

**UNSUPERVISED APPROACHES TO SYNTACTIC
ANALYSIS OF WEB SEARCH QUERIES**

Rishiraj Saha Roy

**UNSUPERVISED APPROACHES TO SYNTACTIC
ANALYSIS OF WEB SEARCH QUERIES**

*Thesis submitted to the
Indian Institute of Technology, Kharagpur
For award of the degree*

of

Doctor of Philosophy

by

Rishiraj Saha Roy

Under the joint supervision of

**Prof. Niloy Ganguly, IIT Kharagpur
and**

Dr. Monojit Choudhury, Microsoft Research India



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DECEMBER 2014

©2014 Rishiraj Saha Roy. All rights reserved.

APPROVAL OF THE VIVA-VOCE BOARD

Date: / / 20

Certified that the thesis entitled “**UNSUPERVISED APPROACHES TO SYNTACTIC ANALYSIS OF WEB SEARCH QUERIES**” submitted by RISHIRAJ SAHA ROY to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Prof. Sudeshna Sarkar)
(Member of DSC)

(Prof. Pabitra Mitra)
(Member of DSC)

(Prof. Anirban Mukherjee)
(Member of DSC)

(Prof. Niloy Ganguly)
(Supervisor 1)

(Dr. Monojit Choudhury)
(Supervisor 2)

(Prof. Vasudeva Varma)
(External Examiner)

(HoD, CSE)
(Chairman)

CERTIFICATE

*This is to certify that the thesis entitled “**UNSUPERVISED APPROACHES TO SYNTACTIC ANALYSIS OF WEB SEARCH QUERIES**”, submitted by **RISHI-RAJ SAHA ROY** to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy, is a record of bona fide research work carried out by him under our supervision and guidance. The thesis, in our opinion, is worthy of consideration for the award of the degree of Doctor of Philosophy in accordance with the regulations of the Institute. To the best of our knowledge, the results embodied in this thesis have not been submitted to any other University or Institute for the award of any other Degree or Diploma.*

Niloy Ganguly

Professor

Computer Science and Engineering

IIT Kharagpur

Monojit Choudhury

Researcher

Microsoft Research India

Date:

DECLARATION

I certify that

- a. The work contained in this thesis is original and has been done by myself under the general supervision of my supervisor.
- b. The work has not been submitted to any other Institute for any degree or diploma.
- c. I have followed the guidelines provided by the Institute in writing the thesis.
- d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
- f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

RISHIRAJ SAHA ROY

ACKNOWLEDGMENTS

I earnestly take this opportunity to thank everybody who has contributed directly or indirectly in realizing this thesis. First of all, I want to thank my family, especially my parents, my wife Amrita, and my sister, who have always been by my side and have inspired me and helped me in every possible way during my PhD. I thank IIT Kharagpur and the Department of Computer Science and Engineering (CSE) for providing me with the opportunity of pursuing my PhD, along with research facilities which were of the very best quality. I sincerely thank my supervisors Prof. Niloy Ganguly and Dr. Monojit Choudhury, without whom this thesis would not have taken shape. They have not only helped me with technical guidance on my research, but more importantly, they have taught me several ways of becoming a better individual. They have also helped me with internship opportunities at Microsoft Research India, which has an excellent environment for research. I am grateful to Srivatsan Laxman and Kalika Bali, our collaborators at Microsoft Research India, for providing me with insights from their respective areas of expertise and helping me broaden my perspective. I would like to thank Bo-June (Paul) Hsu and Kuansan Wang (Microsoft Research Redmond, USA), Matthias Hagen (Bauhaus Universität Weimar, Germany), and Rohan Ramanath (Carnegie Mellon University, USA) for helping me at various stages of code and data preparation.

I sincerely thank the undergraduate students who worked with me on various problems over the duration of my PhD – Nikita Mishra, Naveen Kumar Singh, Smith Agarwal, Anusha Suresh, Rahul Katare, Yogarshi Vyas, and Dastagiri Reddy.

I have been extremely fortunate to be a part of the Complex Networks Research Group (CNeRG), headed by Prof. Ganguly at CSE, IIT Kharagpur, where I made several friends. The alumni – Abyayda (the suffix ‘da’ indicates ‘elder brother’ in the Bengali language), Animeshda, Animesh, Bivasda, Joydeepda, Saptarshida and Subratada (Animeshda and Bivasda are currently faculty members in our Department) – have been forever ready to give us advice whenever we have approached them. I would also like to thank the present members – Abir, Koustav, Mayank, Parantapa, Rajibda, Sandipan, Soumajit, Souravda, Souvik, Sudiptada, Suman, Swadhin and Tanmoy – who make the CNeRG Lab a vibrant and enjoyable place. I am particularly grateful to Saptarshida, for greatly helping me write this thesis. I would like to thank my friends Arghya, Bibhas, Dharmendra, Subhasish, and especially Sumanta for making my stay in our hostel (Acharya Jagadish Chandra Bose Hall of Residence) a fun-filled and memorable one.

I express my sincere gratitude to the faculty members and staff of CSE, IIT Kharagpur. I was supported by Microsoft Corporation and Microsoft Research India under the Microsoft Research India PhD Fellowship Award. I am thankful to several organizations such as Microsoft Research India and NIXI, and to the organizers of conferences such as Evolang, SIGIR and WWW, for financial assistance towards my attending the conferences.

Finally, I am grateful to God for his blessings and for giving me the strength to persevere throughout the long and arduous journey towards a PhD.

Rishiraj Saha Roy

Kharagpur, India

ABSTRACT

The co-evolution of the Web and commercial search engines, and the inability of such search engines to process natural language (NL) questions, have resulted in search queries being formulated in a syntax which is more complex than a bag-of-words model, but more flexibly structured than sentences conforming to NL grammar. In this thesis, we take the first steps to understand this unique syntactic structure of Web search queries in an unsupervised framework, and apply the acquired knowledge to make important contributions to Information Retrieval (IR). First, we develop a query segmentation algorithm that uses query logs to discover syntactic units in queries. We find that our algorithm detects several syntactic constructs that differ from NL phrases. We proceed to augment our method with Wikipedia titles for identifying long named entities. Next, we develop an IR-based evaluation framework for query segmentation which is superior to previously employed evaluation schemes against human annotations. Here, we show that substantial IR improvements are possible due to query segmentation. We then develop an algorithm that uses only query logs to generate a nested (or hierarchical) query segmentation, where segments can be embedded inside bigger segments. Importantly, we also devise a technique for directly applying nested segmentation to improve document ranking. Subsequently, we use segment co-occurrence statistics computed from query logs to find that query segments broadly fall into two classes – content and intent. While content units must match exactly in the documents, intent units can be used in more intelligent ways to improve the quality of search results. More generally, the relationship between content and intent segments within the query is vital to query understanding. Finally, we generate large volumes of artificial query

logs constrained by n -gram model probabilities estimated from real query logs. We perform corpus-level and query-level comparisons of model-generated logs with the real query log based on complex network statistics and (crowdsourced) user intuition of real query syntax, respectively. The two approaches together provide us with a holistic view of the syntactic complexity of Web search queries which is more complex than what n -grams can capture, but yet more predictable than NL.

Keywords: Query understanding, Query syntax, Query segmentation, Query intent, Query complexity

Contents

Table of Contents	xv
Author’s Biography	xix
List of Figures	xxiii
List of Tables	xxv
1 Introduction	1
1.1 Motivation	1
1.2 The Bing query log data	4
1.3 Functional aspects of Web search queries	6
1.4 Structural aspects of Web search queries	9
1.5 Dynamical aspects of Web search queries	9
1.6 Objectives and approach of the thesis	11
1.7 Contributions of the thesis	13
1.8 Organization of the thesis	15
2 Literature Review	17
2.1 The bag-of-words model and beyond	18
2.2 Identifying syntactic units	23
2.3 User intent and role annotation	27
2.4 Deeper syntactic analysis	31
2.5 Model-generated queries	33
2.6 Queries as a distinct language	36
2.7 Scope of further work	37
3 Discovering Syntactic Units by Flat Query Segmentation	41
3.1 Introduction	41
3.2 Query segmentation algorithm	43
3.2.1 Enhancement with Wikipedia titles	45
3.3 Query segmentation evaluation	47
3.3.1 The evaluation framework	48
3.4 Dataset and compared algorithms	53

3.4.1	Test set of queries	53
3.4.2	Document pool and RJs	55
3.4.3	Segmentation algorithms	57
3.4.4	Public release of data	57
3.5	Experiments and observations	57
3.5.1	IR experiments	58
3.5.2	Performance under traditional matching metrics	60
3.5.3	Inferences	62
3.6	Related issues	66
3.6.1	Motivation for a new dataset	66
3.6.2	Retrieval using Bing	67
3.6.3	Inter-annotator agreement	69
3.7	Enhancement with POS tags	70
3.7.1	POS tagging	71
3.7.2	Lexicon augmentation scheme	72
3.7.3	Experiments	74
3.8	Conclusions	77
4	Discovering Syntactic Units by Nested Query Segmentation	79
4.1	Introduction	79
4.2	Issues with flat segmentation	82
4.2.1	Limitations of flat segmentation	82
4.2.2	Advantages of nested segmentation	83
4.3	Terms and definitions	84
4.3.1	Types of segmentation	85
4.3.2	Types of distances	85
4.4	Algorithm for nested query segmentation	87
4.4.1	Splitting flat segments to discover syntax within a flat segment	89
4.4.2	Joining flat segments to discover syntax across flat segments	90
4.5	Using nested segmentation in IR	92
4.5.1	Re-ranking using the tree and document distances	92
4.5.2	Rank aggregation of original and new ranks	94
4.5.3	Re-ranking baselines	95
4.6	Datasets	96
4.6.1	For performing nested segmentation	96
4.6.2	For re-ranking documents	96
4.7	Experiments and results	98
4.7.1	Experimental setup	99
4.7.2	Results and observations	99
4.7.3	Systematic investigation of variations in algorithm	105
4.7.4	Comparison with past work	107
4.8	Related research	110
4.9	Conclusions	111

5	Role Induction of Syntactic Units: Content and Intent	113
5.1	Introduction	113
5.2	Distributional properties of NL function words	115
5.2.1	Datasets	116
5.2.2	Metric	117
5.2.3	Frequency as a function word indicator	117
5.2.4	Co-occurrence statistics as function word indicators	119
5.2.5	Inverse document frequency	123
5.3	Intent units of Web search queries	126
5.3.1	Operational definitions	126
5.3.2	Experimental results	129
5.4	Labeling intent units in query context	132
5.4.1	Evaluating in-query labeling using human annotations	133
5.4.2	Evaluating in-query labeling using clickthrough data	136
5.4.3	Verification of the operational definitions	140
5.4.4	Use of content and intent labeling in IR	143
5.5	A taxonomy of intent units in Web search queries	146
5.6	Related work	149
5.6.1	Intent units as explicit facet indicators	151
5.7	Conclusions	151
6	Understanding Syntactic Complexity of Web Search Queries	153
6.1	Introduction	153
6.2	Statistical language modeling for queries	156
6.2.1	Query generation process	156
6.2.2	Measuring model perplexity	159
6.2.3	Experimental results	159
6.2.4	Interpretation	160
6.3	Complex network modeling for queries	161
6.3.1	Network definition and construction	162
6.3.2	Topological properties of WCNs	163
6.3.3	Stability of WCNs	166
6.3.4	Comparison of real and model-generated query WCNs	169
6.4	User intuition of real queries	173
6.4.1	Experimental setup using crowdsourcing	174
6.4.2	Results and observations	176
6.4.3	Interpretation	178
6.5	Discussion	179
6.6	Synthetic Web search queries	181
6.7	Conclusions	184

7	Conclusions and Future Work	187
7.1	Summary of the contributions	187
7.2	Directions of future work	192
7.3	Final words	193
	Bibliography	195

Author's Biography

Rishiraj Saha Roy received a B.E. (Hons.) degree in Information Technology from Jadavpur University, Kolkata, in 2007, and an M.Tech. degree in Information Technology from Indian Institute of Technology Roorkee in 2009. He has been pursuing Ph.D. at the Department of Computer Science and Engineering, IIT Kharagpur, since December 2009. He was awarded the Microsoft Research India Ph.D. Fellowship in 2011. He is a part of the Complex Network Research Group (CNeRG) at IIT Kharagpur. Even though his Ph.D. is on Web search query log analysis, his general research interests include Information Retrieval, Complex Networks, Machine Learning, Text Mining, Natural Language Processing, and Linguistics.

Publications made out of this thesis

(listed in reverse chronological order)

1. Rishiraj Saha Roy, Smith Agarwal, Niloy Ganguly and Monojit Choudhury, “Syntactic Complexity of Web Queries through the Lenses of Language Models, Networks and Users”, *communicated to PLOS ONE*, Public Library of Science (under review). (Long Paper)
2. Rishiraj Saha Roy, Anusha Suresh, Niloy Ganguly and Monojit Choudhury, “Nested Query Segmentation for Information Retrieval”, *communicated to ACM Transactions on Information Systems (TOIS)* (under review). (Long Paper)
3. Rishiraj Saha Roy, Rahul Katare, Niloy Ganguly, Srivatsan Laxman and Monojit Choudhury, “Discovering and Understanding Word Level User Intent in Web Search Queries”, in *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, 2014 (in press). (Long Paper)
4. Rishiraj Saha Roy, Yogarshi Vyas, Niloy Ganguly and Monojit Choudhury, “Improving Unsupervised Query Segmentation using Parts-of-Speech Sequence Information”, in *Proceedings of the 37th Annual ACM SIGIR Conference on Research and*

- Development on Information Retrieval (SIGIR '14)*, 6 – 11 July 2014, Gold Coast, Australia, pages 935 – 938. (Short Paper)
5. Rishiraj Saha Roy, Rahul Katare, Niloy Ganguly and Monojit Choudhury, “Automatic Discovery of Adposition Typology”, in *Proceedings of the 25th International Conference on Computational Linguistics (Coling '14)*, 23 – 29 August 2014, Dublin, Ireland, pages 1037 – 1046. (Long Paper)
 6. Rishiraj Saha Roy, M. Dastagiri Reddy, Niloy Ganguly and Monojit Choudhury, “Understanding the Linguistic Structure and Evolution of Web Search Queries”, in *Proceedings of the 10th International Conference on the Evolution of Language (Evolang X)*, 14 – 17 April 2014, Vienna, Austria, pages 286 – 293. (Long Paper)
 7. Rohan Ramanath, Monojit Choudhury, Kalika Bali and Rishiraj Saha Roy, “Crowd Prefers the Middle Path: A New IAA Metric for Crowdsourcing Reveals Turker Biases in Query Segmentation”, in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, 4 – 9 August 2013, Sofia, Bulgaria, pages 1713 – 1722. (Long Paper)
 8. Rishiraj Saha Roy, Anusha Suresh, Niloy Ganguly and Monojit Choudhury, “Place Value: Word Position Shifts Vital to Search Dynamics”, in *Posters of the 22nd International World Wide Web Conference 2013 (WWW '13)*, 13 – 17 May 2013, Rio de Janeiro, Brazil, pages 153 – 154 (*companion*). (Short Paper)
 9. Rishiraj Saha Roy, “Analyzing Linguistic Structure of Web Search Queries”, in *Doctoral Consortium of the 22nd International World Wide Web Conference (WWW '13)*, 13 – 17 May 2013, Rio de Janeiro, Brazil, pages 395 – 399 (*companion*). (Short Paper)
 10. Rishiraj Saha Roy, Niloy Ganguly, Monojit Choudhury and Srivatsan Laxman, “An IR-based Evaluation Framework for Web Search Query Segmentation”, in *Proceedings of the 35th Annual ACM SIGIR Conference on Research and Development on Information Retrieval (SIGIR '12)*, 12 – 16 August 2012, Portland, USA, pages 881 – 890. (Long Paper)
 11. Rishiraj Saha Roy, Monojit Choudhury and Kalika Bali, “Are Web Search Queries an Evolving Protolanguage?”, in *Proceedings of the 9th International Conference on the Evolution of Language (Evolang IX)*, 13 – 16 March, 2012, Kyoto, Japan, pages 304 – 311. (**Best Research Poster Award**) (Long Paper)
 12. Rishiraj Saha Roy, Niloy Ganguly, Monojit Choudhury and Naveen Kumar Singh, “Complex Network Analysis Reveals Kernel-Periphery Structure in Web Search Queries”, in *Proceedings of the 2nd ACM SIGIR Workshop on Query Representation and Understanding 2011 (QRU '11)*, 28 July 2011, Beijing, China, pages 5 – 8. (Short Paper)

-
13. Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman and Monojit Choudhury, “Unsupervised Query Segmentation Using only Query Logs”, in *Posters of the 20th International World Wide Web Conference 2011 (WWW '11)*, 28 March - 1 April 2011, Hyderabad, India, pages 91 – 92 (*companion*). (Short Paper)

List of Figures

1.1	Long queries pose a challenge to commercial Web search engines.	2
1.2	A snapshot of our Bing query log.	5
1.3	Query length distributions on our query log.	5
1.4	Analogy of queries using a search engine as a medium.	6
3.1	Distribution of multiword segments across segmentation strategies.	64
3.2	Augmenting unsupervised query segmentation with POS sequences.	71
4.1	Nested segmentation tree.	80
4.2	Illustrating our approach for nested query segmentation.	89
4.3	Variation in parameters on SGCL12.	104
4.4	Variation in parameters on TREC-WT.	105
4.5	Performance examination of algorithm variations on both datasets.	106
5.1	AP@500 with frequency as the function word indicator for English.	118
5.2	Performance of co-occurrence statistics with respect to frequency.	122
5.3	Evaluation of labeling units against human markup.	135
5.4	Illustrating difference in click overlaps.	138
5.5	Evaluation of labeling units against click data.	139
5.6	A Venn diagram for the intent unit taxonomy.	147
6.1	Illustration of a WCN for queries.	163
6.2	Sample CDD for a real query log.	164
6.3	Connected Ψ^3 and Ψ^4 motifs with LNMCs from the real log.	166
6.4	Sample CDD at different network sizes.	168
6.5	Sample CDD for query LMs.	171
6.6	(Colour online) Degree distributions for word and segment networks.	171
6.7	Solved examples for annotation task posted on AMT.	176
6.8	Industry authors in papers on query log analysis.	182

List of Tables

2.1	Corpus details for term dependence and proximity models.	22
3.1	Example of generation of quoted versions for a segmented query.	50
3.2	Segmentation algorithms compared on our framework.	56
3.3	Oracle results for IR-based evaluation of segmentation algorithms.	58
3.4	Matching metrics with <i>BQV</i> as reference.	59
3.5	PMI-Q and Li et al. [137] with respect to matching and IR metrics.	59
3.6	Kendall-Tau coefficients between IR and matching metrics.	62
3.7	Effect of β on IR performance.	66
3.8	IR-based evaluation using Bing API.	68
3.9	Inter-annotator agreement on features as observed from our experiments. . .	69
3.10	Sample clusters produced by Bie-S.	72
3.11	IR performance with different tagsets.	75
3.12	Gaining, unaffected and losing queries.	75
3.13	Example queries showing IR improvement.	76
3.14	Number of iterations and the optimal α	76
4.1	Joining and splitting of flat segments for nested segmentation.	88
4.2	Penalty cases for query word pairs.	93
4.3	Details of datasets used.	97
4.4	Examples of nested segmentations for queries.	99
4.5	Performance comparison of flat and nested segmentations.	100
4.6	Break-up of nDCG gains (over unsegmented query) by length.	102
4.7	Performance of re-ranking strategies.	103
4.8	List of parameters used in re-ranking.	103
4.9	Comparison with Huang et al. [99].	108
4.10	Height distributions for nested segmentation tree.	109
5.1	Details of NL corpora.	116
5.2	Definitions of the different function word indicators.	119
5.3	AP for frequency and co-occurrence statistics.	121
5.4	Comparison of IDF with other indicators for AP@200.	125
5.5	Sample units at top ranks when sorted in descending order by TCE.	127
5.6	AP of each of the indicators for intent unit detection in Web queries.	130

5.7	Ranks assigned to query intent units by the seven different statistics.	131
5.8	General examples of segmented and labeled queries.	134
5.9	Segment match versus document relevance for content units.	142
5.10	Segment match versus document relevance for intent units.	142
5.11	IR evaluation (nDCG@10) of content-intent labeling using the Bing API. . .	144
5.12	Examples of intent units for the <i>restrict</i> class.	148
5.13	Examples of intent units for the <i>rank</i> class.	149
5.14	Examples of intent units for the <i>mixed</i> class.	150
6.1	Queries generated by different models.	157
6.2	Perplexity and counts of n -grams and n -terms.	160
6.3	Network statistics for various network sizes.	167
6.4	Ψ^3 and Ψ^4 Motif signatures at various network sizes (local co-occurrence). .	169
6.5	Mean network statistics for the query LMs.	169
6.6	Ψ^3 and Ψ^4 Motif signatures for the query LMs.	170
6.7	Mean network statistics for word and segment models.	172
6.8	AMT experiment details.	177
6.9	A summary of absolute ratings obtained through AMT.	178
6.10	A summary of relative ratings obtained through AMT.	179

Chapter 1

Introduction

A Web search engine is popularly perceived as a *Window to the Web* – millions of users around the world interact everyday with search engines to satisfy diverse information needs. Web users communicate their information need to a search engine through *queries*. Over the years, as the Web has grown larger with more and more detailed information now available online, requirements of users have also become more complex. These two processes are highly intertwined, and the co-evolution of the Web and search engines has resulted in user queries being formulated in a unique linguistic style, markedly distinct from the parent natural language (NL) in which the Web documents are composed. The fact that search engines do not really “understand” or “process” NLs drives average Web users to specify their queries in a language that has a syntax far simpler than NL, but perhaps more complex than the commonly assumed bag-of-words model.

1.1 Motivation

Despite substantial progress in the field of information retrieval (IR) in the last thirty years, commercial search engines like Bing and Google still rely heavily on the degree of query term match to assess the relevance of documents. In other words, a document that contains a higher number of query words is likely to be more relevant to the query. This idea

WEB IMAGES VIDEOS NEWS MAPS MORE

bing ms office guide book buy online

2020,00,000 RESULTS Narrow by language Narrow by region

Buy / Sell Books Online - Free local OLX classifieds. Ads
www.OLX.in
Search and post classified ads. Try Now!

Sell On Snapdeal.com
sellers.snapdeal.com
Setup Your **Online** Store on Snapdeal Sell Your Product to 20mn Consumers

Yatra.com™ Official Site
Domesticflights.yatra.com
Get Cheap Rates On All Airlines For Limited Period Only. **Book & Save.**

Microsoft Office - Microsoft Word, Outlook & Excel...
office.microsoft.com/en-in
Office Downloads · Try Office 365
Free Microsoft Office clip art, images, templates, how-to articles, downloads, help, and training for Microsoft Office Word, ... OFFICE ONLINE

Microsoft® Office Specialist Study Guide Office 2003 ...
www.microsoft.com/learning/en-us/book.aspx?id=7389
23-06-2004 · Take an **online** skills test ... as well as dozens of **books** in the Quick Course ... **Microsoft Office Specialist Study Guide**. 2007 Microsoft Office ...

Office - Office.com
office.microsoft.com
Try or buy Office 365 for Home or Business, ... and on the web with Office Online for everywhere in between. ... Microsoft Store; Follow us. Office Blogs; Twitter;

Microsoft Press Books
www.microsoft.com/learning/en-us/microsoft-press-books.aspx
... and **online books** allow you to access the information you need to learn ... Training **Guide books**. ... Review our proposal guidelines in Microsoft Office Word ...

Buy MS Office 2010 Training Guide by : MS Office 2010 ...
www.infbeam.com/Books/office-2010-training-guide-s-jain/...
Buy MS Office 2010 Training Guide book by online. MS Office 2010 Training Guide book price, MS Office 2010 Training Guide reviews & ratings, ISBN: 8183334068, EAN ...

Buy Microsoft Office 2013 suites and Office 365 ...
office.microsoft.com/en-us/buy
OFFICE ONLINE ... Choose ... Buy Office for Business . Compare Office suites; Common questions; Free Office trial; System requirements; Office 365 University. At ...

PAID SEARCH RESULTS

Not Applicable

Non-relevant

Relevant

Non-relevant

Relevant

Non-relevant

ORGANIC SEARCH RESULTS

Figure 1.1: Long queries pose a challenge to commercial Web search engines.

follows from the *bag-of-words model*, where both the query and the document are treated as unordered sets of constituent terms. In practice, this principle is used only to select the candidate set of relevant documents, and the final ranked list of results is produced by using more sophisticated algorithms like PageRank [39] (to assess the importance of the website hosting the document) or by leveraging additional user information like clicks, page dwell times and reformulations [109]. This approach generally works quite well for short queries (say, up to about three words) and frequent queries (also referred to as *head queries*) for which the search engine quickly learns about the preferred pages from user behavior. However, quality of search results degrade noticeably when the queries are relatively infrequent (also called *tail queries*) and slightly longer (say, four to ten words). We note that query terms being rare, by itself, is not a sufficient criterion to pose difficulty for a search engine, as it is quite possible that the pages containing these terms are also quite rare and hence leave no scope for confusion. But when the queries are long, it is possible that some of the

words of the query are frequent and also reside on popular websites, which subsequently results in upward shifting of non-relevant pages in the final ranked list. The lack of sufficient user information caused by the rarity (of the query as a whole) therefore makes matters difficult for the search engine. Obtaining the ideal ranking is crucial to user satisfaction because it is known that the searcher usually examines only the top few results properly, and only glances over the lower results, a phenomenon usually known as the *position bias* [54].

We present our case with a representative example in Figure 1.1. The figure shows results obtained through the Bing search engine for the query `ms office guide book buy online` (issued on 28 February 2014). We note that two types of results are usually presented in response to a user query: *organic* (or natural) results, and *paid* results (or advertisements). While the former are retrieved from the World Wide Web by the search engine’s usual algorithm, sponsors pay search engines to get their advertisement pages displayed when certain keywords appear in the query. In this research, we are concerned only about organic search results. Coming back to our example, we observe that the *user intent* behind the query is to buy a reference book for using Microsoft Office, online. However, results at ranks one, three and six all intend to sell the Microsoft Office software instead, and hence are clearly non-relevant to the user. This is caused by the fact that the non-relevant pages also contain most of the query words like `microsoft`, `office`, `buy` and `online`, and these pages come from a trusted website like office.microsoft.com.

Such challenges can be addressed if one tries to look deeper into the query, and identifies structural relationships that were previously overlooked due to the oversimplification of the bag-of-words model. By the identification of “structural relationships”, we refer to processes that involve locating sets of words that are mutually related. Also, it is important to understand dependencies between these sets of words with respect to the original user intent. Revisiting our example, it is intuitive that the above problems could be minimized if one knew that the `microsoft office guide book` was the actual object of interest, and not `microsoft office` itself. Also, inside the unit, `microsoft office` and `guide book` are expressions whose words could not be permuted freely in the document. Next, the `guide book` is the *topic* of the query and the words `buy online` are added by the user to formulate his/her intent more specifically. For example, the user would not be interested to `read` the book online. So, knowing the topic of the query along with the user intent, the search engine could understand that only matching the words `microsoft`,

office, buy and online, without the word guide or book (or its synonyms), would not be meaningful. Further, words like buy and online need not even match in the document – for example, the catalogue page of a relevant book in an e-commerce store would be a very relevant result.

The fundamental goal of this research is to carefully scrutinize the proposition stated above through *syntactic analysis of queries*, and to leverage our findings to improve IR. Wherever applicable, we will focus on *unsupervised* techniques that are generally applicable to queries from all domains. Supervised techniques in Web query analysis typically face the non-trivial challenge of building a good human-labeled training corpus that has appropriate coverage from several query domains [168]. A complete linguistic study encompasses the dimensions of *structure*, *function* and *dynamics* [8, 51]. Structure, in turn, deals with aspects of syntax (study of principles by which sentences are constructed) and semantics (study of meanings associated with sentences) [47, 49]. We have observed interesting conceptual similarities and differences between search queries and NLs on all the three perspectives, which we shall discuss in this chapter. But in this thesis, we prefer to investigate deep into a single aspect, and focus on query syntax. Thus, we aim to understand syntactic relationships between individual words and groups of words within queries¹. But before we proceed, it is necessary to describe our query log data.

1.2 The Bing query log data

For all our experiments, we use a query log sampled from Bing Australia² in May 2010. This raw data slice consists of 16.7M ($M = \text{Million}$) queries. Each query is accompanied by a clicked URL, a unique hash of the URL, and the click count (number of times the URL was clicked by users for the query). A random snapshot of the data showing distinct queries is presented in Figure 1.2. We subsequently extracted 11.9M queries from the raw data such that the queries were composed of ASCII characters only and were of length between two and ten words. The justification for imposing a filter based on query length is as follows. One word queries do not show evidence of syntax, and very long queries

¹By *queries*, *search queries*, *Web search queries* and *Web queries*, we will refer to the same concept.

²<http://www.bing.com/?cc=au>, Accessed 19 May 2014.

google service station for sale in sydney	http://www.business2sell.com.au/	3B02132E1295C8CE94A0	1
gor mulder skinner slavefic	http://www.squidape.org/peja/cai-bin/viewuser.php?uid=233	DC2E88CB8CD2B295747BB	1
gordan lightfoot gords gold blogspot	http://temamuzik.blogspot.com/2008/06/gordon-lightfoot-gords-qold-1975.html	949466FE1673CDDB2FD5	1
gosford car sales manns rd	http://www.aussieweb.com.au/details.aspx?id=1451072	FC60A17F3D32B92FEF6F	2
gosford hospital address	http://www.directory.nsw.gov.au/showOrqUnit.asp?id=%7BF9147229-45DC-4A49	868A6206CEE84D563311	2
govment one payments in feb	http://www.treasurer.gov.au/DisplayDocs.aspx?doc=factsheets/2009/001.htm	C82AC1E43146D2985271	1
gpo hide network icon	http://www.petri.co.il/forums/showthread.php?t=33466	FCE2E87D5F01EAEDE662	1
gps receiver for ipax hx2790	http://www.audion-mm.com/v2/index.php?option=com_content&task=view	C88079B8BD078	2
graceful bamboo potted photo	http://www.shweeashbamboo.com/Bamboo%20Care%20and%20Maintenance.htm	5D810902639F50A3F9C2	1
grade 3 science only animals	http://www.rrsd.mb.ca/LICT/grade%203/plantsLE.doc	5A361C40F5026DA328A3	1
gradient gif	http://au.geocities.com/kiwichickieddiemae/colournoise.html	0DDBC0067640E453A93	1
graduate legal positions 2010	http://www.jobs.com.au/legal-jobs/6917750/articled-traineeship-2010/	E604F8D816F1D40F9F7D	3
graduate recruitment program det	http://www.teach.nsw.edu.au/grp/	ED6EA8EE13AC4A7C1F1E	1
graduate teacher placements perth wa	http://employment.byron.com.au/jobs/education-jobs-in-wa-goldfields.html	769069D677F12A597A3B	1
grame gibbons photos	http://garamegibbonsphotos.com/	9FEA39A63A8AE6ACC80	1
graffiti drawings	http://www.youtube.com/watch?v=3B9ozArubn0	DB7C683DC4FF14BF1742	4
graffiti removal jobs vacant gold coast	http://mycareer.com.au/queensland-rail/profile/	A8A90163099FA76F30CE	1
gramd theft auto iv cheats	http://cheats.ign.com/ob2/068/827/827005.html	43D0124BB6AEFF5912CC	1
grams in a tsp	http://wiki.answers.com/Q/How_many_grams_in_a_tsp_of_sugar	2F372317B328D1F80A33	1
grand theft auto vice city stories website	http://en.wikipedia.org/wiki/Grand_Theft_Auto:_Vice_City_Stories	CB5AFF5E8536DACD224A2	1
grand traverse michigan	http://www.grandtraversemall.com/html/Mallinfo.asp	6FD48C94720699C73A5D	1
grand vitara 2007 parts	http://www.autopartswarehouse.com/models/suzuki*grand_vitara*makemodel.html	EE3BD2C4D626BD43AA2F	1
grandmix 2008 torrent	http://www.torrentreactor.net/torrents/2691735/Ben-Liebrand-Grandmix-1983-2008	26E90582D06A1852A515	1

Figure 1.2: A snapshot of our Bing query log.

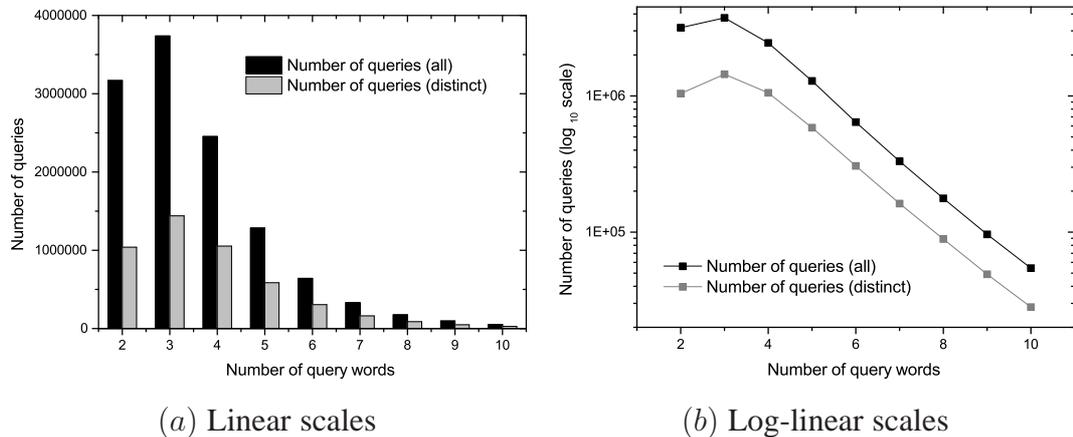


Figure 1.3: Query length distributions on our query log.

(having more than ten words) are typically machine generated messages or excerpts from NL text, and need separate query processing techniques. There are 4.7M unique queries among the extracted 11.9M queries – but in order to preserve the log properties arising out of the natural power law frequency distribution of queries [170], duplicates were retained for all experiments. The length distribution (defined in terms of number of words per query) for our extracted set of 11.9M queries is shown in Figure 1.3. The mean query lengths for this set are 3.58 words (all queries) and 3.77 words (distinct queries).

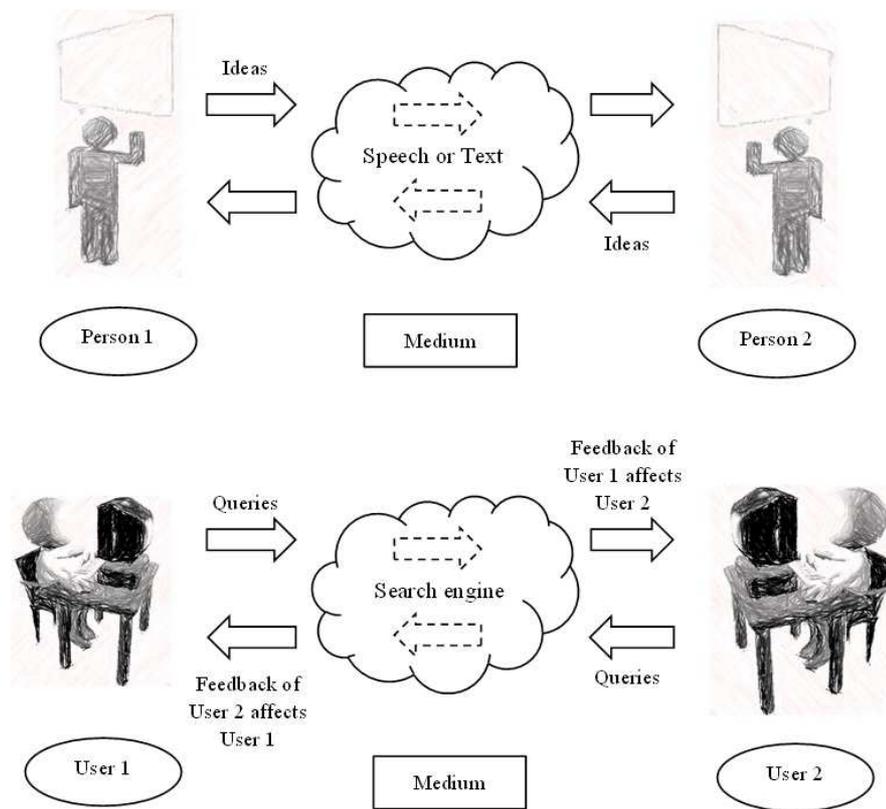


Figure 1.4: Analogy of queries using a search engine as a medium.

1.3 Functional aspects of Web search queries

We will now present a comparison between queries and NLS from structural, functional and dynamical perspectives. We present a synthesis of the salient ideas emerging from these three orthogonal perspectives on queries that point to the strong parallels between Web search queries and a rudimentary language system, often termed as *protolanguage*. At the very outset we want to note that there is a fundamental difference between the function of human language and that of queries: while human language is used for communication between two human beings (presumably) having very similar cognitive capabilities, queries are used as the means of communication between a human user and a search engine, which are incomparable not only in terms of their cognitive capabilities, but also in their biological and cultural history of language use. This asymmetry between the communicating agents in the context of queries can raise serious doubts about our basic proposition of queries being

a distinct language. However, we believe that there is an alternative and perhaps a more realistic interpretation of this communicative behavior, which is to assume that ultimately queries are actions of the users on a shared environment represented by the search engine. While the users might believe that they are communicating their information need to the search engine through their queries which responds with answers (which of course is true), the search engine's response is dependent completely on the actions and responses of other users in the past (Figure 1.4). This is especially true for modern commercial search engines which extensively rely on user queries, URL clicks and explicit feedbacks on relevance of documents for learning and improvement of the search models. Thus, we can visualize this situation as an indirect communication between two human users mediated through a shared environment (or channel) which is the search engine. It is a well-known fact that the channel (such as the structure of our articulatory and perceptual devices) has a profound effect on the structure and dynamics of the evolving language.

Web search queries are small fragments of texts (symbols) that are used to communicate the information need of an individual to a search engine. In this regard, the basic function of queries is similar to that of languages, which is transmission of information. Hockett [93] proposed thirteen design features of a communication system. NLS possess all these features and in this section we see that a large number of these features are present in queries as well. Some of these features, such as *semanticity*, *arbitrariness*, *discreteness* and *duality of patterning*, are exhibited in queries by virtue of the fact that the building blocks of queries are, after all, words – which are also the basic units of NLS. However, with respect to some of the other features, NLS and queries are analogous in their function. We discuss the other design features here.

Vocal-auditory channel: All spoken human language is produced using the vocal tract and auditory channel. While the role of vocal-auditory channel is currently irrelevant for queries, they are produced and perceived by writing (typing) and reading of text.

Broadcast transmission and directional reception: Human language can be heard if it is within the range of another person's auditory channel. Additionally, a listener, who shares the same time and space of the speaker, has the ability to determine the source of a sound by binaural direction finding. In the case of Web search, queries issued by a user are recorded in the search engine log files. The engine uses these logs to generate *query*

completions for another user. This way, a query can be potentially broadcast to millions of new users. A new user can *choose* to be receptive of these completions (similar to signals) by enabling this specific feature on their search engine.

Rapid fading: Waveforms of spoken language dissipate over time and do not persist. A hearer can only receive specific auditory information at the time it is spoken. This feature is related to the modality of language, and as queries are mainly textual, they are therefore less ephemeral than spoken language.

Interchangeability: A person has the ability to both speak and hear the same signal. Anything that a person is able to hear, s/he has the ability to reproduce through spoken language. Similarly, users have the ability to understand and reformulate somebody else's query. If a person has seen a query, s/he can also use that query.

Total feedback: Speakers have the ability to hear themselves speak. Through this, they are able to monitor their speech production and internalize what they are producing through language. Searchers also know what queries they have issued, and can monitor them and internalize their uses.

Specialization: Human language sounds are specialized for communication, that is, humans speak mainly to transmit information. Query words too are specialized for specific information needs of the user.

Displacement: NL has the ability to refer to things in space and time and communicate about things that are currently not present. Queries also allow the users to seek information about past and future events or objects.

Productivity: NL allows for the creation of new and unique meanings of utterances from previously existing utterances and sounds. Likewise, a pre-existing set of distinct words (around $1.2M$ in our dataset) can be combined to formulate unseen queries.

Traditional transmission: Human language is not completely innate and acquisition depends in part on the learning of a language. Searchers can also learn how to formulate queries from search experts, search engine guidelines, guide books [183], search engine feedback and mimicking other users.

Thus, from a purely functional perspective, Web search queries are very similar to NL.

1.4 Structural aspects of Web search queries

Barr et. al. [22] were one of the first to study the linguistic structure of Web search queries. They examine the applicability of part-of-speech tagging to English Web queries and the value of such tagging for enhancing search results. They show that 70% of all words in queries are nouns, followed by adjectives (7.1%) and prepositions (3.7%). They also show that a very large percentage of queries are noun phrases, and not unstructured collections of terms. They manually label a set of queries with these tags to train a Brill tagger, and achieve only about 70% accuracy, the poor figure reflecting the unique syntactic structure of queries. They also try to classify search queries into grammatical classes based on the syntax of part-of-speech tag sequences. They also scrutinize the practical applicability of leveraging query-trained part-of-speech taggers for IR applications. They show that part-of-speech information can be a useful feature in machine-learned search result ranking. Their experiments also include the potential use of the tagger in selecting words for omission or substitution in query reformulation. They conclude that leveraging the unique linguistic structure of web-search queries can improve search experience. This thesis deals with structural aspects of search queries, and Sections 2.2 through 2.4 provide a more detailed review of this space.

1.5 Dynamical aspects of Web search queries

Search engines are *complex adaptive systems* that are able to communicate with humans and evolve at two levels – algorithms and models. Search engines have come a long way since the first generation search systems [145]. Even though search engine companies rarely publish parts of their internal algorithms, the huge volume of Web IR literature over the last decade is an indication enough that search algorithms have evolved. Algorithmic evolution for search engines include more sophisticated machine learning algorithms for ranking and use of a higher number of features for retrieval. These changes are analogous

to agents undergoing anatomical modifications over several years – like the unique structure of the vocal tract and the descended position of the larynx in humans [87].

Evolution of the model, on the other hand, is information that the search system learns from user interactions to present better results in the future. Models are learnt by the search engine through constant user feedback and preferences gathered during the course of Web searches. More and more Web data is crawled to make better document models. More query logs are used to build better query analysis models. Query log analysis can be used to study individual search behavior, query duplications, user sessions and query correlations [204]. Clickthrough data [109, 229] and pseudo-relevance feedback [237] are also used by search engines to enrich their models of relevance.

This evolution of models is similar to the cultural transmission of language. Adaptation of the search engine is a population-level phenomenon, where individual users are agnostic to the fact that their interactions with the search engines indeed affect the response of the search engine for other users and vice versa. Cultural transmission for the language of queries can be considered from two aspects: Experts teaching novices how to search, and new users learning search tips and tricks from the collective knowledge of the Web, or relevant books [183] – like traditional language transmission. An individual's competence in language is derived from data which is itself a consequence of the linguistic competence of other individuals [206]. Modern theories of cultural evolution recognize that cultural traditions are socially transmitted from person to person between and within generations [212]. Individual click data and search engine usage affect the engine as a whole. Users unknowingly affect the response of the engine towards other users, effectively transmitting information of some kind through the engine (Figure 1.4).

Incorporation of user feedback has tremendously improved the performance and perception of the popular commercial search engines. While the algorithmic components of a search engine rarely make any attempt to understand NL or complex queries, search engines can intelligently process very complex queries just by learning from past user behavior. This gives an average user the impression that the search engine is indeed getting smarter, and consequently they are motivated to formulate more complex queries. This results in a population-level snowball effect leading to increase in the structural complexity of the queries.

Summary of linguistic analogy. We have highlighted some similarities that Web search queries are observed to share with an evolving language, giving evidence from three different aspects. First, the function and some of the basic features of queries are similar to that of NL. Second, the structure of the queries are in between that of a random bag-of-words model and a full-fledged NL form. Nevertheless, this syntax seems to be evolving in complexity. Third, the evolutionary dynamics of queries is analogous to models of cultural evolution for language. Although this evolution is actually an outcome of the interactions between the users via the search engine, it seems as though the search engine is itself evolving in this process. Given this context, it is an interesting research problem to understand the linguistic syntax of queries, which may lay down stepping stones for obtaining new insights on the evolution of human language.

1.6 Objectives and approach of the thesis

The concrete objectives of this thesis are three-fold, and are stated below:

(1) To discover the syntactic units of Web search queries.

Web search queries have evolved to follow a unique syntactic structure that represents user intent in a way distinct from NL document syntax. We wish to discover this underlying syntactic structure by analyzing large volumes of query logs. Specifically, we wish to detect relationships between words within a query, and to understand how such relationships can be utilized to produce a more informed representation of the query than the simple bag-of-words model.

(2) To understand the roles played by such units in search queries.

Once we discover the syntactic units of search queries, we wish to understand the roles these units play with respect to the search process. For example, nouns and verbs play different roles in a typical NL sentence, and understanding these roles is vital to the correct interpretation of the sentence. Similarly, we expect that different classes of units in queries will have different roles, and appropriate interpretation of these roles will give us new insights into better query understanding and intelligent retrieval.

(3) To understand the complexity of syntactic structure in Web search queries.

Even though it has been a common perception that Web search queries have been growing in syntactic complexity, a precise quantification of the same has been lacking in the literature. In this thesis, we wish to objectively measure the complexity in query syntax from multiple perspectives. Such a study opens up the avenue for understanding the change in query complexity over several years. This would, in turn, enable us to understand the evolution of linguistic syntax for search queries.

The specific approaches that have been followed in the thesis to achieve the above objectives are stated below:

(1) Unsupervised query segmentation using query logs, and its effectiveness at improving information retrieval.

Syntactic units of search queries must be discovered without attempting to project document structure onto queries. This process of dividing queries into their constituent syntactic units is called query segmentation, and all segmentation approaches till date have relied on some form of document resources to accomplish this task. Next, such algorithms do not specify how to independently apply the knowledge of discovered query segments to retrieval, and thus do not leave scope for a generic IR-based evaluation across algorithms. Finally, almost all segmentation algorithms restrict themselves to non-hierarchical structure, i.e. they try to simply partition a query into a non-overlapping sequence of words. We try to address these issues in the following ways: (i) Deduce an unsupervised query segmentation algorithm that primarily uses query logs as the input resource; (ii) Propose an IR-based evaluation framework for query segmentation; (iii) Design an algorithm for hierarchical or nested query segmentation that can discover richer query structure, again relying only on query logs; and, (iv) Propose an approach for deterministically applying the knowledge of hierarchical query segmentation to improve document ranking.

(2) Unsupervised role induction of discovered query segments.

Query segmentation is the first step to query understanding, and its scope goes beyond simple multiword expression detection. The intuitive next stage would be to deduce the roles discovered segments perform in search queries with respect to the retrieval and

ranking processes. Past approaches to such role induction have mostly dealt with specific types of queries, like noun phrase queries and named entity queries, or were supervised approaches restricted to domains like jobs and movies. Our approach here will try to perform the following tasks: (i) Propose a classification scheme for query segments based on their roles in the query, that would be generally applicable for queries from all domains; and, (ii) Develop a lightweight and unsupervised technique for labeling query segments based on our taxonomy.

(3) Analysis of syntactic complexity of search queries.

There has been no quantifying study that discusses the syntactic complexity of search queries. The approach that we take to close this research gap is as follows: (i) Investigate the perplexity of some simple query generation models to understand the syntactic complexity of queries; (ii) Further, investigate the properties of word co-occurrence networks built real and generated queries to obtain a corpus-level understanding of syntactic complexity; and, (iii) Examine human intuition about query-level syntax using crowdsourcing experiments with real and generated queries.

1.7 Contributions of the thesis

In this thesis, we have developed and evaluated unsupervised approaches to query segmentation and segment role induction. We have also tried to quantify the syntactic complexity of Web search queries using multiple perspectives. The specific contributions are summarized below.

(1) Development and evaluation of flat and nested query segmentation algorithms that rely on query logs and a named entity list:

We have proposed an unsupervised method of flat query segmentation that uses Web queries as the primary resource, which helps discover syntactic units of queries that often differ from NL phrases. We have enhanced this technique with Wikipedia titles to detect relatively rarer named entities. Next, we overcame several conceptual challenges to design and implement the first IR-based evaluation framework for query segmentation, that es-

tablished the usefulness of segmentation in IR. Finally, we have proposed an unsupervised algorithm for nested query segmentation and the first strategy to use the nested query representation for improving IR performance. We have shown that the tree structure inherent in the hierarchical segmentation can be used for effective re-ranking of results.

(2) Development and evaluation of a framework for classifying query segments as content or intent:

We have proposed that all query segments can be classified as content or intent. While content segments must match exactly in documents, intent units can act as indicators of user intent and can be used in other ways (like restricting and re-ranking retrieved pages) to improve result quality. Simple counts and entropies of word co-occurrence distributions, estimated from only query logs, can be used for effective unsupervised labeling of content and intent units. A taxonomy for mined intent units has been presented, providing readers with a qualitative analysis of the nature of such units. We have tried to consolidate ongoing works on associating intents with query words by providing an overarching framework.

(3) Proposal of a framework for understanding syntactic complexity of queries:

We have provided corpus and query level setups for examining whether queries can be said to be a distinct linguistic system. We have built artificial logs based on statistical language models, and subsequently used complex network models and native speaker intuition (general Web search users), to quantify their deviation from real data. Our combined approach is successful in bringing out the fact that n -gram statistics are inadequate for modeling queries, and the ideal generative model has to imbibe both syntactic and semantic constraints specific to Web queries. Finally, overall results obtained can indeed be considered positive cues in favor of acceptance of our original hypothesis of Web search queries evolving into a distinct language.

Thus, to summarize the contributions of this thesis in a single sentence, we have (i) developed and evaluated unsupervised algorithms for flat and nested query segmentations, (ii) developed and evaluated a framework for labeling query segments as content or intent, and (iii) developed a framework for understanding syntactic complexity of search queries based on word co-occurrence networks and Web user intuition.

1.8 Organization of the thesis

The rest of the thesis is organized as follows.

In **Chapter 2**, we provide a review of past literature, while identifying research gaps and scope for further work.

In **Chapter 3**, we develop an unsupervised flat query segmentation algorithm using query logs and enhance it with Wikipedia titles. We also devise an IR-based evaluation framework for flat query segmentation, and show that the proposed algorithm has the best retrieval performance.

In **Chapter 4**, we develop an unsupervised nested query segmentation algorithm and show how improvements in document ranking can be obtained by appropriately leveraging the tree representation of the query arising out of the segmentation.

In **Chapter 5**, we present the concepts of content and intent segments along with in-depth discussions, and provide an unsupervised labeling strategy based on segment co-occurrence statistics computed from query logs.

Chapter 6 presents a framework for understanding the syntactic complexity of search queries by comparing corpus-level and query-level statistics between real and language model-generated artificial query logs.

Finally, **Chapter 7** concludes the thesis by summarizing the contributions and indicating a few issues for future work that have been opened up by the studies in this thesis.

Chapter 2

Literature Review

Scientific and engineering innovation, coupled with decreasing hardware costs and increasing commercial benefits, have made search engines extremely powerful over the last two to three decades [140]. Almost all Web users today regularly visit a search engine page and have their queries “answered” within a few milliseconds. Thus, not surprisingly, the same span of time has seen a significant amount of research being conducted on almost every aspect of Web-based retrieval. The general area of this thesis is *query analysis*, an important aspect of Web IR¹. Research on query analysis tries to infer as much information as possible from the small number of words contained in the input query. In the next few sections, we outline relevant research on the *syntactic* aspects of query analysis, that are concerned with learning the relationships among query words and how these can be exploited for better retrieval. While initial research assumed the simple bag-of-words independence model used in Boolean retrieval, the field has since progressed to experimenting with advanced techniques that include dependence models, query segmentation, and intent analysis. Most of these works include ideas that specifically suit the context of (Web) IR and are not directly borrowed from natural language processing (NLP). This need for basic independent analysis for query understanding strengthens the hypothesis of queries possessing unique linguistic style.

As stated in Chapter 1, the objectives of this thesis are to develop algorithms and eval-

¹Other vital facets include Web crawling, document indexing, and user interface design.

uation frameworks for query segmentation and segment role induction, and to study the syntactic complexity of Web search queries. Hence, we primarily survey past research in related areas and highlight scope of further work. We note that in this thesis, we only deal with *unstructured* text queries and documents. Research on *structured* and *semi-structured retrieval* (like retrieval from relational databases and XML retrieval [77, 167]) is out of the scope of this work.

2.1 The bag-of-words model and beyond

Boolean model

One of the earliest retrieval models in IR is the *Boolean retrieval model* [193]. In this model, the query is formulated as a Boolean expression of words, which means that the query words are combined with Boolean operators like NOT, AND and OR². An example of a query in the Boolean model would be `madonna AND (life OR bio)`, which would imply that the user is looking for documents with the term `madonna`, and any one of the terms `life` or `bio`. The Boolean model treats each query (and each text document) as an unordered set of words, or, more commonly a *bag-of-words*. In effect, only the presence or absence of each query term (as specified in the Boolean query) matters. Thus, in Boolean retrieval, a document either matches a query, or it does not, and there is no concept of document ranking.

Vector space model

The *vector space model* [198] is a notable improvement over the Boolean model where documents can be ranked in response to a query. The ranking is performed with respect to the *term weights* of the matched query words in the documents. Concretely, every query or every document is viewed as an n -dimensional vector, where n is the size of the corpus vocabulary. The i^{th} entry in the vectors of the query and the document contains the term weight of the vocabulary term with index i in the query and the document, respectively. One component of the term weight is the *term frequency* (TF) of the word (in the query

²Throughout this work, *words* and *terms* are used interchangeably.

or in the document, as applicable). However, certain words may be present in almost all the documents and hence may not have much discriminating power with respect to the ranking. To scale down the weights of such terms in the vectors, the term weight includes a second factor called the inverse document frequency (IDF). The IDF of a term with respect to a corpus is defined as the logarithm of the ratio of the total number of documents in the corpus to the number of documents that contain the term. Thus, very common terms will have a low IDF. The term weight is usually a product of the (often normalized) TF and the IDF, and such retrieval models are often referred to as using the TF-IDF ranking scheme. The *similarity score* between the query and the document is usually defined to be the *cosine similarity* of the query and document vectors. Documents with higher cosine similarity with the query are ranked higher in the final results. The vector space model and the TF-IDF ranking scheme are extremely popular in IR and several other applications [7, 18, 91, 132, 180, 214], and their variants are in active use even today [90, 96, 146, 163]. Nevertheless, the vector space model still considers a query as a bag-of-words, and ignores the significance associated with relative word ordering.

Term weighting

An important concept associated with the vector space model is *term weighting*, that assigns different weights to different query terms. This is a significant improvement within the bag-of-words framework, and term weighting has since been vital in almost all IR applications. For example, Robertson and Jones [187], Salton and Buckley [195], Kwok [127] and Greiff [79], define progressively improved expressions for term weighting based on various usage statistics in the corpus. Term weighting has been equally important in *probabilistic models of IR* like the Okapi BM25 [111, 112, 186], where documents are ranked by probabilistic estimates of whether they have content relevant to the information need, given the query and the document representations. Term weighting can also be used for the *reduction* of descriptive verbose queries by neglecting terms with lower weights [125, 131], but alternative approaches have proven to be more effective at this task [27, 126].

Language models

Statistical *language models* (LM) are one of the early mechanisms to allow direct incorporation of query term order into the retrieval process [176, 207]. In the LM framework,

each document is treated as a sample text from a language, and each query as a text generation process. Document ranking is performed based on the probabilities of generating a query from the LMs of the retrieved documents. An LM basically computes a probability distribution over any sequence of words drawn from a vocabulary. Thus, a document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words frequently [140]. The simplest probabilistic LM, the *unigram model*, does not take into account conditioning by word context, and estimates probabilities of each term independently. Hence, it is conceptually equivalent to the bag-of-words representation. However, *bigram language models* condition word probabilities on the previous term and hence use the notion of relative word ordering. For a *trigram model*, the probability of a new word depends on the probabilities of the preceding two words. Specifically, Song and Croft [207] report performance improvements on two datasets using word pairs along with the unigram model, and indicates that further benefits may be obtained using the trigram model.

Lafferty and Zhai [128] were the first to propose *query language models*. They show that combining past work on document models with query generative LMs can lead to substantial IR benefits. Such benefits have been shown to be most significant for short queries. Estimating a model for each query, they evaluate the LM in an IR setup with the goal of minimizing the total risk involved. Moreover, they suggest that LMs for queries can be used in modeling user preferences, query contexts, synonymy and word senses. The concept of (query) LMs has remained popular and has since been used in a variety of real applications [64, 99, 137, 216], where usually the bigram and trigram models suffice.

Dependence models

Even though the idea that modeling term dependencies in documents is useful for IR was around since 1983 [196, 235], it was not until after twenty years that dependencies between *query terms* began being considered. One of the pioneering works was done by Gao et al. [74], who extend the earlier language modeling approach [176, 207] based on the unigram model by relaxing the independence assumption. They introduce a query term dependence or linkage factor as a hidden variable, and model the dependencies as acyclic, planar and undirected graphs. According to their generative model, a query is created from a document in a two-step process. First, the linkages are generated, and second, each query

term is generated sequentially, constrained by the learnt linkages. Their unsupervised approach for learning and applying query term dependencies to IR significantly outperform the probabilistic models [111, 112, 189] and the traditional unigram and bigram-based language models [176, 207, 211] on publicly available data.

Another influential work on query term dependence models, based on Markov Random Fields (MRF) [119], is by Metzler and Croft [148]. Their framework allows use of advanced text features like phrases (word sequences) by weighting the document score in response to a query based on the occurrences of single terms, ordered phrases, and unordered phrases in the document. They propose three models of term dependence: the independence model (bag-of-words), the sequential dependence model (SDM) (adjacent query words are dependent on each other), and the full dependence model (FDM) (all query words are dependent on each other). The SDM is thus similar to the bigram LM [207]. While both SDM and FDM improve significantly over the independence model, SDM is generally seen to perform better for longer queries, while FDM is usually better for shorter queries. This seems to indicate that long range dependencies are not very frequent in search queries, and the more computationally efficient SDM is preferred more as queries become longer.

Subsequent work by Bendersky et al. [28] explore the noun phrase dependence model (NDM) (where *noun phrases* in queries are considered as features for the MRF retrieval model [148]) and the two-stage dependence model (TDM) (where noun phrases longer than two words are subdivided into smaller expressions). NDM and TDM assume all terms within the boundaries of a noun phrase chunk [1] to be dependent upon each other, and no dependencies to exist between chunks. The TDM is observed to outperform the earlier models, while requiring even less computation time than the SDM. Like the MRF model, term dependence has also been successfully incorporated into the Divergence From Randomness (DFR) retrieval model [171]. In another interesting work, term dependence has been used in query reduction for verbose queries [169]. Verbose queries are usually similar to NL sentences, and the authors use syntactic features extracted from dependency parses of verbose queries to rank terms in order of their importance.

Term proximity models

In our last topic in this section, we will look at term proximity models which reward

Table 2.1: Corpus details for term dependence and proximity models.

Corpus	Paper Used	TREC Queries	#Queries	Avg. Length
WSJ87-92	Hou et al. [97]	1999 Ad Hoc Track	50	5.4
AP88-89	Hou et al. [97]	1999 Ad Hoc Track	50	5.4
Robust	Hou et al. [97]	2004 Robust Track	100	4.1
WT10g	Hou et al. [97]	2001 Ad Hoc Track	50	4.9
	He et al. [88]	2000-'01 Web Track	100	4.2
	Zhao et al. [246]	2000-'01 Web Track	100	4.2
TREC8	Zhao et al. [246]	2004 Robust Track	250	2.7
GOV2	He et al. [88]	2004-'06 Terabyte Track	150	3.1
	Zhao et al. [246]	2004-'06 Terabyte Track	150	3.1
Blog06	He et al. [88]	2006-'08 Blog Track	150	2.1
	Zhao et al. [246]	2006-'08 Blog Track	150	2.1
ClueWeb Category B	He et al. 2011 [88]	2009 Relevance Feedback Track	50	2.1
ClueWeb 2009	Vuurens and de Vries [224]	1992-'99 Ad Hoc, 2011-'13 Web Tracks	550	3.5

documents that contain query terms close to each other, a significant deviation from the lines of work presented earlier that rely directly and indirectly on occurrence frequencies (or probabilities) of words and n -grams. The underlying assumption in term proximity models is that relevant documents will have several query words occurring nearby one another. There is no generally agreed-upon definition of term “proximity”; one can compute the smallest span that contains all the query words in the document, or one can aggregate the average distance between all the matched query terms in the document. Tao and Zhai [217] first explore the concept of term proximity and perform a thorough exploration of five such intuitive heuristics. To show the utility of term proximity in IR, they integrate a term proximity factor into the BM25 probabilistic retrieval model [189] and the KL-divergence retrieval model [128] and report statistically significant improvements. They conclude that the minimum distance between any two pairs of matched query words is maximally correlated with document relevance. Cummins and O’Riordan [56] examine seven more proximity functions, and use a genetic programming framework to *learn* their best combination function, which is subsequently integrated into a retrieval model. Their findings agree with those of Tao and Zhai [217] regarding the best correlation of minimum document distance of query term pairs with relevance, but they emphasize the importance

of considering *all* pairs of query terms. Their results show improvements for both short and long queries over the chosen baselines. More recently, He et al. [88] continue in the same paradigm of incorporating a proximity function into a probabilistic retrieval model. Their proximity functions based on window-based n -gram frequency counts and co-occurrence probabilities show improvement over BM25 [189], its bigram and trigram variants, and the MRF retrieval model [148].

Bai et al. [20] raise an important question of identifying lengths of sequences within which term proximities should be considered. They focus on long queries and find that the proximity of sequences of three to five words is the most effective, and suggest that sequences of such length are indicative of *user intent*. They also show improvement when n -gram sequences are appropriately weighted in proportion to their frequency in query logs. Song et al. [208] propose a different approach of using query term proximity, and do not look at pairwise term closeness in the document. Instead, they first group sets of query words into non-overlapping sequences, and view these sequences as providing context for the constituent terms. The relevance contribution of a query term occurrence in the document is measured by how many query terms occur in its context and how compact the *span* [217] is. They replace term frequency in the BM25 model [189] by the accumulated relevance contribution of the span containing the term. Details of the collections used by some of the most recent literature on dependence and proximity models is presented in Table 2.1.

2.2 Identifying syntactic units

Having seen how query representation has progressed from the simple bag-of-words model to incorporating term dependence and proximity, we will now survey the research that tries to identify syntactic units inside a query. By syntactic units, we refer to groups or sequences of words that are coherent from some perspective. For example, they may either be named entities like names of people, places or objects, multiword expressions or noun phrases of English or simply expressions that, if treated as a unit, help in the retrieval of more relevant pages. Breaking a query into such syntactic units is (ideally) beneficial to the search engine; it is not necessary that they will always have well-defined meaning, or

appear “well-formed” to a human user. We have already had glimpses of such attempts in the research covered so far, for example, in the work by Bendersky et al. [28], where in order to infer a two-stage dependence model, the query needed to be partitioned into a noun phrase and the remaining part of the query, or in Song et al. [208], where grouping words into coherent sequences was essential for modeling term proximity. Here we will first cover works on named entity recognition and noun phrase detection, before we move on to the relatively popular and more general theme of query segmentation.

Named entity recognition

Named entities like names of people and movies in queries can also be viewed as syntactic units, and identifying them can help significantly in improving IR precision or generating good query suggestions. We note that techniques from the general topic of named entity recognition in NL documents are hardly applicable in the context of queries, because of reasons like lack of capitalization, adequate context, and grammatical syntax. Research has been performed on named entity recognition in queries (NERQ), with Guo et al. [81] first bringing the problem into focus, who report that 70% of search queries contain named entities. They propose a probabilistic approach to identify them using query log data as the only resource, based on the Latent Dirichlet Allocation model [35]. Further, the authors also automatically classify the detected entities into predefined classes like *movies*, *games* and *music*. Subsequently, Du et al. [63] note that query sessions data contain valuable context information and use it to improve upon the method proposed by Guo et al. [81].

Noun phrase detection

Identifying noun phrase units like `year of the horse`, `present political scene` or `the wall of shame` is often useful for better retrieval and other applications like query expansion. Just as for named entity recognition, the lack of grammar and context make the task non-trivial for queries. The effectiveness of such syntactic query parsing for IR was first promoted by Zhai [239], who provides a probabilistic model for detecting noun phrases in queries. The author shows that indexing noun phrases along with words systematically produces better retrieval performance. Usually, noun phrase detection techniques involve the use of an English part-of-speech tagger (POS) trained on NL corpora [28, 82, 243], due to the lack of large volumes of human POS annotated query logs.

While the efficacy of such a method may seem questionable, Hagen et al. [82] report 99% precision and 90% recall for two such taggers when evaluated on a test set of 1000 queries. This is possibly explained by the highly skewed class distribution of query words, where 70% of the words are nouns, and adjectives at 7% come a distant second.

Bendersky et al. [28] and Hagen et al. [82] use noun phrase detection as a step towards query segmentation (discussed later), and show direct IR benefits of the process. Lima and Pederson [58] provide a fresh perspective and generate a syntactic parse of the query using an EM algorithm based on a probabilistic context-free grammar. The parsed query is then used for noun phrase recognition. However, their grammar consisting of 300 hand written rules, and the complexity of the overall algorithm, make their approach difficult to be representative and operational at Web scale. This may be a possible reason why the paper, presenting the very novel idea of a *query grammar* did not spark substantial research, apart from the work by Manshadi and Li [142] where the authors formulate a probabilistic phrase structure grammar for tagging product queries. Carvalho et al. [57] adopt a purely statistical approach and use word n -gram statistics for finding noun phrases, and show improvements in retrieval performance when phrasal knowledge is incorporated into the retrieval model.

Query segmentation

Query segmentation is by far the most prominent line of research towards the goal of syntactic partitioning of query words, with more than twenty papers published till date that are directly on the topic. Even though this “volume” of research may not be enough to make this an independent “field” in query analysis, it nevertheless highlights the recognition of the need for such techniques from researchers around the world. Query segmentation partitions the entire query into a sequence of non-overlapping words, and is not restricted to finding named entities or noun phrases. An example of a segmented query is `windows xp | home edition | hd video | playback`, where the pipes (|) represent segment boundaries.

Query segmentation was proposed by Risvik et al. [185], where the authors use frequencies and mutual information [52] of n -grams learnt from Web documents and query logs to come up with meaningful segmentations for queries. In the next ten years, we see a

plethora of work on Web search query segmentation using diverse resources like Wikipedia titles [28, 82, 84, 216], Web documents [29, 99, 185, 216, 240], Google n -grams [28, 82–84], clickthrough data [120, 137], and search result snippets [37]. Distinct algorithmic approaches like eigenspace similarity [240], conditional random fields [238], support vector machines [29] and expectation maximization [137, 216] have been used to accomplish this task, often using word co-occurrence statistics and normalized frequencies [37, 82–84, 114, 168, 185]. The concept of query LMs, as presented earlier, have also been used to perform query segmentation [99, 137, 216]. Unsupervised methods [37, 82–84, 114, 137, 168, 185, 216, 240] have outnumbered supervised techniques [28, 29, 238, 241, 242], as the latter relies on human annotations for training which is quite expensive to obtain in large volumes. Moreover, it is always difficult to achieve good query coverage from various domains in the datasets used for supervised learning [168]. Query segmentation has also been applied to domains other than Web search, like e-Commerce [120, 168] and patent search [73].

Query segmentation has generally been evaluated by comparing the machine output against a set of queries segmented by humans [29, 82, 84, 137, 216, 241, 242]. The basic assumption underlying this evaluation scheme is that humans are capable of segmenting a query in a “correct” or “the best possible” way, which, if exploited appropriately, will result in maximum benefits in IR performance. This is probably motivated by the extensive use of human judgments and annotations as the gold standard in the field of NLP (like POS labeling and phrase boundary identification). However, this idea has several shortcomings. Among those who validate query segmentation against human-labeled data, most [29, 37, 83, 84, 137, 216, 240–242] report accuracies on a corpus of 500 queries released by Bergsma and Wang in 2007 (BWC07) [29] sampled from the 2006 AOL query log [170]. BWC07 facilitated comparison of various segmentation algorithms based on matching metrics against human annotations. However, BWC07 suffered from limitations like consisting of noun phrase queries only, containing several ambiguous queries (no consensus among annotators on 40% of the queries), and having a few spelling and character encoding errors. As Hagen et al. [84] rightly note, none of the above reasons should be regarded as “shortcomings” in the work by Bergsma and Wang, since their goal was simply to evaluate their noun phrase segmentation algorithm on a dataset which they would make publicly available, and *not* to create a benchmark dataset corpus, which it subsequently became. Subsequently, Hagen et al. [84] create a much larger and cleaner dataset of about

50,000 queries (Webis-QSeC-10), also sampled from the 2006 AOL log, which is two orders of magnitude greater than BWC07 and free from all the drawbacks identified with BWC07. The queries in Webis-QSeC-10 are accompanied by ten annotations each and collected through crowdsourcing, and then thoroughly cleaned and filtered in a series of principled steps to be as representative of the entire log as possible. Currently, about 10% of this dataset is publicly available³.

2.3 User intent and role annotation

In this section, we discuss methods for understanding user intent, another dimension of query analysis, which has attracted a fairly large amount of research. The intention behind a query (like finding some specific site, or performing an online transaction) is vital to inferring the kind of pages that the user would be interested in viewing. Originally, after Broder's seminal paper on intent classification [40] twelve years back, the notion of intent was associated with the query as a whole. In recent years, the focus has shifted to identifying specific words in queries that explicitly carry user intent. Role annotation is closely associated with this line of research, where query words are marked up according to the part they play in the query.

Query-level intent

Broder [40] classifies user intent behind most queries as being *informational*, *navigational*, or *transactional*. Informational queries are issued to find information on a topic and are the classic type of queries handled traditionally by IR systems. But for Web search engines, users often have other special needs – resulting in navigational queries (queries issued for finding a specific URL to navigate to) and transactional queries (used to find places on the Web for performing transactions like buying or selling items, and uploading or downloading files). Even though informational queries still formed the majority (about 73%), navigational (about 26%) and transactional (about 36%) ones had a fair share. This taxonomy was well-accepted by the community and Broder's paper was followed by several attempts at manual and automatic classification into the above intent

³<http://www.webis.de/research/corpora>, Accessed 31 May 2014.

classes [76, 129, 133, 175, 190]. We briefly outline a few of these here.

Jansen et al. [104] built an automated classifier which classifies queries into these broad classes with an accuracy of about 74%. They also state that there are queries with vague or multiple intent, giving rise to a need for probabilistic classification. Ashkan et al. [13] focus on commercial queries (a subset of transactional queries) in which the user is interested in making an immediate or a future purchase. Other transactional queries are said to be of a non-commercial nature. In this work, the researchers claim that advertisement clickthrough data, when combined with the associated query log information and the content of the search engine result pages, can be effective in identifying user intent in the context of commercial queries. Predicting advertisement click rates for future queries is provided as a potential application. Baeza-Yates et al. [15] use various supervised and unsupervised techniques like support vector machines (SVM) [215] and probabilistic latent semantic analysis (PLSA) [95] for examining user intent. The goal or motivation behind a query is classified into informational, non-informational, and ambiguous. Queries are chosen from various categories like business, computers, sports, and science. They claim that supervised methods perform well given user goals and query categories, but unsupervised techniques help in subsequent validation, refinement, and selection of the one which best suits the users' needs. Proper intent identification also has ramifications on the appropriate evaluation of the search system [191, 192]. Over the years, researchers have identified the need to have finer intent classes like jobs, products, travel and people [98, 136].

Word level intent

The search query is a translation of the user's intent into a short sequence of keywords. This imposes great value on every word in the query from the perspective of a search engine. Recent research has started focusing on specific words that are direct indicators of user intent, and trying to identify and leverage them for better result quality. Identifying such words as performing the roles of intent carriers can be termed as *role annotation* for the query words. Wang et al. [225] were one of the first to explore this line of thought, where they refer to such words as *query aspect* words (such as *pictures*, *video*, *download*, *lyrics*, *games* and *movie*). They propose that each aspect represents one particular user information need. These are often left implicit by users, which leads to underspecified queries. They mine these aspects from search sessions data, and suggest

that using them can help users in (re-)formulating their queries better.

Yin and Shah [231] and Yin et al. [233] use of the terms *intent phrases* and *intent words* for the above class of words. They focus on named entity queries where one part of the query is the named entity, and the other part (intent word or phrase) indicates generic and popular user intent for that class of entities. They loosely define intent words for a category as those words that *co-appear* with many entities of a category in user queries. For example, `pictures`, `movies` and `songs` are some of the intent words for the category `actors`). The goal in Yin and Shah [231]’s work is to automatically build a taxonomy of such intent words of *specific classes* from query logs, while Yin et al. [233] try to extract structured information about the query entities from users’ search trails (post-search browsing behavior).

Li [135] propose that noun phrase queries are composed of *intent heads* and *intent modifiers*. For example, the query `alice in wonderland 2010 cast` can be said to be composed of `alice in wonderland, 2010` and `cast`, where `cast` is proposed to be the *intent head* (main user intent), while the other parts are termed as *intent modifiers*. The author uses *supervised* methods based on Markov and semi-Markov conditional random fields to identify these units and annotate their roles. They evaluate their approach on queries from *three domains* – jobs, movies and national parks.

Yu and Ren [236] present two intent roles, *kernel-objects* (like `harry potter` and `omega watch`) and *modifiers* (like `game` and `song`), for query words to better interpret query syntax. They term queries with an identifiable kernel object and modifier as *role explicit queries*, and focus on detecting such queries and labeling query words according to their framework. They show that a supervised classifier based on features like query length, presence or absence of named entities and interrogative words can achieve a precision of about 90% on identifying role explicit queries. The authors then use search sessions data to mine candidate modifiers. Subsequently, they propose a simplified word *n*-gram role model which estimates the generating probability of a role-explicit query from a query log and then performs intent role annotation, achieving more than 73% in terms of word annotation accuracy. Thus, while named entities, nouns and intent modifiers in past work would generally be the kernel, aspect words, intent words and phrases, and intent heads would usually be the modifiers in Yu and Ren’s framework.

In a work recently published in the speech community [85], the authors employ search query click logs to extract intent information using weakly-supervised methods. They use clicked URLs for supervision and extend knowledge graphs based on relational information. The posterior probabilities estimated from the graphical model can map discovered intents (like `play` and `show trailer` for movies) to search queries. These queries are subsequently used as additional training examples to augment bootstrapped relation identification models.

An associated line of research with a significant volume of work in the last few years involves *entities, attributes, classes, instances, and relations* [9, 102, 156, 159–161, 184, 221], with notable work being done by Paşca et al. [155, 156, 159–161, 221]. While entities refer to items like people, places, movies and books, attributes are specific properties of the relevant entities (like age, area, cast and title). Entities can be generalized to classes (the entity `aspirin` belongs to the class `medicine`). Specific entities belonging to a class are also referred to as instances of that class (`aspirin` in this case). Finally, a relation articulates the connection between an entity and any of its attributes. For example, the relation `is the capital of` connects the attribute `Paris` with the entity `France`. In our context, entities and attributes can be considered to be roles played by words or groups of words in queries, analogous to named entities and intent phrases [231, 233], intent modifiers and intent heads [135], or kernel objects and modifiers [236]. The general goal of the line of work on entities and attributes is to building structured representations of knowledge from the huge amount of unstructured data on the Web. Apart from enabling search engines to present users with a more informed set of results, this would facilitate identification of interesting relationships among various entities and can be helpful in a large set of Web search applications. In general, most of the supervised and unsupervised approaches rely on the approximate matching of relevant distributional patterns or templates [4, 165] in queries and Web documents for extracting entities [102, 155, 156, 161, 172, 221] and attributes [9, 157–160, 162, 184]. Finally, Lin et al. [138] associate intent with *verbs* or *actions* in queries like `buy` and `download`, and define actions as “empirically observable, direct manipulation or information request on an entity”.

Query intent can also be represented through a set of *facets*, like spatial and time sensitivities, genre, topic and scope [76, 154]. These are aspects that can be attributed to the query as a whole. Proper identification of such facets has been shown to improve the per-

formance of query intent classification [76].

2.4 Deeper syntactic analysis

In this section, we will briefly look at research that has been directed at understanding the linguistic syntax of Web queries. Linguistic analysis and annotation of queries based on its parent NL, like POS tagging [22, 26, 72, 169], has been an important direction of research. These studies reveal interesting syntactic properties and trends, such as more than 70% of the query words are nouns [22] and NL question queries are on the rise [166].

POS tagging for queries

Use of POS tagging for *query analysis* is not a very recent idea. Allan and Raghavan, in 2002 [10], showed how to use POS patterns to mitigate the problem of ambiguity for very short queries. In their study, they find frequent POS patterns near one-word queries and convert them to clarification questions. These questions, based on statistical language models, are shown to reduce query ambiguity a substantial number of times. Barr et al. [22] provided the first measurements on POS tagging of Web search queries. Examining a sample of 3, 283 queries, they found that a large majority (about 70%) of all query words are nouns, while adjectives are the second most used POS with a presence of 7%. The authors also show that as expected, the Brill Tagger [38] achieves a low accuracy of about 48% when trained on NL, which improves to about 79% when trained on queries. Nonetheless, taggers trained on NL corpora have been shown to be useful in a variety of real scenarios [25, 28, 82, 152]. Hagen et al. [82] report a precision of 99% and a recall of 90% when the Stanford Tagger [219, 220] is used in the task of detecting noun phrase queries. Accurate POS tagging of queries is still an active area of research. Recently, Ganchev et al. [72] show that transfer of POS tags from URL snippets retrieved by the search engine to the query words can significantly improve POS tagging accuracies.

Understanding syntactic complexity

Mean query lengths have been slowly but steadily rising over the years (from 2.2 in 2000 [209] to 3.5 words in 2006 [170] to 3.77 in 2010 (own data)), which implies

that the number of relatively longer queries has also been increasing. This is generally believed to be an indication that queries are becoming more complex as well. Technically though, length may not always be directly proportional to complexity. For example, the queries `eternal sunshine of the spotless mind` and `code to turn text blue excel` both have six words. But the latter is more complex than the former, which is simply the name of a movie and is thus easily processed. However, this brings to light the lack of clarity on (syntactic) “complexity” itself, and there have not been any studies till date that has defined or quantified such complexity. We note that a complex query is distinct from a complex search task [205], and the latter can be solved with a sequence of “simple” queries. “Query complexity” is used in a different sense in Radinsky and Ailon [179], where the authors refer to the complexity of the order of pages that need to be judged by assessors. Also, here we refer to the complexity in the syntactic structure of the query from a language perspective, and not in the older sense of the term, which referred to the degree of usage of advanced search operators like AND, MUST APPEAR or PHRASE [92, 105].

Nevertheless, the notion of query complexity, in some cases, can be correlated with the concept of query “difficulty”, and a few studies have been directed at the latter topic [12, 86, 152, 234]. For example, Hauff et al. [86] define a difficult query as one that has a low IR performance (as measured by a metric like average precision [197]). They provide ways to predict the difficulty of a query, which can help the search engine in taking non-standard avenues for difficult queries, like applying query expansion, suggesting alternative search terms, adjusting advertisements, or returning results from specialized collections. They show that their method, based on the discrepancy between the query and document language models, outperforms past approaches, and validate their hypotheses on two large Web collections. Mothe and Tanguy [152] propose a *linguistic analysis* of the query text for predicting query difficulty. The authors find correlations between morphological (like number of words and average word length), syntactic (like syntactic depth and span) and semantic (like degree of polysemy) features of queries, and the average precision and recall scores obtained by systems for these queries. Their results encourage the use of linguistic processing in IR systems [130]. Aligned with this line of thought, Liu et al. [139] classify queries by *readability*, or *reading level*, and show that queries from users of different age levels (like tenth grade, undergraduate and graduate) can be effectively separated using

an SVM [218] based on syntactic and semantic features. The authors also propose that improvement in search experience can be obtained by matching the query and documents not only by content but also by level.

2.5 Model-generated queries

Biemann [32] showed the importance of model-generated sentences for the discovery and understanding of NL syntax. A good generative model for NL sentences is able to synthesize realistic text with high probability. Such a model can account for several distinct syntactic properties of language, which is more than simply a body of “time-linear sequence of symbols”. The complexity of such a generative model is a reflection of the syntactic complexity of the language itself. He introduced a random text generation model that agrees well with NL with respect to word frequency distribution, word length, sentence length and co-occurrence neighborhood. His model was not constrained by any *a priori* probability distribution – the corpus characteristics emerged from a two-level process involving one parameter for the word generator and one parameter for the sentence generator. His model was the first random text generator that modelled sentence boundaries beyond inserting a special blank character at random. Rather, sentences are modelled as a path between sentence beginnings and sentence endings which imposes restrictions on the words at sentence beginnings and endings. The model was proposed to be a simple but plausible model for the emergence of large-scale language characteristics, without assuming an underlying grammar or semantics. The model-generated sentences were found to be non-sensical in meaning, but maintained some of the distributional properties of NL corpora. Further, Biemann showed that statistics of word co-occurrence networks [61, 69] provide a reliable framework for comparing real and model-generated corpora.

There have been attempts at question generation in NL [89, 150] and in the general IR framework [14]. Azzopardi et al. focus on methods for building simulated queries in six European languages and explore their quality against real queries. Using probabilistic query generation from document language models, they explore factors which may influence the generation of queries, and propose a model with improved document and term selection properties, showing that simulated queries can be generated that are comparable

to real queries. The authors claim that building models of querying behavior provides a deeper insight into the querying process so that better retrieval mechanisms can be developed to support the user.

However, there has only been one major contribution that is adapted specifically to synthetic Web search queries – the research presented by Li et al. [134]. Their generated query log⁴ (named *QRU-1*) contains 2,030 queries that were generated by applying a string transformation method on 100 base queries. The base queries were sampled from the query sets used for the TREC 2009 and 2010 Web Tracks. As a guarantee of goodness, they report that 70% of these queries were actually found in a separate log from Bing.

The goal of the work by Li et al. [134] was to create synthetic Web queries that would benefit the community in research on query representation and understanding. The QRU-1 dataset was constructed based on the topics (queries) that were developed during the TREC 2009 and 2010 Web Tracks. For each of the hundred topics used in Web Tracks in these two years, the authors assign approximately twenty similar queries. The similar queries assigned to the original TREC topic represent the same user intent, but are expressed in different forms, including synonyms, stemming variations, spelling errors and abbreviations. The synthetic queries in the QRU-1 dataset are automatically generated from a model trained from Bing search log data with the title of the TREC topic as the input. Subsequently, a manual cleaning of the artificial queries was also performed and unlikely queries are discarded, based on predetermined guidelines. The guidelines were as follows: (a) The generated query represents the same user intent as the original query. The original TREC topics are often ambiguous, and may contain more than one sense. The generated query was to be retained if it represents any of the senses, or if it is judged by the annotator as representing a likely sense; (b) It is likely to be input by users, including typographical errors. Interestingly, the authors observed that 70% of the remaining similar queries actually occur in another Bing search log. Since the specific queries were generated artificially instead of being directly collected from the search log, the authors claim that the privacy of the searchers is not violated.

The dataset of synthetic queries can be used in tasks like query rewriting [244], query suggestion [43], query segmentation [84] and query expansion [228]. As an example, the

⁴<http://bit.ly/1cSHSfP>, Accessed 1 June 2014.

authors demonstrate the usage of their queries for improving the relevance of web search results using query reformulation [110]. To study the effectiveness of generated queries for query reformulation, the authors record the retrieval performance of the most effective formulation of the query (including the original topic and the generated queries) for a given TREC topic. Thus, they simulate the actions of an oracle user who always selects the best-performing query among all the proposed candidates, and calculate the upper bound on the retrieval effectiveness that can be achieved using the QRU-1 dataset. They conducted retrieval experiments using the web corpus ClueWeb Category B⁵, which is a set of approximately 50 million pages. They use two retrieval models, the standard BM25 retrieval model [188] and the state-of-the-art sequential dependence model (SDM) [148]. Retrieval experiments are implemented using the Indri open-source search engine ⁶ and evaluated using the MAP, nDCG and ERR metrics [46]. For each of these metrics, the authors reported the following statistics for the retrieval models: (a) the baseline metric value achieved by the original query; (b) the best metric value achievable by either the original query or one of the generated queries; (c) the percentage of generated queries that outperform the original query; and (d) the percentage of queries, whose performance is improved by using at least one of the generated queries.

Their results demonstrate that the generated queries can significantly improve the retrieval performance both for the retrieval models. By using the QRU-1 dataset, the retrieval effectiveness was improved for about two thirds of the queries. The authors note, however, that not all of the generated queries are equally helpful. Therefore, automatic query selection for effective reformulation seems to be an important research topic. Even though the quality of the generated queries were good, this is too small a log to be useful for practical applications like attribute extraction [159].

Pasca et al. [159] introduce a weakly supervised approach for the acquisition of entity attributes from query logs, by automatically expanding the set of real queries from which attributes are extracted with *additional synthetic queries* which have not yet been submitted to the search engine. Expanding the input data produces a more general approach, which can be applied to existing methods for attribute acquisition from query logs for increased coverage. Application of past extraction patterns to the new set of queries permits the

⁵<http://www.lemurproject.org/clueweb09.php/>, Accessed 16 November 2014.

⁶<http://www.lemurproject.org/indri/>, Accessed 16 November 2014.

acquisition of additional attributes that would otherwise be missed due to the absence of real issued queries containing such instance-attributes pairs.

In order to generate new queries, Pasca aggregates known queries into templates associated with known phrase fillers. The known phrase fillers for each template are then expanded into new candidate phrase fillers. New queries are generated based on query analysis alone, and is thus more scalable compared to a document-based analysis. Among the inferred queries, the ones of higher interest to attribute extraction are those derived from a query template that fixes either a potential attribute or a potential instance. In experiments using a large set of anonymized search queries, the synthetic queries allow for the acquisition of accurate attributes over an evaluation set of 75 instances introduced previously.

2.6 Queries as a distinct language

In a series of studies that have been cited more than a total of 2000 times, Jansen et al. and Spink et al. [103, 106, 107, 209] were the first to refer to Web search queries as a unique language. In their papers, they provided the first measurements of several aspects of query logs. They find that a small fraction of search terms are used with very high frequency, even though the size of the overall vocabulary is very large. This large vocabulary is partly created by a large number of spelling errors, non-English terms, and Web URLs. Interestingly, they remark that general Web users “talk” in their searches in their own way. They emphasize the need for further study of both ends of the word rank frequency distribution, and of other “linguistic characteristics of Web queries so that user query language can be anticipated and supported”. They also note that the distribution of topics in Web search queries does not resemble the distribution of the corresponding subjects in Web pages. Encouragingly, they emphasize that studies investigating the linguistic aspects of search queries have potential to benefit IR systems and Website development.

In independent studies from those by Spink et al. and Jansen et al., Guichard [80] and Dessalles [60] note that Web search queries seem to resemble a “protolanguage”, a linguistic state that bridges the gap between a wholly alingual state and the full possession of language [30]. This was motivated by their observation that users often enter words

in grammatically incoherent orders when formulating their search queries. Much more recently, Huang et al. [99] observe that search queries are composed in a very different linguistic style from that of the document body. They identify the scope for leveraging this style discrepancy for search query processing on a large scale, and show improvements in tasks like spelling correction and query segmentation using language models estimated from large volumes of query logs.

2.7 Scope of further work

In this chapter, we have provided a literature survey on the evolving perspectives on search query syntax. We have seen how IR systems that originally treated queries as bags-of-words in the Boolean and vector space models, have gradually learnt to incorporate relative term ordering through language models, dependence models and term proximity models. Specifically, we have looked at past efforts on (i) identifying syntactic units in Web queries, (ii) annotating roles of syntactic units in queries and their relationships to query intent, and (iii) understanding the syntactic complexity of search queries. In (i), we first reviewed research on named entity recognition and noun phrase detection in queries (where the aim was to identify units that were typically named entities and noun phrases), followed by query segmentation, which partitions the entire query into non-overlapping sequences of words. In (ii), we saw how the overall query intent is now being associated with specific words mentioned by the user, and surveyed attempts at automatically inferring roles of such words within queries (for example, intent heads, intent phrases and query modifiers). In (iii), we found that a number of researchers in the past have indicated that queries possess linguistic features of their own. Parallely, we also saw that a group of researchers believe that since essentially queries are borrowing from a parent NL, techniques like POS tagging that work for the parent NL should be applicable for queries as well. Finally, we looked at principles of understanding the syntactic complexity of NL text, and the role that artificial synthesis can play in determining such complexity. We now outline a few directions for further work along each of these three lines.

Past approaches to query segmentation miss out on the unique syntactic properties of queries due to a bias towards projecting NL syntax on to queries. A better query segmen-

tation algorithm that primarily relies on query logs to discover syntactic units in queries needs to be developed and applied to document retrieval. At this stage when query segmentation has been around for about a decade, the research community would also be benefited by a generic retrieval-based evaluation framework, that can conveniently decouple the algorithm and the evaluation modules. This is motivated by the fact that the end user of segmentation is the search engine and human annotations may not always reflect the best partitioning from an IR perspective. Finally, development of a nested, or hierarchical query segmentation strategy using query logs that can discover segments embedded inside bigger segments, along with a mechanism for using it for improving IR, can address the problems of granularity and word adjacency requirements that are associated with flat, or non-hierarchical segmentation.

There is a need to consolidate several proposals for a segment role induction framework that talks about related concepts but are applicable to different classes of queries. Such an overarching framework should be generally applicable for all types of queries, and not restricted to categories like noun phrase queries, named entity queries, or entity-attribute queries. It has been proposed that the Web (document) space and user (query) space must be modeled separately [102]. Thus, to identify true user intent, a role induction framework must rely only on query logs, and must be able to label units in the context of a query with minimum runtime overhead for query processing. It is also preferable if the framework uses unsupervised learning to associate roles with segments. Finally, it is important to make direct connections between the role induction strategy and its application to improve the quality of search results.

Several works that posit linguistic approaches to query understanding (for example, POS tagging) are based on the fundamental assumption that queries issued in a certain language, say English, will borrow grammatical artifacts of that language (like nouns and noun phrases). This assumption is biased for the following reason: a noun in English is called a noun because it follows a particular syntactic distribution; it is quite unlikely that the same word will behave as a noun in a query either from the point of statistical distribution or its cognitive interpretation by the users. Thus, if queries are to be understood linguistically, they should be analyzed from the first principles rather than superimposing the grammatical syntax of NLS and thereby masking their true syntactic properties. On the other hand, while some researchers have hypothesized that queries form a new and

unique linguistic system, there is no systematic and comprehensive study of the syntactic properties of Web queries that can convincingly bring out this fact. The challenge, of course, is to identify the unique syntactic features of an NL that make it different from any random or artificially generated sequence of symbols. Thus, a holistic approach based on the first principles is required that can provide an estimate of the syntactic complexity of search queries, and quantify its proximity to its parent NL.

With a detailed understanding of the state-of-the-art, we now move on to report our contributions in this thesis. In the next two contributory chapters, we study algorithms and evaluation strategies for non-hierarchical and hierarchical query segmentation. Specifically, Chapter 3 proposes a (flat) query segmentation algorithm using query logs and Wikipedia titles, and develops a retrieval-based evaluation framework for assessing the potential of a segmentation algorithm. In Chapter 4, we devise a nested query segmentation algorithm based on only query logs, and present a deterministic way of leveraging the hierarchical query syntax to improve document ranking. Chapter 5 focuses on inducing and understanding simple roles that segments perform in search queries. The final contributory chapter, Chapter 6, analyzes the syntactic complexity of search queries at corpus and query-levels using model generated datasets.

Chapter 3

Discovering Syntactic Units by Flat Query Segmentation

3.1 Introduction

This chapter and the next deal with query segmentation and techniques to use it for improving retrieval and ranking. Specifically, this chapter deals with flat, or non-hierarchical query segmentation, where we propose an algorithm based on query logs and Wikipedia titles, and a retrieval-based evaluation framework. The next chapter studies nested, or hierarchical query segmentation, and a method for directly using it to improve document ranking. A background study which brought forward the importance of the problems, as well as the existing literature related to the problems, have been presented in Chapter 2.

Query segmentation is the process of dividing a query into its individual syntactic units [137]. To be specific, a *flat* segmentation strategy partitions the query into a sequence of non-overlapping words. For example, the search query `singular value decomposition online demo` can be broken down into the constituent syntactic units of `singular value decomposition` and `online demo`. All Web documents containing the individual terms `singular`, `value` and `decomposition` are not necessarily relevant for this query. Rather, one can almost always expect to find the segment

singular value decomposition in the relevant documents. In contrast, although online demo is a segment, finding the phrase or some variant of it may not affect the relevance of the document. Hence, the scope of query segmentation goes beyond the detection of multiword named entities. Rather, segmentation leads to a better understanding of the query and is crucial to the search engine for improving IR performance.

Previous research has expressed and addressed the need for identification of these units. Towards this end, various external resources such as Webpages [29, 99, 185, 216, 240], search result snippets [37] and Wikipedia titles [28, 82, 84, 216] have been used. Although these methods can help in retrieval, query expansion and query suggestion, we strongly believe that they miss out on the unique syntactic properties of queries due to a bias towards projecting NL syntax on queries. Thus, we think that the linguistic syntax of queries is distinct from that of the parent NL (i.e., English, in our case); the first step towards understanding this syntax is to understand the nature of the constituent word groups. These word groups should be identified solely on the basis of queries, because use of external resources raises the risk of projecting NL syntax onto the queries; and a proper understanding of this syntax coupled with automatic techniques for parsing it can lead to significant performance improvements in various IR tasks. In this work, we take the first steps to unravel the syntax of queries by proposing an unsupervised method for query segmentation that primarily uses query logs. As we shall see, the segments identified by our method do not necessarily align with NL segments, yet it is clear that they are meaningful.

Next, we note that there is a broad consensus in the literature that query segmentation can lead to better retrieval performance [28, 29, 84, 137, 216]. However, most automatic segmentation techniques [29, 37, 84, 137, 216, 240] have so far been evaluated only against queries segmented by human annotators. Such an approach implicitly assumes that a segmentation technique that scores better against human annotations will also automatically lead to better IR performance. We challenge this approach on multiple counts. First, there has been no systematic study that establishes the quality of human segmentations in the context of IR performance. Second, grammatical syntax in queries is not as well-understood as NL sentences where human annotations have proved useful for training and testing of various NLP tools. This leads to considerable inter-annotator disagreement when humans segment search queries [216]. Third, good quality human annotations for segmentation can be difficult and expensive to obtain for a large set of test queries. Thus, there is

a need for a more direct IR-based evaluation framework for assessing query segmentation algorithms. This is the second objective of this contributory chapter.

The rest of the chapter is organized as follows. In Section 3.2, we provide our algorithm for query segmentation using query logs and Wikipedia titles. Section 3.3 introduces our evaluation framework and its design philosophy. Section 3.4 presents the dataset and the segmentation algorithms compared on our framework. Section 3.5 discusses the experimental results and insights derived from them. In Section 3.6, we discuss a few related issues about our dataset construction and inter-annotator agreement. In Section 3.7, we present ideas for bringing together the allied concepts of query segmentation and NL chunking. We conclude this chapter by summarizing our contributions and suggesting future work in Section 3.8.

3.2 Query segmentation algorithm

We are given a query log, which is a large collection of search queries. Let us consider a candidate MWE $\mathcal{M} = \langle w_1 w_2 w_3 \dots w_n \rangle$ where w_j -s denote the words constituting \mathcal{M} . Let $\{q_1, q_2, q_3, \dots, q_k\}$ denote the subset of queries in the log that contain all the words of \mathcal{M} , though not necessarily occurring together as an n -gram. Our premise is that search queries can be viewed as bags of multiword expressions (MWEs), which is to say that any permutation of the MWEs constituting a particular search query will effectively represent the same query. Thus, to test if an observed n -gram is an MWE, we could ask the question if the constituents of an MWE appear together more frequently than they would under a bag-of-words null model. We now formalize this intuition in a new test of significance for detecting MWEs in Web search queries.

Let X_i be the indicator variable for the event “ \mathcal{M} occurs in the query q_i ”, and let $Prob[X_i = 1]$ denote the probability of this event. Also, let l_i be the length of query q_i in words. Then, there are $(l_i - n + 1)$ locations where \mathcal{M} can be positioned in q_i , and for each choice of location there are $(l_i - n)!$ ways of permuting the remaining $(l_i - n)$ non-MWE words of q_i . Thus, we can write the probability of $[X_i = 1]$ under the bag-of-words model (our null model) as follows:

$$Prob[X_i = 1] = \frac{(l_i - n + 1) \times (l_i - n)!}{l_i!} = \frac{(l_i - n + 1)!}{l_i!} \quad (3.1)$$

We define $X = \sum_i X_i$ (which models the number of times the words of \mathcal{M} appear together in the k queries). We use Hoeffding's Inequality [94] to obtain an upper-bound δ on the probability of $[X = N]$, where N denotes the observed value of X in the data (also referred to as the frequency of \mathcal{M}):

$$Prob[X \geq N] \leq \exp\left(-\frac{2(N - E(X))^2}{k}\right) = \delta \quad (3.2)$$

where, the expectation $E(X)$ is given by $E(X) = \sum_i Prob[X_i = 1]$. We obtain δ for each n -gram (or candidate MWE) \mathcal{M} and define $-\log_e \delta$ as the MWE score for \mathcal{M} :

$$Score(\mathcal{M}) = -\log_e \delta = \frac{2(N - E(X))^2}{k} \quad (3.3)$$

If δ is small, then the surprise factor is higher, indicating a greater chance of \mathcal{M} being an MWE, and vice versa. We note that unigrams have a score of zero, since their observed and expected frequencies are equal.

Since queries are generally shorter than NL sentences, only bigrams and trigrams are considered ($n = 2, 3$). For computational reasons, we compute the MWE scores only for n -grams whose constituent words have each appeared in at least α queries in the log (where α is a user-defined threshold). We add an n -gram to the list of significant n -grams if its MWE score exceeds β (a second user-defined threshold). In our experiments we used $\alpha = 10$ and $\beta = 0.6k$ (where k is the number of queries in which all the words of the n -gram occur, though not necessarily together). We note that in this case, β is specific to every MWE and there is no global threshold. Choosing β in such a way allows us to be more selective with the lexicon entries with respect to statistical significance than a global threshold. We now have a list of significant n -grams and their associated MWE scores. We use this list to perform unsupervised query segmentation as follows: First, we compute a score for each possible segmentation by adding the MWE scores of individual segments.

Then, we pick the segmentation that yields the highest segmentation score. Here we use a dynamic programming approach to search over all possible segmentations.

There are two important novelties in the method: (a) A decision is made on the significance of an MWE only on the basis of the number of queries which contain all the terms of the MWE, thus disallowing frequently misleading unigram statistics to interfere with the decision, and (b) the segmentation procedure is capable of segmenting queries using only query logs, not relying on any other external resource. On manual examination of the segmentation results, we found that many segments discovered by our scheme do not seem intuitive to humans, because the human concept of segmentation is largely influenced by NL grammar. For example, the query `how to spot a fake bill` is segmented as `how to | spot a fake | bill` by our method (pipes mark segmentation boundaries). While `a fake bill` is a noun phrase, and therefore, a valid “segment” according to the Standard English grammar, one cannot deny the fact that `how to` expresses a class of intent in queries and is found to be associated with diverse concepts such as `save money`, `play guitar` or `make tea`. Interestingly, `spot a fake`, which makes very little sense as an MWE, is in fact quite commonly seen in queries expressing a generic action phrase applicable to diverse objects such as `video`, `gucci bag` or `mona lisa painting`. Some other examples of generic query intents discovered by this method are `information about`, `difference between` and `history of the`.

The proposed algorithm is also capable of detecting named entities such as `windows media player` and `nikon d5000`, including relatively infrequent ones like `very hungry caterpillar`. However, longer and rarer named entity identification requires world knowledge and can be addressed by using external resources such as Wikipedia, though adequate care has to be taken so that the generic intent phrases are not lost in the process. We next outline the proposed procedure for augmenting our segmentation algorithm using Wikipedia titles.

3.2.1 Enhancement with Wikipedia titles

We now explain how to augment the output of our algorithm (or any n -gram score

Algorithm 1 Wiki-Boost(Q' , W)

```

1:  $W' \leftarrow \emptyset$ 
2: for all  $w \in W$  do
3:    $w' \leftarrow \text{Seg-Phase-1}(w)$ 
4:    $W' \leftarrow W' \cup w'$ 
5: end for
6:  $W'$ -scores  $\leftarrow \emptyset$ 
7: for all  $w' \in W'$  do
8:    $w'$ -score  $\leftarrow MI(w')$  based on  $Q'$ 
9:    $W'$ -scores  $\leftarrow W'$ -scores  $\cup w'$ -score
10: end for
11:  $U$ -scores  $\leftarrow \emptyset$ 
12: for all unique unigrams  $u \in Q'$  do
13:    $u$ -score  $\leftarrow \text{probability}(u)$  in  $Q'$ 
14:    $U$ -scores  $\leftarrow U$ -scores  $\cup u$ -score
15: end for
16:  $W'$ -scores  $\leftarrow W'$ -scores  $\cup U$ -scores
17: return  $W'$ -scores

```

aggregation based segmentation algorithm) with Wikipedia titles¹. We call this procedure *Wiki-Boost*. Inputs required are a list of queries Q' already segmented by the original algorithm (say, *Seg-Phase-1*) and W , the list of all stemmed Wikipedia titles (4, 508, 386 entries after removing one-word entries and those with non-ASCII characters at the time of experimentation in April 2011). We compute the Mutual Information (MI) score [185] of an n -segment Wikipedia title w' (segmented by *Seg-Phase-1*) by taking the higher of the MI scores of the first $(n - 1)$ segments with the last segment *and* the first segment and the last $(n - 1)$ segments. The frequencies of all n -grams are computed from Q' . Scores for unigrams are defined to be their probabilities of occurrence. Thus, the output of the *Wiki-Boost* is a list of MI-scores for each Wikipedia title in W .

Following this, we use a second segmentation strategy (say, *Seg-Phase-2*) that takes as input q' (the query q segmented by *Seg-Phase-1*) and tries to further join the segments of

¹<http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-all-titles.gz>, Accessed March 24, 2014

q' such that the product of scores of the candidate output segments, computed based on the output of *Wiki-Boost*, is maximized. A dynamic programming approach is again found to be helpful in searching over all possible segmentations in *Seg-Phase-2*. The output of *Seg-Phase-2* is the final segmentation output.

3.3 Query segmentation evaluation

Now that we have a working flat segmentation algorithm in place, we now move on to our framework for an IR-based evaluation. We propose a retrieval-based evaluation framework for query segmentation that requires only human relevance judgments (RJs) for query-URL pairs for computing the performance of a segmentation algorithm – such relevance judgments are anyway needed for training and testing of any IR engine. A fundamental problem in designing an IR-based evaluation framework for segmentation algorithms is to decouple the effect of segmentation accuracy from the way segmentation is used for IR. This is because a query segmentation algorithm breaks the input query into, typically, a non-overlapping sequence of words (segments), but it does not prescribe how these segments should be used during the retrieval and ranking of the documents for that query. We resolve this problem by providing a formal model of query expansion for a given segmentation; the various queries obtained can then be issued to any standard IR engine, which we assume to be a black box.

We conduct extensive experiments within our framework to understand the performance of several state-of-the-art query segmentation schemes [84, 137], our own algorithms and segmentations by three human annotators. Our experiments reveal several interesting facts such as: (a) Segmentation is actively useful in improving IR performance, even though submitting all segments (detected by an algorithm) in double quotes to the IR engine degrades performance; (b) All segmentation strategies, including human segmentations, are yet to reach the best achievable limits in IR performance; (c) In terms of IR metrics, some of the segmentation algorithms perform as good as the best human annotator and better than the average/worst human annotator; (d) Current match-based metrics for comparing query segmentation against human annotations are only weakly correlated with the IR-based metrics, and cannot be used as a proxy for IR performance; and (e) There is scope for improvement

for the matching metrics that compare segmentations against human annotations by differentially penalizing the straddling, splitting and joining of reference segments. In short, the proposed evaluation framework not only provides a formal way to compare segmentation algorithms and estimate their effectiveness in IR, but also helps us to understand the gaps in human annotation-based evaluation. Finally, the framework provides valuable insights regarding the segmentations that can be used for improvement of the algorithms.

3.3.1 The evaluation framework

We now present our framework for the evaluation of query segmentation algorithms based on IR performance. Let \mathbf{q} denote a search query and let $\mathbf{s}^{\mathbf{q}} = \langle s_1^{\mathbf{q}}, \dots, s_n^{\mathbf{q}} \rangle$ denote a segmentation of \mathbf{q} such that a simple concatenation of the n segments equals \mathbf{q} , i.e., we have $\mathbf{q} = (s_1^{\mathbf{q}} + \dots + s_n^{\mathbf{q}})$, where $+$ represents the concatenation operator. We are given a segmentation algorithm \mathcal{A} and the task is to evaluate its retrieval performance. We require the following resources:

1. A test set \mathcal{Q} of unquoted search queries.
2. A set \mathcal{U} of documents (or URLs) out of which search results will be retrieved.
3. Relevance judgments $r(\mathbf{q}, u)$ for query-URL pairs
 $(\mathbf{q}, u) \in \mathcal{Q} \times \mathcal{U}$. The set of all relevance judgments are collectively denoted by \mathcal{R} .
4. An IR engine that supports quoted queries as input.

The resources needed by our evaluation framework are essentially the same as those needed for the training and testing of a standard IR engine, namely, queries, a document corpus and a set of relevance judgments. Akin to the training examples required for an IR engine, we only require relevance judgments for a small and appropriate subset of $\mathcal{Q} \times \mathcal{U}$ (each query needs only the documents in its own pool to be judged) [222].

It is useful to separate the evaluation of segmentation performance, from the question of how to best exploit the segments to retrieve the most relevant documents. From an

IR perspective, a natural interpretation of a segment could be that it consists of words that must appear together, in the same order, in documents where the segment is deemed to match [29]. This can be referred to as *ordered contiguity matching*. While this can be easily enforced in modern IR engines through use of double quotes around segments, we observe that not all segments must be used this way (see Metzler and Croft [148] for related ideas and experiments in a different context). Some segments may admit more general matching criteria, such as *unordered or intruded contiguity* (e.g., a segment `a b` may be allowed to match `b a` or `a c b` in the document). The case of unordered intruded matching may be restricted under *linguistic dependence* assumptions (e.g., `a b` can match `a of b` or `b in a`). Finally, some segments may even play non-matching roles (e.g., when the segment specifies user intent, like `how to` and `where is`). Thus, there may be several different ways to exploit the segments discovered by a segmentation algorithm. Within the same query, different segments may also need to be treated differently. For instance, in the query `cannot view | word files | windows 7`, the first one might be matched using intruded ordered occurrence (`cannot properly view`), the second may be matched under a linguistic dependency model (`files in word`) and the last under ordered contiguity.

Intruded contiguity and linguistic dependency may be difficult to implement for the broad class of general Web search queries. Identifying how the various segments of a query should be ideally matched in the document is quite a challenging and unsolved research problem. On the other hand, an exhaustive expansion scheme, where every segment is expanded in every possible way, is computationally expensive and might introduce noise. Moreover, current commercial IR engines do not support any syntax to specify linguistic dependence or intruded or unordered occurrence based matching. Hence, in order to keep the evaluation framework in line with the current IR systems, we focus on ordered contiguity matching which is easily implemented through the use of double quotes around segments. However, we note that the philosophy of the framework does not change with increased sophistication in the retrieval system – only the expansion sets for the queries have to be appropriately modified.

The theoretical framework proposed for matching terms inside segments at the document side is more general, and can potentially handle ordered, unordered and intruded matches and linguistic dependencies. However, current commercial search engines cannot

Table 3.1: Example of generation of quoted versions for a segmented query.

Segmented query	Quoted versions
we are the people song lyrics	we are the people song lyrics
	we are the people "song lyrics"
	we are "the people" song lyrics
	we are "the people" "song lyrics"
	"we are" the people song lyrics
	"we are" the people "song lyrics"
	"we are" "the people" song lyrics
	"we are" "the people" "song lyrics"

handle such advanced matching scenarios due to the additional time complexity introduced. Hence, in our implementation, we only considered strictly ordered matches for segment terms, supported in any standard text search engine (generally through the use of double quotes, often referred to as *phrase matching*).

We propose an evaluation framework for segmentation algorithms that generates all possible quoted versions of a segmented query (see Table 3.1) and submits each quoted version to the IR engine. The representative query example has been chosen by hand to illustrate the task. The corresponding ranked lists of retrieved documents are then assessed against relevance judgments available for the query-URL pairs. The result quality of the best-performing quoted version is used to measure the retrieval performance of the query segmentation algorithm.

Quoted query version generation

Let the segmentation output by algorithm \mathcal{A} be denoted by $\mathcal{A}(\mathbf{q}) = \mathbf{s}^{\mathbf{q}} = \langle s_1^{\mathbf{q}}, \dots, s_n^{\mathbf{q}} \rangle$. We generate all possible *quoted versions* of the query \mathbf{q} based on the segments in $\mathcal{A}(\mathbf{q})$. In particular, we define $\mathcal{A}_0(\mathbf{q}) = (s_1^{\mathbf{q}} + \dots + s_n^{\mathbf{q}})$ with no quotes on any of the segments, $\mathcal{A}_1(\mathbf{q}) = (s_1^{\mathbf{q}} + \dots + "s_n^{\mathbf{q}}")$ with quotes only around the last segment $s_n^{\mathbf{q}}$, and so on. Since there are n segments in $\mathcal{A}(\mathbf{q})$, this process will generate 2^n versions of the query, $\mathcal{A}_i(\mathbf{q})$, $i = 0, \dots, 2^n - 1$. We note that if $b_i = (b_{i1}, \dots, b_{in})$ be the n -bit binary representation of i , then

$\mathcal{A}_i(\mathbf{q})$ will apply quotes to the j^{th} segment $s_j^{\mathbf{q}}$ iff $b_{ij} = 1$. We deduplicate this set, because $\{\mathcal{A}_i(\mathbf{q}) : i = 0, \dots, 2^n - 1\}$ can contain multiple versions that essentially represent the same quoted query version (when single words are inside quotes). For example, the query versions "harry potter" "game" and "harry potter" game are equivalent in terms of the input semantics of an IR engine. The resulting set of unique quoted query versions is denoted by $\mathcal{Q}_{\mathcal{A}}(\mathbf{q})$.

Document retrieval using IR engine

For each $\mathcal{A}_i(\mathbf{q}) \in \mathcal{Q}_{\mathcal{A}}(\mathbf{q})$ we use the IR engine to retrieve a ranked list \mathcal{O}_i of documents out of the document pool \mathcal{U} that matched the given quoted query version $\mathcal{A}_i(\mathbf{q})$. The number of documents retrieved in each case depends on the IR metrics we will want to use to assess the quality of retrieval. For example, to compute an IR metric at the top k positions, we would require that at least k documents be retrieved from the pool.

Measuring retrieval against relevance judgments

Since we have relevance judgments (\mathcal{R}) for query-URL pairs in $\mathcal{Q} \times \mathcal{U}$, we can now compute IR metrics such as normalized Discounted Cumulative Gain (nDCG) [108], Mean Average Precision (MAP) [197] or Mean Reciprocal Rank (MRR) [223] to measure the quality of the retrieved ranked list \mathcal{O}_i for query \mathbf{q} . We use @ k variants of each of these measures which are defined to be the usual metrics computed after examining only the top- k positions. For example, we can compute nDCG@ k for query \mathbf{q} and retrieved document-list \mathcal{O}_i using the following formula:

$$\text{nDCG@}k(\mathbf{q}, \mathcal{O}_i, \mathcal{R}) = r(\mathbf{q}, \mathcal{O}_i^1) + \sum_{j=2}^k \frac{r(\mathbf{q}, \mathcal{O}_i^j)}{\log_2 j} \quad (3.4)$$

where \mathcal{O}_i^j , $j = 1, \dots, k$, denotes the j^{th} document in the ranked-list \mathcal{O}_i and $r(\mathbf{q}, \mathcal{O}_i^j)$ denotes the associated relevance judgment from \mathcal{R} .

Oracle score using best quoted query version

Different quoted query versions $\mathcal{A}_i(\mathbf{q})$ (all derived from the same basic segmentation $\mathcal{A}(\mathbf{q})$ output by the segmentation algorithm \mathcal{A}) retrieve different ranked lists of documents \mathcal{O}_i . As discussed earlier, automatic apriori selection of a good (or the best) quoted query version is a difficult problem. While different strategies may be used to select a quoted query version, we would like our evaluation of the segmentation algorithm \mathcal{A} to be agnostic of the version-selection step. To this end, we select the best-performing $\mathcal{A}_i(\mathbf{q})$ from the entire set $\mathcal{Q}_{\mathcal{A}}(\mathbf{q})$ of query versions generated and use it to define our *oracle score* for \mathbf{q} and \mathcal{A} under the chosen IR metric [131]. For example, the oracle score $\Omega(\cdot, \cdot)$ for $nDCG@k$ is as defined in the equation below:

$$\Omega_{nDCG@k}(\mathbf{q}, \mathcal{A}) = \max_{\mathcal{A}_i(\mathbf{q}) \in \mathcal{Q}_{\mathcal{A}}(\mathbf{q})} nDCG@k(\mathbf{q}, \mathcal{O}_i, \mathcal{R}) \quad (3.5)$$

where \mathcal{O}_i denotes the ranked list of documents retrieved by the IR engine when presented with $\mathcal{A}_i(\mathbf{q})$ as the input when processing query \mathbf{q} and segmentation algorithm \mathcal{A} , where $\mathcal{A}_i(\mathbf{q})$ is described during quoted query version generation and i indexes the set of quoted versions. We note that $\mathcal{Q}_{\mathcal{A}}(\mathbf{q})$ (set of unique quoted query versions) always contains the original unsegmented version of the query. \mathcal{R} refers to the set of all relevance judgments. We refer to such an $\Omega(\cdot, \cdot)$ as the *Oracle score*. The evaluation metric $nDCG@k$ is defined in Equation 3.4 where k refers to the number of documents retrieved for each query and $nDCG$ is computed after looking at the top- k documents only.

This forms the basis of our evaluation framework. We note that there can also be other ways to define this oracle score. For example, instead of seeking the best IR performance possible across the different query versions, we could also seek the minimum performance achievable by \mathcal{A} irrespective of what version-selection strategy is adopted. This would give us a lower bound on the performance of the segmentation algorithm. However, the main drawback of this approach is that the minimum performance is almost always achieved by the fully quoted version (where every segment is in double quotes) (see Table 3.8). Such a lower bound would not be useful in assessing the comparative performance of query segmentation algorithms.

QVRS computation

Once the oracle scores are obtained for all queries in \mathcal{Q} , we can compute the average oracle score achieved by \mathcal{A} . We refer to this as the Quoted Version Retrieval Score (QVRS) of \mathcal{A} with respect to test set \mathcal{Q} , document pool \mathcal{U} and relevance judgments \mathcal{R} . For example, using the oracle with the $\text{nDCG}@k$ metric, we can define the QVRS score as follows:

$$QVRS(\mathcal{Q}, \mathcal{A}, \text{nDCG}@k) = \frac{1}{|\mathcal{Q}|} \sum_{\mathbf{q} \in \mathcal{Q}} \Omega_{\text{nDCG}@k}(\mathbf{q}, \mathcal{A}) \quad (3.6)$$

Similar QVRS scores can be computed using other IR metrics, such as $\text{MAP}@k$ and $\text{MRR}@k$. For our experiments, we report results using $\text{nDCG}@k$, $\text{MAP}@k$, and $\text{MRR}@k$, for $k = 5$ and 10 as most Web users examine only the first five or ten results.

3.4 Dataset and compared algorithms

In this section, we describe the dataset used and briefly introduce the algorithms compared on our framework.

3.4.1 Test set of queries

We selected a random subset of 500 queries from our query log (Section 1.2). We used the following criteria to filter the logs before extracting a random sample: (1) Exclude queries with non-ASCII characters, (2) Exclude queries that occurred fewer than 5 times and more than 15 times in the logs, and (3) Restrict query lengths to between five and eight words. Shorter queries rarely contain multiple multiword segments, and when they do, the entire queries are mostly single named entities that can be easily detected using dictionaries. Moreover, traditional search engines usually give satisfactory results for short queries. On the other hand, queries longer than eight words (only 3.24% of all queries in our log) are usually error messages, complete NL sentences or song lyrics, that need to be addressed separately.

A note on query frequency. We would like to clarify here that query frequency was not used as a filter to build our training corpus. However, while constructing a small (relative to the size of the training log) test corpus of 500 queries, we chose to impose a frequency restriction for the range five to fifteen, i.e., only queries having a count in this range were considered for sampling. This is because we wanted to focus on relatively rarer queries in the log, which would not have sufficient clickthrough data and hence improved query analysis techniques like segmentation are meaningful to the search engine. The upper (fifteen) and lower (five) bounds of this range were determined heuristically for building the original candidate set for randomly sampling 500 queries. Specifically, we observed that lowering the threshold beyond five introduced several apparently non-sensical queries and ones with typographical errors. Since we did not want to involve manual cleaning of the test set, and since such queries are not desirable while building a benchmark collection, we chose to retain the lower frequency threshold at five.

We denote this set of 500 queries by \mathcal{Q} , the test set of unsegmented queries needed for all our evaluation experiments. The average length of queries in \mathcal{Q} (our dataset) is 5.29 words. The average query length was 4.31 words in the Bergsma and Wang 2007 Corpus² (henceforth, BWC07) [29, 84]. Each of these 500 queries were independently segmented by three human annotators (Computer Science graduate students each issuing around 20-30 search queries per day in the age group 25-30 years) who were asked to mark a contiguous chunk of words in a query as a *segment* if they thought that these words together formed a coherent semantic unit. The annotators were free to refer to other resources and Web search engines during the annotation process, especially for understanding the query and its possible context(s). We shall refer to the three sets of annotations (and also the corresponding human annotators) as H_A , H_B and H_C .

It is important to mention that the queries in \mathcal{Q} have some amount of word level overlap, even though all the queries have very distinct information needs. Thus, a document retrieved from the pool might exhibit good term level match for more than one query in \mathcal{Q} . This makes our corpus an interesting testbed for experimenting with different retrieval systems. There are existing datasets, including BWC07, that could have been used for this study. However, refer to Section 3.6.1 for an account of why building this new dataset was

²<http://bit.ly/1fBJ5O9>, Accessed 31 March 2014.

crucial for our research.

3.4.2 Document pool and RJs

Each query in \mathcal{Q} was segmented using all the nine segmentation strategies considered in our study (six algorithms and three humans). For every segmentation, all possible quoted versions were generated (total 4,746) and then submitted to the Bing API³ and the top ten documents were retrieved. We then deduplicated these URLs to obtain 14,171 unique URLs, forming \mathcal{U} . As mentioned earlier, we used nine strategies for retrieval and found that on an average, adding the quoted query versions by any one of the strategies to the versions generated by the remaining eight strategies resulted in about one new quoted query version for every two queries. These new versions may or may not introduce new documents to the pool. We observed that for 71.4% of the queries there is less than 50% mean overlap between the top ten URLs retrieved for the different quoted versions. This indicates that different ways of quoting the segments in a query does make a difference in the search results. By varying the pool depth (ten in our case), one can roughly control the number of relevant and non-relevant documents entering the collection.

For each query-URL pair, where the URL has been retrieved for at least one of the quoted versions of the query (approx. 28 per query), we obtained three independent sets of relevance judgments from human users. These users (again, graduate students in the 25 – 30 year age group) were different from annotators H_A , H_B and H_C who marked the segmentations, but having similar familiarity with search systems. For each query, the corresponding set of URLs was shown to the users after deduplication and randomization (to prevent position bias for top results), and the users were asked to mark whether the URL was *irrelevant* (score = 0), *partially relevant* (score = 1) or *highly relevant* (score = 2) to the query. We then computed the average rating for each query-URL pair (the entire set forming \mathcal{R}) over the three annotators, which has been used for subsequent nDCG, MAP and MRR computations. nDCG [108] is defined as in Equation 3.4. MAP [197] and MRR [223] are defined as follows:

³<http://msdn.microsoft.com/en-us/library/dd251056.aspx>, Accessed 31 March 2014.

Table 3.2: Segmentation algorithms compared on our framework.

Algorithm	Training data
Li et al. [137]	Click data, Web n -gram probabilities
Hagen et al. [84]	Web n -gram frequencies, Wikipedia titles
Proposed	Query logs
Proposed + Wiki	Query logs, Wikipedia titles
PMI-W [84]	Web n -gram probabilities (used as baseline)
PMI-Q	Query logs (used as baseline)

$$MAP = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{\sum_{k=1}^n P@k \times rel(k)}{\text{Number of relevant documents}} \quad (3.7)$$

$$MRR = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{Rank_q} \quad (3.8)$$

where $|Q|$ is the number of queries in our query test set Q , n is the number of search results, $P@k$ is the *precision* at rank k (i.e., the fraction of the k retrieved documents that are also relevant), $rel(k)$ is zero or one according as the document at rank k is non-relevant or relevant, and $Rank_q$ is the rank of the first relevant or “correct” result for query q . Since we needed to convert our graded relevance judgments to binary values for computing $MAP@k$, URLs with ratings of 1 and 2 were considered as relevant (responsible for the generally high values) and those with 0 as irrelevant. For MRR, only URLs with ratings of 2 were considered as relevant. Please refer to Table 3.9 in Section 3.6.3 for inter-annotator agreement figures and other related discussions. We will refer to this dataset as **SGCL12** (last name initials of the four inventors Saha Roy, Ganguly, Choudhury and Laxman, and the year in which this work was performed (2012)) in future sections and chapters.

3.4.3 Segmentation algorithms

Table 3.2 lists the six segmentation algorithms that have been studied in this work. Li et al. [137] use the expectation maximization algorithm to arrive at the most probable segmentation, while Hagen et al. [84] show a simple frequency-based method produces a performance comparable to the state-of-the-art. Our proposed technique (Section 3.2) uses only query logs for segmenting queries. As discussed earlier, the technique can be improved if Wikipedia titles are used for the detection of long named entities (Section 3.2.1). The Point-wise Mutual Information (PMI)-based algorithms [84, 114, 185] are used as baselines. The thresholds for PMI-W and PMI-Q were chosen to be 8.141 and 0.156 respectively, that maximized the *Seg-F* (Section 3.5.2) on our development set [84].

3.4.4 Public release of data

The test set of search queries along with their manual and some of the algorithmic segmentations, the theoretical best segmentation output that can serve as an evaluation benchmark (BQV_{BF} in Section 3.5.1), and the list and contents of URLs that serve as our document corpus is available for public use⁴. The relevance judgments for the query-URL pairs have also been made public which will enable the community to use this dataset for evaluation of any new segmentation algorithm.

3.5 Experiments and observations

In this section we present experiments, results and the key inferences made from them.

⁴<http://cse.iitkgp.ac.in/resgrp/cnerg/qa/querysegmentation.html>, Accessed 31 March 2014.

Table 3.3: Oracle results for IR-based evaluation of segmentation algorithms.

Metric	Unseg. query	Li et al. [137]	Hagen et al. [84]	Proposed	Proposed + Wiki	PMI-W	PMI-Q	H_A	H_B	H_C	BQV_{BF}
nDCG@5	0.688	0.752*	0.763*	0.745	0.767*	0.691	0.766*	0.770	0.768	0.759	0.825
nDCG@10	0.701	0.756*	0.767*	0.751	0.768*	0.704	0.767*	0.770	0.768	0.763	0.832
MAP@5	0.882	0.930*	0.942*	0.930*	0.945*	0.884	0.932*	0.944	0.942	0.936	0.958
MAP@10	0.865	0.910*	0.921*	0.910*	0.923*	0.867	0.912*	0.923	0.921	0.916	0.944
MRR@5	0.538	0.632*	0.649*	0.609	0.650*	0.543	0.648*	0.656	0.648	0.632	0.711
MRR@10	0.549	0.640*	0.658*	0.619	0.658*	0.555	0.656*	0.665	0.656	0.640	0.717

The highest value in a row (excluding the BQV_{BF} column) and those with no statistically significant difference with the highest value are marked in **boldface**. The values for algorithms that perform better than or have no statistically significant difference with the *minimum* of the human segmentations are marked with *. The paired t -test was performed and the null hypothesis was rejected if the p -value was less than 0.05.

3.5.1 IR experiments

For the retrieval-based evaluation experiments, we use the Lucene⁵ text retrieval system, which is publicly available as a code library. In its default configuration, Lucene does not perform any automatic query segmentation, which is very important for examining the effectiveness of segmentation algorithms in an IR-based scheme. Double quotes can be used in a query to force Lucene to match the quoted *phrase* (in Lucene terms) exactly in the documents. Starting with the segmentations output by each of the six algorithms as well as the three human annotations, we generated all possible quoted query versions, which resulted in a total of 4,746 versions for the 500 queries. In the notation of Section 3.3, this corresponds to generating $\mathcal{Q}_{\mathcal{A}}(\mathbf{q})$ for each segmentation method \mathcal{A} (including one for each human segmentation) and for every query $\mathbf{q} \in \mathcal{Q}$. These quoted versions were then passed through Lucene to retrieve documents from the pool. For each segmentation scheme, we then use the oracle described in Section 3.3 to obtain the query version yielding the best result (as determined by the IR metrics – nDCG, MAP and MRR computed according to the human relevance judgments). These oracle scores are then averaged over the query set to give us the QVRS measures.

⁵<http://lucene.apache.org/java/docs/index.html>, Accessed 31 March 2014.

Table 3.4: Matching metrics with BQV as reference.

Metric	Unseg. query	Li et al. [137]	Hagen et al. [84]	Proposed	Proposed + Wiki	PMI-W	PMI-Q	H_A	H_B	H_C	BQV_{BF}
Qry-Acc	0.044	0.056	0.082*	0.058	0.094*	0.046	0.104*	0.086	0.074	0.064	1.000
Seg-Prec	0.226*	0.176*	0.189*	0.206*	0.203*	0.229*	0.218*	0.176	0.166	0.178	1.000
Seg-Rec	0.325*	0.166*	0.162*	0.210*	0.174*	0.323*	0.196*	0.144	0.133	0.154	1.000
Seg-F	0.267*	0.171*	0.174*	0.208*	0.187*	0.268*	0.206*	0.158	0.148	0.165	1.000
Seg-Acc	0.470	0.624	0.661*	0.601	0.667*	0.474	0.660*	0.675	0.675	0.663	1.000

The highest value in a row (excluding the BQV_{BF} column) and those with no statistically significant difference with the highest value are marked in **boldface**. The values for algorithms that perform better than or have no statistically significant difference with the *minimum* of the human segmentations are marked with *. The paired t -test was performed and the null hypothesis was rejected if the p -value was less than 0.05.

Table 3.5: PMI-Q and Li et al. [137] with respect to matching and IR metrics.

Metric	nDCG@10	MAP@10	MRR@10	Qry-Acc	Seg-Prec	Seg-Rec	Seg-F	Seg-Acc
PMI-Q	0.767	0.912	0.656	0.341	0.448	0.487	0.467	0.810
Li et al. [137]	0.756	0.910	0.640	0.375	0.524	0.588	0.554	0.810

The highest value in a column is marked in **boldface**.

The results are summarized in Table 3.3. Different rows represent the different IR metrics that were used and columns correspond to different segmentation strategies. The second column (marked “Unseg. Query”) refers to the original unsegmented query. This can be assumed to be generated by a trivial segmentation strategy where each word is always a separate segment. Columns 3-8 denote the six different segmentation algorithms and 9-11 (marked H_A , H_B and H_C) represent the human segmentations. The last column represents the performance of the **best quoted versions** (denoted by BQV_{BF} in table) of the queries which are computed by *brute force*, i.e. an exhaustive search over all possible ways of quoting the parts of a query (2^{l-1} possible quoted versions for an l -word query) irrespective of any segmentation algorithm. The results are reported for two sizes of retrieved URL lists (k), namely five and ten.

The first observation we make from the results is that human as well as all algorithmic segmentation schemes consistently outperform unsegmented queries for all IR metrics.

Second, we observe that the performance of some segmentation algorithms are comparable and sometime even marginally better than some of the human annotators. Finally, we observe that there is a considerable scope for improving IR performance through better segmentation (all values less than BQV_{BF}). We found that applying the proposed query segmentation algorithm (using Wikipedia titles) can be potentially useful to 334 out of the 500 queries in our test set on $nDCG@10$, i.e., about 67% of the queries can be benefited. For the remaining 166 queries, the non-segmented query will retrieve equally good results as a segmented one. This exact number will vary somewhat on the exact segmentation algorithm used and the associated metric, but from the general result trends of statistical significance, the variation will not be very high. The inferences from these observations are stated later in this section.

3.5.2 Performance under traditional matching metrics

In the next set of experiments, we study the utility of traditional matching metrics that are used to evaluate query segmentation algorithms against a gold standard of human segmented queries (henceforth referred to as the *reference* segmentation). These metrics are listed below [84]:

1. **Query accuracy (*Qry-Acc*):** The fraction of queries where the output matches exactly with the reference segmentation.
2. **Segment precision (*Seg-Prec*):** The ratio of the number of segments that overlap in the output and reference segmentations to the number of output segments, averaged across all queries in the test set.
3. **Segment recall (*Seg-Rec*):** The ratio of the number of segments that overlap in the output and reference segmentations to the number of reference segments, averaged across all queries in the test set.
4. **Segment F-score (*Seg-F*):** The harmonic mean of *Seg-Prec* and *Seg-Rec*.
5. **Segmentation accuracy (*Seg-Acc*):** The ratio of correctly predicted boundaries and non-boundaries in the output segmentation with respect to the reference, averaged

across all queries in the test set.

We computed the matching metrics for various segmentation algorithms against H_A , H_B and H_C . According to these metrics, “Proposed + Wiki” turns out to be the best algorithm which agrees with the results of IR evaluation. However, the average Kendall-Tau rank correlation coefficient [118] between the ranks of the strategies as obtained from the IR metrics (Table 3.3) and the matching metrics⁶ was only 0.75. This indicates that matching metrics are not perfect predictors for IR performance. In fact, we discovered some costly flaws in the relative ranking produced by matching metrics. One such case was rank inversions between Li et al. [137] and PMI-Q. The relevant results are shown in Table 3.5, which demonstrate that while PMI-Q consistently performs better than Li et al. [137] under IR-based measures, the opposite inference would have been drawn if we had used any of the matching metrics.

In Bergsma and Wang [29], human annotators were asked to segment queries such that segments matched exactly in the relevant documents. This essentially corresponds to determining the best quoted versions for the query. Thus, it would be interesting to study how traditional matching metrics would perform if the humans actually marked the best quoted versions. In order to evaluate this, we used the matching metrics to compare the segmentation outputs by the algorithms and human annotations against BQV_{BF} . The corresponding results are presented in Table 3.4. The results show that matching metrics are very poor indicators of IR performance with respect to the BQV_{BF} . For example, for three out of the five matching metrics, the unsegmented query is ranked the best. This shows that even if human annotators managed to correctly guess the best quoted versions, the matching metrics would fail to estimate the correct relative rankings of the segmentation algorithms with respect to IR performance. This fact is also borne out in the Kendall-Tau rank correlation coefficients reported in Table 3.6. Another interesting observation from these experiments is that *Seg-Acc* emerges as the best matching metric with respect to IR performance, although its correlation coefficient is still much below one.

⁶This coefficient is 1 when there is perfect concordance between the rankings, and -1 if the trends are completely reversed.

Table 3.6: Kendall-Tau coefficients between IR and matching metrics.

Metric	Qry-Acc	Seg-Prec	Seg-Rec	Seg-F	Seg-Acc
nDCG@10	0.432	-0.854	-0.886	-0.854	0.674
MAP@10	0.322	-0.887	-0.920	-0.887	0.750
MRR@10	0.395	-0.782	-0.814	-0.782	0.598

The highest value in a row is marked in **boldface**.

3.5.3 Inferences

Segmentation is helpful for IR. By definition, $\Omega(\cdot, \cdot)$ (i.e., the oracle) values for every IR metric for any segmentation scheme are at least as large as the corresponding values for the unsegmented query. Nevertheless, for every IR metric, we observe significant performance benefits for all the human and algorithmic segmentations (except for PMI-W) over the unsegmented query. This indicates that segmentation is indeed helpful for boosting IR performance. Thus, our results validate the prevailing notion and some of the earlier observations [28, 137] that segmentation can help improve IR.

Human segmentations are a good proxy, but not a true gold standard. Our results indicate that human segmentations perform reasonably well on IR metrics. The best of the human annotators beats all the segmentation algorithms, on almost all the metrics. Therefore, evaluation against human annotations can indeed be considered as the second best alternative to an IR-based evaluation (though see below for criticisms of current matching metrics). However, if the objective is to improve IR performance, then human annotations cannot be considered a true gold standard. There are at least three reasons for this, as explained below.

First, in terms of IR metrics, some of the state-of-the-art segmentation algorithms are performing as well as human segmentations (no statistically significant difference). Thus, further optimization of the matching metrics against human annotations is not going to improve the IR performance of the segmentation algorithms. Thus, evaluation on human annotations might become a limiting factor for the current segmentation algorithms.

Second, the IR performance of the best quoted version of the queries derived through our framework is significantly better than that of human annotations (last column, Table 3.3). This means that humans fail to predict the correct boundaries in many instances. Thus, there is a scope for improvement for human annotations.

Third, IR performance of at least one of the three human annotators (H_C) is worse than some of the algorithms studied. In other words, while some annotators (such as H_A) are good at guessing the “correct” segment boundaries that will help IR, not all annotators can do it well. Therefore, unless the annotators are chosen and guided properly, one cannot guarantee the quality of annotated data for query segmentation. If the queries in the test set have multiple intents, this issue becomes an even bigger concern.

Matching metrics are misleading. As discussed earlier and demonstrated by Tables 3.4 and 3.6, the matching metrics provide unreliable ranking of the segmentation algorithms even when applied against a true gold standard, BQV_{BF} , that, by definition, maximizes IR performance. This counter-intuitive observation can be explained as follows: either the matching metrics, or the IR metrics (or probably both) are misleading. Given that IR metrics are well-tested and generally assumed to be acceptable, we are forced to conclude that the matching metrics do not really reflect the quality of a segmentation with respect to a gold standard. Indeed, this can be illustrated by a simple example.

Example. Let us consider an example query to be the looney toons show cartoon network, whose best quoted version turns out to be "the looney toons show" "cartoon network". The underlying segmentation that can give rise to this quoted version and therefore can be assumed to be the reference is:

Ref: the looney toons show | cartoon network

The segmentations

(1) the looney | toons show | cartoon | network

(2) the | looney | toons show cartoon | network

are equally bad if one considers the matching metrics of $Qry-Acc$, $Seg-Prec$, $Seg-Rec$ and $Seg-F$ (all values being zero) with respect to the reference segmentation. $Seg-Acc$ values for the two segmentations are $3/5$ and $1/5$ respectively. However, the BQV for (1) ("the looney" "toons show" cartoon network) fetches better pages than the BQV of (2) (the looney toons show cartoon network). So the segmentation (2)

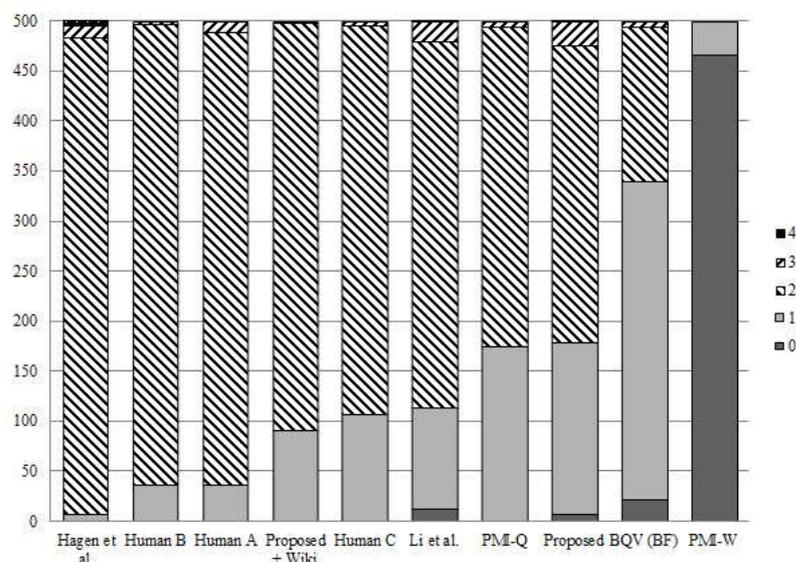


Figure 3.1: Distribution of multiword segments across segmentation strategies.

provides no IR benefit over the unsegmented query and hence performs worse than (1) on IR metrics. However, the matching metrics, except for *Seg-Acc* to some extent, fail to capture this difference between the segmentations.

Distribution of multiword segments across queries gives insights about effectiveness of segmentation strategy. The limitation of the matching metrics can also be understood from the following analysis of the multiword segments in the queries. Figure 3.1 shows the distribution of queries having a specific number of multiword segments (for example, 1 in the legend indicates the proportion of queries having *one* multiword segment) when segmented according to the various strategies. We note that for Hagen et al. [84], H_B , H_A and “Proposed + Wiki”, almost all of the queries have two multiword segments. For H_C , Li et al. [137], PMI-Q and the proposed method, the proportion of queries that have only one multiword segment increases. Finally, PMI-W has almost negligible queries with a multiword segment. BQV_{BF} is different from all of them and has a majority of queries with one multiword segment. Now given that the first group generally does the best in IR, followed by the second, we can say that out of the two multiword segments marked by these strategies, only one needs to be quoted. PMI-W, as well as unsegmented queries, are bad because these schemes cannot detect the one crucial multiword segment, quoting which improves the performance. Nevertheless, these schemes do well for matching met-

rics against BQV_{BF} because both have a large number of single word segments. Clearly this is not helpful for IR. Finally, the proposed scheme, without Wikipedia titles, performs poorly despite being able to identify a multiword segment in most of the cases because it is not identifying the one that is important for an exact-match based retrieval performance.

Hence, the matching metrics are misleading due to two reasons. First, they do not take into account that splitting a useful segment (i.e., a segment which should be quoted to improve IR performance) is less harmful than joining two unrelated segments. Second, matching metrics are, by definition, agnostic to which segments are useful for IR. Therefore, they might unnecessarily penalize a segmentation for not agreeing on the segments which should not be quoted, but are present in the reference human segmentation. While the latter is an inherent problem with any evaluation against manually segmented datasets, the former can be resolved by introducing a new matching metric that differentially penalizes splitting and joining of segments. However, we would like to emphasize here that with the IR system expected to grow in complexity in the future (supporting more flexible matching criteria), the need for an IR-based evaluation like ours' becomes imperative.

Based on our new evaluation framework and corresponding experiments, we observe that "Proposed + Wiki" has the best performance. Nevertheless, the algorithms are trained and tested on different datasets, and therefore, a comparison amongst the algorithms might not be entirely fair. This is not a drawback of the framework and can be circumvented by appropriately tuning all the algorithms on similar datasets. However, the objective of the current work is not to compare segmentation algorithms; rather, it is to introduce the evaluation framework, gain insights from the experiments and highlight the drawbacks of human segmentation-based evaluation.

Effect of choice of β on algorithm performance. The parameter β (Section 3.2) in our segmentation algorithm controls the number of entries that enter the segmentation lexicon and has values specific to each n -gram instead of a global threshold. We had tuned β on a development set to maximize the segmentation accuracy against a set of human annotations, trying out values from 0 to k through steps of $0.1k$, i.e, 0, $0.1k$, $0.2k$, and so on upto k . The way we have defined β , the ideal value of β depends on the query log and not on the document collection. However, tuning β to optimize different metrics will result in different optimal values of beta. For example, when we tried to optimize β

Table 3.7: Effect of β on IR performance.

β	nDCG@10
0	0.756
0.1k	0.754
0.2k	0.753
0.3k	0.753
0.4k	0.752
0.5k	0.751
0.6k	0.751
0.7k	0.750
0.8k	0.747
0.9k	0.747
k	0.743

The highest value in a row is marked in **boldface**.

on $nDCG@10$, an IR metric, we obtained the best β to be 0, which effectively means no pruning on statistical significance. However, in a practical situation, due to memory and time complexity concerns, it may not be possible to admit all n -grams into the lexicon, and having β provides us with a principled way of pruning the lexicon entries. The effect of variation in β on nDCG@10 is shown in Table 3.7.

3.6 Related issues

In this section, we will briefly discuss a few related issues that are essential for understanding certain design choices and decisions made during the course of this research.

3.6.1 Motivation for a new dataset

TREC data has been a popular choice for conducting IR-based experiments throughout the past decade. Since there is no track specifically geared towards query segmentation, the queries and *qrels* (query-relevance sets) from the ad hoc retrieval task for the Web Track would seem the most relevant to our work. However, 74% of the 50 queries in the 2010 Web

track ad hoc task⁷ had less than three words. Also, when these 50 queries were segmented using the six algorithms, half of the queries did not have a multiword segment. As discussed earlier, query segmentation is useful but not necessarily for all types of queries. The benefit of segmentation may be observed only when there are multiple multiword segments in the queries. The TREC Million Query Track, last held in 2009, has a much larger set of 40,000 queries⁸, with a better coverage of longer queries. But since the goal of the track is to test the hypothesis that a test collection built from several incompletely judged topics is more useful than a collection built using traditional TREC pooling, there are only about 35,000 query-document relevance judgments for the 40,000 queries. Such sparse *qrels* are not suitable here – incomplete assessments, especially for documents near the top ranks, could cause crucial errors in system comparisons. Yet another option could have been to use BWC07 as \mathcal{Q} and create the corresponding \mathcal{U} and \mathcal{R} . However, this query set is known to suffer from several drawbacks [84]. A new dataset for query segmentation⁹ containing manual segment markups collected through crowdsourcing has been recently made publicly available (after we had completed construction of our set) by Hagen et al. [84], but it lacks query-document relevance judgments. These factors motivated us to create a new dataset suitable for our framework, which has been made available online for use by the research community (see Section 3.4.4).

3.6.2 Retrieval using Bing

Microsoft’s Bing is a large-scale commercial Web search engine that provides an API service. Instead of Lucene, which is too simplistic, we could have used Bing as the IR engine in our framework. However, such a choice suffers from two drawbacks. First, Bing might already be segmenting the query with its own algorithm as a preprocessing step. Second, there is a serious replicability issue: the document pool that Bing uses, i.e. the Web, changes dynamically with documents added and removed from the pool on a regular basis. This makes it difficult to publish a static dataset with relevance judgments for all appropriate query-URL pairs that the Bing API may retrieve even for the same set of queries. In

⁷<http://trec.nist.gov/data/web10.html>, Accessed 31 March 2014.

⁸<http://trec.nist.gov/data/million.query09.html>, Accessed 31 March 2014.

⁹<http://bit.ly/1mETNX5>, Accessed 31 March 2014.

Table 3.8: IR-based evaluation using Bing API.

Metric	Unseg. query	All quoted for Proposed + Wiki	Oracle for Proposed + Wiki
nDCG@10	0.882	0.823	0.989*
MAP@10	0.366	0.352	0.410*
MRR@10	0.541	0.515	0.572*

The highest value in a row is marked in **boldface**. Statistically significant ($p < 0.05$ for paired t -test) improvement over the unsegmented query is marked with *.

view of this, the main results were reported in this thesis using Lucene.

However, since we used Bing API to construct \mathcal{U} and corresponding \mathcal{R} , we have the evaluation statistics using the Bing API as well. In Table 3.8 we present the results for nDCG@10, MRR@10 and MAP@10 for “Proposed + Wiki”. The table reports results for three quoted version-selection strategies: (i) Unsegmented query only (equivalent to each word being within quotes), (ii) All segments quoted, and (iii) *QVRS* (oracle for “Proposed + Wiki”). For all the three metrics, *QVRS* is statistically significantly higher than results for the unsegmented query. Thus, segmentation can play an important role towards improving IR performance of the search engine. We note that the strategy of quoting all the segments is, in fact, detrimental to IR performance. This emphasizes the point that how the segments should be matched in the documents is a very important research challenge. Instead of quoting all the segments, our proposal here is to assume an oracle that will suggest which segments to quote and which are to be left unquoted for the best IR performance. Philosophically, this is a major departure from the previous ideas of using quoted segments, where issuing a query by quoting all the segments implies segmentation to be a way to generate a fully quoted version of the query (all segments in double quotes). This definition severely limits the scope of segmentation, which ideally should be thought of as a step forward towards better query understanding.

Table 3.9: Inter-annotator agreement on features as observed from our experiments.

Feature	Pair 1	Pair 2	Pair 3	Mean
Qry-Acc	0.728	0.644	0.534	0.635
Seg-Prec	0.750	0.732	0.632	0.705
Seg-Rec	0.756	0.775	0.671	0.734
Seg-F	0.753	0.753	0.651	0.719
Seg-Acc	0.911	0.914	0.872	0.899
Rel. judg.	0.962	0.959	0.969	0.963

For relevance judgments, only pairs of (0, 2) and (2, 0) were considered disagreements.

3.6.3 Inter-annotator agreement

Inter-annotator agreement (IAA) is an important indicator for reliability of manually created data. Table 3.9 reports the pairwise IAA statistics for H_A , H_B and H_C . Since there are no universally accepted metrics for IAA, we report the values of the five matching metrics when one of the annotations (say H_A) is assumed to be the reference and the remaining pair (H_B and H_C) is evaluated against it (average reported). As is evident from the table, the values of all the metrics, except for *Seg-Acc*, is less than 0.78 (similar values reported in Tan and Peng [216]), which indicates a rather low IAA. The value for *Seg-Acc* is close to 0.9, which to the contrary, indicates reasonably high IAA [216]. The last row of Table 3.9 reports the IAA for the three sets of relevance judgments (therefore, the actual pairs for this column are different from that of the other rows). The agreement in this case is quite high.

There might be several reasons for low IAA for segmentation, such as lack of proper guidelines and/or an inherent inability of human annotators to mark the correct segments of a query. Low IAA raises serious doubts about the reliability of human annotations for query segmentation. On the other hand, high IAA for relevance judgments naturally makes these annotations much more reliable for any evaluation, and strengthens the case for our IR-based evaluation framework which only relies on relevance judgments. We note that ideally, relevance judgments should be obtained from the user who has issued the query. These have been referred to as *gold* annotations in previous research [21], as opposed to

silver or *bronze* annotations which are obtained from expert and non-expert annotators respectively who have not issued the query. Gold annotations are preferable over silver or bronze ones due to relatively higher IAA. Our annotations are silver standard, though very high IAA essentially indicates that they might be as reliable as gold standard. The high IAA might also be due to the unambiguous nature of the queries.

3.7 Enhancement with POS tags

Having discussed our primary algorithm and evaluation framework for flat query segmentation, we will explore an interesting idea and discuss how segmentation relying on word association scores (WAS) can be improved by using POS sequences to detect relatively rare segments. Conceptually, query segmentation is analogous to *chunking* of NL text. Chunking breaks sentences into syntactic structures like noun phrases or prepositional phrases [3]. Automatic text chunking is usually performed by learning part-of-speech (POS) patterns from large volumes of human annotated corpora [123, 181]. The annotations, in turn, are performed using linguistic rules guided by the grammar of the language. Chunking, however, is distinct from the identification of multiword expressions (MWEs) like `the last straw`, which are word sequences whose meanings are non-compositional [78]. Such MWEs are generally identified using word association scores (WAS) like PMI [52] or LLR [65]. Lack of well-defined grammatical syntax and absence of human POS-annotated query logs have led researchers to use *MWE detection techniques* to perform unsupervised *query segmentation* [82]. However, several potential query segments (chunks) are not MWEs (e.g., `buy online`, `how to`, `driving rules`). Our work aims to bridge this gap between the concepts and techniques behind query segmentation.

Our generic strategy to augment WAS-based segmentation techniques with POS information is as follows. First, we construct a lexicon of potential word n -grams from the corpus (say, a query log). This is usually the first step in a WAS-based query segmentation algorithm [84, 216]. Then we identify underlying POS sequences (or POS n -grams) of the lexicon entries, and count their frequency of occurrence. A modified score is then computed for each word n -gram which is a combination of its original WAS and the lexicon frequency of its POS n -gram. New entries are introduced into the lexicon according to this

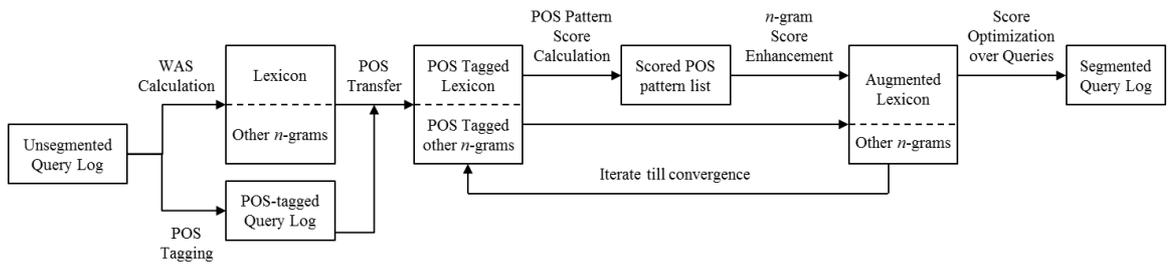


Figure 3.2: Augmenting unsupervised query segmentation with POS sequences.

modified score. This process is iterated till convergence of the lexicon. This *augmented lexicon* is used for query segmentation, where the newly derived scores perform the role of the original WAS. This lexicon augmentation is an *offline* process and thus does not add any runtime overhead to the segmentation process.

We conduct experiments using an English POS tagger based on the Penn Treebank (PTB) tagset and a recently proposed compact universal tagset [173]. We also experiment with a tagset that has been induced from the query log in a completely unsupervised fashion [31]. Our results show that POS information from all the three tagsets can lead to significant performance improvement for an unsupervised segmentation algorithm.

Figure 3.2 presents a schematic of the proposed framework to combine WAS and POS information for unsupervised query segmentation. Our method requires a POS tagger for queries, a WAS to be computed from a query log, and a lexicon augmentation scheme.

3.7.1 POS tagging

POS tagging is defined as the process of assigning POS labels to the words of a text fragment based on the context. For example, if the input text fragment is `the yellow book`, the corresponding POS labels would be `the_DT yellow_JJ book_NN` (i.e., Determiner, Adjective and Noun respectively). The framework proposed here is not tied to any specific POS tagging strategy or tagset. To bring out this fact, here we perform experiments with two different taggers – the supervised Stanford Log-Linear POS Tagger [219], and a fully unsupervised POS induction technique using graph clustering based on Bie-

Table 3.10: Sample clusters produced by Bie-S.

Cluster 1:	bake, casserole, dessert, fry, meatballs, ...
Cluster 2:	athletics, baseball, cycling, football, golf, ...
Cluster 3:	army, citizenship, customs, defence, government, ...
Cluster 4:	battlefield, diablo, godfather, hitman, sims, ...

mann [31]. The Stanford Tagger uses the PTB tagset that has 36 tags¹⁰. Recently, Petrov et al. [173] proposed a universal tagset (UTS) which contains 12 tags and provided a mapping between the PTB (and many other) tags and UTS tags. In order to understand the effect of granularity of the tagset, we also run experiments for the UTS tagset, which are obtained by one-to-one mappings of the PTB tags of the queries labeled by the Stanford Tagger.

Since English Web search queries do not necessarily follow the syntax of the English language, the appropriateness of tagsets such as PTB or UTS for tagging Web search queries is questionable. Therefore, we also experiment with a completely unsupervised POS induction technique based on graph clustering by Biemann [31] that induces the tagset as well as the tagger from the first principles without making any assumptions about the syntactic structure of the language. Moreover, the technique automatically generates the number of tags (clusters). The original method is simplified for queries so as to assign a unique tag to a word (by suitably removing the Viterbi tagging step in [31]), irrespective of the context. This ensures a fast and lightweight tagger that is suitable in a Web search setting. We refer to this tagger (and the associated tagset) as Bie-S (S = Simplified). Table 3.10 shows parts of sample clusters generated by the Bie-S algorithm on our query log. As we can see, clusters are focused around *topics* like food, sports, governance, and video games. The method resulted in 405 distinct tags.

3.7.2 Lexicon augmentation scheme

Intuition. Traditional unsupervised query segmentation algorithms use a WAS to build a lexicon of meaningful n -grams [216], which is subsequently used to generate the most likely segmentation for a query. Such methods fail to identify rare word n -grams as poten-

¹⁰<http://bit.ly/JY5lwb>, Accessed 31 March 2014.

tial segments. The rarer n -grams, nevertheless, almost always follow the same syntactic structure (or POS sequence pattern) as the frequent ones, and their rarity is by virtue of the rarity of the words rather than the underlying syntactic construction. This fundamental observation led us to the intuition that the WAS of rarer word n -grams could be boosted up if the underlying POS pattern is observed frequently in the set of segments originally extracted by a WAS. In other words, we can learn the common syntactic structures of the segments by extracting statistically significant word co-occurrences, and then in turn, use this knowledge to extract rarer segments. This intuition, which is the primary contribution of this work, is formalized in the following steps.

1. Given a query log Q , the queries are POS tagged using a tagger. Also, a WAS is computed for every unique word n -gram, \mathbf{w} , appearing in Q .
2. An initial lexicon L_0 is constructed with the word n -grams (say, the rolling stones) that have $\text{WAS} \geq \delta$, a user-defined threshold. Let L_i be the lexicon after the i^{th} iteration of the algorithm.
3. Every entry in L_i is assigned a unique POS tag sequence based on how that word n -gram was tagged in Q (say, the_DT rolling_VBG stones_NNS). In the rare case when the same word n -gram is tagged differently in different queries, we assign the most common POS tags to words in that n -gram sequence as found in Q .
4. For each POS n -gram (or POS pattern) P_j (say, DT-VBG-NNS), we count the number of times P_j appears in L_i . Let us denote this by $\text{count}(P_j, i)$.
5. We define a score for P_j as follows:

$$\text{score}(P_j, i + 1) = \text{score}(P_j, i) \ln(e + \alpha e^{-i/\ln(1 + \text{count}(P_j, i))}) \quad (3.9)$$

where iteration $i \geq 0$ and α is a tuning parameter. We define $\text{score}(P_j, 0) = 1$.

6. The WAS for every unique \mathbf{w} in Q is then combined with its corresponding pattern score as shown in Equation 3.10:

$$\text{score}(\mathbf{w}, i + 1) = \text{score}(\mathbf{w}, i) \times \text{score}(P_{\text{POS}(\mathbf{w})}, i) \quad (3.10)$$

where $\text{POS}(\mathbf{w})$ is the index j of the underlying POS pattern of \mathbf{w} . We define $\text{score}(\mathbf{w}, 0)$ to be $\text{WAS}(\mathbf{w})$.

7. L_{i+1} is constructed by including all \mathbf{w} for which $score(\mathbf{w}, i) \geq \delta$.
8. Steps 3 to 8 are then repeated until convergence, where $L_i \equiv L_{i-1}$.

The multiplicative factor in Equation 3.9 is based on the proximity transformation function used by Tao and Zhai [217], which has all the mathematical characteristics to suit the current purpose: (a) the value of the function diminishes with each successive iteration, which is necessary because otherwise eventually all n -grams will enter the lexicon; (b) as i grows, this factor approaches unity, which ensures convergence; (c) this factor is proportionate to the logarithm of $count(P_j, i)$, which is usually desirable because frequency distributions of n -grams typically follow power laws.

We use our proposed segmentation algorithm without Wikipedia titles to prevent masking the effect of POS sequences. In our POS-augmented approach, we *do not* use the initial lexicon L_0 to segment queries; rather we use the lexicon $L_{\hat{i}}$ where \hat{i} is the iteration at which convergence occurs. We refer to the segmentation produced using L_0 as the original segmentation *Orig*, over which we aim to improve.

3.7.3 Experiments

Our Bing query log (Section 1.2) was POS tagged using the Stanford Tagger (using both PTB and UTS tags) as well as the Bie-S algorithm. For evaluating segmentations generated by our approach, we use our retrieval-based approach on the set of 500 queries described earlier (SGCL12, Section 3.4). Queries 1 to 250 have been used as the development set and 251 to 500 as the test set.

Table 3.11 reports nDCG, MAP and MRR for the original algorithm and the POS-augmented strategy for the three tagsets used. All the tagsets result in improvements over the original segmentation, which is statistically significant for nDCG@5. This implies that many better pages are presented in the top-five slots, which is very important for a Web search setting. The improvements are because of the meaningful but rare n -grams that are discovered by our POS-based method and were originally missed by the WAS alone. At convergence, the PTB, UTS and Bie-S tagsets added $337k$, $447k$ and $452k$ ($k =$

Table 3.11: IR performance with different tagsets.

Metric	Orig	PTB	UTS	Bie-S
nDCG@5	0.743	0.751 [†]	0.751 [†]	0.751 [†]
nDCG@10	0.747	0.753	0.752	0.752
MAP	0.901	0.905	0.905	0.905
MRR	0.587	0.601	0.598	0.602

Statistically significant improvement over *Orig* is marked using [†] (one-tailed paired *t*-test, $p < 0.05$).

Table 3.12: Gaining, unaffected and losing queries.

Tagset	Gain	Same	Loss
PTB	67 (+0.048)	150	33 (−0.060)
UTS	57 (+0.055)	162	31 (−0.068)
Bie-S	67 (+0.042)	140	43 (−0.050)

Numbers in parentheses indicate the average gain/loss in nDCG@10 for each class of queries with respect to the original segmentations.

thousand) new word n -grams to L_0 respectively. Mean IR performances on the test set for the three tagsets are almost exactly the same (a gain-loss analysis reported later reveals some differences). With respect to this application, the UTS tagset does *not* result in any *loss of information* when the 36 PTB tags are collapsed to the 12 UTS tags.

Table 3.12 reports the numbers of gaining, unaffected and losing queries (in terms of nDCG@10, with respect to the original segmentation without POS information) for each tagset for the optimum α -s. We observe that our method benefits a significant proportion of the queries (23 – 27%), much higher than the fraction of queries for which the nDCG value drops (12 – 17%). The three tagsets affect the relatively same number of queries in all the three ways, even though the number of queries negatively affected is slightly higher for the Bie-S tagset. Since these queries are relatively rare with query frequency

Table 3.13: Example queries showing IR improvement.

Tagset	Segmented Query	nDCG@10
Orig	picture in picture lcd tv	0.606
PTB, UTS	picture in picture lcd tv	0.788
Bie-S	picture in picture lcd tv	0.747
Orig	samsung i900 omnia free games	0.691
PTB, UTS, Bie-S	samsung i900 omnia free games	0.810
Orig	richard burns rally pc cheats	0.675
PTB, UTS, Bie-S	richard burns rally pc cheats	0.751

Tagset	PTB	UTS	Bie-S
Lexicon convergence iteration	30	70	30
Segmentation convergence iteration	80	70	90
Peak IR performance iteration	50	50	10
Optimum α	100	10	1000

The lowest value in a row is marked in **boldface**.

Table 3.14: Number of iterations and the optimal α .

between five and fifteen¹¹, improvement on a significant fraction of these queries would be of considerable interest to commercial Web search engines. Relative magnitudes of average gains and losses appear comparable.

Table 3.13 shows representative queries that undergo segmentation change due to the augmented lexicon with a consequent IR benefit. It is evident that all the three tagsets are able to detect relatively rarer n -grams (for example, `picture in picture` and `samsung i900`) which did not feature in the initial lexicon. Our method can also adjust n -gram scores so as to insert new segment boundaries at better locations from an IR perspective (`richard burns rally` \rightarrow `richard burns | rally`).

¹¹<http://cse.iitkgp.ac.in/resgrp/cnegr/qa/querysegmentation.html>, Accessed 31 March 2014.

Convergence. Even after the convergence of the lexicon, it is possible that the segmentations change over further iterations because the scores of the items in the lexicons can continue to change, albeit converging towards specific values. Therefore, we explore an alternative convergence criterion, which is when the segmentation of the queries stabilize for our development set. Nevertheless, we observed that the segmentations so obtained do not necessarily lead to maximum IR performance (say, in terms of nDCG@10). In Table 3.14 we report the number of iterations required for these two types of convergence – lexicon and segmentation, and also the number of iterations after which peak IR performance was achieved. For all our experiments, the parameter α was tuned on the development set using grid-search for maximizing nDCG@10, and the optimal values for each tagset are also reported in Table 3.14. We observe that Bie-S, which is a *deterministic* and hence a *fast* approach, takes only 10 iterations to reach its peak IR performance. This is comparable to the nDCG of other tagsets, whereas the other approaches take 50 rounds. This is definitely a big advantage for the unsupervised POS induction approach. For all the tagsets, the nDCG@10 at segmentation convergence is slightly less than the peak value, though this difference is not statistically significant.

Frequent POS patterns. The ten most frequent patterns in the lexicons for the PTB and the UTS tagsets turned out to be NN NN, NN NN NN, JJ NN NN, JJ NN, NN NNS, NN NN NNS, NN IN NN, FW FW, JJ JJ NN, JN NN NNS, and NOUN NOUN, NOUN NOUN NOUN, ADJ NOUN NOUN, ADJ NOUN, NOUN ADP NOUN, NOUN VERB, NOUN NOUN VERB, VERB NOUN, ADJ ADJ NOUN, NOUN VERB NOUN respectively. The Bie-S tags are system-generated and hence are not readily interpretable.

3.8 Conclusions

In this chapter, we have proposed an unsupervised method of query segmentation that uses Web queries as the only resource. The method unravels syntactic units of queries that are distinct from NL phrases. We have shown how our segmentation algorithm can be enhanced by using lists of named entities like Wikipedia titles. End-user of query segmentation is the retrieval engine; hence, it is essential that any segmentation algorithm should be evaluated in an IR-based framework. In this chapter, we have also overcome several conceptual chal-

lenges to design and implement the first such scheme of evaluation for query segmentation. Using a carefully selected query test set and a group of segmentation strategies, we show that it is possible to have a fair comparison of the relative goodness of each strategy as measured by standard IR metrics. The proposed framework uses resources which are essential for any IR system evaluation, and hence does not require any special input. Our entire dataset – complete with queries, segmentation outputs and relevance judgments – has also been made publicly available to facilitate further research by the community. Moreover, we gain several useful and non-intuitive insights from the evaluation experiments. Most importantly, we show that human notions of query segments may not be the best for maximizing retrieval performance, and treating them as the gold standard limits the scope for improvement for an algorithm. Also, the matching metrics extensively used till date for comparing against gold standard segmentations can often be misleading. We would like to emphasize that in the future, the focus of IR will mostly shift to tail queries. In such a scenario, an IR-based evaluation scheme gains relevance because validation against a fixed set of manual segmentations may often lead to overfitting of the algorithms without yielding any real benefit. Finally, we show how the gap between the techniques used for the conceptually similar processes of chunking and segmentation can be reduced using POS sequence information from query logs, with our results showing significant improvement with all the three tagsets. In the next chapter, we will see how nested, or hierarchical query segmentation, provides a more powerful representation of the query which can be used to overcome several challenges faced by flat segmentation.

Chapter 4

Discovering Syntactic Units by Nested Query Segmentation

4.1 Introduction

As we have seen in the last chapter, flat query segmentation [29, 82, 137] partitions complex queries into syntactic units made of non-overlapping word sequences. An example of such a *flat* or *non-hierarchical segmentation* is shown below:

```
windows xp home edition | hd video | playback
```

where pipes (|) represent flat segment boundaries. In flat segmentation, it is hard to specify the appropriate *granularity* or the expected length of the segments. For example, algorithms that prefer shorter segments may split the first segment into `windows xp` and `home edition`, while others may choose not to break the sequence `video playback`. This issue of an “ideal” granularity creates confusion in a retrieval-based setting. First, whether longer or shorter segments should be preferred purely depends on the query and document pair in question during the search process. Hence, a flat segmentation algorithm consistently adopting either of the two strategies (long or short segments) will fail in several contexts. Next, when the generated segment (say, `windows xp`

home edition) is matched only partially in the document (say, as office xp home edition or windows xp pro edition), a flat segmentation algorithm relying on exact (or approximate) string matching fails to understand that the latter case is much more relevant than the former.

These difficulties of granularity associated with flat segmentation can effectively be addressed if we allow nesting or embedding of segments inside bigger segments. For instance, instead of a flat segmentation, our running example query could be more meaningfully represented as follows:

((windows xp) home) edition) ((hd video) playback)

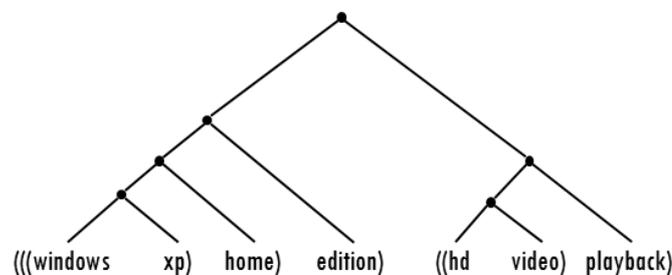


Figure 4.1: Nested segmentation tree.

Here, the atomic segments, i.e., windows xp and hd video, are progressively joined with other words to produce larger segments as follows – windows xp home, windows xp home edition, and hd video playback. We shall refer to this process as *nested* (or *hierarchical*) *query segmentation*. The hierarchy in this form of syntactic analysis is better visualized through a *nested segmentation tree* as shown in Figure 4.1. It is intuitive from this representation that windows xp and hd video are non-negotiable (atomic units) when it comes to matching within documents, and the strength of ties between word pairs can be said to weaken as they move farther in terms of the path through the tree. This observation, in fact, forms the basis of our re-ranking scheme that is aimed at addressing the issue of non-exact segment matching in documents.

In this work, we systematically develop an algorithm and an evaluation methodology for nested query segmentation, which, unlike the framework proposed in the previous chapter, is a strategy that can actually be used to apply nested segmentation to IR. Our nested

segmentation algorithm is based on some very simple yet powerful local statistical and linguistic information. Through a detailed evaluation of the various aspects involved and using two different datasets, we demonstrate that nested segmentation is not only a more informative representation of the query, but also can be exploited to gain better IR performances especially for slightly long queries (greater than or equal to three words). Note that nested segmentation (or *chunking*), which is a very intuitive representation of natural language (NL) sentences [2] and more specifically *phrase structure grammar* [49], has hardly been used for representing queries. A possible reason for the low attention paid to this problem could be that the deduction of hierarchical syntax in NL sentences heavily relies on accurate POS tagging of the words and an underlying grammar. More importantly, such an analysis adds a non-trivial runtime overhead during query processing. Furthermore, there is no prevalent notion of grammatical syntax for Web search queries which could provide a sound basis for a hierarchical syntax.

In absence of linguistic cues, we adopt a purely statistical approach. The intuitions behind our approach are as follows. State-of-the-art flat segmentation algorithms involve a word association score optimization over all the words of the query, and hence flat segments contain vital information that should be utilized effectively. Our objective is to discover more detailed query syntax by finding interesting relationships *within* flat segments, and *between* different flat segments. Syntax *within* flat segments is determined by an exhaustive search over lower order constituent n -grams, and such an approach is feasible in this context because the lengths of flat segments rarely exceed five words. The relative strengths of bigrams straddling flat segment boundaries is exploited in inferring the relationships *between* different flat segments. Relevant bigram statistics, again, are already available. These strategies help us discover the hierarchical syntax within a query, which is subsequently harnessed during document re-ranking. This document re-ranking strategy, in turn, is our instrument for directly applying nested segmentation to improve result quality.

Contributions of this work. This research is the first to harness the power of deep query syntax through nested segmentation and use it to improve ranking. A highlight of our approach is a principled way of dealing with cases where certain words of a query (segment) are absent in the documents, i.e., an exact match of a segment is not found. Specifically, in this research, keeping the above perspectives in mind, we (a) develop an *unsupervised and lightweight* technique for nested segmentation that uses query logs as the

only resource; (b) design a *deterministic* document re-ranking strategy exploiting the nested representation of the query; (c) demonstrate that the use of nested segmentation can lead to significant improvement in document re-ranking over the state-of-the-art flat segmentation strategies.

The rest of this chapter is organized as follows. Section 4.2 discusses current techniques in flat query segmentation, their limitations, and the corresponding benefits of nested segmentation. Basic concepts and necessary terminology are defined in Section 4.3. Section 4.4 presents our algorithm for generating nested segmentations. Next, in Section 4.5, we discuss the technique for using nested segmentation to improve result ranking. We describe datasets used in Section 4.6. Section 4.7 describes the experimental results and observations. Section 4.8 reviews research on proximity and dependence models, indirectly related to this work. Section 4.9 concludes this chapter by summarizing our contributions and highlighting future research directions.

4.2 Issues with flat segmentation

In this section, we explain the limitations associated with flat segmentation and how nested segmentation can conceptually overcome such shortcomings.

4.2.1 Limitations of flat segmentation

There are two important conceptual deficiencies of flat segmentation: its definition and its use in IR. These two issues are, in fact, very closely related because it seems impossible to posit a definition of a segment without an IR model in place. More often than not, the definition of segmentation is presented vaguely as grouping of syntactic units [29, 137]. Ultimately, it is the segmentation strategy that provides an implicit definition of the concept. Needless to say, such definitions and guidelines leave out scope for a subjective interpretation of segments leading to low inter-annotator agreement on manually annotated queries (about 58 to 73% on most metrics [216]). In our previous chapter, we have highlighted issues with evaluation against human annotations. However, the problem is deeper than

just being an outcome of imprecise definition. Rather, it stems from the fact that the notion of segments cannot be defined in the absence of an IR model – because unlike NL, there are no cognitive correlates of segments, like phrases or clauses, in queries. At best, an annotator can be asked to group multiword (named) entities together, which drastically reduces the scope of segmentation and makes it equivalent to the problem of (named) entity identification in queries.

There is no clear consensus on the best use of segmentation in retrieval or ranking models, although there have been proposals such as the use of dependence models [28], language models [137] and double quotes [82]. A commonly assumed restrictive principle in this context is that the words of the same segment must appear adjacent to each other in the document. This has resulted in the use of double quotes (as operators to ensure exact matches) to surround segments in several experimental frameworks (our own work and [29, 82]). However, as we have seen in the previous chapter, putting quotes around all segments degrades performance, and while use of quotes for certain segments yields better results, detection of these segments at runtime is still a hard task. Finally, the use of exact segment matching leaves the following important question unanswered: how does one deal with the situation when the exact segment is only partially found in the document? A “segment found/not found” type of binary scoring would not be the best choice, as we have seen through our running example that some of the words may be entirely replaceable (*edition*) while others are not (*windows*). Current proposals of using flat segmentation for IR [28, 82, 137] do not provide guidelines for handling such cases explicitly. In general, quoting-based strategies severely limit the scope of segmentation and effectively narrow it down to multiword entity detection.

4.2.2 Advantages of nested segmentation

The aforementioned problems, in essence, are manifestations of the deeper issue of granularity at which segmentation needs to be done, i.e., whether to prefer longer or shorter segments, and whether this choice is context-sensitive. These problems vanish if we allow hierarchical or nested segmentation, where the human annotator or the algorithm is allowed to mark meaningful units in a hierarchical fashion and is required to go as deep as possible

preserving the semantics of the query. This will result in multi-level segmentation where at the lowest level, we will have multiword expressions for which quoting or exact matching would make sense during retrieval (e.g., `windows xp` and `hd video` in Figure 4.1), whereas at higher levels it would make more sense to employ less strict matching where the terms are expected to be closer in a relevant document but not necessarily adjacent to each other (for example, a few words may be allowed to intrude between the pair `video` and `playback`).

Nesting is conceptually identical to hierarchical chunking [2,3] or phrase structure parsing of NL sentences [49]. For example, a complex noun phrase (`(a flight) (from Indianapolis) (to Houston)`) can be *chunked* by parenthesizing smaller units. Thus, similarly nesting query segments can effectively resolve the problem of granularity. In the context of queries, a straightforward algorithm for nested segmentation would be to continue splitting a query or segments until certain boundary conditions are met. However, as we will show, this approach overlooks the rich local syntax present in the query which can be used to customize nested segmentation.

Huang et al. [99] introduce a simple algorithm for hierarchical query segmentation as an application for Web scale language models. However, they do not suggest how nested segmentation could be used in IR. It is worth mentioning that term proximity models [56,217] and term dependence models [74,148], which are based on the fundamental assumption that certain query terms are expected to occur in close proximity in the relevant documents, are obliquely related to the concept of segmentation, because terms that are within a segment or appear closer in a segmentation tree can be expected to appear closer in the relevant documents. We shall borrow some of these ideas to build our re-ranking framework.

4.3 Terms and definitions

In this section, we formally define the types of segmentation and the different distances used to build up the algorithms and re-ranking models in the subsequent sections.

4.3.1 Types of segmentation

Flat segmentation. A flat segmentation for a query is defined as a partitioning of the query words into non-overlapping word sequences. Each sequence of terms between two segment boundaries is called a *flat segment*. In general, a flat segment corresponds to a meaningful syntactic unit within the query.

Nested segmentation. A nested segmentation for a query q is defined as a recursive partitioning of q such that each partition is either an indivisible (possibly multiword) unit or another nested segment. The partitions are marked using parentheses, and so a nested segmentation is represented as a complete parenthesization of the words in q . For example, `((windows xp) home) edition) ((hd video) playback)` is a possible nested segmentation for the corresponding query. By convention, parentheses are always present around single words, and at the ends of the query.

Note that this definition does not enforce a strict *binary partitioning* of the query; it is often possible that an atomic unit is composed of more than two words (bed and breakfast). The query can also be constituted of multiple disparate concepts, like `(price comparison) ((ps3) (nintendo) (xbox))`, where more than two elements (ps3, nintendo and xbox) are conceptually at the same level.

A *nested segmentation tree* is an alternative representation of nested segmentation, where the query terms are leaf nodes and every multiword segment is represented by an internal node whose children include *all and only* the nodes corresponding to the words or other segments that constitute this segment. Figure 4.1 graphically illustrates this concept. This tree representation not only provides an intuitive visualization of nested segmentation, but is also useful in defining the topological distance between a pair of terms in the query.

4.3.2 Types of distances

Document distance. The distance between a pair of words in a document can be considered as the difference in the positions of the two words in the document, or equivalently, one more than the number of intervening words. Since a pair of words can occur multiple

times in a given document, the notion of distance, so defined, is ambiguous. Consequently, various proximity heuristics have been proposed in the past to compute the effective distance between two words in a document [56,217]. These include the minimum, maximum and mean of the distances between all paired occurrences of the two words in the document.

Let a and b be two terms in the query q , which are also present (matched) in a retrieved document \mathcal{D} . Past research [56,217] has shown that amongst the various proximity heuristics, *minimum distance* has the highest inverse correlation with document relevance, i.e., the lower the minimum distance between a and b in \mathcal{D} , the higher the chances that \mathcal{D} is relevant to q . However, past measures do not directly reward a document if it has multiple instances of a and b occurring within low distances of each other. Let there be k instances of *ordered* pairwise occurrences of a and b (ordered pairs of positions of a and b , (p_1, p_2) where $p_1 < p_2$) in \mathcal{D} at minimum distances $d_1, d_2, \dots, d_i, \dots, d_k$, such that the d_i -s are in ascending order. We combine the ideas of minimum distance and multiple occurrences of a term pair to formulate the following definition of accumulative inverse document distance (*AIDD*) for a and b in document \mathcal{D} :

$$AIDD(a, b; \mathcal{D})_{a \neq b} = \frac{1}{d_1} + \frac{1}{d_2} + \dots + \frac{1}{d_k} \quad (4.1)$$

By this method, a document with several (a, b) pairs close by will have a high *AIDD*. Since our concept is based on minimum distance, we do not need a document length normalizer. A threshold on k is nevertheless necessary to avoid considering *all* pairwise distances of a and b , as distant pairs could be semantically unrelated. Further, to remove unrelated occurrences from computation, we score matches only if the pair occurs within a given window size, *win*, i.e., we do not consider d_i when it exceeds *win*.

We compute the pairwise distances using position vectors (pv) of a and b in \mathcal{D} [56]. For example, $pv(a) = \{1, 5, 10\}$ and $pv(b) = \{2, 3\}$ mean that a has occurred in positions one, five and ten and b in two and three (in \mathcal{D}), respectively. We currently ignore sentence boundaries while computing *AIDD*. Such a style of computation of pairwise distances can lead to re-counting of specific instances of a and b . For example, the three minimum distance pairs in this case would be $(1, 2)$, $(1, 3)$ and $(5, 3)$. Here, with patterns like ". . . a a b b b c . . ."), one could address the problem by choosing the optimum distance

pair (a, b) using dynamic programming. This entails search in exponential time over the entire document, limited by the number of occurrences of the less frequent word. However, such an approach has been shown to be less effective than the simple case when re-counting is tolerated (see [56] for a more detailed discussion). Moreover, such patterns are quite rare in running text of documents.

Query distance. The query distance $qd(a, b; q)$ between two terms a and b in a query q is defined as the difference between the positions of a and b in q , or equivalently, one more than the number of intervening words in the query. For instance, for our running example query, the distance between `xp` and `video` is four. In special cases when the same word appears multiple times in a query (`johnson and johnson home page`), each term instance is treated as distinct during pairwise comparisons.

Tree distance. The tree distance $td(a, b; n(q))$ between two terms a and b in $n(q)$, the nested segmentation of a query q , is defined as the shortest path (i.e., the number of hops) between a and b (or vice versa) through the nested segmentation tree for q . A tree ensures a unique shortest path between a and b , which is through the common ancestor of a and b . For example, $td(\text{xp}, \text{video}; n(q) \text{ in Figure 4.1}) = 7$. The minimum possible tree distance between two words is two. We hypothesize that term pairs having low tree distance must appear close together in the document. Note that td between a and b can vary for the same q , depending on $n(q)$. As with query distance, when the same word appears multiple times in a query, each word instance is treated as distinct during pairwise comparisons.

4.4 Algorithm for nested query segmentation

The goal behind devising a principled nested segmentation strategy is to discover deep syntactic relationships in a query, which are often present *within* a flat segment, and/or *between* multiple flat segments. We do not propound simple top-down (begin with the query as a single unit and continue splitting till all units are single words) or bottom-up (begin with each word as a single unit and continue merging till the whole query becomes one unit) approaches for deducing the hierarchical syntax in a query because such methods are naïve and do not involve any optimization step over all the words of the query. State-of-the-art flat

Table 4.1: Joining and splitting of flat segments for nested segmentation.

Step	Syntactic representation of the query
Input flat seg	windows xp home edition hd video playback
Parenthesized	(windows xp home edition) (hd video) (playback)
Split	((windows xp home) (edition)) (hd video) (playback)
Split	(((windows xp) (home)) (edition)) (hd video) (playback)
Join	(((windows xp) (home)) (edition)) ((hd video) (playback))
Join and output	((((windows xp) (home)) (edition)) ((hd video) (playback)))

All text except new segment markers are greyed out.

segmentation algorithms like Hagen et al. [84], Li et al. [137] and our methods (Chapter 3) involve principled optimization criteria leading to the discovery of flat segments, and a good nesting strategy should exploit this knowledge to the best capacity.

There are three primary constraints or features of a query segmentation algorithm that need to be considered before designing an algorithm for this purpose. First, the accuracy and robustness (i.e., reasonable performance on a wide variety of queries); second, the speed (segmentation is an online process and therefore to be practically useful, it must have a very short turnaround time); and third, lack of annotated data. It might be worthwhile to elaborate a little on this last point. It may be argued that if we can get sufficient queries annotated by human experts for nested segmentation, the data could be used for supervised learning of nesting algorithms. Indeed, most of the NL parsing algorithms do rely on supervised learning on human-annotated treebanks [144]. However, there is an important difference between these two cases. NL parsing is guided by an underlying (context-free or phrase structure) grammar which linguists have designed through years of systematic analysis of NLS. The annotators, who are themselves trained linguists, use the knowledge and framework of the grammar to annotate the tree syntax for sentences. Likewise, the parsing algorithms search in the space of all possible parse trees that conform to this grammar. Queries do not follow grammatical rules, or at the least no such grammar has been formulated or deciphered till date. Neither do we have annotators who are experts or native speakers of the “query language”. Therefore, structural annotation of queries [25, 26, 199] has always been subjective, often leading to low inter-annotator agreement. Moreover, creation of annotated data, for example, the treebanks for NLS, takes a tremendous amount of

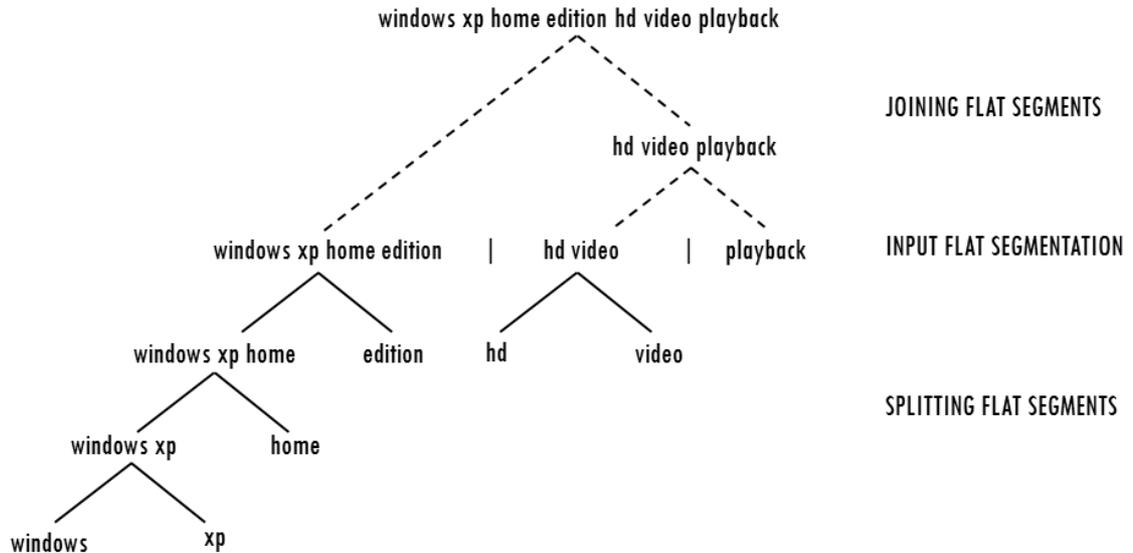


Figure 4.2: Illustrating our approach for nested query segmentation.

time and effort. It is also not straightforward to ascertain whether NL parsing algorithms can be efficiently adapted for fast online processing.

Approach. Since flat segmentation is a well-researched problem, we develop our algorithm for nested segmentation by starting with a flat segmentation of the query and trying to *split* within a flat segment and *join* adjacent flat segments recursively. Since flat segments are rarely longer than four to five words, nesting can be done rather fast with some clever manipulations of low order n -gram statistics ($n = 2, 3$). Thus, in our setup, given a flat segmentation for a query as input, a nesting strategy consists of the following two steps: (a) Split individual flat segments recursively till atomic units are obtained; (b) Join adjacent flat segments recursively till the whole query is one single unit. The split and the join steps are independent of each other and can be performed in any order. This process is illustrated in Table 4.1 and Figure 4.2 with the help of our running example query.

4.4.1 Splitting flat segments to discover syntax within a flat segment

Our main motivation for designing simple segment nesting strategies stems from the fact that most flat segmentation algorithms compute some form of scores for n -grams as a key

step of their respective methods (generally $n \leq 5$) [84, 216]. In doing so, most often the scores of the contiguous lower order n -grams ($n-1, n-2, \dots$) are also known. We exploit these scores to deduce the syntax within a flat segment. Any word association measure can be used to score an n -gram in our method (in our experiments, we use the method discussed in the previous chapter (Equation 3.3)).

We adopt a simple greedy approach in this research. The n -gram that has the highest association score within a flat segment (where the number of words in the n -gram is less than the number of words in the corresponding flat segment) is immediately grouped together as a unit, i.e. a *sub-segment*. In this work, we restrict n to a maximum of three, i.e. we search for highest scoring bigrams and trigrams exhaustively within a flat segment. We define a sub-segment as a smaller segment created by the division of a larger segment. Recursively, this newly grouped sub-segment's left and right n -grams (possibly null) and the sub-segment itself are processed in the same greedy fashion till every string to be processed cannot be divided further. For example, in the flat segment `windows xp home edition`, `windows xp home` has the highest score among the five possible n -grams (two trigrams and three bigrams). Thus, it is grouped together first. Since `edition` cannot be processed further, we repeat the search within `windows xp home` and group `windows xp` inside it, which leads to the following parenthesized form for the original flat segment: `((windows xp) home) edition`. For the flat segment `the legend of zelda twilight princess`, we have `legend of zelda` grouped first (with `legend of` being grouped inside it in a subsequent step) followed by `twilight princess`. This sequence thus results in the embedded syntax: `(the) ((legend of) zelda) (twilight princess)`.

4.4.2 Joining flat segments to discover syntax across flat segments

Joining flat segments is essential to completing the nested segmentation tree, which in turn ensures a path between every pair of words in the query. At first sight, it seems that to be able to make a decision regarding the joining of two flat segments with m and n words respectively, one needs to have $(m+n)$ -order statistics. However, we found an elegant way to join segments using two simple local statistics explained next.

Bigram statistics of words at segment boundary. The bigram at a flat segment boundary, i.e. the last word of a flat segment and the first word of the next flat segment, can be effectively used to take the segment joining decision. In our running example, if we wish to decide whether to join windows xp home edition and hd video, or hd video and playback, we check the relative order of the scores of the (ordered) bigrams formed by the underlined words only. The bigram with the higher score (in this case video playback) dictates which pair should be joined. This process is similarly repeated on the new parenthesised segments obtained until the whole query forms one unit. This local and context insensitive approach may seem to fail in cases, and we do not claim that using bigrams only is sufficient in this process. Nevertheless, as we shall see, it works quite well in practice. In this research, we use the well-established concept of pointwise mutual information (PMI) [84, 114, 185] to score bigrams. Let $\mathcal{B} = \langle w_1 w_2 \rangle$ be a bigram constituted of words w_1 and w_2 . $\text{PMI}(\mathcal{B})$ is defined as follows:

$$\text{PMI}(\mathcal{B}) = \log_2 \frac{p(w_1 w_2)}{p(w_1)p(w_2)} \quad (4.2)$$

where $p(w_1 w_2)$, $p(w_1)$ and $p(w_2)$ refer to the probabilities of occurrences of \mathcal{B} , w_1 and w_2 in the *query log*, i.e. the number of queries each of them is present in, normalized by the total number of queries in the log.

Determiners, conjunctions and prepositions. It often happens that the last (or the first) word in a segment is a determiner, conjunction or preposition (DCP). In these cases, it is almost always meaningful to combine such a segment with the next segment (or the previous segment) to make a meaningful *super-segment* (a larger segment created by the joining of two smaller segments). Examples are (bed and) (breakfast) and (sound) (of music). In our algorithm, we prioritize such cases over the bigram scores during the joining process. The list of DCP used is freely available online¹.

¹<http://www.sequencepublishing.com/cgi-bin/download.cgi?efw>, Accessed 6 April 2014.

4.5 Using nested segmentation in IR

The use of flat query segmentation in IR has been based upon the concept of proximity, which states that two words which are in the same segment should also occur within a short distance of each other in the relevant documents; whereas words in different flat segments need not necessarily occur close to each other [185]. A stricter but more popularly assumed and experimented version of this hypothesis is that words within a flat segment should occur next to each other exactly in the same order in the relevant document as in the query [216]. This is typically implemented through the use of double quotes around segments, which most search engines interpret as an instruction for exact phrase match. As discussed earlier, this severely limits the scope of query segmentation and often results in misleading conclusions. Nevertheless, there is no obvious analogy between quoting of flat segments and that of nested segments, because it is unclear as to which level of nesting the quotes should be applied. More importantly, quoting is against the basic philosophy of nested segmentation because then we are not harnessing the true benefits of the hierarchical representation of the query terms.

4.5.1 Re-ranking using the tree and document distances

Here we define a score *Re-rank Status Value*² (*RrSV*) of every document \mathcal{D} that was retrieved and ranked in response to an unsegmented query q . The *RrSV* for each such document is determined based on the following principle – *a pair of words that have a low tree distance in the nested representation of the query should not have a high document distance*. In other words, while re-ranking a document, the document distance (Section 4.3.2) between a pair of words should be penalized by a factor *inversely* proportional to their tree distance. We recall that tree distance between two words a and b in a query q , $td(a, b; n(q))$ is the path between a and b in the nested segmentation ($n(q)$) tree of q , and the document distance between a and b in a document \mathcal{D} , $AIDD(a, b; \mathcal{D})$, is defined by Equation 4.1. The *RrSV* for a document \mathcal{D} is thus defined as

²The nomenclature is inspired by the Retrieval Status Value (*RSV*) of a document with respect to a query, which is a term that is popular in IR literature [140].

$$RrSV_{\mathcal{D}} = \sum_{\substack{t_i, t_j \in q \cap \mathcal{D} \\ t_i \neq t_j \\ td(t_i, t_j; n(q)) < \delta}} \frac{AIDD(t_i, t_j; \mathcal{D})}{td(t_i, t_j; n(q))} \quad (4.3)$$

where t_i -s are query terms matched in the document and $n(q)$ is the nested segmentation for q . However, we do not wish to penalize the case when the words are close by in the document and are relatively far apart in the tree. This is because it is always preferable to have all query words close by in the document [56]. Rather, we want to penalize a document only when specific word pairs (those that have a low tree distance) have high document distance. In our example, we would penalize the case `windows` and `xp`, which are close by in the tree, have a high document distance. We will not care when `windows` and `playback`, which have a high tree distance are nearby in the document or not. These situations and the corresponding desired penalties are presented in Table 4.2. This analysis drives us to create a tree distance threshold (cut-off) parameter δ . In other words, if $td(a, b; n(q)) < \delta$, only then is the word pair a and b considered in the computation of $RrSV$.

Table 4.2: Penalty cases for query word pairs.

Tree distance	Document distance	Penalty
Low	Low	Low
Low	High	High
High	Low	X
High	High	X

X marks represent *don't care* conditions.

We experimented with a number of variations of incorporating penalty into our re-ranking formulation, and found that the simple method of thresholding the tree distance works best. This is because the tree distance acts as a normalizer for the document distance, and thus proportionately “rewards” (or inversely, penalizes, in the vice versa case) the document distance when the tree distance is low (if tree distance is two, only halving the AIDD; if the tree distance is three, dividing the AIDD by three). When the tree distance exceeds the threshold, we arrive at the *don't care* cases.

The set of documents retrieved by a search engine by issuing the unsegmented query will be re-ranked in *descending* order of this $RrSV$. Let the ranks assigned to the document \mathcal{D} by the original ranker and our re-ranking strategy be $R_{orig}(\mathcal{D})$ and $R_{new}(\mathcal{D})$ respectively. Then, according to our strategy, for two documents \mathcal{D}_1 and \mathcal{D}_2 , if $RrSV_{\mathcal{D}_1} > RrSV_{\mathcal{D}_2}$, then $R_{new}(\mathcal{D}_1) < R_{new}(\mathcal{D}_2)$, i.e. \mathcal{D}_1 will be ranked higher up in the new ranked list than \mathcal{D}_2 . The intuition here is that if a document \mathcal{D}_1 accumulates a higher value of $RrSV$ than document \mathcal{D}_2 , then \mathcal{D}_1 has a relatively higher number of occurrences of query terms having a low tree distance close together inside its text than \mathcal{D}_2 .

4.5.2 Rank aggregation of original and new ranks

The set of documents that we re-rank are originally retrieved from a collection in response to an unsegmented query using well-established IR ranking principles based on term frequencies and inverse document frequencies, and we wish to give due weight to the old ranks. We aggregate or fuse these ranks in the following manner to obtain an aggregated score $S_{rank-agg}$ for every document \mathcal{D} [5]:

$$S_{rank-agg}(\mathcal{D}, R_{orig}, R_{new}, w) = \left(w \times \frac{1}{R_{new}(\mathcal{D}) + 1} \right) + \frac{1}{R_{orig}(\mathcal{D}) + 1} \quad (4.4)$$

where the weight w (assigned to the new rank) is a heuristically tuned scaling factor representing the relative “importance” of the new ranking. The documents are finally ranked in *descending* order of $S_{rank-agg}$ to produce the final aggregated rank R_{final} . Formally, if $S_{rank-agg}(\mathcal{D}_1) > S_{rank-agg}(\mathcal{D}_2)$, then $R_{final}(\mathcal{D}_1) < R_{final}(\mathcal{D}_2)$, i.e. \mathcal{D}_1 will be ranked higher up in the final aggregated ranked list than \mathcal{D}_2 . Setting w to zero or a very large value nullifies the effects of the new and original ranking respectively.

There are several other approaches to rank aggregation [67, 70, 151] and one of several proposed approaches could produce the best results in a given setup. However, that is not the focus of this research and we adopt one of the relatively recent, simple and popular techniques in this work that allows us to tune the effects of the original and rankings.

The authors of Agichtein et al. [5] adapt a simple and robust method of merging the rank

orders. The main reason for ignoring the original *scores* and considering only the *ranks* is that since the feature spaces and learning algorithms are different, the scores are not directly comparable. They experimented with a variety of merging functions on a development set of queries. They found that a simple rank merging heuristic combination works well, and is robust to variations in score values from original rankers. The query results are ordered by decreasing values of the final score to produce the final ranking. One special case of this model arises when setting w to a very large value, effectively forcing the new ranking to be preferred over the old ranking - an intuitive and effective heuristic that they used as an experimental baseline. Applying more sophisticated ranker combination algorithms may result in additional improvements, and is left as future research. The approach above assumes that there are no interactions between the underlying features producing the original ranking and the new features.

4.5.3 Re-ranking baselines

We now introduce three baselines for comparing the performance of our re-ranking strategy for nested segmentation. Flat segmentation is the first of these baselines, where we extend our notion of using pairwise term proximity to words within flat segments only. The other two baselines are natural variants of the re-ranking equation (Equation 4.3) that require investigation – one where only document distances are considered, and the other where the tree distance is replaced by the simple query distance (Section 4.3.2).

Flat segmentation. This re-ranking technique is based on the notion that words within a flat segment are expected to appear near each other in the relevant documents [216]. Let q be a query that has p flat segments, S_1 to S_p . The *RrSV* computation in this case is restricted only to intra-segment term pairs, i.e.,

$$RrSV_{\mathcal{D}} = \sum_{k=1}^p \sum_{t_i, t_j \in S_k \cap \mathcal{D}, t_i \neq t_j} AIDD(t_i, t_j; \mathcal{D}) \quad (4.5)$$

Document distances only. This strategy is based on the principle that proximities between all pairs of query terms are equally important. The re-ranking score is thus simplified

as shown below:

$$RrSV_{\mathcal{D}} = \sum_{t_i, t_j \in q \cap \mathcal{D}, t_i \neq t_j} AIDD(t_i, t_j; \mathcal{D}) \quad (4.6)$$

Document and query distances. This method assumes that only terms close by in the query are required to be near each other in the document, and thus takes into account the query distance qd . Hence, Equation 4.3 is suitably modified to:

$$RrSV_{\mathcal{D}} = \sum_{t_i, t_j \in q \cap \mathcal{D}, t_i \neq t_j} \frac{AIDD(t_i, t_j; \mathcal{D})}{qd(t_i, t_j; q)} \quad (4.7)$$

4.6 Datasets

In this section, we describe the datasets that we have used. We divide this section into two parts: (a) data needed for performing nested segmentation of queries, and (b) data needed to apply and evaluate our strategies with respect to IR.

4.6.1 For performing nested segmentation

As discussed, our nested segmentation algorithm requires a query log as the only resource, for computing various n -gram scores. For our experiments, we use our two to ten-word queries from Bing Australia query log as discussed in Section 1.2. We use the Porter Stemmer [178] to stem the queries before the computation of the n -gram scores.

4.6.2 For re-ranking documents

In order to ensure the replicability of our results, we report our IR evaluation on publicly available datasets only (Table 4.3) and use open source retrieval systems.

Table 4.3: Details of datasets used.

Dataset Name	Number of queries	Average words per query	Average RJs per query	Search system
SGCL12	500	5.29	28.34	Lucene
TREC-WT	75	3.43	34.39	Indri

SGCL12. We use the dataset that we created for evaluating various flat segmentation algorithms (Section 3.4) because it consists of slightly longer queries (five to eight words) where segmentation is meaningful from an IR perspective. Since we also showed that flat segmentation can potentially lead to substantial nDCG improvement on SGCL12, this dataset is very appropriate for evaluating nested segmentation, and to show improvements over flat segmentation. Note that the queries in the SGCL12 dataset also have flat segmentation annotations from various algorithms and human experts³. As in the previous chapter, we use the commercially popular open source Apache Lucene⁴ (same version 3.4.0 chosen for comparability of results) to search this collection. Queries 1 – 250 were used as the development set for tuning model parameters (k , win , δ and w) and queries 251 – 500 were used as the test set.

TREC-WT. TREC topics, especially those belonging to the Web Track (WT) (last held in 2012) and the Million Query Track (MQT) (last held in 2009) are the ideal proxy for real Web search queries. All the data related to TREC-WT is public⁵. However, the topics of WT are very short (average length of 2.32 words for 2012⁶) and therefore, not very appropriate for evaluation of nested segmentation. The issue with the MQT (2009)⁷ is the sparseness of RJs, which is more acute for slightly longer queries. We pulled out the 500

³In this chapter, in order to prevent digression, we do not discuss human annotations for nested segmentation. However, we explored the effectiveness of crowdsourcing for this task. Through carefully designed control experiments and Inter Annotator Agreement metrics for analysis of experimental data, we showed that crowdsourcing may not be a suitable approach for nested query segmentation because the crowd seems to have a very strong preference towards balanced binary trees.

⁴<http://lucene.apache.org/core/>, Accessed 6 April 2014.

⁵<http://trec.nist.gov/data/webmain.html>, Accessed 16 November 2014.

⁶<http://trec.nist.gov/data/web2012.html>, Accessed 6 April 2014.

⁷<http://trec.nist.gov/data/million.query09.html>, Accessed 6 April 2014.

longest queries from the MQT (2009) having at most ten words. 491 of these queries had no associated RJ. Moreover, of all the queries that have length greater than or equal to five words, only 42 have at least one RJ.

Nevertheless, in order to conduct nested segmentation experiments on the widely used TREC data, we accumulated queries from the 2009 to 2012 WT, and retained the queries that had three or more words⁸ (100 queries out of a total of 200). The highest number of words in this query set is five, even though it would have been better to have longer queries for truly appreciating the benefits of nested segmentation. Relevance judged documents for these queries are present in the ClueWeb09 collection⁹; we used the open source Indri¹⁰ to search this collection through the provided API and retrieved the top 100 documents. The queries for which there are no relevant documents in the top 100 results were removed from the dataset. We will refer to this remaining set of 75 queries as the TREC-WT. These queries, on an average, had 34 RJs within the top 100 results (Table 4.3). RJs for all TREC-WT queries, downloadable from the respective track websites (*qrels*), have been appropriately collapsed to a 3-point scale (0, 1, 2). Queries 1 – 35 queries were used as the development set for tuning model parameters and queries 36 – 75 queries were used as the test set (see <http://bit.ly/13StKUN> for the ordered query set).

4.7 Experiments and results

In this section, we first report the specifics of our experimental setup and present the detailed results about our re-ranking strategy. In particular, we report the results of the following experiments: (a) effectiveness of nested segmentation over flat segmentation, (b) effect of query lengths, (c) effect of re-ranking strategies, (d) effect of parameter tuning, (e) effect of algorithmic variants, and (f) comparison with past work.

⁸Nested segmentation can only benefit queries with at least three words.

⁹<http://lemurproject.org/clueweb09/>, Accessed 6 April 2014.

¹⁰<http://www.lemurproject.org/indri/>, Accessed 6 April 2014.

Table 4.4: Examples of nested segmentations for queries.

Flat segmentation	Nested segmentation
garden city shopping centre brisbane qld	((garden city) (shopping centre)) (brisbane qld)
the chronicles of riddick dark athena	(the ((chronicles of) riddick)) (dark athena)
sega superstars tennis nintendo ds game	((sega superstars) tennis) ((nintendo ds) game)
samurai warriors 2 empires walk throughs	((samurai warriors) 2) empires) (walk throughs)
as time goes by sheet music	(as (time goes) by) (sheet music)

4.7.1 Experimental setup

We used the outputs of four flat segmentation algorithms – our proposed algorithms, Hagen et al. [84], and Li et al. [137], as input to the nested segmentation algorithm. Final nested segmentations for these queries were obtained as output. Documents are retrieved using the unsegmented queries, and subsequently re-ranked using the proposed technique (Section 4.5) and the baselines (Section 4.5.3). Results are compared in terms of popular IR evaluation metrics: nDCG and MAP (Equations 3.4 and 3.7). nDCG was computed for the top- k retrieved documents (represented with @ k suffix, where k is 5, 10 and 20). For computing MAP, URLs with ratings > 0 were considered as relevant. MAP values are computed on the top-30 documents for SGCL12 and the top-40 documents for TREC-WT (depending upon the approximate pool depth of 28 and 34 respectively (Table 4.3)). For each setting, the four parameters (Table 4.8) were optimized using the grid search technique for maximizing nDCG@10 on the development set and the best set of values were applied on the test set, which are reported in this section. Our proposed re-ranking method is found to be robust to parameter variation, as shown later in the text.

4.7.2 Results and observations

To provide a qualitative feel of nested segmentation outputs on typical queries, we provide some representative nested segmentations generated by our algorithm for SGCL12 queries in Table 4.4. Table 4.5 presents our *main findings* – the performance of nested segmentation in comparison with unsegmented queries and flat segmentation. Since the TREC-WT dataset was quite small compared to SGCL12, we report average values over ten runs with random train-test splits of 35 and 40 queries respectively, while preserving the query word

Table 4.5: Performance comparison of flat and nested segmentations.

Dataset	Algo	Hagen et al. [84]		Li et al. [137]		Proposed Flat		Proposed Flat+Wiki	
		Flat	Nested	Flat	Nested	Flat	Nested	Flat	Nested
SGCL12	Unseg								
nDCG@5	0.6839	0.6815	0.6982	0.6913	0.6989	0.6977	0.6976	0.6746	0.7000 [†]
nDCG@10	0.6997	0.7081	0.7262 [†]	0.7144	0.7258 [†]	0.7189	0.7274	0.7044	0.7268 [†]
nDCG@20	0.7226	0.7327	0.7433 [†]	0.7366	0.7437 [†]	0.7389	0.7435	0.7321	0.7433 [†]
MAP	0.8337	0.8406	0.8468 [†]	0.8404	0.8469 [†]	0.8411	0.8481 [†]	0.8423	0.8477
TREC-WT	Unseg								
nDCG@5	0.1426	0.1607	0.1750 [†]	N. A.*	N. A.	0.1604	0.1752 [†]	0.1603	0.1767 [†]
nDCG@10	0.1376	0.1710	0.1880 [†]	N. A.	N. A.	0.1726	0.1882 [†]	0.1707	0.1884 [†]
nDCG@20	0.1534	0.1853	0.1994 [†]	N. A.	N. A.	0.1865	0.2000 [†]	0.1889	0.2010 [†]
MAP	0.2832	0.2877	0.3298 [†]	N. A.	N. A.	0.3003	0.3284 [†]	0.3007	0.3296 [†]

The higher value among flat and nested segmentations is marked in **bold**. Statistical significance of nested segmentation (under the one-tailed paired t -test, $p < 0.05$) over flat segmentation *and* the unsegmented query is marked using [†].

* We are unable to report the performance of Li et al. [137] on TREC-WT due to unavailability of outputs and code, and associated difficulties in reimplementing due to use of proprietary data.

length distribution. For each algorithm, *Flat* refers to the baseline re-ranking strategy (Section 4.5.3) when applied to the query (flat) segmented by the corresponding algorithm, and *Nested* refers to the proposed re-ranking strategy (Section 4.5.1) when applied to the nested segmentation of the query (Section 4.4) generated when the corresponding flat segmentation was used as the start state. We observe that nested segmentation, when using the proposed re-ranking scheme, significantly outperforms the state-of-the-art flat segmentation algorithms in all the cases. Importantly, improvements are observed for both the datasets on all the metrics. This indicates that one should not consider proximity measures for *only* the pairs of terms that are within a flat segment. We also note that both the flat and nested segmentations perform better than the unsegmented query, highlighting the general importance of query segmentation. Henceforth, because of its superior performance over the other flat segmentation methods, we will assume the input flat segmentation for the nested segmentation algorithm as the output by “Proposed Flat+Wiki”, unless otherwise mentioned in the text.

The results in Table 4.5 and 3.3 are computed under different evaluation frameworks. While document distance-based re-ranking is the underlying philosophy in Table 4.5, the oracle score for a quoting-based retrieval has been reported in Table 3.3. This is why the metric values are different in the two tables for the same algorithms. However, for the unsegmented query, the results appear different only because of the computation precision (three decimal places in Table 3.3, resulting in 0.701 nDCG@10 while four decimal places in Table 4.5 giving 0.6997), but are essentially the same.

Effect of query length. To understand the potential of nested segmentation, it is important to see for how many queries in each length group it results in improved retrieval performance. In Table 4.6, we report the number of queries of a particular length in our datasets ($\#Q$), the number among these Q that show a positive gain in nDCG@10 ($\#Gain Q$), the associated percentage of queries and the average gain (A. G.) on nDCG@10 computed over all queries of a particular length that show performance improvement over the original unsegmented query. We observe that for almost all length groups, nested segmentation improves a strong majority of the queries. The mean improvement is slightly more for queries in the medium length zone (5- and 6-word queries). We found that longer queries in our dataset (for example, `you spin my head right round right round and eternal sunshine of the spotless mind watch online`) generally contain song lyrics or long named entities that require exact document matches and hence nesting is often not required, and may be detrimental in certain cases. Corresponding figures for flat segmentation (lower half of table) are observed to be lower.

In the current Web search scenario, slightly longer queries are generally harder to solve, with keyword matches retrieving several spurious results. To be specific, the percentage of long queries (≥ 5 words) in our Bing Australia query log is 26.65% (distinct queries only) – a significant number when the total search volume is considered. Thus, we can no longer undermine the impact nested segmentation can have on Web search. In total, while $\simeq 49\%$ queries are benefited by flat segmentation for SGCL12 and $\simeq 45\%$ for TREC-WT, the numbers rise to $\simeq 61\%$ for SGCL12 and $\simeq 48\%$ for TREC-WT in case of nested segmentation. Importantly, the mean improvements (over the unsegmented queries) in nDCG@10 for benefited queries are 0.1084 for SGCL12 and 0.2185 for TREC-WT in case of nested segmentation; corresponding values for flat segmentation are lower: 0.0876 (SGCL12) and 0.2053 (TREC-WT).

Table 4.6: Break-up of nDCG gains (over unsegmented query) by length.

SGCL12 (Nested segmentation)					TREC-WT (Nested segmentation)				
Length	#Q	#Gain Q	#Gain Q%	A. G.	Length	#Q	#Gain Q	#Gain Q%	A. G.
5	387	235	60.72	+0.1103	3	52	22	42.31	+0.1695
6	91	58	63.74	+0.1006	4	14	9	64.29	+0.2842
7	14	9	64.29	+0.1401	5	9	5	55.56	+0.3156
8	8	4	50.00	+0.0414	-	-	-	-	-
SGCL12 (Flat segmentation)					TREC-WT (Flat segmentation)				
Length	#Q	#Gain Q	#Gain Q%	A. G.	Length	#Q	#Gain Q	#Gain Q%	A. G.
5	387	193	49.87	+0.0887	3	52	21	40.38	+0.1868
6	91	42	46.15	+0.0772	4	14	9	64.29	+0.2071
7	14	6	42.86	+0.1166	5	9	4	44.44	+0.2987
8	8	3	37.50	+0.1061	-	-	-	-	-

Comparison of re-ranking strategies. Table 4.7 compares re-ranking strategies. Here *Doc* refers to the baseline re-ranking method that uses only document distances (Section 4.5.3), *Query* refers to the scheme using document and query distances, *Tree* refers to the proposed re-ranking strategy using the nested segmentation tree (Section 4.5.1). We observe that scaling *AIDD* by the tree distance generally improves the results over the unscaled version. This shows the importance of the tree distance in bringing out the relationship between query terms. In other words, the nested segmentation tree provides a more principled and meaningful estimation of proximity between query terms, which can be systematically exploited during re-ranking of documents for significant performance gains. We observed that the number of queries on which *Doc*, *Query* and *Tree* perform the best are 102, 94, 107 (SGCL12, 250 test queries) and 30, 29.7, 30.8 (TREC-WT, 40 test queries, averaged over ten splits) respectively. The numbers do not add up to 250 (SGCL12) or 40 (TREC-WT) because multiple models may produce the best output for the same query. Thus, the *Tree* model helps greater numbers of queries for both datasets.

Tunable Parameters. We now systematically study the effect of variation of the four tunable parameters on the re-ranking performance. Table 4.8 lists the tunable parameters. Variation patterns on the development set of SGCL12 and TREC-WT are reported in Fig-

Table 4.7: Performance of re-ranking strategies.

Dataset	SGCL12			TREC-WT		
Metric	Doc	Query	Tree	Doc	Query	Tree
nDCG@10	0.7193	0.7255	0.7268[†]	0.1801	0.1798	0.1884[†]
MAP	0.8398	0.8472	0.8477[†]	0.3237	0.3189	0.3296[†]

The highest value among the *Doc*, *Query* and *Tree* re-ranking strategies is marked in **boldface**. Statistical significance of the *Tree* strategy under the one-tailed paired *t*-test ($p < 0.05$) over the lowest value among the three is marked using [†].

Table 4.8: List of parameters used in re-ranking.

Notation	Parameter
k	Number of minimum distances considered
win	Window size
δ	Tree distance cut-off
w	New rank weight

ures 4.3 and 4.4. For examining a particular parameter for a specific re-ranking strategy, others are fixed at the point of global maximum. *Doc* and *Query* refer to the baseline re-ranking strategies using only document, and document and query distances respectively (Section 4.5.3). *Tree* refers to the proposed re-ranking method based on the nested segmentation tree (Section 4.5.1). Wherever applicable, the *Tree* re-ranking model outperforms the *Doc* and *Query* models systematically. From plots (Figures 4.3 and 4.4) (a) and (b), we see that preferred values of k and win are five and four respectively for SGCL12 and one and three for TREC-WT, and increasing them further brings semantically unrelated word pair occurrences into the *RrSV* computations. Figures 4.3 (c) and 4.4 (c) show the effect of varying δ – the tree distance cut-off value; very low δ essentially means ignoring the tree hierarchy, and thus leads to poor performance for SGCL12. For SGCL12, the result stabilizes for $\delta \geq 5$, and increasing δ further almost has no effect on the results as there are very few word pairs that will have a tree distance greater than five or six for a typical

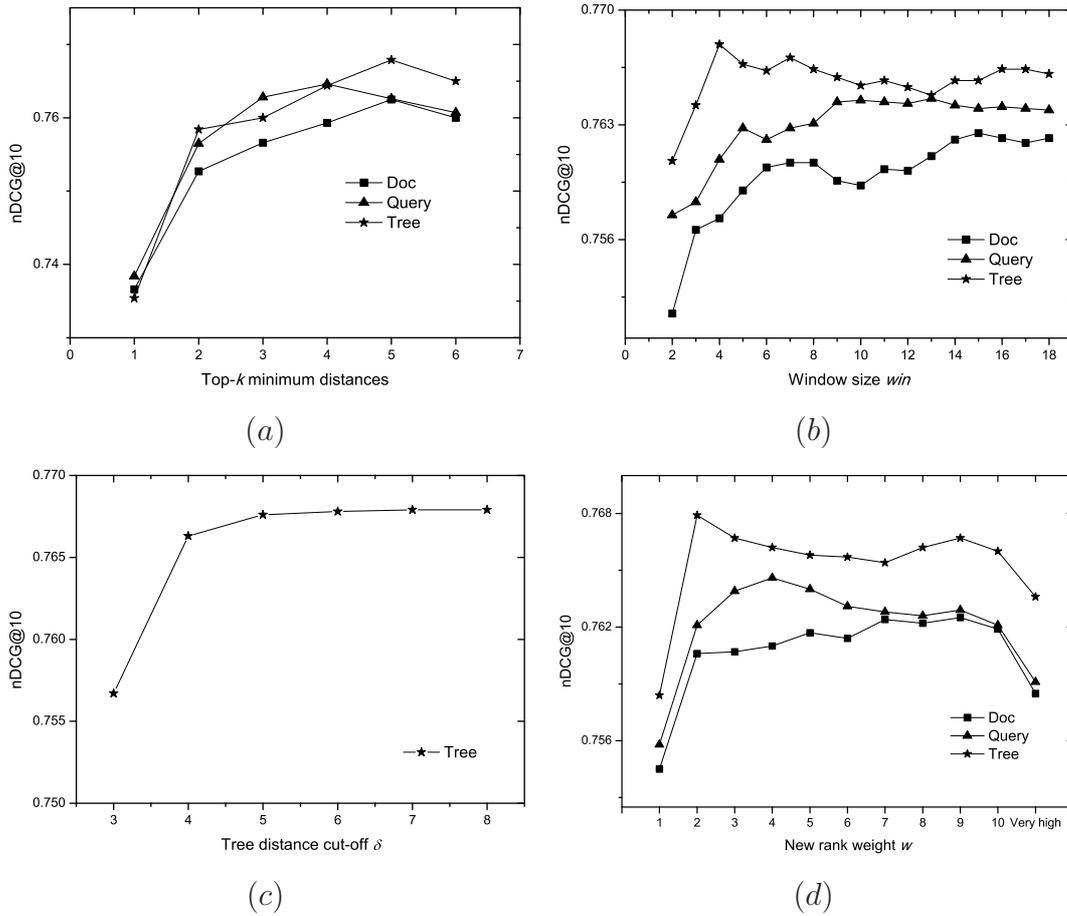


Figure 4.3: Variation in parameters on SGCL12.

query. Thus, having this parameter in the system setup is optional; but if one chooses to use δ for finer control on the results, one must be careful as to not set it to a very low value. However, we note that $\delta = 3$ is ideal for TREC-WT, and greater δ is applicable only for $\simeq 30\%$ of the queries, which are of length greater than three words. Finally, setting the new rank weight w to two is found to be the best for SGCL12. Setting w to zero logically translates to ignoring the new ranking, and would result in the performance of the original query, which is always poorer than when re-ranking is applied (*Unseg* in Table 4.5). Using a large value for w ($\simeq 1,000$) implies ignoring the old ranking. This is found to produce the best results for TREC-WT, emphasizing the importance of our re-ranker. Thus, one should decide on the weights to assign to the original ranker and that derived by the nested representation of the query after an empirical analysis on a tuning set.

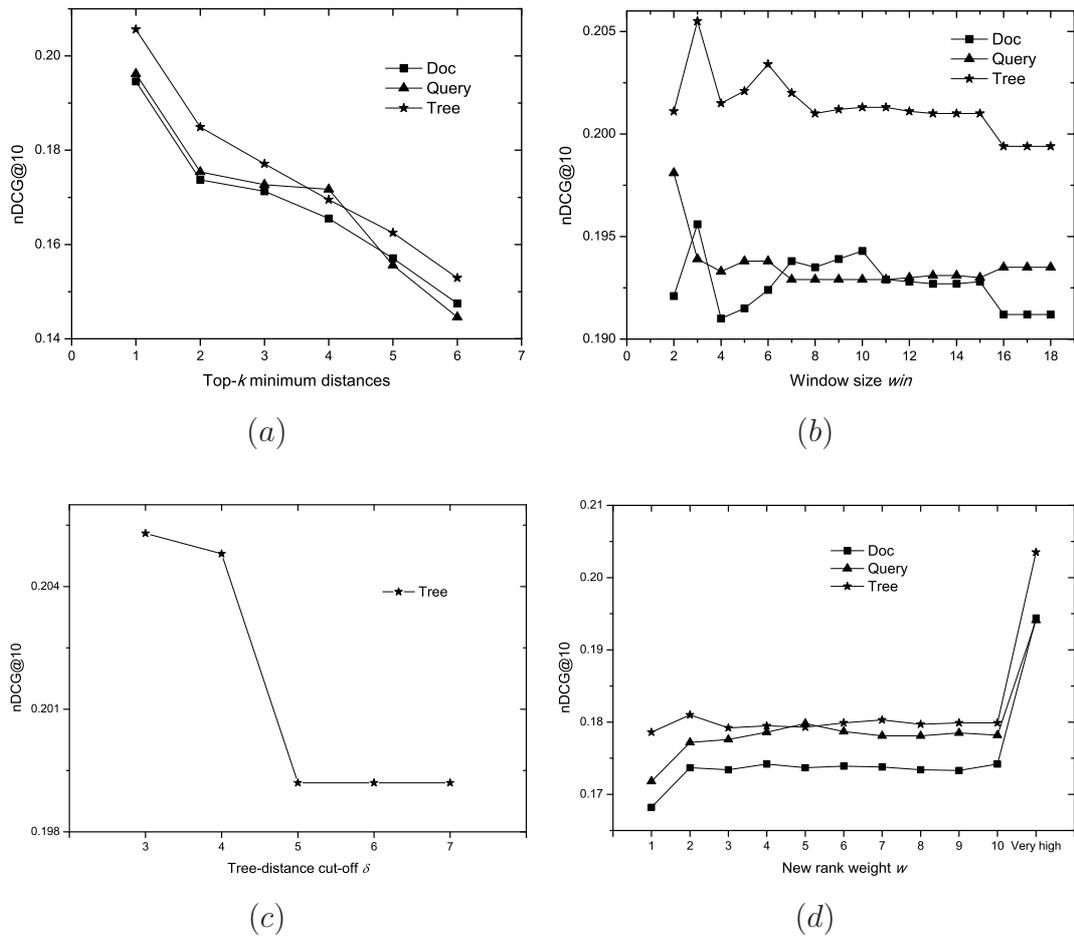


Figure 4.4: Variation in parameters on TREC-WT.

4.7.3 Systematic investigation of variations in algorithm

Our overall algorithm entails the systematic exploration of certain variations. First, instead of a greedy approach, one can opt for an optimized strategy to split a flat segment. In this approach, every possible way of breaking a flat segment is considered, such that the constituent sub-segments are 1-, 2- or 3-grams only, and the partitioning that leads to the best combined score is selected¹¹. These partitions are assumed to be the atomic units of the base flat segment. If a flat segmentation is purely based on an optimal combination of individual segment scores, then each segment, by itself, is an optimal way of combining

¹¹Addition is the combination operator for the scores owing to the logarithmic space in which they are defined [84].

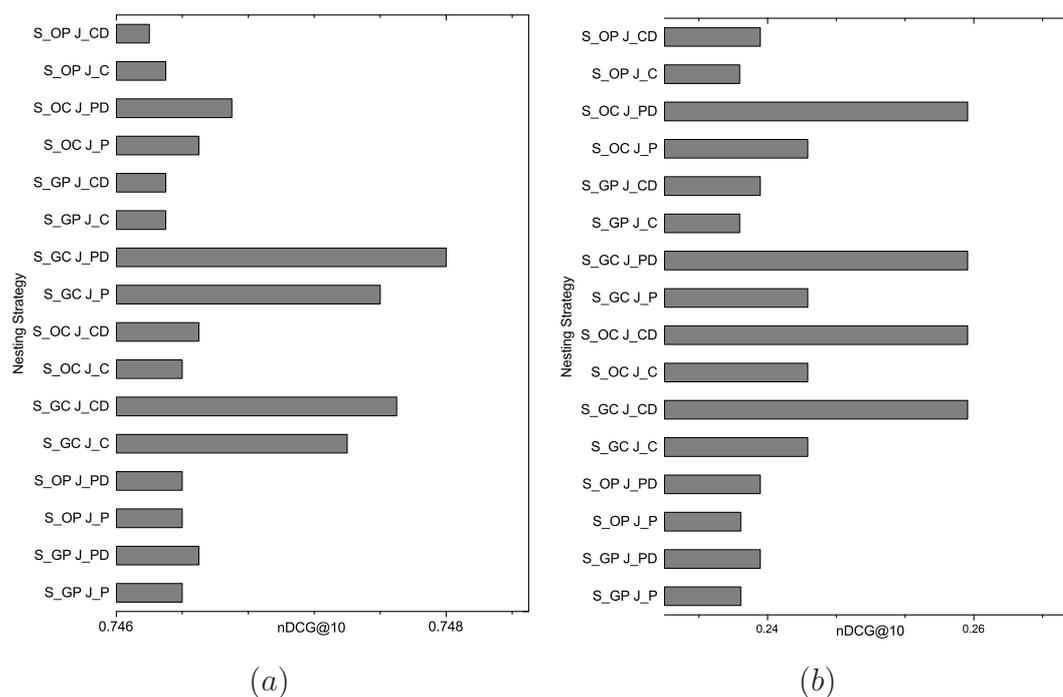


Figure 4.5: Performance examination of algorithm variations on both datasets.

its constituent words. In such a case, the optimized strategy of splitting would not have any effect on a flat segment. On the other hand, if a flat segment is deduced through matching against a list of named entities or a domain-specific multiword lexicon, getting smaller strings based on the scores is likely. Note that it is quite possible that the greedy and optimized approaches produce the same final output.

Second, one can use our co-occurrence based word association score (Equation 3.3) for scoring bigrams for *joining* smaller segments instead of PMI. This gives rise to two choices in the joining phase. Third, the definition of PMI can be appropriately extended to score n -grams when $n > 2$ [185], and can thus be used during the *splitting* process instead of our score. This gives to two choices during the splitting phase. Fourth, *joining* segments may be purely on the basis of bigram scores and DCP (preference to determiners, conjunctions and prepositions during joining, Section 4.4.2) need not be considered during the merging process. This leads to two more choices during the joining phase. We shall systematically represent and refer to these nested segmentation strategies as $S_{UV}J_{XY}$, where U is G or O for greedy and optimized approaches for splitting flat segments respectively, V and X

are C or P respectively for our co-occurrence based score and PMI scores for splitting and joining respectively. Y is D if DCP is considered during joining, else null. Thus, $S_{GC}J_P$ refers to the case where greedy splitting is done using our scores, and joining is done using PMI scores without considering DCP. If DCP is considered, the corresponding representation will be $S_{GC}J_{PD}$. In this manner, we have $2 \times 2 \times 2 \times 2 = 16$ combinations in all for nested segmentation (greedy or optimized splitting, splitting or joining with PMI or our scores and optional preference to DCP) for each input flat segmentation. Choice of these different nesting strategies is examined here.

The best performance of these variants are computed by appropriately tuning model parameters on the development set, and reported in Figure 4.5. We observe that the proposed nesting strategy $S_{GH}J_{PD}$ outperforms all the other nested segmentation strategies for both SGCL12 and TREC-WT (there are three other strategies with comparable performance for TREC-WT). In general, it is observed that during splitting, greedy approaches work better. This is due to the fact that the greedy approaches are almost always able to split a multiword segment further leading to deep nested syntax that is more informative. On the other hand, while joining, giving preference to DCP turns out to be a better choice. Interestingly, PMI scores are more useful for joining segments and our scores are better at splitting segments. This also falls in line with the assumptions underlying the usage of these scoring methods; the concept of PMI is more meaningful for examining relative strengths of pairs of words only, and thus has been more frequently used for marking segment breaks (and hence non-breaks) by observing PMI scores of adjacent word pairs [84, 114]. In contrast, our score is aimed at grading how well a group of words gel together as an expression. However, we observe that differences between strategies are not statistically significant, which highlights the flexibility of the algorithm outline.

4.7.4 Comparison with past work

For comparing our algorithm with past work, we reimplement the nested segmentation strategy of Huang et al. [99] (Section 4.2.2), which is based on SPMI (Segment Pointwise Mutual Information). A query (and its segments thereafter) is iteratively split into two halves based on an SPMI threshold until the minimum SPMI reaches a termination

threshold. We emphasize again that Huang et al. do not provide any methodology for using nesting for IR. While evaluating their algorithm, Huang et al. observed that the anchor text language model, obtained using the Microsoft Web n -gram Services¹², performed better than title, body and query models. Hence, we choose the anchor text language in our experiments as well, for fairness and comparability. The tunable parameter in their algorithm is the SPMI termination threshold α , which is required for stopping further nesting. As suggested in their paper, α needs to be tuned by optimizing one of the three matching metrics (*Exact match*, *Cover*, *Violation*) against manual annotations. Since the authors do not specify the best of these metrics, we choose to maximize *Exact match*. For this purpose, we ask three human annotators A , B and C to discover and annotate important phrasal segments from the queries of SGCL12 and TREC-WT [99]. The annotators were Computer Science undergraduate and graduate students between 22 – 28 years of age, each issuing around 20 – 30 Web queries per day. Using this policy, we observed a slightly poorer performance of their algorithm with respect to our proposed strategy. Subsequently, for fairness, we also tuned α to maximize the nDCG@10 value on the development set. Results are presented in Table 4.9. Values obtained for the three annotators were quite close to each other, and hence only their average is reported. Tuning α using manual annotations and nDCG@10 is indicated by *Anno* and *IR* respectively.

Table 4.9: Comparison with Huang et al. [99].

Dataset	SGCL12			TREC-WT		
Metric	Proposed Algo	Huang et al. (Anno)	Huang et al. (IR)	Proposed Algo	Huang et al. (Anno)	Huang et al. (IR)
nDCG@10	0.7284	0.7224	0.7240	0.1884	0.1845	0.1918
MAP	0.8481	0.8456	0.8461	0.3296	0.3263	0.3368

The highest values in rows (for each dataset) are marked in **bold**.

Our algorithm is slightly superior to Huang et al. on both nDCG@10 and MAP on SGCL12. We recollect that the SPMI threshold for Huang et al. was chosen so as to maximize nDCG@10, and hence the lower IR performance is not due to the choice of an unsuitable threshold. We observed that while the average tree height is 2.96 for our method,

¹²<http://bit.ly/bFKSxz>, Accessed 6 April 2014.

the same is about 2.23 for Huang et al. (for SGCL12). Note that due to the strict binary partitioning at each step for Huang et al., one would normally expect a greater average tree height for this method. Thus, it is the inability of Huang et al. to produce a suitably deep tree for most queries (inability to discover fine-grained concepts) that is responsible for its somewhat lower performance on the metrics. Huang et al., however, perform better on TREC-WT on both the metrics. More importantly, both nesting strategies faring favorably (none of the differences are statistically significant) bodes well for the usefulness of nested segmentation for IR in general. The tree height distributions for the two algorithms are given in Table 4.10 for both datasets (IR optimization for α in Huang et al.).

Table 4.10: Height distributions for nested segmentation tree.

Dataset	Algorithm	1	2	3	4	5
SGCL12	Proposed	0	99	327	71	3
	Huang et al.	59	292	124	23	2
TREC-WT	Proposed	15	46	13	1	0
	Huang et al.	37	30	7	1	0

Values denote the numbers of queries for each algorithm that attain a tree height equal to the column headers.

Summary of results. We now summarize our main findings: (a) Nested segmentation significantly outperforms state-of-the-art flat segmentation baselines when using segment syntax to re-rank documents based on term proximity; (b) nested segmentation improves performance for a majority of the queries, for both datasets; (c) distances in the nested segmentation tree are more effective at re-ranking than using only document and query distances; (d) exhaustive experimentation with parameter variation shows systematic consistency of tree distance-based re-ranking over other models; (e) exploration of fifteen algorithmic variations of our method for generating nested segmentations shows that the proposed technique produces the best results; (f) comparison of our results with previous work [99] shows that the proposed method is better on annotation-based thresholding, while achieving comparable performance on IR-based thresholding.

4.8 Related research

In essence, the nested segmentation tree specifies a complete term dependence syntax, and suggests that term pairs having a low tree distance should be in close proximity in the document. Our research, thus, lies along the confluence of ideas from proximity, dependence models for IR, and query segmentation. In this section, we present a brief review of proximity and dependence models, which though are not directly related to our work, can potentially benefit nested segmentation-based retrieval strategies. Work on segmentation has been reviewed in Section 4.2.

Term Proximity Models. The notion that document relevance is directly improved by query terms appearing close to each other has its roots in the NEAR operator in the Boolean retrieval framework [117] by which a user can specify that two query terms should occur near each other in the document. It has fueled a plethora of research on term proximity over the years, and primarily involves incorporating proximity heuristics into a traditional retrieval model to show a performance improvement. Tao and Zhai [217] systematically explored the proximity measures that had been proposed till date, and found that the minimum document distance between *any* two terms of a query is best correlated with relevance. They also make the important conclusion that any naïve combination of the existing ranking function with a proximity measure is unlikely to be fruitful. Cummins and O’Riordan [56] further propose more heuristics and show that ideally the minimum distance between *all* pairs of terms should be examined. They also propose that any particular measure is often unlikely to give the best overall results, and that the optimal combination needs to be learnt from the data. The proximity concept has also been generalized to term sequences rather than pairs only [20, 88, 208], which has brought with it new challenges like assigning relative weights to such sequences [20]. In fact, flat segmentation strategies roughly fall under this philosophy, with the underlying assumption that proximities (or more strictly, adjacencies) are important only within flat segments. We have shown in our experiments that such a model is easily outperformed, and the tree-based model suggests which of the long range dependencies are crucial to query semantics.

Term Dependence Models. Traditional retrieval models like BM25 assume independence between query terms, even though the idea that certain dependencies are important

for efficient retrieval is hardly new, including ideas based on tree syntax [235]. Gao et al. [74] propose a language model-based framework for predicting term dependencies by assuming that a query is generated from a document in two stages; first the linkages are formed and then the terms that satisfy the linkage constraints. Metzler and Croft [148] propose that term dependencies in a query can be classified into a sequential dependence model (SDM), where adjacent terms are related, and a full dependence model (FDM) where all the terms are inter-dependent. Their results show that significant improvements in IR are possible by formally modeling dependencies and the FDM outperforms the SDM on most corpora. Concepts of term dependence [148] have also been found useful in query segmentation by Bendersky et al. [28] and relatively newer retrieval models [171]. The nested segmentation tree based retrieval is much less computationally intensive than Gao et al. [74] and more informed than Metzler and Croft [148]. The tree not only encodes the term dependencies, but also provides an effective way of weighting long range dependencies in search queries.

4.9 Conclusions

The primary contribution of this chapter lies in proposing a strategy to use nested segmentation of Web search queries for improving IR performance. We have shown that the tree syntactic structure inherent in the hierarchical segmentation can be used for effective re-ranking of result pages ($\simeq 7\%$ nDCG@10 improvement over unsegmented query for SGCL12 and $\simeq 40\%$ for TREC-WT). Importantly, since n -gram scores can be computed offline, our algorithms have minimal runtime overhead. The only resource used for performing nested segmentation is a query log, which is always available to search engines. Thus, we believe that they can be practically useful for large-scale Web search systems. While the concept of flat query segmentation has been around for more than a decade, there is very little work that show a significant IR benefit of directly applying the process. Therefore, it has been a long standing debate whether query segmentation is at all useful in practice for IR. Our results clearly demonstrate that hierarchical segmentation can bring in substantial IR gains for slightly long queries. In addition to the unsegmented query, we have used state-of-the-art flat segmentation algorithms Li et al. [137] and Hagen et al. [84],

and the nested segmentation algorithm by Huang et al. [99], as competitive baselines. We are currently exploring this line of research further by reimplementing the term proximity model by Vuurens and de Vries [224] and comparing our approach with them. The research by Vuurens and de Vries [224] is the state-of-the-art term proximity model and produces comparable performance with the best dependence model as well. In the two chapters on query segmentation, we saw how we can effectively identify syntactic units from search queries. In the next chapter, we will try to understand roles that these units play in queries, and how we can automatically infer such roles in an unsupervised setup.

Chapter 5

Role Induction of Syntactic Units: Content and Intent

5.1 Introduction

We have seen in the previous chapters how queries are composed of syntactic units or segments. Extending this idea of query syntax further, we propose that words or multiword syntactic units in queries basically perform two roles – *content words* represent the central topics of queries, while *intent words*, are articulated by users to refine their information needs concerning the content words. The class of content units include, but are not restricted to named entities (like `brad pitt`, `titanic` and `aurora borealis`) – anything that is capable of being the topic of a query would be the content unit in the context of that query. For example, `blood pressure`, `marriage laws` and `magnum opus` are legitimate examples of content words or units. Intent words or intent units, on the other hand, present vital clues to the search engine regarding the specific information sought by the user about the content units. For instance, intent units like `home page`, `pics` and `meaning`, all specify unique information requests about the content units. The queries `brad pitt website`, `brad pitt news` and `brad pitt videos` all represent very different user intents. It is not hard to see that while content units need to be matched inside document text for relevance, it is possible to leverage the knowledge of intent units

to improve user satisfaction in better ways. For example, words like `pics`, `videos` and `map` can all trigger relevant content formats to directly appear on the result page. Words like `near` and `cheap` may be used to sort result objects in the desired order. These ideas motivate us to focus on the discovery and understanding of query intent units.

Appropriately understanding the distinction between the two classes of words and concretizing these notions of intent and content required rigorous manual analysis of large volumes of query logs on our part. During this process, we observed that intent units share corpus distributional properties similar to function words of NL. NLS generally contain two categories of words – *content* and *function* [201]. In English, nouns, verbs, adjectives and most adverbs constitute the class of content words. On the other hand, pronouns, determiners, prepositions, conjunctions, interjections and other particles are classified as function words. While content words express meaning or *semantic content*, function words express important grammatical relationships between various words within a sentence, and themselves have little lexical meaning. The distinction between content and function words, thus, plays an important role in characterizing the syntactic properties of sentences [48, 71, 101]. Distributional postulates that are valid for function word detection, like the co-occurrence patterns of function words being more diverse and unbiased than content words, seemed to be valid for query intent units as well. Following these leads, we first segment queries by our flat segmentation algorithm, and compute the relevant distributional properties, namely, co-occurrence counts and entropies, for the obtained query units¹. We found that the units which exhibit high values of these indicators indeed satisfy our notions about the class of intent units. Subsequently, we systematically evaluated our findings against human annotations and clickthrough data (which represent functional evidence of user intent) and substantiate our hypotheses.

In hindsight, we understand that while NL function words have little describable meaning (like `in`, `of` and `what`) and only serve to specify relationships among content words, well-defined semantic interpretations can be attributed to most intent words (like `map`, `pics` and `videos`). Intent words, even though effectively lacking purpose without the presence of a content word(s) in the same query, carry weight of their own within the query. Thus, content and intent units play slightly different roles in the query from the

¹Computation of co-occurrence statistics is not easily interpretable with the output of nested segmentation, and hence we use flat segmentation outputs in this work.

roles of content and function words in NL sentences.

The objective of this chapter is to identify and characterize content and intent words in Web search queries, and it is organized as follows. In Section 5.2, we begin with a verification of the efficacy of corpus-based distributional statistics towards function word identification and through rigorous experimentation over five languages, discover that *co-occurrence counts and entropies* are the most robust indicators of function words in NL. Having convinced ourselves of the power of co-occurrence statistics in detecting function words across diverse languages, we apply similar techniques to discover intent units in Web search queries (Section 5.3). This is followed by a simple algorithm to label content and intent units in the context of individual queries and subsequent evaluations using human annotations and clickthrough data (Section 5.4). Observing that co-occurrence statistics locate quite a diverse set of intent units, we attempt to provide a taxonomy of such units based on their relationships with content words (Section 5.5). Finally, we present concluding remarks (Section 5.7).

5.2 **Distributional properties of NL function words**

Function words play a crucial role in many NLP applications. They are used as features for unsupervised POS induction and also provide vital clues for grammar checking and machine translation. In this section, we first re-examine this popular hypothesis that the most frequent words in a language are the function words. By *function words or units* we refer to all the closed-class lexical items in a language, e.g., pronouns, determiners, prepositions, conjunctions, interjections and other particles (as opposed to open-class items, e.g., nouns, verbs, adjectives and most adverbs). We note that the statistics presented here are applicable for both single-word (*in, about*) as well as multiword (*how to, because of*) function units from corpora, though the latter demands chunking of the NL text. We perform all the NL experiments on unsegmented (or unchunked) sentences and hence report the results for detection of single word function units. Nevertheless, Web search queries, on which we mainly focus, have been suitably segmented by our algorithm.

Language	Corpus source	S	N	V	Function word list source	F
English	Leipzig Corpora ^a	1M	19.8M	342157	Sequence Publishing ^b	229
French	-do-	1M	19.9M	388221	Built by extracting pronouns, determiners, prepositions, conjunctions and interjections from POS-tagged corpora available at WaCKy ^c	289
Italian	-do-	1M	20M	434680	-do-	257
Hindi	-do-	0.3M	5.5M	127428	Manually constructed by linguists and augmented as above with POS-tagged corpora available at LDC ^d	481
Bangla	Crawl of <i>Anandabazar Patrika</i> ^e	0.05M	16.2M	411878	-do-	510

^a<http://corpora.informatik.uni-leipzig.de/download.html>, Accessed 18 May 2014.

^b<http://www.sequencepublishing.com/academic.html#function-words>, Accessed 18 May 2014.

^c<http://wacky.sslmit.unibo.it/doku.php?id=download>, Accessed 18 May 2014.

^d<http://www ldc.upenn.edu> (Catalog #LDC2010T24 and #LDC2010T16 for Hindi and Bangla), Accessed 18 May 2014.

^e<http://www.anandabazar.com/>, Accessed 18 May 2014.

Table 5.1: Details of NL corpora.

5.2.1 Datasets

For the NL experiments, we shall look at five languages from diverse families: English, French, Italian, Hindi and Bangla. English is a *Germanic* language, French and Italian are *Romantic* languages, and Hindi and Bangla belong to the *Indo-Aryan* family. Therefore, any function word characterization strategy that works across these languages is expected to work for a large variety of languages.

The details of the corpora used for these five languages are summarized in Table 5.1. S , N and V respectively denote the *numbers* of sentences, words and unique words present in the corpus, and F denotes the number of function words present in the gold standard list used. The sentences were uniformly sampled from larger datasets. M in the value columns denotes million. S , N , V and F denote the *numbers* of all sentences, all words, unique words (vocabulary size) and function words, respectively. We have made the lists of function words publicly available². We note that the Indian languages have almost twice as many function words as compared to the European ones. This is due to morphological

²http://cse.iitkgp.ac.in/~rishiraj/Function_words_of_5_languages.zip, Accessed 16 November 2014.

richness and the existence of large numbers of modal and vector verbs.

5.2.2 Metric

In a distributional property-based function word detection approach, the output is a ranked list of words sorted in descending order of the corresponding indicator value. Here we adopt a popular metric, *Average Precision* (AP) [17, 197], used in IR for the evaluation of ranked lists. More specifically, let w_1, w_2, \dots, w_n be a ranked list of words sorted according to some corpus statistic, say, frequency. Thus, if $i < j$, then frequency of w_i is greater than the frequency of w_j . *Precision at rank k* , denoted by $P@k$, is defined as

$$P@k = \frac{1}{k} \sum_{i=1}^k f(w_i) \quad (5.1)$$

where, $f(w_i)$ is 1 if w_i is a function word, and is 0 otherwise. This function can be computed based on the gold standard lists of function words. Subsequently, *average precision at rank n* , denoted by $AP@n$, is defined as

$$AP@n = \frac{1}{n} \sum_{k=1}^n P@k \quad (5.2)$$

$AP@n$ is a better metric than $P@k$ because $P@k$ is insensitive to the rank at which function words occur in the list. In our tables, we report $AP@n$ averaged over \mathcal{N} corpus sub-samples, which is given by $\frac{1}{\mathcal{N}} \sum_{r=1}^{\mathcal{N}} (AP@n)_r$ where $(AP@n)_r$ is the $AP@n$ for the r^{th} sub-sample.

5.2.3 Frequency as a function word indicator

Frequency (Fr) is often used as an indicator for detecting function words, but the following factors affect its robustness.

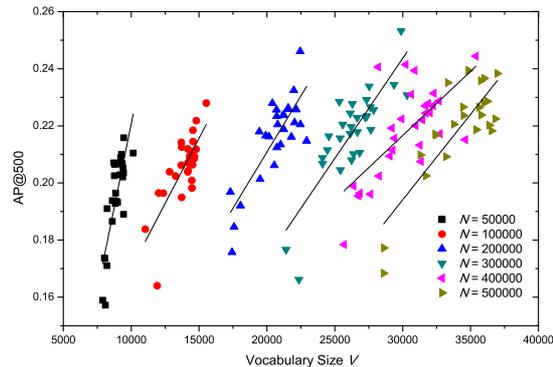


Figure 5.1: AP@500 with frequency as the function word indicator for English.

Corpus size. If the corpus size is not large, many function words will not occur a sufficient number of times. For example, even though `the` and `in` will be very frequent in most English corpora, meanwhile `and` and `off` may not be so. As a result, if frequency is used as a function word detector with small datasets, we will have a problem of low recall [17]. In our experiments, we measure corpus size, N , as the total number of words present in the corpus.

Corpus diversity. If our language corpus is restricted, or sampled only from specific domains, words specific to those domains will have high frequencies and will get detected as function words. For example, the word `government` will be much more frequent in political news corpora than `although`. The number of unique words in a corpus, or the vocabulary size, V , is a good indicator of its diversity. For restricted domain corpora, V grows much more slowly with N than in an open domain corpus.

Experiments and results

For our frequency-based experiments, we create 200 sub-samples from the original corpora. We choose 10 different values of N , and for each N choose 20 different samples such that we get a different V each time. For each sub-sampled corpora, we compute frequency of each word and sort words in decreasing order of frequency. Then we compute AP@200, AP@500 and AP@1000 with respect to the gold standard lists of function words (Table 5.1). A representative set of results is shown in Figure 5.1, for various V and N , with

Indicator	Symbol	Definition
Frequency	Fr	Frequency of a word in the corpus
Left co-occurrence count	LCC	Number of distinct words appearing to the immediate left of a word
Left co-occurrence entropy	LCE	Entropy of the left co-occurrence distribution
Total co-occurrence count	TCC	Number of distinct words appearing to the immediate left and right of a word
Total co-occurrence entropy	TCE	Entropy of the total co-occurrence distribution
Right co-occurrence count	RCC	Number of distinct words appearing to the immediate right of a word
Right co-occurrence entropy	RCE	Entropy of the right co-occurrence distribution

Table 5.2: Definitions of the different function word indicators.

linear regression lines. We see this same trend for all the languages, as well as for AP@200 and AP@1000. For a fixed N , AP increases with V , which means that the performance of the frequency-based strategy works better when the corpus has high diversity. We also observe that, in general, the performance gets better as N increases. However, for a fixed V , increasing N effectively means increasing the number of sentences without increasing the diversity of the corpus. Regression lines in Figure 5.1 suggest that for the same V , a higher N would lead to a lower AP .

5.2.4 Co-occurrence statistics as function word indicators

After having a feel of the issues faced when using frequency as a function word indicator, we introduce other properties of function words that may help in a more robust detection. We observe the following interesting characteristics about the syntactic distributions of function and content words in NL, which can be summarized by the following two postulates.

Postulate I. Function words, in general, tend to co-occur with a larger number of distinct words than content words. What can occur to the immediate left or right of a content word is much more restricted than that in the case of function words. We hypothesize that even if a content word, e.g., *government*, has high frequency owing to the nature of the domain, there will be only a relatively few words that can co-occur immediately after or before it. Thus, the co-occurrence count may be a more robust indicator of function words.

Postulate II. The co-occurrence patterns of function words are less likely to show bias

towards specific words than those for content words. For example, `and` will occur beside several other words like `school`, `elephant` and `pipe` with more or less an equally distributed co-occurrence count with each of these words. In contrast, the co-occurrence distribution of `school` will be skewed, with more bias towards `to`, `high` and `bus` than `over`, `through` and `coast`, with the list of words occurring beside `school` also being much smaller than that for `and`.

In order to test Postulate I, we measure the number of distinct words that occur to the immediate left, right and either side of each unique word in the sub-sampled corpora. We shall refer to these statistics as *left*, *right* and *total co-occurrence counts* (LCC, RCC and TCC) respectively. To test Postulate II, we compute the *entropy* [203] of the co-occurrence distributions of the words occurring to the *left*, *right* and either side of a word w :

$$\text{Entropy}(w) = - \sum_{t_i \in \text{context}(w)} p_{t_i|w} \log_2(p_{t_i|w}) \quad (5.3)$$

where, $\text{context}(w)$ is the set of all words co-occurring with w either to the left, the right or either side, and $p(t_i|w)$ is the probability of observing word t_i in that specific context window in the sentence, defined as below:

$$p(t_i|w) = \frac{p(t_i w)}{p(w)} = \frac{\text{No. of times } t_i \text{ occurred with } w}{\text{No. of times } w \text{ occurred in a sentence}} \quad (5.4)$$

Context. In this chapter, the *left*, *right* and *total contexts* of a word w respectively denote the immediately preceding (one) word, immediately succeeding (one) word and both the immediately preceding and the immediately succeeding words for w respectively, in sentences of the corpus. The definition of context (i.e., whether it includes the preceding or the succeeding one or two or three words) will change the absolute values of our results, but all the trends in the results are expected to remain the same.

This probability in Equation 5.3 can be computed simply by counting the frequency of the appropriate bigrams normalized by the frequency of w . We shall refer to these statistics as *left*, *right* and *total Co-occurrence Entropy* (LCE, RCE and TCE respectively). We would expect LCC, RCC or TCC of function words to be higher than that of content words

Language	Metric	Typology	Fr	LCC	LCE	TCC	TCE	RCC	RCE
English	AP@200	Pre-	0.663	0.702*	0.729*	0.684*	0.679*	0.637	0.527
	AP@500		0.453	0.477*	0.493*	0.468*	0.464*	0.439	0.365
	AP@1000		0.314	0.328*	0.336*	0.324*	0.319	0.305	0.259
French	AP@200	Pre-	0.594	0.642*	0.648*	0.615*	0.611*	0.553	0.501
	AP@500		0.390	0.430*	0.438*	0.405*	0.398	0.357	0.313
	AP@1000		0.264	0.290*	0.296*	0.273	0.269	0.242	0.212
Italian	AP@200	Pre-	0.611	0.639*	0.645*	0.636*	0.620	0.606	0.601
	AP@500		0.422	0.433*	0.423	0.438*	0.423	0.411	0.395
	AP@1000		0.299	0.295	0.290	0.299	0.291	0.282	0.268
Hindi	AP@200	Post-	0.682	0.614	0.510	0.698*	0.694*	0.716*	0.713*
	AP@500		0.497	0.458	0.394	0.511*	0.505	0.523*	0.521*
	AP@1000		0.368	0.345	0.306	0.379*	0.371	0.383*	0.380*
Bangla	AP@200	Post-	0.648	0.684*	0.691*	0.730*	0.763*	0.741*	0.757*
	AP@500		0.522	0.543*	0.537*	0.579*	0.599*	0.589*	0.603*
	AP@1000		0.415	0.428*	0.422	0.454*	0.470*	0.463*	0.475*

The highest value in a row is marked in **boldface**. Statistically significant improvement over frequency is marked by *. The paired *t*-test was performed and the null hypothesis was rejected if *p*-value < 0.05.

Table 5.3: AP for frequency and co-occurrence statistics.

due to *Postulate I*; similarly, due to *Postulate II* we can expect the LCE, RCE or TCE to be higher for function words than for content words. The definitions of these indicators are summarized in Table 5.2.

Experiments and results

We now sort the list of all words in descending order of each of the seven indicators. We then compute metrics AP@200, AP@500 and AP@1000 for these seven lists. To bring out the performance difference of each of the six co-occurrence features with respect to frequency, we plot (in Figure 5.2) the following performance measure against *N*:

$$\text{Value plotted} = \frac{\text{Metric for indicator} - \text{Metric for Fr}}{\text{Metric for Fr}} \quad (5.5)$$

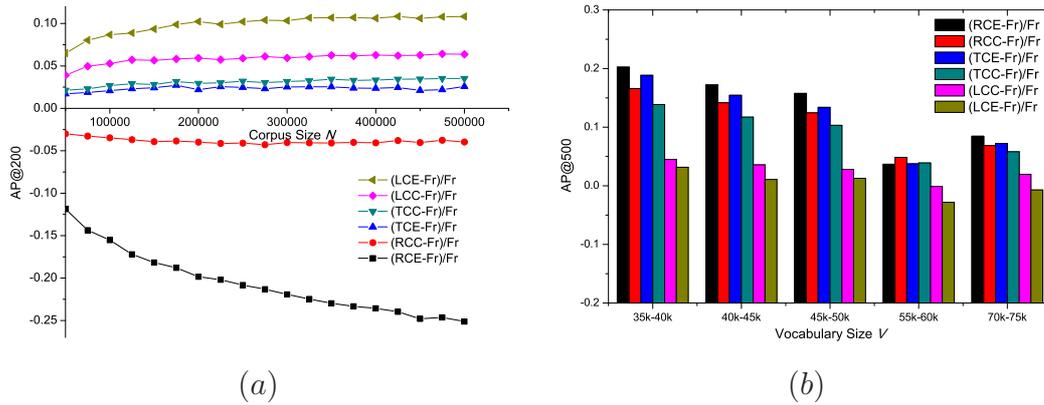


Figure 5.2: Performance of co-occurrence statistics with respect to frequency.

The x -axis can now be thought of as representing the performance of frequency. In Figure 5.2 (a), for a particular N , the data points were averaged over all (N, V) pairs (we had 20 (N, V) pairs for each N). For Figure 5.2 (b), we kept N fixed at 500000. The general trends were the same for AP@500 and AP@1000. The observations (both N and V variation) for French and Italian were similar to that of English, while those for Hindi and Bangla were similar to each other. Table 5.3 reports AP values for all statistics for the five languages. From Table 5.3, where the values are averaged over 200 (N, V) pairs for each language, we see that for all the languages, AP for some of the co-occurrence statistics are higher than AP obtained using frequency.

Regular improvements over frequency. From the plots and Table 5.3, it is evident that some of the co-occurrence statistics consistently beat frequency as indicators. In fact, as evident from Figure 5.2, use of co-occurrence statistics results in systematic improvement over frequency with variations in N and V , and hence, are very robust indicators. Among the co-occurrence statistics, entropy is generally observed to be more powerful than simple counts. This justifies that Postulate II is indeed a stricter characteristic of function units.

The best indicator depends upon language typology. A very interesting fact that came out of these experiments is that the left co-occurrence statistics (LCE and LCC) generally outperform the right for English, French and Italian, whereas the reverse is true for Hindi and Bangla (RCE and RCC are the best). This is due to the fact that English, French and Italian are prepositional languages whereas Hindi and Bangla are postpositional. In

a prepositional language, function words generally precede content words. Therefore, the lexical categories (and hence the exact numbers of lexical items) that can succeed a function word is restricted. For instance, only nouns or articles can follow words like *in* and *of* in English. On the other hand, there is no restriction on the class of words that can precede a function word. Hence function words in a prepositional language can be expected to have significantly higher left co-occurrence counts (and hence higher entropies). Similarly, the opposite is valid for postpositional languages. Thus, co-occurrence statistics have potential in predicting the *adposition typology* of a new language.

As an aside, we note that categorizing a language by its adposition typology helps in addressing several challenges in linguistics and NLP. Understanding the adposition typologies for less-studied languages by manual analysis of large text corpora can be quite expensive, yet automatic discovery of the same has received very little attention till date. Using our principle, we performed some experiments and showed that languages can be classified as prepositional or postpositional based on the rank correlations derived from entropies of word co-occurrence distributions. We experimented with 23 languages from ten diverse families, 19 of which were correctly classified by our technique.

Total co-occurrence: A safe choice. It is not always possible to know the typology of a language in advance. Thus, it may not be clear *a priori* whether to depend on left or right co-occurrence statistics. The nice point here is that the total co-occurrence statistics (TCE and TCC) are almost always better than frequency (Table 5.3). This makes them safe indicators to rely on when not much is known about the language syntax.

5.2.5 Inverse document frequency

A *stop word* is a term that is popular in IR which is used to denote a word that does not have sufficient discriminative power. Such a stop word cannot be used by the retrieval system to distinguish between relevant and non-relevant documents. Even though the concepts of stop words in IR and function words in NL understanding are fundamentally different in function, it nevertheless turns out that there is a significant level of overlap among these sets. See, for example, one of the lists of English stop words, used in the popular SMART IR system [194], at <http://bit.ly/8vBrVF> (Accessed 18 May 2014.). We note that the overlap

is caused by general domain stop words.

Thus, it is worthwhile to explore techniques used in stop word detection to our problem. The concept of Inverse Document Frequency (IDF) is traditionally used to mark stop words in IR systems. The IDF of a word w is defined as

$$\text{IDF}(w) = \log_{10} \frac{1 + |d|}{|d_w|} \quad (5.6)$$

where $|d|$ is the number of documents in the document collection d , and $|d_w|$ is the number of documents containing w . In the SMART system [194], some combination of term frequency (TF) ($\text{Fr}(w)$) and IDF, known as TF-IDF, is measured for every word-document pair (w, d) . One of the popular ways of defining TF-IDF is shown below:

$$\text{TF-IDF}(w, d) = \begin{cases} 0 & \text{if } \text{Fr}(w, d) = 0 \\ \text{TF}(w, d) \times \text{IDF}(w) & \text{otherwise} \end{cases} \quad (5.7)$$

where $\text{TF}(w, d)$ is the normalized term frequency of w in document d and is defined as

$$\text{TF}(w, d) = 1 + \log_{10}(1 + \log_{10}(\text{Fr}(w, d))) \quad (5.8)$$

where $\text{Fr}(w, d)$ is the raw frequency of w in d . The higher number of documents that a word is present in, the lower is its IDF. Stop words, by virtue of their relative abundance, have low IDF and hence low TF-IDF values. For measuring the effectiveness of TF-IDF of a word as a corpus-level indicator, we generalize it from being a document-specific value by computing the mean TF-IDF for every document containing that word.

EuroParl Corpus. The existence of multiple documents is necessary for computing IDF-related measures, i.e. the NL corpus should be segmented into discrete documents. The Leipzig Parallel Corpora used for the previous experiments contain all the sentences in a single large document, which deems it unfit for evaluating the performance of IDF. Fortunately, in version five (and earlier) of another widely used NL corpus, the EuroParl³

³<http://www.statmt.org/europarl/>, Accessed 18 May 2014.

Indicator	English	French	Italian
IDF	0.435	0.360	0.400
TF-IDF	0.035	0.020	0.030
Fr	0.571	0.564	0.547
LCC	0.678	0.667	0.633
LCE	0.722	0.649	0.648
TCC	0.648	0.623	0.601
TCE	0.673	0.609	0.593
RCC	0.592	0.524	0.540
RCE	0.492	0.454	0.508

The two minimum values in a column are marked in **boldface**.

Table 5.4: Comparison of IDF with other indicators for AP@200.

dataset [121], the corpus is fragmented into thousands of documents (approximately 5000 documents for each language). However, the EuroParl corpus, being Parliament proceedings of European countries, does not contain datasets for Hindi and Bangla. Hence, we report findings on English, French and Italian only.

Experiments and results. For a fair evaluation, we need to recompute AP values for all indicators for the EuroParl dataset and contrast them with IDF and TF-IDF. Note that while ranked lists for frequency and co-occurrence statistics were obtained by sorting words in descending order of these indicators, a reverse sorting (ascending order) is necessary for IDF and TF-IDF (stop words have low IDF). We summarize our results in Table 5.4 (AP@200). Trends observed for AP@500 and AP@1000 are exactly the same.

From Table 5.4, we see that TF-IDF performs the worst, followed by IDF. But even for IDF, the difference in performance with the next better indicator is always substantial. Thus, we infer that these measures are clearly unsuitable for function word detection. On manually analyzing the ranked lists for understanding the poor performance of IR measures, the reason was clearly understood. IDF and TF-IDF pull out stop words that do not offer discriminating evidence for ranking documents in response to a query. A majority of these words at the top positions turn out to be content words like `resume`, `declare`,

and `adjourns` (except the few most frequent function words). Note that the corpus is from a restricted domain (Parliament proceedings), and the domain-specific stop words negatively impact the performance of IDF-based measures. We recollect that the same reason is one of main drawbacks of using frequency as an indicator (Section 5.2.3). The best performance again comes from co-occurrence statistics (mostly entropy), highlighting their robustness even in restricted domain datasets.

Through the experiments on the EuroParl corpus, we have shown that IDF and TF-IDF are not good function word indicators in restricted domain corpora, which is often the case with many NLP applications. However, IDF and TF-IDF are expected to perform better in a more general setting, for example, when all Web documents form the document collection. Nevertheless, our indicators based on co-occurrence counts and entropies are observed to perform well across all scenarios. Hence, they can be used for function word detection from Web corpora as well. Also, we note again that while low TF-IDF has been shown to be an effective stopword detector, the concepts of function words and stopwords are fundamentally different.

5.3 Intent units of Web search queries

In this section, we apply our robust function word identification strategies to query logs and observe the resultant partitioning of words. We find that the top ranking words according to co-occurrence statistics align well with our notion of intent units (Section 5.1). For all our experiments on queries, we use our query log sampled from Bing Australia (Section 1.2).

5.3.1 Operational definitions

We study and classify *segments* (units) for Web search queries. In our study, we used our flat query segmentation algorithm (Section 3.2), which uses query logs and Wikipedia titles as the input resources, to identify query segments. For facilitating representation and understanding for certain annotations, segment boundaries are marked by parentheses in this chapter, like `(public schools) (new york)`, instead of the usual `pipes(|)`.

Ranks 1-10	Ranks 11-20	Ranks 21-30	Ranks 51-60	Ranks 91-100
in	with	for sale	home	time
the	lyrics	is	de	your
and	by	what is	pictures of	book
for	from	best	music	show
of	2010	vs	uk	la
free	online	video	jobs	myspace
to	new	2009	black	baby
on	at	my	song	james
how to	2008	pictures	news	cheap
a	download	school	about	does

Table 5.5: Sample units at top ranks when sorted in descending order by TCE.

We apply the segmentation algorithm on all the queries and compile a list of unique units (about 1.3M in number) that occur in our query log. For each unit, we measure its frequency, the three co-occurrence counts and the corresponding entropies.

To give a feel of the units that are pulled up, we present some examples in Table 5.5 when sorted in descending order of TCE. Only 26 out of the top 100 units for queries are function words of English. We understand that it can be hard to make a definite distinction between content and intent units solely on a qualitative basis. So before we can have any further quantitative evaluation of our indicators, we must have in place *operational definitions* of content and intent units in queries that can help concretize the notion of a word being content or intent with respect to a query. An empirical validation of the proposed operational definitions is presented in Section 5.4.3.

Content units in Web search queries. They carry the core information requirement within a Web search query. Just like the role of content units in NL sentences, removing these units makes the query lose its central idea. For this reason, content units need to be *matched* within the documents for effective retrieval. For example, `titanic`, `age of empires` and `ford cars` are all content units.

Intent units in Web search queries. They specify user intent in Web search queries. They *need not match* exactly at the document side, and the search engine can have intelligent techniques for using such units to increase the relevance of result pages. For example, `music`, `online`, and `for sale` are some commonly encountered intent units. Analogous to NL, removal of these units removes vital details about query semantics. We note that function units in NL (like `and`, `of` and `in`) can play similar roles in queries, and hence fall under this category.

These definitions of content and intent words, and the condition of matching in document text, are extremely vital to principles in semantic search. We emphasize that the definitions of content and intent are always necessarily operational – content segments need to be matched in the document text during the retrieval process, while the search engine can have intelligent techniques to process intent segments to improve relevance of result pages. Thus, what has to be treated as content today can become an intent segment after a few years if the (semantic) search system develops a more improved way to handle that segment than searching for it in the document text. This is where it differs from other similar frameworks, which are static and more like the entity-attribute model [160].

For example, in a query like `london wedding` or `london population`, we would treat `wedding` (or `population`) to be a content word and not an intent word (`london` would be a content too), because in the current search scenario, there is almost no way to infer the “intent” `wedding` or `population` from a page without matching the term within the document text. `Population` could become an intent word the day when annotations or other features of a Semantic Web enable the engine to infer the answer (i.e. the population of a city or country) even without the presence of the word on the retrieved page. But `population` is, and would always remain, an attribute of a country or a city (which is the entity). Current search engines provide direct answers to queries like `london population today` but those are summaries generated from a document pool created by traditional matching. In contrast, for queries like `london weather`, `london place` and `london life` (generally all Web queries are in lowercase), `london` would be content (as it is the topic of the information need) and `weather`, `place` or `life` would be intent as there exist ways today (search engines may use them or not) to infer information relevant to these contexts without direct matching. Say, for example, knowledge graphs enable the search engine to know that temperature, rainfall, and humidity are aspects of

weather (as can be employment, poverty and cleanliness aspects of city life) and can be scraped off pages to provide consolidated information on weather and life. Intent words like place or location can be used to understand the preferred content type, like bringing up relevant maps. In summary, the collection of all intent words or units is a *dynamic set* completely defined for a particular span of time by the state-of-the-art (semantic) search technologies available during that span of time.

5.3.2 Experimental results

We note that it is not possible to build an exhaustive list of such intent units for queries. So in order to have a suitable gold standard set created by humans for future validation of results, we first need a representative sample unit set. These can be manually classified as intent units (or content units). To avoid bias towards any particular indicator, we took the union of the top 1000 units when sorted by each indicator. We asked three human annotators *A*, *B* and *C* to mark these 1215 query segments as “intent” or “content” with the above operational definitions as guidelines. All of our annotators were graduate students in the age group of 25 – 35 years and were well-acquainted with Web search, each issuing about 20 – 30 queries per day. Out of the 1215 segments, *A*, *B* and *C* marked 607, 646 and 548 units as intent respectively. Now we assume the units marked as “intent” by each annotator separately as the gold standard. Then, similar to the method followed in NL, we sort the list of all units in descending order of each of the seven indicators and compute the AP@200, AP@500 and AP@1000 for these ranked lists. Results are presented in Table 5.6.

Superiority of total co-occurrence. Just like NL, co-occurrence statistics consistently beat the performance of frequency. When the ranked list is small (200 units), the right (*A* and *C*) or left (*B*) co-occurrence statistics gives the best accuracy. On the other hand, for longer lists (500 and 1000 units), the total co-occurrence count (*A*) and entropy (*B*) always perform the best. In general, total co-occurrence statistics are generally the best or the second-best, with improvements over frequency in all cases. These trends are observed across all the annotators, thus underlining the adequacy of the operational definitions. We observed that *C* was more strict in labeling units as intent (markedly lower AP values than *A* and *B*). This can be understood from the following example units that are marked

Annotator	Metric	Fr	LCC	LCE	TCC	TCE	RCC	RCE
<i>A</i>	AP@200	0.622	0.654	0.639	0.696	0.653	0.701	0.668
	AP@500	0.462	0.495	0.498	0.548	0.519	0.513	0.479
	AP@1000	0.335	0.348	0.331	0.421	0.400	0.343	0.305
<i>B</i>	AP@200	0.719	0.812	0.854	0.850	0.852	0.793	0.777
	AP@500	0.528	0.617	0.631	0.665	0.674	0.590	0.567
	AP@1000	0.381	0.416	0.408	0.488	0.491	0.388	0.363
<i>C</i>	AP@200	0.434	0.458	0.488	0.490	0.494	0.542	0.535
	AP@500	0.338	0.361	0.359	0.401	0.385	0.392	0.381
	AP@1000	0.252	0.261	0.253	0.322	0.308	0.260	0.243

The two highest values in a row are marked in **boldface**.

Table 5.6: AP of each of the indicators for intent unit detection in Web queries.

as intent by *A* and *B* but not by *C* – *driver*, *kids*, *tutorial*, *program* and *custom*. All of these do carry user intent in queries, but not in a direct fashion like the more general units like *movies*, *define* and *games* (labeled as intent by all three).

Intent units which tend to occur at the beginning of the query have low LCC and LCE (e.g. *how to*, *what does* and *define*). Similarly, there are examples like *mp3*, *for sale* and *blog*, which typically occur only at the end in queries, displaying the opposite behavior. Such extreme cases are rare in NL, because words that begin or end a sentence also frequently occur at other positions. Thus, left or right co-occurrence alone are insufficient for extracting intent units in queries, highlighting the importance of total co-occurrence statistics.

Rank adjustments by co-occurrence statistics. In Table 5.7, we compare the ranks of a few units with respect to the seven different statistics. Content units like *wedding* can have very high frequency owing to the popularity of the event or concept; however, co-occurrence statistics help push such candidates lower down the list (from Rank 138 in frequency to out of the top-200 by all other indicators). Next, we see that intent units like *blog* and *define*, which rank around 500 by frequency move much higher up the

Unit	Fr	LCC	LCE	TCC	TCE	RCC	RCE
for sale	16	24	30	27	58	119	2,216
pictures	48	39	35	56	45	93	53
mp3	109	75	93	115	221	487	1,712
blog	490	164	87	294	127	1,323	945
biography	824	278	110	561	171	5567	4,009
how to	4	80	77	8	32	2	11
wedding	138	363	377	295	438	240	447
make a	188	3,953	209,164	213	923	66	40
what does	316	2,275	1,517	174	734	56	294
define	503	1,727	1,098	199	51	70	22

Table 5.7: Ranks assigned to query intent units by the seven different statistics.

ranked list when appropriate co-occurrence statistics are used. Hence, average precision is generally observed to increase for co-occurrence-based features. We note that the rank of `make a` by LCE is 209,164. This is because `make a` is preceded by only a handful of segments like `how to` or `way to`. Thus, it has a very restricted left co-occurrence distribution and hence a very low LCE. This pushes its rank by LCE so far down. Other indicators are seen to have balancing effects on words with such skewed distributions.

A note on segmentation errors. First names like `james` co-occur with several different family names and acquire a high rank (Table 5.5). We would not have observed them this high up in the lists had the segmentation algorithm always been able to group together entire names. For example, popular figures like *james bond* and *james cook* do get grouped together, and as units they do not have such high co-occurrence statistics.

A note on IDF for queries. The concept of IDF (Section 5.2.5) cannot be explored in the context of intent word detection in Web queries (Section 5.3) because even though each query can be considered as a sentence, the concept of a (coherent) *document* is not well-defined. The only notion that comes close is grouping the queries from a single user *session* as a document. However, session segmentation of a query stream is an active area

of research [6, 113] and is beyond the scope of this work.

5.4 Labeling intent units in query context

A segment can act as content or intent in a query depending upon the context. For example, while the segment `video` behaves as an intent unit in most queries, like, `(us open) (video)` (specifying that the desired content type is a video), it is the content unit in the query `(definition of) (video)`. Thus, a labeling scheme is practically useful only if it can label segments as content or intent within a query, and not just in a context-agnostic standalone fashion. In this work, for simplicity, we restrict ourselves to labeling two-segment queries; extension to multi-segment queries is an important future work. Interestingly, two-segment queries (derived from the output of our flat segmentation algorithm) form a significant proportion of our Bing log ($\simeq 44\%$).

As a first step, we define an *intent-ness score* $IS(u)$ for every unit u that appears in the query log. Since all our indicators hold clues towards the *intent-ness* of a unit, this score is calculated as a simple log-linear combination of the indicators as

$$IS(u) = \log_2(\text{Fr}(u)) + \log_2(\text{LCC}(u)) + \text{LCE}(u) \\ + \log_2(\text{TCC}(u)) + \text{TCE}(u) + \log_2(\text{RCC}(u)) + \text{RCE}(u) \quad (5.9)$$

Logarithms of Fr , LCC , TCC and RCC are taken to make them comparable in value to the entropies (c.f. Equation 5.3), which are already in logarithmic space. Since intent units are expected to obtain higher individual feature values than content units, the former is also expected to achieve higher intent-ness scores. However, we understand that there could be more appropriate methods of feature combination [59] like learning weights with linear regression models, but such methods require supervision (while all the techniques used in this research are unsupervised) and will require detailed experimentation.

Algorithm. The segment with the lower IS in a query is marked as content ($\setminus c$). The intuition behind this is that a query must have at least one content unit, and the IS of a con-

tent unit is expected to be lower than that of an intent unit. If the score of the other unit in the query exceeds that of a user-defined threshold δ , it is marked as intent ($\backslash i$). Otherwise, the second unit is also labeled as content. Since the absolute number of intent units in the query log is expected to be low in comparison to the number of content units, simply labeling the unit with the higher IS as intent, without a threshold, would result in too many false positives. We note that if the intent-ness score (IS) of a segment is below the threshold δ , it will always be labeled as content. Obtaining an intent-ness score below the threshold essentially means that there is insufficient evidence in the query log for labeling this unit as intent. Thus, our tagging algorithm labels two-segment queries as *either* content segment-intent segment (equivalently intent segment-content segment), *or* as content segment-content segment. We denote the first set of queries as *content-intent queries* (for example, (brad pitt)\c (home page)\i, (pictures of)\i (digestive system)\c and (how to)\i (paraglide)\c) and the second set of queries to be *content-content queries* ((brad pitt)\c (jennifer aniston)\c, (digestive system)\c (carbohydrates)\c and (paraglide)\c (safety equipment)\c).

5.4.1 Evaluating in-query labeling using human annotations

Experiment. Our test data comprised of 2600 unique two-segment queries (segmented by our flat segmentation algorithm), randomly sampled from all the two-segment queries in our entire Bing log. These queries were not used for training. We asked our three annotators A , B and C , who had previously annotated individual segments (Section 5.3.2), to annotate 1000 queries each by marking the segments as content or intent units, as they deem fit, in accordance with the operational definitions. If the segmentation was incorrect, they were supposed to provide the correct segmentation and then mark the content and intent units. Queries that had more or less than two segments after annotation were not considered for further steps. In order to measure inter-annotator agreement (IAA), we had ensured that there are 200 queries common for all the annotators A , B and C $((1000 - 200) \times 3 + 200 = 2600)$ queries. Some general sample annotations, not restricted to this dataset of two-segment queries, are shown in Table 5.8.

For content unit labeling in queries, in general, our method can be improved by using

Human Labeled Query	Machine Labeled Query
(roger federer)\c (pics)\i	(roger federer)\c (pics)\i
(cranes)\c (for sale)\i	(cranes)\c (for sale)\i
(star trek)\c (wikipedia)\i	(star trek)\c (wikipedia)\i
(britney)\c (biography)\i	(britney)\c (biography)\c
(ethan hawke)\c (movies)\i	(ethan) (hawke) (movies) [†] *
(adobe flash)\c (download)\i	(adobe flash)\c (download)\i
(free)\i (video converters)\c	(free video)\i (converters)\c [†]
(hotels)\c (near)\i (airport)\c	(hotels) (near) (airport)*

[†] Error in segmentation algorithm.

* Machine unable to label more than two-segment queries.

Table 5.8: General examples of segmented and labeled queries.

rules and resources for identifying named entities (NE) like names of people, organizations and places using NE lists such as Yago, DBpedia and Freebase. However, that would make our method partly supervised or informed. Hence, we did not try those out. But for practical applications, it would be imperative to fine-tune the algorithm using such rules. Usually lists will work only for the relatively well-known entities, and if our segmentation algorithm can correctly group (rare or popular) entities, our content-intent tagger will also make the correct decision most of the time as such entities will have restricted co-occurrence distributions and will be correctly marked as content, even if it does not appear on the popular NE lists.

Results and observations

Percentage Inter-annotator Agreement (IAA) on the labels, i.e., percentages of units on which annotators agree on the content-intent labels, are 83.99, 77.06 and 77.32 for $A - B$, $B - C$, and $C - A$ respectively. All annotators marked about 70% of the units as content and the rest 30% as intent. The corresponding values for Cohen’s Kappa (κ) [53], a stricter metric for IAA that considers the effect of chance agreements, are 0.62, 0.45 and 0.46. A κ close to 0.5 indicates statistically significant IAA between annotators.

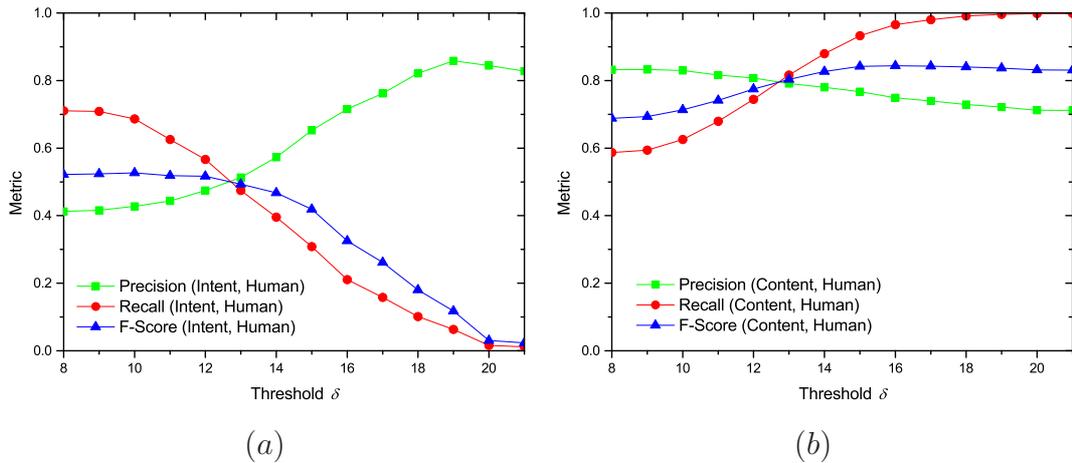


Figure 5.3: Evaluation of labeling units against human markup.

For simplicity, from now on we use only the 2400 queries for which we have exactly one annotation, for our analysis. Out of these 2400 queries, 1356 queries (56.5%) were labeled as content-intent and 1044 queries (43.5%) were labeled as content-content by our annotators. We first compute our labeling accuracy by penalizing cases where our algorithm predicts an opposite set of labels for content-intent queries. Results show that our algorithm achieves a labeling accuracy of 78.79% (82.28% for *A*, 78.67% for *B*, and 75.43% for *C*) ($\delta = 13$, as determined through experiments presented later). This is particularly high considering that the IAA is also roughly 80%. This means that we predict the opposite set of labels only about 20% of the times; to be specific, for 271 queries (out of 1356 queries). The mistakes typically occur in those cases where the content unit is very popular and achieves a significantly high intentness score, while the intent unit is relatively uncommon. For example, in the query (`finland`) (`bed and breakfast`), `finland` is marked as intent by the annotator and `bed and breakfast` as content, while our algorithm labels wrongly as the reverse. According to our framework, `bed and breakfast` is the main topic of the query and hence acts as content, whereas the location `finland` represents user intent (see *source* specifiers, Section 5.5).

Effect of Threshold. We evaluated the labeling algorithm against the test set at different values of δ . For this purpose, we computed the precision, recall and F-Score [197] for intent and content units, as defined below.

$$\text{Precision}(\text{Intent units}) = \frac{\#(\text{Units correctly labeled as intent})}{\#(\text{Units labeled as intent})} \quad (5.10)$$

$$\text{Recall}(\text{Intent units}) = \frac{\#(\text{Units correctly labeled as intent})}{\#(\text{Units labeled as intent by annotators})} \quad (5.11)$$

$$\text{F-Score}(\text{Intent units}) = \frac{2 \times \text{Precision}(\text{Intent units}) \times \text{Recall}(\text{Intent units})}{\text{Precision}(\text{Intent units}) + \text{Recall}(\text{Intent units})} \quad (5.12)$$

The precision, recall and F-score for content units are defined similarly. We note that these metrics are computed by looking at the aggregate pool of content-intent and content-content queries, i.e. all the 2400 queries. Figures 5.3 (a) and (b) show the curves obtained when these metrics are plotted by varying δ for intent and content unit detection, respectively. The optimum δ turns out to be about 13 (value used in the previous experiments for computing labeling accuracies). Our content labeling has a much higher precision than intent labeling, but this is correlated to the fact that the natural proportion of content units in a query log is expected to be much higher than that for intent units. As one would expect, there is a trade-off between precision and recall.

5.4.2 Evaluating in-query labeling using clickthrough data

Till now, we have postulated and identified the distributional characteristics of the lexical categories of the query language, i.e. content units and intent units. As in NL, lexical categories in queries must also have their specific *functions*. In fact, our notions of content and intent units are based on their functions, which is *a content unit denotes the core information need of the user* and *an intent unit further modifies the information need in one of many possible ways* (Section 5.3.1). We asked the question if we can mathematically model and compute the functional characteristics of these units and provide further evidence for their existence. One possible way to study the functions of the units is to analyze click data. A click is representative of the function or the role of the unit in a query because it leads to the purpose of issuing the query, i.e. land on a (possibly) relevant page.

Human judgments can often be very expensive to obtain on a Web scale. Fortunately, clickthrough logs can also help us in large-scale automatic evaluation of our content-intent

labeling algorithm. The basic idea is as follows: Consider two content units c_1 and c_2 (say `tom cruise` and `anjelina jolie`) and two intent units i_1 and i_2 (say `movies` and `home page`). The queries c_1 , $c_1 i_1$ and $c_1 i_2$ (or c_2 , $c_2 i_1$, and $c_2 i_2$) are closely related because the core information need, which is c_1 (or c_2), is the same for all of them. Therefore, we can expect to see a good amount of overlap among the URLs clicked for each of them. On the other hand, the queries i_1 (if it makes sense), $c_1 i_1$ and $c_2 i_1$ (or i_2 , $c_1 i_2$ and $c_2 i_2$) are very different in their information needs. Hence, we can expect very little, if not zero, overlap among the URLs clicked for them. Thus, one way to define the *information content* of a unit u is to collect all queries containing u and compute the overlap between clicked URLs for these queries. A low overlap would imply that u is usually an intent unit, and a high overlap indicates that u is generally a content unit. This concept is illustrated through an example in Figure 5.4. The exact procedure of using clickthrough logs to arrive at a labeling of a two-segment query is explained next.

Modeling click overlap

A precise quantification of the amount of overlap between two sets of URLs is non-trivial because exact string match to compare URLs is unreliable. For instance, the pair of URLs www.puzzle.com and www.puzzle.com/demo/help.html are very closely related, but do not match exactly at string level. On the other hand, partial string-level matches can also be misleading. For example, URLs en.wikipedia.org/wiki/fox and en.wikipedia.org/wiki/guitar have no logical overlap. Therefore, we first show how to identify the overlaps between pairs of URLs (with respect to a particular query, as in Figure 5.4), and then use these overlap values to compute the overlap between two *sets* of URLs. Let a URL \mathcal{U} be created by the concatenation of a number of strings s_{u_i} . Drawing upon intuition, we propose that the overlap between a pair of URLs $\mathcal{X} \equiv s_{x_1}/s_{x_2}/s_{x_3}/\dots/s_{x_k}/\dots/s_{x_{n_1}}$ and $\mathcal{Y} \equiv s_{y_1}/s_{y_2}/s_{y_3}/\dots/s_{y_k}/\dots/s_{y_{n_2}}$ depends on the following factors: the length (as measured by the number of strings delimited by slashes) of the prefix up to which the URLs match exactly (k), the number of times the URLs have been clicked for the query under consideration (click counts $c_{\mathcal{X}}$ and $c_{\mathcal{Y}}$), the lengths of the URLs n_1 and n_2 (as measured by the number of strings delimited by slashes), and a quantity we term as the Inverse URL frequency (IUF). This last factor is helpful in identifying very general domain pre-

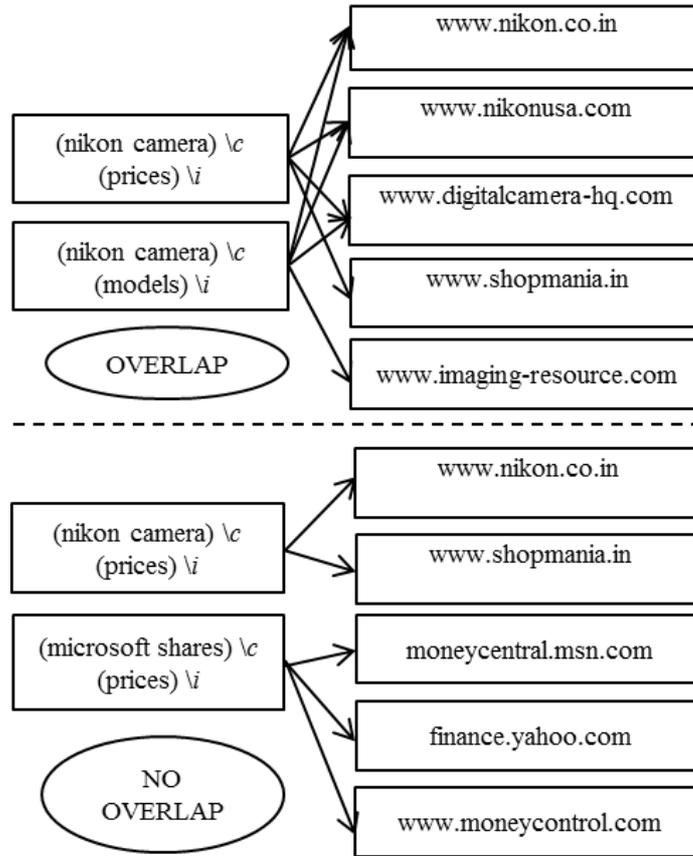


Figure 5.4: Illustrating difference in click overlaps.

fixes such as en.wikipedia.org which should contribute minimally to the overlap score (c.f. the concept of IDF in Section 5.2.5). We define the IUF of a URL prefix s as follows (c.f. Equation 5.6 for justification):

$$\text{IUF}(s) = \log_{10} \frac{1 + |U|}{|U_s|} \quad (5.13)$$

where $|U|$ is the number of distinct URLs in our log and $|U_s|$ is the number of distinct URLs with prefix s . The overlap o between \mathcal{X} and \mathcal{Y} is directly proportional to the IUF of the first string of the common prefix ($s_{\mathcal{X}_1}$ or $s_{\mathcal{Y}_1}$), the number of *common* clicks obtained by *both* the URLs ($\min(c_{\mathcal{X}}, c_{\mathcal{Y}})$), and the length of the common prefix (k). On the other hand, it is inversely proportional to the sum of n_1 and n_2 , i.e. the sum of the lengths of the two URLs (in terms of constituent strings). For the last factor, we use the mean length of

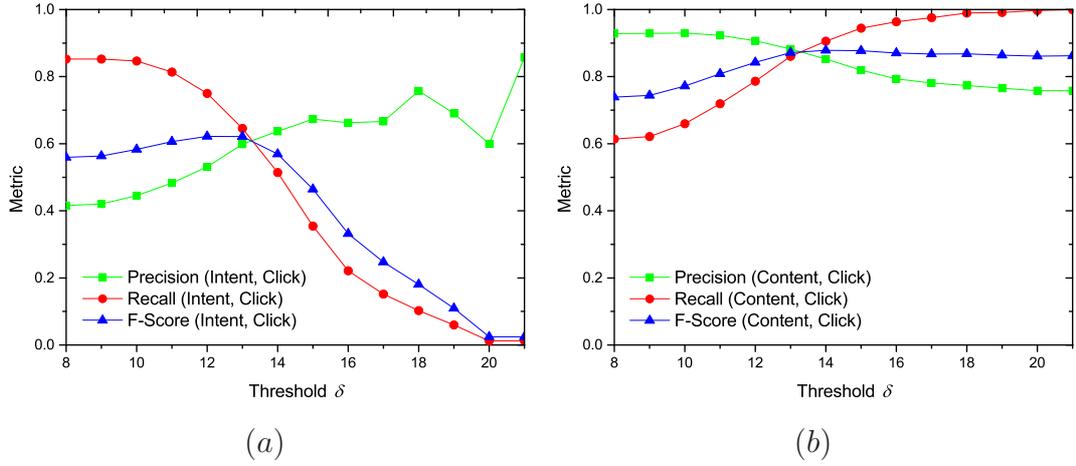


Figure 5.5: Evaluation of labeling units against click data.

the two URLs as the combining factor, i.e. $\frac{n_1+n_2}{2}$. We thus define the overlap o between \mathcal{X} and \mathcal{Y} as a simple combination of the factors as (assuming the constant of proportionality to be one):

$$\begin{aligned}
 o(\mathcal{X}, \mathcal{Y}) &= \text{IUF}(s_{\mathcal{X}_1}) \times \min(c_{\mathcal{X}}, c_{\mathcal{Y}}) \times k \times \frac{1}{\frac{n_1+n_2}{2}} \\
 &= \text{IUF}(s_{\mathcal{X}_1}) \times \min(c_{\mathcal{X}}, c_{\mathcal{Y}}) \times \frac{2k}{n_1 + n_2}
 \end{aligned} \tag{5.14}$$

The contributing factors could be combined in a better way to define the resultant overlap. However, it is not the focus of this research and we record it as future work. To compute the click overlap of a set of URLs S , we compute the mean of the pairwise overlaps of all URLs in S . For each content or intent unit u , a value of o can thus be derived. The final labeling is done as follows:

“For a given two-segment query q , the unit with the lower overlap $o(u)$ is treated as an intent unit, and the one with the higher $o(u)$ as content.”

Results and observations

The identification of the different behaviors of click overlaps for content and intent units opens up the possibility of not being tied to manual annotations for evaluation. We checked the percentage IAA of labeling done using click overlap formulation (unit with lower overlap is intent, the other is content) with the manual annotators and found it to be 73.09%, 71.65% and 68.23% for *A*, *B* and *C* respectively, which are similar to our earlier IAA values (Section 5.4.1). We then checked the precision, recall and F-Score (Equations 5.10 through 5.12) for our labeling algorithm with the output produced by click data modeling. The definition of recall, however, is appropriately modified to

$$\text{Recall(Intent units)} = \frac{\#(\text{Units correctly labeled as intent})}{\#(\text{Units labeled as intent by click data})} \quad (5.15)$$

A similar change is made for content recall. Figures 5.5 (a) and (b) show the corresponding plots obtained by varying threshold δ , for intent and content units respectively. These results are markedly similar to the results produced by evaluating against human annotated data (Figure 5.3), which justifies our choice of using clickthrough data as an alternative evaluation strategy.

5.4.3 Verification of the operational definitions

While every relevant document for a query must contain the content units, this is not necessarily true for intent units. For example, in the query (*jaguar x8*) (*for sale*), the user expects every relevant document to contain the content unit *jaguar x8*, but this is not true of the intent unit *for sale*. This was the basis on which our operational definitions for content and intent units were formulated. We verified the validity of this notion on our IR corpus (Section 3.4). Since this dataset contains queries accompanied by relevant documents, it is appropriate for verifying our operational definitions.

We used the segmented versions of the queries as output by our flat segmentation algorithm, and subsequently labeled the 383 two-segment queries with content and intent tags using our algorithm. We now wish to observe the presence, and the distribution, of

content and intent segments in relevant documents associated with each query. Since exact string matching for segments in documents can often be misleading (the segment `hp aio printers` can be present in the document as `hp printer aio`), we formulated the following three-point (0 – 2) scoring criteria for *approximate* segment matches in documents. Exact and exact stemmed matches (each word of the segment stemmed by the Porter Stemmer [177]) would be rated as *SM 2* (Segment Match Grade 2). *SM 1* is awarded if the stemmed segment was present in a *modified* form in the document. We define segment *modification* as a 1-insertion, a 1-substitution, a 1-deletion or a 1-transposition (1-indicating at *one position* only) of the stemmed form. The above operations, as applied to a multiword expression $\mathcal{M} = \langle a b c d \rangle$, are explained next. We note that there are some overlaps among these sets, but since all are assigned the same score (*SM 1*), it does not make a difference. We do not deal with n -modifications in this work, where $n > 1$.

- 1-insertions. All new segments formed by inserting one new word in an intermediate position of the original segment. 1-insertions for $\mathcal{M} = \{\langle a x b c d \rangle, \langle a b x c d \rangle, \langle a b c x d \rangle\}$, where x is any word.
- 1-substitutions. All new segments formed by substituting one word in the original segment by a new word. 1-substitutions for $\mathcal{M} = \{\langle x b c d \rangle, \langle a x c d \rangle, \langle a b x d \rangle, \langle a b c x \rangle\}$, where x is any word.
- 1-deletions. All new segments formed by deleting one word from the original segment. 1-deletions for $\mathcal{M} = \{\langle b c d \rangle, \langle a c d \rangle, \langle a b c \rangle\}$.
- 1-transpositions. All new segments formed by swapping the positions of one pair of adjacent words in the original segment. 1-transpositions for $\mathcal{M} = \{\langle b a c d \rangle, \langle a c b d \rangle, \langle a b d c \rangle\}$.

The segment is searched in the document text of each document in the query pool (average pool depth for this dataset is about 30) and the subsequent match (or non-match) is rated as *SM 0*, *SM 1* or *SM 2*. Each document in our document pool is associated with an RJ of 0 (non-relevant), 1 (partially relevant) or 2 (relevant). Since each segment searched is tagged as content or intent, we can now build the following 3×3 matrices for degree-of-match versus degree-of-relevance, accumulated for all segments of a particular type (Tables 5.9 and 5.10).

Content	SM 0	SM 1	SM 2
RJ 0	19.113	2.647	7.285
RJ 1	21.566	3.844	13.491
RJ 2	15.328	3.022	13.704

Table 5.9: Segment match versus document relevance for content units.

Intent	SM 0	SM 1	SM 2
RJ 0	20.458	0.360	4.530
RJ 1	30.747	0.390	6.959
RJ 2	27.127	0.630	8.799

Table 5.10: Segment match versus document relevance for intent units.

The absolute counts of the specific cases in the matrix cells were normalized by the sum of the values in the entire table, and converted into percentages. The first rows of the tables are greyed out because matches in non-relevant documents are not of interest to us. The second and the third rows imply that the document was at least partially relevant to the query. If we consider exact and partial matches (*SM 1* or *SM 2*) for these two rows, we see that the corresponding total percentage for content units ($\simeq 34\%$) is almost double of that for intent units ($\simeq 17\%$). Moreover, we note how the absence of segments affects document relevance. For content segments, in only $\simeq 37\%$ cases was the document at least partially relevant (*RJ 1* or *RJ 2*) when the segment was absent in the document, while the corresponding number for intent segments is as high as $\simeq 58\%$. Both of these observations indicate that while matching a content segment in a document is crucial to improving IR performance, an intent segment *need not* always match (exactly or partially) for the document to be relevant – thus validating our operational definitions. It is important to note that the way current Web documents and commercial search engines are designed (emphasising presence or match of keywords), it is very difficult to obtain substantial evidence for pages that do not contain the intent units and yet are relevant to the query. However, it is intuitive that such pages exist on the Web, and one of the main objectives of semantic search is to

discover these pages.

5.4.4 Use of content and intent labeling in IR

Labeling segments as content or intent is only half of the work required for our ideas to be useful in a practical scenario. The second half is the functional aspect, i.e. to be somehow able to use these labels during the IR process for better ranking or result presentation. We note that there can be several ways of doing this, and search engines are possibly doing some of these today. For example, specifying `video` or `pics` or `map` with content units almost certainly puts video or image or map content at the top, instead of the usual “ten blue links”.

We devise a simple and generic application for our labeling strategy, in line with our operational definitions of content and intent. Our intuition lies in the definitions themselves: while content segments need to be matched exactly within documents, intent units need not match exactly in the document text for relevance. Current search engines support use of the double quotes operator (“.”) to force exact phrase match in the document. Exact match refers to perfect ordering of segment words in the document, without word insertions, deletions, transpositions, substitutions or other linguistically informed flexible matching criteria (like synonyms). However, it is known that users rarely use quotes in their queries to use this feature (only about 8% of queries in our Bing log), while a much larger fraction of queries (about 71% as reported in Guo et al. 2009 [81]) do have named entities or multiword expressions (`roger federer`, `summa cum laude`) within them. It could also be detrimental to put quotes indiscriminately around all segments. In our opinion, for example, it would be harmful to ensure exact match for intent segments like `how to` or `difference between`, because a page can contain the procedure for something or comparison between items (say, as a table) without having these exact words. Thus, developing an automatic selective quoting strategy based on content and intent markup could be a good way of putting our work to use. To summarize, we state that content units must be quoted while intent units should not be enclosed within double quotes during the search process. Note that quoting for ensuring exact word ordering is meaningful only for multiword segments, as quoting single word units only differentiates between stemmed and

Anno	Perc. queries improved				Avg. for quoting strategy				Avg. gain over original			
nDCG	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
<i>X</i>	34.150	34.960	24.390	47.970	0.771	0.807	0.542	0.882[†]	0.196	0.176	0.227[†]	0.212
<i>Y</i>	39.840	39.020	22.760	50.410	0.709	0.755	0.552	0.818[†]	0.101	0.105	0.202[†]	0.154
<i>Z</i>	34.960	39.020	18.700	50.410	0.723	0.797	0.531	0.858[†]	0.145	0.157	0.213[†]	0.185
<i>Avg.</i>	34.960	40.650	21.140	52.030	0.772	0.830	0.575	0.890[†]	0.151	0.132	0.195[†]	0.161
<i>X</i>	42.280	38.210	14.630	47.150	0.567	0.589	0.230	0.625[†]	0.176	0.182	0.170	0.184
<i>Y</i>	31.710	30.890	14.630	40.650	0.343	0.370	0.145	0.392[†]	0.102	0.098	0.126	0.109
<i>Z</i>	38.210	46.340	15.450	56.100	0.478	0.528	0.217	0.568[†]	0.126	0.116	0.142[†]	0.128
<i>Avg.</i>	32.520	34.150	8.940	39.020	0.359	0.380	0.112	0.397[†]	0.126	0.105	0.090	0.108

S1: Both content and intent units in quotes; **S2**: Content unit quoted, intent unit unquoted; **S3**: Content unit quoted, intent unit deleted; **S4**: Maximum of S2 and S3. The highest value within each set of columns is marked in **boldface**. Statistical significance (two-tailed paired *t*-test, $p < 0.05$) of the highest value within a set of columns over the next best is shown using a [†]. This is applicable only for the middle and the right sets of columns.

Table 5.11: IR evaluation (nDCG@10) of content-intent labeling using the Bing API.

unstemmed word forms (like `brown` and `browning`). In our previous work (Chapter 3), we had shown with an oracle-based approach that quoting helps improve IR performance, but a deterministic quoting strategy is yet to be discovered. We believe that content-intent labeling is the first step towards such a strategy.

We note here that none of the four state-of-the-art researches that tag queries with content-intent like labels [135, 231, 233, 236] provide an IR-based evaluation for their approaches. Moreover, schemes that do use some sort of query tagging to improve retrieval, do not report results on a single dataset so as to be comparable among each other. Thus, we select Microsoft Bing Web Search, a commercial search engine, as our state-of-the-art baseline, accessible through its API⁴. This provides a very challenging baseline, and if we are able to show IR improvement on a reasonable proportion of queries over the Bing API, our method can be said to have substantial practical significance.

We run experiments with our IR evaluation dataset (Section 3.4) that contains 500 queries (5 to 8 words), a corpus of 13959 documents, and about 30 relevance judgments

⁴<http://datamarket.azure.com/dataset/bing/search>, Accessed 18 May 2014.

per query (0 – 2 scale; three annotators). Most queries (383 out of 500) consist of two segments only (according to our segmentation algorithm), which are labeled for content and intent segments by our method. 123 out of these 383 queries have one intent segment and one content segment (remaining 260 are content-content), which forms our final evaluation set. For each of these 123 content-intent queries, we generate the following query variants: (a) both content and intent segments are in quotes ($c-q\ i-q$), (b) content segment is in quotes and intent segment is unquoted ($c-q\ i-u$), and (c) content segment is in quotes and the intent segment is deleted ($c-q\ i-d$). Among these, $c-q\ i-u$ and $c-q\ i-d$ can be said to be “our” proposed methods as $c-q\ i-q$ can be generated without the tagging step by simply quoting both segments. We subsequently use the Microsoft Bing Search API to search our document collection. Essentially, we use the Bing search API to retrieve the top-10 URLs from the Web for our query versions (three quoting variants and the original query) and then search our corpus for these URLs and their corresponding relevance judgments. Since the original corpus was also constructed using the Bing API, all the documents and most of the corresponding relevance judgments were found in the dataset. Next, we compute nDCG (Equation 3.4) and MAP (Equation 3.7) for each query, and report averaged values in Table 5.11. nDCG is computed after observing the first ten results only, as happens in a typical Web search scenario, and hence we report nDCG@10. For computing MAP, relevance judgment ratings of 2 were treated as “relevant” while ratings of 0 and 1 were considered as “non-relevant”. The results are computed for each of the three annotators (named X , Y and Z) and their mean rating, all of which are available in our dataset. We compute the following three statistics for each quoting variant (represented by the three sets of columns in Table 5.11): (a) percentage of queries on which the variant improves over the original query, (b) mean metric value (nDCG@10 or MAP) for the variant, and (c) the mean metric gain over the original query for improved queries. In addition to the three variants, we compute these values for the column $\text{Max}(c-q\ i-u, c-q\ i-d)$ that represents the better of the two variants $c-q\ i-u$ and $c-q\ i-d$ in terms of the metric value (nDCG@10 or MAP). If this strategy gives the best results among the rest, we can say that content-intent labeling has the *potential* for producing substantial improvement over the original query, even with a very strong baseline.

We make the following important observations from Table 5.11: (a) $c-q\ i-u$ and $c-q\ i-d$ together can improve nDCG for more than 50% queries (64 out of 123 queries

for mean rating) has the best performance, (b) $c-q\ i-d$, and $c-q\ i-u$ and $c-q\ i-d$ together result in higher metric gain (both metrics) over the original query than $c-q\ i-q$, and (c) $c-q\ i-u$ generally has the highest IR performance among the three variants. For case (b), we note that taking $\text{Max}(c-q\ i-u, c-q\ i-d)$ increases the number of improved queries, and hence the mean of $\text{Max}(c-q\ i-u, c-q\ i-d)$ can fall below the mean for $c-q\ i-d$. For both metrics, for a large majority of the cases (17 out of 24 cases), $\text{Max}(c-q\ i-u, c-q\ i-d)$ version achieves the best results, and the gains are often statistically significant (applicable for the second and the third sets of columns, 8 out of 16 cases). These results show that tagging segments as content or intent can be leveraged for good IR performance. It is heartening to see that our deterministic $c-q\ i-u$ variant generally achieves the second best performance for the left and the middle sets of columns of percentage queries improved and mean metric values (i.e., the best among the first three columns of deterministic variants in each set) (13 out of 16 cases). This is a direct validation of the success of our operational definition that intent segments need not match exactly within text of relevant documents. We also see evidence that intent segments are not always “deletable” and while they need not match exactly in document text, or can even be absent, they can be used by the search engine in other different ways. This is apparent from the result that even though the $c-q\ i-d$ variant on its own generally performs the poorest among the three variants (16 out of 16 cases), yet for the queries that it improves upon (21 – 24% on nDCG, 9 – 15% on MAP), the gain is quite substantial. This is seen from the performance of this variant in the third set of columns, where it is usually the best among the four variants (6 out of 8 cases).

5.5 A taxonomy of intent units in Web search queries

Roles of units. In order to better understand the roles of intent units in queries, we went through the list of intent units and several hundreds of queries in which they occur. Our study reveals that intent units in Web search queries can be broadly thought of as performing one of two tasks, namely, *restrict* or *rank*. The *restrict* task is concerned with filtering the pool of relevant documents from which the final results are presented. The *rank* task determines the order in which the final results are displayed. These broad categories can be

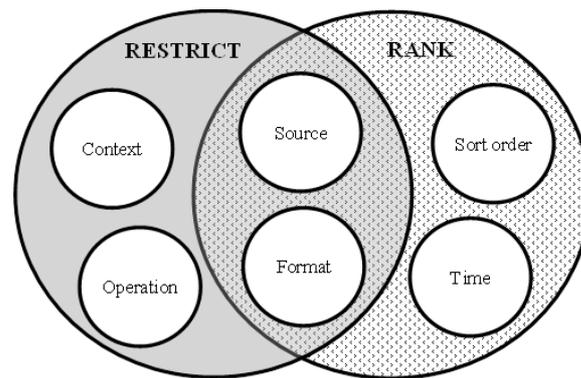


Figure 5.6: A Venn diagram for the intent unit taxonomy.

further subdivided into classes as shown in Figure 5.6 and Tables 5.12, 5.13 and 5.14. In some cases, the distinction between restrict and rank tasks begins to blur, and consequently, the table also presents examples for the *restrict + rank* category.

The Restrict Class. In the *restrict* category, *context specifiers* act as disambiguators for the rest of the query (*book*, *movie*). Similarly, *operation specifiers* are generic action units that specify some action to be performed on or with the content unit(s) (*download*, *install*). They act like an operator with one or more content units as arguments, thus often behaving like unary, binary or multi-nary relations. The intent units in the *other aspects* sub-category mainly specify aspects of particular classes of content (like medicines (*side effects*) and songs (*lyrics*)), in which the user is interested.

The Rank Class. In the *rank* category, *sort order specifiers* indicate that results can be ranked by a parameter of the content unit(s). For example, *near* or *cheap* specifies that results can be ranked in order of some distance or price respectively. *Time specifiers* are used when users have a preference about *when* the pages were published (the *latest news* or *recent updates* about events or products). Most adjectives fall in the rank category, e.g., *free*, *public* and *printable*. These intent units specify the user's preference as to which of the retrieved pages must be ranked higher in the final results' list.

The Intersection Class. The intent units in the intersection of these classes can help in both restrict and rank tasks. For example, *source specifiers* indicate from where the user wants result pages to be retrieved. Real source specifiers are geographical locations (mostly names of countries like *germany* or *australia*). Similarly, virtual source

Context	Operation	Other aspects
book	how to	side effects
movie	what is	benefits
game	where are	reviews
tv show	download	biography
ps2	compare	obituary
soap	difference between	history
windows	buy	applications
scientist	upload	recipe
footballer	install	lyrics
actor	who is	cheats

Table 5.12: Examples of intent units for the *restrict* class.

specifiers indicate online sources (like wikipedia or ebay). *Format specifiers* indicate explicit output formats for the results. They may be direct (file extensions like pdf or mp3) or indirect (photos of and videos of). We propose that these units belong to the *restrict + rank* category because while they try to *restrict* pages to the desired source or type, they also help in the *ranking* of the other results (lower than desired pages). If the desired pages are not available, then the other pages are ranked higher. In either case, the user (generally) still only *prefers* pages of the desired type, and will often look at alternative sources or types if the earlier content was not satisfactory. For example, consider a common source specifier unit such as wikipedia. The user may be only interested in Wikipedia articles (restrict task). Alternatively, the user may just *prefer* a Wikipedia article, but is willing to consider results from other sources as well (rank task).

Discussion. We observe that intent words play very important and diverse roles in Web search queries. Sometimes this distinction of intent from content can become ambiguous. For example, take the query (facebook) (wikipedia), where the user could be looking for the Facebook (content) entry in Wikipedia (intent), or the Wikipedia (content) page on Facebook (intent). Therefore, detection of intent units and understanding their role is very important for IR. A particularly useful scenario for applying our meth-

Sort order	Time	Other preferences
near	latest	online
cheap	recent	free
fast	2012	downloadable
large	new	public
close to	current	exclusive
high-res	last 24 hours	private
shortest	today	black
budget	now	best
popular	last month	printable
best-selling	this week	widescreen

Table 5.13: Examples of intent units for the *rank* class.

ods is *enterprise search*, i.e., searching the entire collection of documents belonging to a particular enterprise (mostly) by its employees. The collection of user intents (and consequently the set of intent units and its distribution) is expected to vary from one enterprise to another. Since additional information such as clickthrough data may not be available (or may be very sparse), often query logs are the only resources for intent analysis in enterprise search. Classification of intent units according to our taxonomy can help in identifying the most important needs within the enterprise. Moreover, our taxonomy can also be used for intent diversification, triggering advertisements in sponsored search, and generating query suggestions. Since the relevance of this taxonomy is mostly application-centric, an *evaluation* of the taxonomy is best conducted through appropriate end-to-end applications by the administrators of the deploying systems.

5.6 Related work

We emphasize that our notion of intent units does not contradict but supports or subsumes much of the related efforts in this area, which use Web documents, query logs and knowl-

Source	Format
wikipedia	pdf
youtube	mp3
espncricinfo	slides
ebay	videos
bestbuy	pictures
facebook	photos
linkedin	images
australia	ppt
india	map
us	torrent

Table 5.14: Examples of intent units for the *mixed* class.

edge bases. One such line of research is on the automatic acquisition of *attributes* of *classes* or *instances* [9, 160, 161]. Our method captures several attributes, like *side effects* (of medicines), *biography* (of important people) and *recipes* (of dishes). However, our technique also detects intent units like *compare* and *how to*, which do not fit in with the current framework of *class-instance-attribute*. Similarities can be observed in the nomenclature of Li [135], where the author states that noun phrase queries are composed of *intent heads* (like *cast*) and *intent modifiers* (like *alice in wonderland*). Intent heads are closely related to attributes and our intent units. Our framework is not limited to noun phrase queries, and can explain other queries like *(how to)\i (meditate)\c*. A framework and taxonomy using *entities* and *intent phrases* have been proposed for understanding name entity queries in Yin and Shah [231] – but our framework is more generic in the sense that it is not restricted to name entity queries only. The motivation of our work is also fundamentally different from the previous studies. Our notion of intent units largely agrees with the term *intent words* [231, 233], proposed for specific domains like *actors*, *musicians*, *cities* and *national parks*. Similar is the case with *modifiers* [236], which are proposed to carry user intent within queries (as opposed to the query *kernel*). Again, our framework applies for all domains of queries and our unsupervised method using co-occurrence statistics can be considered as a low-cost information extraction tech-

nique to detect all categories of such attributes, intent words, intent phrases, intent heads and modifiers.

5.6.1 Intent units as explicit facet indicators

We believe that query intent units mined through our technique are actually *words or segments that the user has included in the query to explicitly indicate his or her intent*, and there is often a one-to-one correspondence between query facets [76] and our intent units. Query facets can be ascribed to the entire query, and include aspects like genre and scope. It is important to note that a segment when behaving as an intent unit can indicate multiple facets at the same time. For example, the unit `mp3` can tell us both that the query is from the *topic* of *music* and that the user has the *objective* of finding a *resource*. Similarly, presence of `imdb` indicates the facets `{topic: movies}` and `{authority sensitivity: yes}` (the latter implying that the query requires an answer from an authoritative source). We believe that intent units can be very useful features for query intent classification, and can deepen our understanding of user intent. Thus, classification of our intent units into various facet classes and using them as features for intent classification are promising directions for future research.

5.7 Conclusions

In this chapter, we have proposed that syntactic units in queries broadly perform two roles – content and intent. While content units define the topics of queries, intents units act as indicators of user intent. We have shown that co-occurrence distributions of units in query logs can be leveraged for differentiating between content and intent units. Our techniques are inspired by their effectiveness in NL text where they can automatically discover function words. We have also shown that automatic labeling of segment roles within queries is possible with reasonable accuracies using simple algorithms based on corpus distributional properties. Results obtained by our generic and lightweight method have been validated by independent evaluations with human annotations and clickthrough data. We have also

shown results where content and intent labeling can be leveraged for improving retrieval performance. A comprehensive classification scheme for mined intent units has been presented, providing readers with a qualitative analysis of the nature of such units. We have proposed that intent units broadly serve two important functions in IR – to *restrict* and *rank* final result pages. Content and intent units can thus be conceptualized as the broad syntactic categories of the query language. In the next and final contributory chapter, we will show how we can objectively quantify this syntactic complexity of Web search queries.

Chapter 6

Understanding Syntactic Complexity of Web Search Queries

6.1 Introduction

Searching information on the World Wide Web by issuing queries to commercial search engines is one of the most common activities engaged in by almost every Web user. The Web has grown extensively over the past two decades, and search engines have kept pace by incorporating progressively smarter algorithms to keep all the information at our fingertips. This co-evolution of the Web and search engines have driven users to formulate more complex and longer queries [170, 209]. Search queries represent a unique mode of interaction between humans and artificial systems, and they differ observably in syntax from that of the parent natural language (NL). This has led researchers to argue that probably queries are acquiring linguistic properties of their own [60, 80, 107, 209]. Arguing from the perspective of the *function* of queries, i.e., communication, and the factors that influence their self-organization, it can be fairly convincingly established that queries are indeed an evolving *linguistic system*. Nevertheless, there is no systematic and comprehensive study of the syntactic properties of Web queries that can convincingly bring out the fact that queries are indeed a “language”. The challenge, of course, is to identify the unique syntactic features of an NL that make it different from any random or artificially generated sequence

of symbols. Fortunately, there are three distinct lines of research that can help address this fundamental question.

One of the oldest statistical characterizations of NL comes from n -gram models that can be used for both generation of utterances or sentences to a certain degree of accuracy, and also for quantifying the predictability (and hence the complexity) of a system of symbols [41]. Such a line of research in the past has been extensively used for analyzing ancient languages like the Indus-script hypothesis [182], studying languages with diverse typological properties [75], and also for understanding non-linguistic systems such as music [62] and the genetic code [143]. Over time, n -gram models have been appropriately generalized or restricted using more sophisticated linguistic features capturing various syntactic and semantic properties (see [24] for a review). Collectively, these models are studied under the broad topic of *Statistical Language Modeling* and extensively used in applications like Automatic Speech Recognition [247], Machine Translation [122], spelling correction [64] and Information Retrieval (IR) [176].

A second approach to characterize a linguistic interactions is to study it from the perspective of the *native speakers' intuition*, which says, to quote Noam Chomsky [49]: “The sentences generated will have to be acceptable to the native speaker”. Though the concept of *native speaker* is debatable and eludes a clear definition [164], in the context of queries it assumes an altogether new dimension.

A third and more recent line of investigation into linguistic systems is through *Complex Network Modeling* of languages, where a language is modeled as a network of *entities* and their *relations* (see [51] for a review). These studies were inspired by similar modeling techniques employed by physicists and biologists, which lead to interesting insights into the systems being modeled. Such studies using network modeling have also revealed some interesting properties of languages [61, 69].

These three lines of investigation are, in fact, complementary, and therefore can be very well used for evaluating each other and getting a more comprehensive picture of a linguistic system. Hence, in this work, we explore the syntactic properties of queries through these three different lenses and cross-validate our findings to come up with interesting conclusions. More specifically, we (1) build n -gram models and also n -term models [211, 230]

from our query log and analyze perplexity (or the predictability) of the models and compare them with that of Standard English, (2) build word co-occurrence networks, the most popular and well-studied network modeling approach for NLS, for the real log and compare the topological properties of the network with those built from artificial logs generated using the n -gram models, and finally, (3) ask ordinary Internet users to rate the acceptability or “real”-ness of the queries generated by the various models. Our study reveals that although queries seem to be more predictable (or less complex) than NL, n -gram models still fall short of generating a rich set of artificial queries. A typical user is able to tell apart a real query from an artificially generated one, even though a trigram-based generative model seems to overfit the data and is capable of confusing the user. The word order in queries seem to be the most important clue helping a user to differentiate between the real and artificially generated queries. Hence, the structure of current Web search queries indicates a linguistic system that has at least a rudimentary word ordering constraint, and several other syntactic and semantic constraints that lie beyond the scope of n -gram and n -term models.

We do not know of any previous systematic study on these aspects of Web search queries, even though features like auto-complete and query suggestions are indirect evidence that predictability of query terms is indeed exploited by Web search engines. On the other hand, this study can not only help in more systematic and principled techniques in relevant applications, but can also have profound impact on the way we view *query understanding* today.

The organization of the rest of this chapter is as follows. First, we describe the generative language models for queries used in this work in Section 6.2. In Section 6.3, we discuss our complex network modeling technique and use the framework to compare the various generative models. Next, in Section 6.4, we present our experimental framework and results for crowdsourcing to measure human intuition of query syntax. We discuss the implications of our results in Section 6.5. We comment on the application of our ideas to synthetic query log generation and evaluation in Section 6.6 and make concluding remarks in Section 6.7.

6.2 Statistical language modeling for queries

The ***n*-gram** language model (LM), which assumes that the probability of the n^{th} word in a sentence depends only on the previous $(n - 1)$ words [41], has been one of the most popularly used generative language models for NL with many applications [66, 100, 122]. Therefore, as the first steps, we evaluate query models based on n -grams (and their variants). An n -gram, in our context, is any continuous sequence of n words w_i from a query. Mathematically, an n -gram LM over sequences of words is defined by a Markov chain of order $(n - 1)$ [141], and the probability of a k -word query is given by:

$$P(w_1 w_2 \dots w_k) = \prod_{i=1 \dots k} P(w_i | w_{i-1} \dots w_{i-n+1}) \quad (6.1)$$

The required probabilities can be initialized through training on the real query log from the relative *counts*

$$P(w_i | w_{i-1} \dots w_{i-n+1}) = \frac{\text{count}(w_i \dots w_{i-n+1})}{\text{count}(w_{i-1} \dots w_{i-n+1})} \quad (6.2)$$

An ***n*-term** [211, 230] is an *unordered set* of n words, all of which occur in a query, but not necessarily next to each other. Since queries have been considered to be bags-of-words in several contexts [176, 195, 198, 207], a systematic exploration of n -terms becomes necessary. Since queries are short (mean length was close to four words for distinct queries in our log), it is not practical to look beyond trigrams (order-3 n -gram model).

6.2.1 Query generation process

We now explain our process of generating artificial queries using n -grams or n -terms. We explore seven generative models for queries, viz. 1-gram, 2-gram, 3-gram, 2-term, 3-term, 2-term-GR and 3-term-GR, where GR refers to “Greedy Reordering” (where the n -grams are reordered in a greedy manner in descending order of their generation probabilities). Example queries generated by each of these models are shown in Table 6.1. Words in query

Table 6.1: Queries generated by different models.

Model	Example queries
1-gram	a dhcp ephemeral detailing map rc2 western pacific kennedy anzac center catering civ integrate you guide
2-gram	create user account on roads access 2003 not working with info party poker table tennis player
3-gram	shaolin kung fu panda a brief history of witchcraft a thousand miles sheet music for webservers
2-term	acer 5310 for flash player adobe acrobat free download windows housing thailand what is cost of housing
3-term	adelaide entertainment explained seating plan anti software virus windows vista adobe photoshop 7 for mac
2-term-GR	flash player for acer 5310 adobe acrobat free download windows what is cost of housing thailand
3-term-GR	adelaide entertainment seating plan explained anti virus software windows vista adobe photoshop 7 for mac

logs were stemmed using the Porter Stemmer [177] prior to the query generation process. In the examples shown in Table 6.1, word stems have been replaced by random unstemmed versions of the word for readability. 1-gram queries typically do not make much sense; they are just some random words thrown in to create a query, albeit maintaining the real word probability distribution. The notion of local coherence immediately becomes clear as one moves to 2-grams. The crossovers from one bigram to the next seem smooth, but as a whole the queries are generally not meaningful. Clearly, the chances of a query being meaningful in its entirety diminishes with increasing query length. The same thing can be observed for trigrams; but since several queries only have three, four or five words (can be generated with only one, two or three trigrams, respectively), such queries are often realistic. For 2-terms and 3-terms, we have similar observations but it is also apparent that the sequencing of the words is not very natural. Another point to note is that such models often contain word repetitions. Both these aspects are fall-outs of the relaxation of the strict ordering constraint. This naturally motivated us to check equivalents of n -term models when query

words were deduplicated and some simple reordering strategy was applied. Queries of GR models appearing better than their unordered counterparts reflect this intuition.

Query generation process. We denote the query length (in words) as L , a random variable that can take integral values between two and ten. *Query length distribution* refers to the probability distribution of L , which is empirically estimated from the log. All the generation models described here are made to follow this distribution. The process for generating artificial queries using n -grams is as follows (the process is exactly analogous for n -terms): first a value l of L is sampled from the query length distribution. Then, an n -gram is first chosen stochastically in proportion to its probability. An extra word is added to the query trying to extend the previous string of $(n - 1)$ words. This new word is chosen probabilistically from all the n -grams which have their first $(n - 1)$ words as the string concerned. This process is continued till the desired query length, i.e., l , is reached. When a query generation process is unable to add a new word to a partially generated query using an n -gram model, then it *backs off* to an $(n - 1)$ -gram generative model to generate the word. Let us understand the process with one of the example queries for 3-grams, a thousand miles sheet music for webservers. First, we randomly sample a query length seven from the real log query length distribution, which implies that the generated query will have seven words. Next, the 3-gram a thousand miles is stochastically sampled from the list of all real trigrams based on its occurrence probability. We now try to extend the query by looking at all 3-grams that start with the 2-gram thousand miles. Using similar stochastic sampling as above, we obtain the trigram thousand miles sheet. Next, a search for 3-grams beginning with miles sheet fails, and hence we *back off* to the 2-gram model and sample for 2-grams starting with sheet, which produces sheet music. We resume our search for trigrams using the rightmost 2-gram and add the words for and webservers in consecutive iterations. No backing off was required in the last two steps. Since we have now reached the desired query length of seven, the query generation process stops.

Greedy reordering (GR). We observe that queries generated by 2-term and 3-term LMs often contain a set of coherent and meaningful words, but their order is not what a human user would normally type. Also, often there are repetitions of the same word in a query. Both of these are outcomes of the way the models are defined. Thus, to refine these models, we deduplicate the query words and perform a *greedy reordering* of the words of

2-term and 3-term models in a query. We call these new models 2-term-GR and 3-term-GR respectively. Reordering is done as follows. After deduplication of the words of the query, we find the local n -gram ($n = 2$ for 2-term and $n = 3$ for 3-term queries) consisting of n words from the query that has the highest n -gram probability. We then extend this new partial query by iteratively choosing a word from the remaining words in the original query, which has the highest local n -gram probability when appended to the last $(n - 1)$ words of the new partial query. If we are unable to extend a partial query because no such local n -grams exist, we back off to $(n - 1)$ -order models until the query can be extended. Instead of a greedy approach, we could have aimed for a globally optimized reordering of the query or a Viterbi search. We leave such reordering strategies as future work.

6.2.2 Measuring model perplexity

Perplexity is one of the most common metrics used for evaluating n -gram systems [19]. It can be intuitively thought of as the *weighted average* number of choices that a random variable can take. Thus, perplexity of an n -gram model tells us that if, on an average, a string of $(n - 1)$ words of the language are known, how many words are likely to occur in the next position. In other words, it tries to model how “perplexed” a user would be in guessing the n^{th} word after seeing a string of $n - 1$ words. A higher perplexity value for a language model thus implies less certainty in the user’s mind about its predictability. The perplexity of a probability distribution $p(x)$ of a random variable X is defined as $2^{H(X)}$, where $H(X)$ is the entropy of X and is given by $H(X) = \sum_{x \in X} p(x) \log_2 p(x)$.

The entropy of a particular $(n - 1)$ -gram is the entropy of the probability distribution over all words that can appear in the n^{th} position given that the first $(n - 1)$ words are fixed. This entropy, raised to the power of two, gives the corresponding perplexity.

6.2.3 Experimental results

Table 6.2 reports the perplexity of the different models for NL and Web search queries (the GR models are reorderings of the corresponding n -gram and n -term models and hence do

Table 6.2: Perplexity and counts of n -grams and n -terms.

Model	NL	Queries	NL	Queries
	(Perplexity)	(Perplexity)	(Counts)	(Counts)
1-gram	1,406.593	6,417.283	0.3M	0.2M
2-gram	193.722	104.337	3.5M	1M
3-gram	17.663	5.430	9.7M	1.1M
2-term	893.851	384.945	48.1M	4.2M
3-term	N.A.*	23.360	N.A.*	24.8M

* Dictionary runs out of memory even with 64 GB of RAM.

not have separate entries in the table). For NL, the corpus used contained 1M randomly sampled sentences from newswire data¹ in 2010. Newswire text was chosen because, in general, they contain cleaner NL sentences than random Web data. For comparability of NL values with queries, we kept the dataset size similar for the latter by randomly sampling 1M queries from our dataset. To preserve the natural frequency distribution, duplicates were not removed from either dataset. The NL text was case-folded and only alphanumeric characters (and whitespace) were retained. The words in both corpora were stemmed using the Porter Stemmer² [177]. Perplexity values for Standard English reported in Brown et al. [42] are obtained from corresponding cross-entropy values and hence are not directly comparable to those in Table 6.2.

6.2.4 Interpretation

It is quite interesting to note that while the perplexity of the unigram model for queries is much higher than that of NLs, the perplexity of bigrams and trigrams show just the opposite trend. The explanation for this surprising trend is as follows. We observed in our data that the rate of encountering a new word queries is much higher (about one per 20 words) than NL (about one per 58 words). Hence, the unigram distribution of queries is more diverse

¹<http://corpora.uni-leipzig.de/download.html>, Accessed 18 May 2014.

²<http://tartarus.org/martin/PorterStemmer/>, Accessed 18 May 2014.

than NL. In fact, queries have a much larger specialized or peripheral vocabulary and very small core vocabulary as compared to NLs. This unique feature makes the perplexity of the unigram model very high for queries.

On the other hand, queries are also repeated and their repetition frequency is known to follow a power-law distribution [170]. In NL, sentences are rarely repeated exactly, except for phrases like `Thank you` and `Good morning`. In our dataset, the number of duplicates (repeated at least once) in NL and queries were found to be 447 and 164185 respectively. Furthermore, the mean sentence length for NL and queries were found to be 18.159 and 3.980 respectively. These two factors play a crucial role in bringing down the bigram and trigram perplexities for queries. One interesting conclusion of these findings on perplexity is that for a random sentence or query, a native speaker (or search engine user in case of queries) will be able to predict a random word present in an NL sentence much more certainly than for a query. On the other hand, if one or more words are shown, it is much easier to predict the rest of the words in a query than in a sentence. This is precisely why an autocomplete feature can work much better for search engines than a word editor. An alternative perspective is as follows: Queries generated using bigrams or trigrams will look much more realistic than sentences generated using the same models.

The perplexities of n -terms are greater than their corresponding n -grams due to the manifold increase in the number of possibilities in the former (Table 6.2). The number of 3-grams is comparable to the number of the 2-grams for queries because of the presence of a significant number of 2-word queries, that do not contribute to 3-gram counts.

6.3 Complex network modeling for queries

Network analysis provides an elegant mathematical framework to study various complex systems [8, 36, 153, 213]. The success of such network-based techniques in the last couple of decades is primarily due to the fact that a network can capture aggregate properties of a system, while considering both local and long range (global) interactions present in a system. Of special interest to us here is the application of network models to linguistics and corpus studies [51, 147]. The most popular and well-studied representation of a language

corpus is the Word Co-occurrence Network (WCN) [33, 50, 61, 69], which we have apply here to study the statistical properties of query logs. In the recent past, such WCNs have also been used for term weighting in IR [34].

6.3.1 Network definition and construction

A WCN for any given text corpus is defined as a network $\mathcal{N} : \langle N, E \rangle$, where N is the set of nodes each labeled by a unique word and E is the set of edges. Two nodes $\{i, j\} \in N$ are connected by an edge $(i, j) \in E$ if and only if i and j “co-occur” in a sentence [33, 69]. Co-occurrence can be defined variously; in this chapter, we define local and global models of co-occurrence as follows.

Local co-occurrence. According to this model of WCN, immediate word neighborhood is considered important and an edge is added between two words if they occur within a distance of two (i.e. separated by zero or one word) in a query.

Global co-occurrence. In this model, an edge is added between two words if they occur within the same query, irrespective of their positions. Thus, a global co-occurrence network will have more edges than a local co-occurrence network.

For both local and global networks, the edges resulting from random collocations are pruned using “restriction” [69] as follows. Let i and j be two distinct words from the corpus. Let p_i, p_j and p_{ij} be the probabilities of occurrence of i, j and the 2-gram $\langle i j \rangle$ (or 2-term $\{i, j\}$), respectively, in the data. Then, in a restricted network, an edge exists if and only if $p_{ij} > p_i p_j$. All networks considered in this study are undirected and unweighted. Figure 6.1 illustrates the concept of WCN by showing the network generated from the toy query log below. Edges pruned due to restriction are shown using dashed lines.

```
samsung focus gprs config
dell laptop extreme gaming config
extreme gaming dell laptop config
buy samsung focus at&t
gprs config at&t samsung focus
```

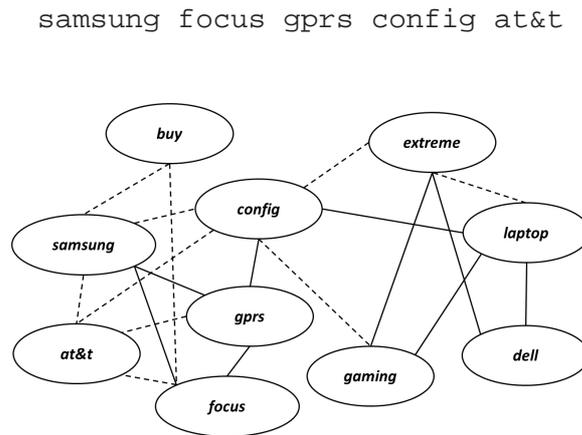


Figure 6.1: Illustration of a WCN for queries.

As per convention [69], all network statistics discussed are computed on the largest connected component (LCC) of the graph. Words in query logs are stemmed before WCNs are built. For LCCs of WCNs generated for $1M$ query samples from our query logs, $|N| \simeq 180,000$ (both for local and global models), while $|E| \simeq 1.5M$ (local) and $|E| \simeq 2.0M$ (global). Thus, these are very sparse networks with average edge density (i.e., probability of having an edge between a random pair of nodes = $|E|/\binom{|N|}{2}$) of the order of 10^{-4} .

6.3.2 Topological properties of WCNs

We now explain the topological properties of word co-occurrence networks that we use for characterizing real and generated query logs, namely, degree distribution, clustering coefficient, average shortest path length, and network motifs.

Degree distribution

The degree of a node in a network is the number of nodes that it is connected to. The degree distribution (DD) of a network is the probability distribution p_k of a node having a degree k . A cumulative degree distribution (CDD) P_k (probability of a node having degree $\geq k$) is more robust to noisy data points and is preferred for visualization. A representative CDD for a query WCN built from $1M$ randomly sampled queries is shown in Figure 6.2

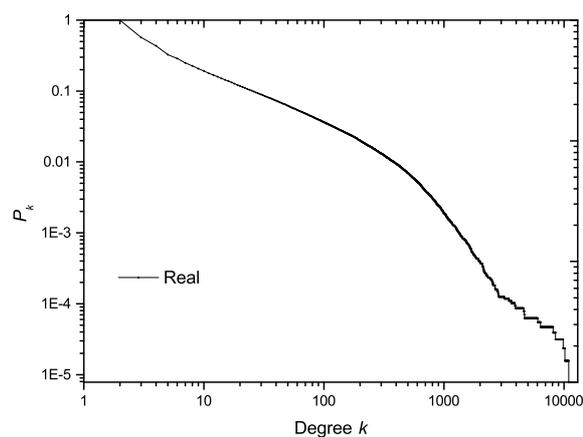


Figure 6.2: Sample CDD for a real query log.

(local model; global model is almost exactly similar) and is found to resemble a two-regime power law. This is also found for NLs [69] and indicates the presence of a *kernel-periphery* structure. A *kernel* is a small subgraph of the network where all nodes have very high degrees. The nodes in the *periphery* have relatively lower degrees than the nodes in the kernel. The majority of the nodes in a query WCN are observed to form very small peripheral clusters which are all connected to the kernel.

Clustering coefficient

Let a node r in the network have k neighbors. Then, $\binom{k}{2}$ edges are possible among its neighbors. The clustering coefficient (CC) of r , CC_r , is the fraction of these edges that actually exist in the network [226]. The $CC_{\mathcal{N}}$ of the entire network \mathcal{N} is defined as the average of CC_r over all $r \in \mathcal{N}$. A high CC indicates that the network consists of one or more dense subgraphs or clusters. The average CCs for the real WCNs (built from $1M$ random queries) are 0.429 (local) and 0.521 (global). The CC for the global network is slightly higher because of the higher edge density. These values are quite high as compared to the CC of an E-R random graph [68,69], which is of the order of its edge density (10^{-4}). CC for similar networks for NL has been reported to be 0.437 [69], and thus show lower density for NL WCNs than those for queries.

Average shortest path length

The shortest path length between a pair of nodes is the minimum number of edges that one must traverse to reach one node from the other. The average shortest path length (ASPL) is defined as the mean of the shortest path lengths between all pairs of nodes in the network that can be reached in a finite number of steps from each other. The ASPL for the WCN built from a $1M$ real query sample is 3.519 (local) and 3.004 (global). The ASPL for the global network is slightly lower because of the higher edge density. These values are quite small for a network with close to 180,000 nodes, and near to the expected ASPL for an E-R random graph of similar size and density (4.253 (local) and 3.830 (global)). ASPL for an E-R random graph is given by $\ln|N|/\ln(\bar{k})$, where \bar{k} is the average degree of the graph, given by $(2 \times |E|)/|N|$ [226]. It has been argued that the low ASPL for similarly constructed NL WCNs (2.67) is an outcome of an optimization of language structure necessary for fast recognition and retrieval of words [69]. As an aside, we also note that a network with high CC, low ASPL and low edge density (all with respect to random graphs) is known as a *small world* [153]. Hence, like social networks and WCN for NLs, WCNs for queries is also a small world.

Network motifs

Network motifs are small subnetworks that are found to occur in significantly higher numbers in real networks than in random networks [23, 115, 116, 149, 200, 227]. For example, cliques with four nodes have an occurrence probability of 10^{-11} in a real WCN, while its expected probability in an E-R random graph [68] with the same number of nodes and edge density, is only 10^{-20} . A Ψ^n -motif is defined as a subgraph of n distinct nodes in the network unique to structural isomorphism. Counting motifs for large graphs is computationally expensive, because beyond Ψ^4 , motifs typically have a very large number of possible isomorphisms. In this study, we only consider connected Ψ^3 and Ψ^4 motifs. Figure 6.3 enumerates all the *connected* Ψ^3 and Ψ^4 motifs. The motifs in this chapter are named following the convention introduced by Biemann et al. [33]. Recently, motif detection has attracted attention as a useful technique to uncover structural design principles of networks in various domains like biochemistry, neurobiology, ecology, and engineering [149, 227].

3-chain (Ψ_1^3)	3-clique (Ψ_2^3)	4-chain (Ψ_1^4)	4-star (Ψ_2^4)	4-loop-out (Ψ_3^4)	4-box (Ψ_4^4)	4-semi-clique (Ψ_5^4)	4-clique (Ψ_6^4)
							
2.941	7.185	4.101	6.057	10.315	7.212	15.463	21.484
<i>motherboard, mercury, element</i>	<i>mercury, element, compound</i>	<i>planet, mercury, motherboard, lan</i>	<i>mercury, messenger, planet, motherboard</i>	<i>mercury, motherboard, lan, card</i>	<i>mercury, mining, iron, element</i>	<i>mercury, water, system, requirements</i>	<i>mercury, water, system, steel</i>

Figure 6.3: Connected Ψ^3 and Ψ^4 motifs with LNMCs from the real log.

Here, we use the algorithm FANMOD [227] to detect Ψ^3 and Ψ^4 motifs. Since motif counts are dependent on the size of a network, they must be suitably normalized by their corresponding expected counts for an E-R random graph model with the same number of nodes and edge density [227]. Since the ratios of the probabilities can be very skewed, we take the natural logarithm of these quantities, which we shall refer to as the *Log Normalized Motif Count* or LNMC. Thus,

$$LNMC(\Psi_i^n) = \log_e \frac{\text{Actual count of } \Psi_i^n}{\text{Expected count of } \Psi_i^n \text{ in E-R graph}} \quad (6.3)$$

where, Ψ_i^n is the i^{th} n -sized motif. For example, following Figure 6.3, Ψ_2^3 and Ψ_3^4 would be the 3-clique and the 4-loop-out respectively. Figure 6.3 (third row) reports the LNMC values for Ψ^3 and Ψ^4 motifs for the real WCN (local co-occurrence model). This vector of eight elements for motifs is referred to as the *motif signature* of the network [33]. These ratios indicate that the probabilities of occurrence of all the connected motifs in a real WCN is several orders of magnitude higher than that in an E-R random graph. Figure 6.3 (last row) also contains real examples of each type of motif from the network. Motifs in WCNs capture semantic relatedness between the words, and certain Ψ^4 motifs like *boxes* and *chains* are representative of semantic concepts like synonymy and polysemy, respectively [33].

6.3.3 Stability of WCNs

The network statistics described above are useful and robust indicators of the structural properties of WCN if and only if the statistics are immune to minor perturbations or random noise in the query log from which they have been constructed. The trends in the statistics

Table 6.3: Network statistics for various network sizes.

Log size	$ N _l$	$ N _g$	$ E _l$	$ E _g$	CC_l	CC_g	$ASPL_l$	$ASPL_g$
W	177,900	177,975	1,526,043	2,089,729	0.429	0.521	3.519	3.320
$W/10$	46,528	46,523	299,496	412,218	0.463	0.555	3.536	3.328
$W/20$	30,398	30,399	169,670	233,932	0.474	0.566	3.567	3.355
$W/50$	17,160	17,162	77,052	106,306	0.493	0.588	3.644	3.410
$W/100$	10,991	10,990	41,270	56,992	0.512	0.611	3.741	3.481

should also remain reasonably fixed when the size of the log is varied. Hence, to analyze the stability of the WCN statistics, we varied the network size by controlling the number of queries from which the network is created. Let W ($= 1M$ here) be the number of sample queries from the entire log (which has $\simeq 10M$ queries) used to build a large WCN. We construct smaller query logs consisting of $W/10$, $W/20$, $W/50$, and $W/100$ queries by random subsampling of the entire log and computing the network statistics. To minimize sampling bias, for each W/s -sized log, the experiments were repeated s times and statistics were averaged over these s instances (not applicable for DD). Table 6.3 reports $|N|$, $|E|$, CC and ASPL for each of these sizes. Figure 6.4 shows the CDD plots for the respective networks (one specific sample from each network size; local co-occurrence model). In this figure, from the top left, we show degree distributions for networks constructed from query logs of size $W/100$, $W/50$, $W/20$, $W/10$ and W . We do not reduce the network size beyond $W/100$ because $W/1,000 \simeq 1,000$ queries which is too small for any reliable network analysis.

The results (mean values over s experiments) in Table 6.3 show that the statistics are extremely robust to network size variation³. Subscripts l and g imply local and global WCNs respectively. Importantly, the standard deviations for the s experiments were found to be very low in all cases, further indicating network stability. Even when the dataset size is increased by two orders of magnitude ($W/100$ through W), the CC and ASPL change only by $\simeq 15 - 16\%$ and $\simeq 5 - 6\%$, respectively. We also note an interesting trend here – both CC and ASPL increase as the network size decreases. This is slightly counterintuitive, because a large network with small ASPL is expected to have a higher CC. However, the trend is explained as follows. As the network grows in size due to increase in number of

³Similar stability for WCNs for NL text has been reported in Biemann et al. [33].

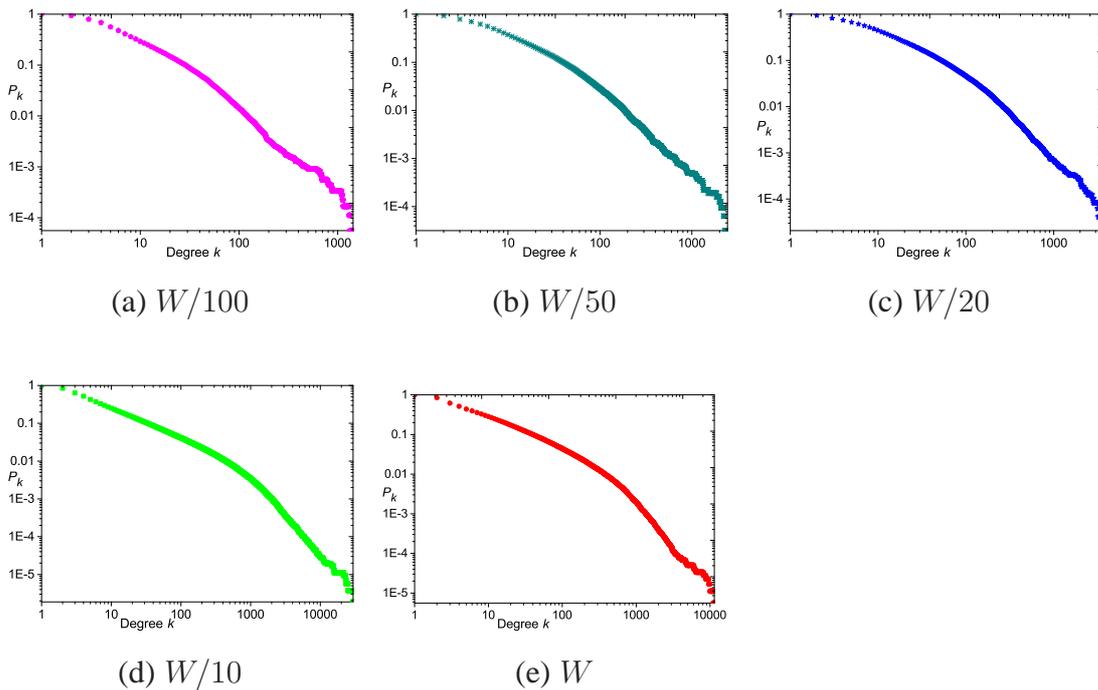


Figure 6.4: Sample CDD at different network sizes.

queries, (a) new nodes join the network with edges connecting to existing nodes; and (b) new edges form between existing nodes. Since new nodes do not immediately have several new connections, the first event decreases the CC and increases the ASPL. The second event decreases the ASPL and increases the CC. However, these rates of increase and decrease of CC and ASPL caused by the two events are not the same. In the case of Web queries, with lots of new and rare words (arising from various proper nouns) continuously joining the network, it is the first event that mostly dominates and is responsible for the drop in CC and increase in ASPL.

Examining the DD, it is evident that the network crystallizes to its final form marked by a two-regime power law, at about $W/10$, which is $100,000 = 0.1M$ queries. The motif statistics for the local network indicate similar stabilization trends and have been shown in Table 6.4. Motif results for the global network also show similar behavior and hence are not reported here. From these results, we infer that for dependable query WCN analysis, one must have at least $0.1M$ queries in their sample.

Table 6.4: Ψ^3 and Ψ^4 Motif signatures at various network sizes (local co-occurrence).

Log size	3-chain	3-clique	4-chain	4-star	4-loop-out	4-box	4-semi-clique	4-clique
W	3.574	7.973	4.637	6.350	10.998	8.003	16.921	23.562
$W/10$	3.063	7.658	4.417	6.261	10.839	7.905	16.375	22.791
$W/20$	2.873	7.172	4.066	5.890	10.251	7.189	15.427	21.460
$W/50$	2.614	6.580	3.587	5.407	9.489	6.215	14.208	19.735
$W/100$	2.371	6.185	3.192	4.915	8.873	5.406	13.296	18.498

Table 6.5: Mean network statistics for the query LMs.

Model	$ N $	$ E $	CC	ASPL	KLD
Real	34,209	242,680	0.623	3.302	0.000
1-gram	28,748	311,955	0.280	2.823	0.349
2-gram	33,257	209,947	0.619	5.011	0.077
3-gram	60,594	292,210	0.591	3.472	0.068
2-term	28,978	227,146	0.630	4.968	0.054
3-term	47,292	249,966	0.634	3.538	0.031
2-term-GR	28,998	230,868	0.628	4.894	0.050
3-term-GR	47,140	249,254	0.632	3.534	0.032

Proximity of local and global co-occurrence networks. An important observation from these experiments on network stability is the relative invariance in the properties of local and global co-occurrence networks for queries. Even though the number of edges in the global network is higher in number, differences in the CC and ASPL values are very small. Moreover, trends observed in degree and motif distributions are also quite similar. Therefore, in subsequent sections, all results will be reported only on local WCNs.

6.3.4 Comparison of real and model-generated query WCNs

We have seen that the statistical properties of the networks are stable as long as the log consists of at least $0.1M$ queries. Therefore, for reliable results, we decided to conduct all our

Table 6.6: Ψ^3 and Ψ^4 Motif signatures for the query LMs.

Model	3-chain	3-clique	4-chain	4-star	4-loop-out	4-box	4-semi-clique	4-clique	M-Diff	M-Sum
Real	2.941	7.185	4.101	6.057	10.315	7.212	15.463	21.484	0.000	74.758
1-gram	2.579	6.428	3.706	4.868	8.987	6.627	13.710	19.072	8.781	65.977
2-gram	2.969	7.369	4.162	6.095	10.464	7.540	15.795	21.954	1.590	76.348
3-gram	2.971	8.049	4.631	6.001	11.383	8.286	17.281	24.153	8.109	82.755
2-term	2.874	7.073	3.989	5.733	9.935	7.210	15.075	21.034	1.835	72.923
3-term	2.907	7.832	4.439	5.859	11.057	7.967	16.832	23.514	6.113	80.407
2-term-GR	2.777	7.060	3.967	5.494	9.884	7.151	15.042	21.020	2.363	72.395
3-term-GR	2.890	7.828	4.431	5.822	11.048	7.954	16.825	23.520	6.132	80.318

The three lowest and the highest values in the M-Diff and M-Sum columns, respectively, are marked in **boldface**.

experiments on logs having $1M$ queries. We sampled the entire real query log to construct 100 subsamples of $1M$ queries each, each subsample *preserving the query length distribution* by words. These will serve as our real logs for all the following experiments. Similarly, we stochastically generated 100 logs, each consisting of $1M$ queries, for each of the seven generative models. We constructed the WCNs for these 8 (real and seven model-generated strategies) $\times 100 = 800$ logs and computed the DD, CC, ASPL and motif signatures for each network. We observed negligible variance in the network statistics across the 100 samples generated from the same model, which further demonstrates the robustness of network modeling. Thus, we report only the average values for $|N|$, $|E|$, CC and ASPL in Table 6.5, and the average LNMC values for the connected Ψ^3 and Ψ^4 motifs in Table 6.6. We note here that the values reported here for the real log do not necessarily match those corresponding to W in the previous section because now the sampling is done preserving the query length distribution by words; the sampling was intentionally random during the experiments on network stability for emphasizing the idea of stability. Henceforth in this text, we will refer to the WCNs generated from the real query logs as *real networks* and the WCNs generated from the model-based query logs as *model-generated networks*.

Since DD cannot be summarized by a single average value, we compute the Kullback-Leibler Divergence (KLD) [124] between the DDs (after applying add-one Laplace smoothing [140]) of the real networks and the DDs of the model-generated networks. These values are also reported in Table 6.5. The smaller the KLD, the closer is the DD of the network to that of the real WCN. Figure 6.5 shows the CDDs for one of the subsamples each from the real and the model-generated networks. The plots have been split into two groups of four

models each, to somewhat mitigate the problem of almost completely overlapping curves.

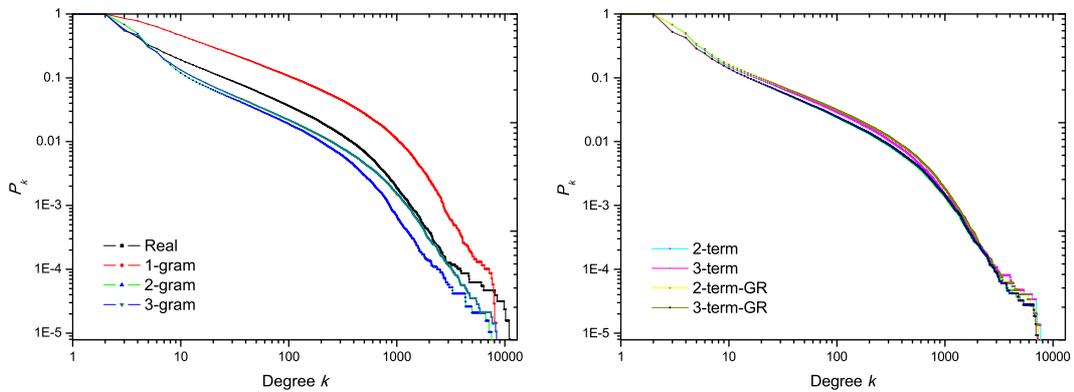


Figure 6.5: Sample CDD for query LMs.

We can observe that even the DD of the 1-gram model is like a two-regime power law (Figure 6.5), which means that DD is the easiest of the network statistics to replicate. For other generated networks, the DD is almost identical to the real network, a fact also apparent from the KLD values in Table 6.5. The 1-gram model also has much lower CC and ASPL. The CC matches for all models where $n \geq 2$, and the ASPL matches only for the 3-gram LM.

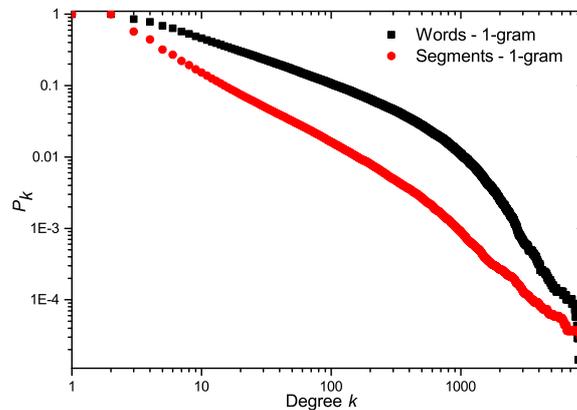


Figure 6.6: (Colour online) Degree distributions for word and segment networks.

Segment co-occurrence networks. We wanted to explore how segment co-occurrence networks would behave in the current setup. As a first step, we generate queries using the 1-gram model but use segment probabilities instead (all the real queries were first segmented

Log	N	E	CC	ASP	KLD
<i>Real</i>	34,209	242,680	0.623	3.302	0.000
1-gram _w	28,748	311,955	0.280	2.823	0.349
1-gram _s	270,687	1,642,641	0.572	3.519	0.523

Table 6.7: Mean network statistics for word and segment models.

using our proposed segmentation algorithm). Also, the query length was computed based on the number of segments, and not words. The same basic properties were computed and the results are present in Table 6.7 and Figure 6.6. We observe that with segment statistics ($1 - gram_s$), even an elementary model like $1 - gram$ has matched up quite closer to real logs than when the corresponding word statistics ($1 - gram_w$) were used (CC and ASPL). This is because segment detection includes sophisticated co-occurrence statistics, and hence such logs already contain several syntactic units present in real logs. It is indicative that using segment statistics could ultimately produce better quality logs. However, the degree distribution shows a higher deviation which needs a more thorough examination. We keep this as a future avenue to explore.

A general observation from the motif signatures is that like NLS [33], it is possible to be close to real networks on Ψ^3 motif counts with 2- and 3-grams. To examine deeper, we computed two aggregate statistics: *M-Diff* – the sum of the absolute differences of the LNMC values of real and the generated networks, and *M-Sum* – the sum of the LNMC values of all the connected motifs. Equations 6.4 and 6.5 show the computations of *M-Diff* and *M-Sum* for a particular LM:

$$M-Diff(LM) = \sum_{k=3}^4 \sum_i |\text{LNMC}(\Psi_i^k)_{Real} - \text{LNMC}(\Psi_i^k)_{LM}| \quad (6.4)$$

$$M-Sum(LM) = \sum_{k=3}^4 \sum_i \text{LNMC}(\Psi_i^k)_{LM} \quad (6.5)$$

It is a well-known fact that the existence of a large number of connected motifs in a network is an indication of its non-randomness [227]. We argue that the larger the sum of

LNMC values for connected motifs, the more structured the network is. However, if this sum exceeds that of the real WCN, it implies that the network is becoming more restrictive than what is natural.

With *M-Diff* values as low as 1.590, 1.835, and 2.363, the 2-gram and 2-term-based models have almost the same motif signatures as the real network. On the other hand, going by the abundance of connected motifs, 3-gram and 3-term-based models seem to be the most restrictive ($>$ Real *M-Sum*). As a supporting evidence, we note that the trigram model has lower perplexity than the unigram or bigram model (Table 6.2). We also note that the n -gram models have motif signatures closer to the real network than the corresponding n -terms. Thus, *relative word ordering in queries is important*.

6.4 User intuition of real queries

One of the important aspects of any NL is the grammatical correctness and coherence of the sentences, which is typically verified through native speakers' judgments [89, 150]. Native speakers can also predict the next word in a sentence given the previous $(n - 1)$ words with a reasonable degree of accuracy [202], which makes them a good point of comparison against n -gram models. Therefore, statistical and network modeling-based analyses of query syntax would not be complete without a *native speaker* evaluation on acceptability of the generated queries. The challenge, however, is to redefine the concept of native speakers in the context of queries, and to design the corresponding query-acceptability task.

If queries are considered as a language, then clearly anybody generating a query can be considered a native speaker of the language. Thus, for our experiments, we deem an average search-engine user as the native speaker of the query language. However, asking a user whether a query is acceptable or not seems quite a meaningless task – any random sequence of keywords could constitute a query that has been issued by a real user, because as such there is no consensus on grammatical constraints on queries. To get around this problem we carefully designed our experiment in the following way:

1. Users were given a triplet having one real query and two generated queries.

2. They were asked to identify the real query in this triplet.
3. The remaining two queries were to be rated on a five-point scale.

To make the comparison meaningful, the three queries shown had some words in common. The selection of query triplets was automated by a program; the program conditionally selected triplets from thousands of query sets if the three chosen queries of a triplet had some words in common. The philosophy behind this evaluation strategy is that if a generated query is sufficiently realistic, the user will have to make a random choice between the generated and the real query. Moreover, the rating points awarded to the queries in the triplet will give us information about the relative quality of the underlying generative models. We did not consider preference judgments [44] for the models as these are useful if one is only interested in the relative performance of the models. Here, on the other hand, we would like to find out the absolute goodness of a model with respect to real queries. This would not be captured through preference-based judgments. Moreover, after selecting the real query, since the users had to *score* the two remaining queries, the ranking within a triplet can be easily inferred.

On a related note, Li et al. [134] describe an experiment where artificial queries were rejected if they were not acceptable to human judges. However, in their setup, new queries were created by string transformation methods from a real query. Users only had to judge if the generated queries had the same intent as the corresponding original ones, and therefore, their experimental framework is not applicable to our problem.

6.4.1 Experimental setup using crowdsourcing

We use crowdsourcing through Amazon's Mechanical Turk⁴ (AMT) for our user experiments. Apart from being a cheap and fast method for gathering large data [11, 45, 84], a Turker (AMT task participant) is expected to be as good as an average search-engine user because AMT experiments are done online and often require one to really conduct Web searches. Hence, crowdsourcing is amenable to our experimental setup. We designed the Human Intelligence Task or HIT (a unit task in AMT) as follows. The seven LMs can

⁴<https://www.mturk.com/mturk/welcome>, Accessed 18 May 2014.

be combined with real queries to create $\binom{7}{2} = 21$ triplets. However, we do not combine *2-term* with *2-term-GR* and *3-term* with *3-term-GR* as they represent the same query words reordered, and we expect the *2-term-GR* and *3-term-GR* models to be generally rated better. This leaves us with 19 combinations of $\{\text{Real query-Model } i\text{-Model } j\}$ triplets. For each of these 19 combinations, we randomly selected 35 triplets such that individual queries had some words in common – making a total of $19 \times 35 = 665$ triplets containing 1,995 queries to be judged. Word stems were replaced by randomly selected surface forms for user readability. To reduce annotator bias, we tightened our guidelines as far as possible, which are specified below. Some solved examples are shown in Figure 6.7.

1. Each question (1, 2, 3, 4, 5) in the datasheet contains three queries – one is issued by a real human user, and the other two are generated by an algorithm.
2. The task is to find the query that is most likely to be issued by a human user and **mark it 5**. For example, `how to play a cd on my computer`.
3. The remaining queries are to be scored on a scale of 0 – 4 according to the following convention.
4. **Mark 4** if you think that the query is generated by an algorithm, but could almost be the real query. For example, queries like `australian currency exchange limit`.
5. **Mark 3** if you think that the query is generated by an algorithm, and it makes sense, but has incorrect grammar or spelling. For example, queries like `i fit to get blood transfusion`.
6. **Mark 2** if you think that the query is generated by an algorithm, and represents incomplete information needs or jumbled units, but could be meaningful if completed or reordered. For example, queries like `ancient rome slaves how did`.
7. **Mark 1** if you think that the query is generated by an algorithm, and parts of the query are coherent, but not as a whole. For example, queries like `creating pdf a file share tab security`.
8. **Mark 0** if you think that the query is generated by an algorithm, and it is totally nonsensical. For example, queries like `and anzac to jungle characters 101`.

Task details are presented in Table 6.8. The HIT consisted of rating all the queries in *five triplets*. Moreover, a set of five annotations was requested for each HIT. Aspects of AMT experiment setups like cost, allowed time for each HIT and task descriptions are crucial to receiving quick and reliable responses. For our research, we followed the general guidelines presented in Alonso and Baeza-Yates [11].

Solved Examples

1.	map weather moreton island all	<input checked="" type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5
	a map victoria australia home kit	<input type="radio"/> 0	<input checked="" type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5
	map of east timor and surrounding islands	<input type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="radio"/> 5
2.	fun relay races for kids	<input type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="radio"/> 5
	for kids fun easter gifts buy where to	<input type="radio"/> 0	<input type="radio"/> 1	<input checked="" type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5
	musical relay races for kids	<input type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input checked="" type="radio"/> 4	<input type="radio"/> 5
3.	white fish and potatoes recipes	<input type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="radio"/> 5
	fish potato big computer child	<input checked="" type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5
	a recipes for cook white fish	<input type="radio"/> 0	<input type="radio"/> 1	<input type="radio"/> 2	<input checked="" type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5

Figure 6.7: Solved examples for annotation task posted on AMT.

6.4.2 Results and observations

The triplets for which the annotation results obtained from AMT were inconsistent in any way (missing rating, and none or multiple 5-point ratings within a triplet) were rejected. These rejected triplets account for the differences between the first and the second columns in Table 6.9. Table 6.9 reports the average rating assigned to a query within a triplet by the five annotators, averaged over all queries from a model (reported under the “Average Rating” column). “#Triplets” count the number of triplets that has the presence of a query from the corresponding LM. The “Real Percentage” column lists the percentage of times a query generated by a model was marked as “Real”. Real queries are detected correctly a large number of times ($\simeq 60\%$). It is notable that among generated queries, those from the 3-gram model were judged as “real” the greatest number of times and have the highest average rating of 3.276. Even though n -term models are poorer than corresponding n -gram models on being judged as real, their average ratings are marginally better than 2-grams. Greedy reordering (GR) is observed to have a markedly positive effect on all the metrics.

In Table 6.10, results are reported in a tournament-like fashion. Ordering of LMs in rows and columns is such that light and dark cells create (approximately) upper and lower triangular matrices. The values are computed from all the triplets that had queries from both models i and j . Cell $[i][j]$ contains the fraction of times model i has won over model j . A dark cell indicates that the row lost significantly to the column (cell value < 0.4).

Table 6.8: AMT experiment details.

Parameter	Details
Task description	Identify a real query hidden among model-generated ones. Then give grades to the remaining queries.
Task keywords	Web search queries, Real and generated queries, Rating queries
No. of triplets in 1 HIT	5
Total no. of triplets	665
Annotations per triplet	5
Alloted assignment time	20 minutes
Actual assignment time	1 minute 24 seconds
Turker qualification	Approval rate > 50 tasks
Turker location	Any
Reward per HIT	\$0.05
Total completion time	7 days

A light cell indicates that the row and the column faired more or less equally well (cell value between 0.4 and 0.6). An unshaded cell indicates that the row won over the column significantly (cell value > 0.6). In this representation, the relative performance of the models becomes evident from the row (left to right, better to worse) or column orderings (top to bottom, better to worse). Not considering real queries which are identified correctly at least 65% of the times against the next best model, all the models have at least one grey cell in their rows (or columns). This indicates that even though the trigram model competes the best against real queries, it is not the best by a very big margin. It is closely followed by the 2-term-GR model. The models in the middle zone are also quite comparable in their performance levels.

Inter-annotator agreement (IAA). In AMT, since a single Turker need not complete all annotations of the entire dataset, conventional ideas of IAA are not applicable. However, the average standard deviation for ratings from five annotators for a particular query is found to be 1.032. Given that the overall rating was to be done on a 6-point scale (0 – 4 and “Real” ratings), an average deviation of one point is within acceptable limits for IAA.

Table 6.9: A summary of absolute ratings obtained through AMT.

Model	#Total Triplets	#Consistent Triplets	Judged “Real”	“Real” Percentage	Average Rating
Real	665	630	380	60.317	4.046
1-gram	210	197	19	9.645	2.406
2-gram	210	204	41	20.098	2.833
3-gram	210	210	59	28.095	3.276
2-term	175	166	30	18.072	2.880
3-term	175	160	25	15.625	2.875
2-term-GR	175	158	35	22.152	3.076
3-term-GR	175	172	38	22.093	3.163

The two highest values in the last two columns are marked in **boldface**.

6.4.3 Interpretation

These results provide us with the following interesting insights: (a) If any string was equally acceptable, real queries would get a “Real” rating only 33.3% of the time by random chance. The fact that real queries get the “Real” rating about 60% of the time implies that users already have a notion of queries being *well-formed*, i.e., the *acceptability* of queries; (b) Trigram generated queries can confuse the user about 28% of the time. In contrast, for NL, speakers can easily identify trigram generated sentences, which are locally readable, but semantically incoherent [33]. This shows that trigrams capture a lot more information than bigrams and probably overfit the data; (c) 2-term and 3-term queries getting lower “Real” percentage scores than 2-gram and 3-gram queries implies that word ordering provides vital clues to the users. To further confirm this hypothesis, we note that for the reordered models (GR), the mean rating jumped from 2.880 to 3.076 (2-terms) and 2.875 to 3.163 (3-terms) from the corresponding n -gram models; (d) Bigrams received a higher “Real” percentage value but a lower average rating than the 2-term and the 3-term models (and also the corresponding GR models). This is because bigrams generate very realistic queries at times, especially when the query length is small, but meaningless ones on other occasions. This is supported by the observation that the standard deviation of the ratings for

Table 6.10: A summary of relative ratings obtained through AMT.

Model	Real	3-gram	2-term-GR	2-term	3-term	3-term-GR	2-gram	1-gram
Real	<i>X</i>	0.655	0.730	0.741	0.752	0.702	0.708	0.832
3-gram	0.345	<i>X</i>	0.520	0.559	0.613	0.514	0.548	0.862
2-term-GR	0.270	0.480	<i>X</i>	<i>X</i>	0.560	0.606	0.688	0.455
2-term	0.260	0.441	<i>X</i>	<i>X</i>	0.500	0.520	0.519	0.762
3-term	0.248	0.387	0.440	0.500	<i>X</i>	<i>X</i>	0.533	0.704
3-term-GR	0.298	0.486	0.394	0.480	<i>X</i>	<i>X</i>	0.444	0.846
2-gram	0.292	0.452	0.313	0.482	0.467	0.556	<i>X</i>	0.423
1-gram	0.169	0.138	0.546	0.238	0.296	0.154	0.577	<i>X</i>

bigrams is 1.476, while it is lower for 2-terms and 3-terms (1.334 and 1.302 respectively); and, (e) 2-term and 3-term getting almost exactly similar average ratings further emphasizes the importance of word ordering. 3-terms are expected to generate semantically more coherent queries, but an obvious lack of ordering hinders their acceptability to Web users.

6.5 Discussion

We now discuss some of the inferences that we draw from the research presented so far in this chapter.

Motif analysis is insightful. Biemann et al. [33] observe that *box* motifs in NL (Figure 6.3) often occur due to *synonymy*, where the pair of diagonally opposite nodes, which are disconnected, are typically synonyms of each other in a broad sense. This is because synonyms will rarely co-occur in the same query, thus leading to an absence of connecting edges in the WCN. We observe that *box* motifs in query logs typically occur with two similar entities (e.g., *titanic* and *spiderman*, both movies) forming a pair of disconnected nodes and two attributes (e.g., *mp3* and *cast*) forming the other pair of disconnected nodes. The other common reason for *box* motifs is spelling mistakes or spelling variations (like *pituitary* and *pitutiary*) at two opposite ends, and related words like *hormone* and *tumor* forming the other two opposite ends of the *box* motif. We also

observe *star* motifs in query logs for a content unit (e.g., `titanic`) in the centre and three intent units (e.g., `cast`, `mp3` and `review`) connected to it. Thus, motifs occur due to syntactic and distributional constraints in the linguistic system, and hence, they are relatively harder to capture through a generative model.

Trigrams overfit the data. User experiments show that trigram models generate quite realistic queries and have surprisingly low perplexity. This is not surprising because the average length of a query being only four, a large number of queries will be generated with only one or two trigrams. This will effectively generate only queries that have been frequently seen in the training query log.

There is scope for better generative models. Since trigram models overfit and bigram models fall short of generating good individual queries, realistic queries can only be generated using more sophisticated models that can capture the structural constraints of queries both at syntactic and semantic levels. Since motif analysis indicates the importance of content-intent relationships in queries, we believe that a better quantification of the distribution of these relationships can lead to improved generative models for queries.

Relative word ordering is important. Researchers in the past have criticized the bag-of-words model for queries [55, 174, 245]. Our analysis strengthens earlier findings by showing the importance of word ordering constraints in queries, as the bag-of-words model-based query generation (using the n -term model) is shown to be inadequate in both network and user experiments.

There is a cognitive model for queries. Finally, it is interesting to note that users, or as we can say, the native speakers of the language of queries, are indeed able to differentiate real queries from artificially generated ones. This shows that an average user has already internalized a cognitive model for queries. Further probing of this cognitive model through psycholinguistic experiments would be an interesting exercise that can provide interesting insights into not only how query syntax is organized but also how a new language might evolve and automatically acquire a syntax of its own.

Finally, we do not know of prior work that takes a holistic approach towards the analysis of the syntactic complexity of Web queries from the first principles. Previous works related

to statistical and network analysis of queries have been reported in the respective sections. Nevertheless, there are two lines of research pertaining to the syntactic analysis of queries.

Linguistic analysis of queries. First, linguistic analysis and annotation of queries at the level of segmentation or chunking [82] and parts-of-speech tagging [22, 25, 26, 72] have been important directions of research since the early 2000s [107]. While these studies reveal interesting syntactic properties and trends, such as more than 70% of the query terms are nouns [22] and NL question queries are on rise [166], they are based on the fundamental assumption that queries issued in a certain language, say English, will borrow grammatical artefacts of that language (i.e., nouns, verbs, noun and verb phrases, etc.). This assumption is biased because a noun in English is called a noun because it follows a particular syntactic distribution; it is quite unlikely that the same word will behave as a noun in a query either from the point of statistical distribution or its cognitive interpretation by the users. Thus, if queries are to be understood linguistically, they should be analyzed from the first principles rather than superimposing the grammatical syntax of NLS and thereby masking their true syntactic properties. Such linguistic analysis can still be useful for practical applications, but they cannot tell us much about the true syntax of Web search queries.

Entities and intents. The second line of research, which we believe is more promising, is the analysis of queries in terms of user intents. Such studies have looked into queries from various perspectives and have come up with various concepts such as entities and attributes [9, 102, 155, 160, 184], kernel-objects and modifiers [236] and query facets [76, 154], to factor the parts of a query and place it within a taxonomy of semantic or syntactic patterns. While it is not possible to review all these studies here, a closer look at the actual network motifs of the WCN for real queries reveal interesting synergy between the concept of *intent words* [138,232] (also called modifiers or attributes) and *content words* (or entities or kernel-objects) which is worth mentioning here.

6.6 Synthetic Web search queries

Web search query logs can be used for developing a large number of important applications, like query recommendation [16], entity extraction [102] and query segmentation (proposed

research). There has been a good amount of research on query log analysis in the last decade. However, almost all of academia is deprived of this extremely potent resource. This is because commercial Web search engines (e.g., Bing, Google and Yahoo!) restrict public access to these logs to protect the privacy of millions of users. In fact, there were serious legal issues when AOL released massive amounts of query log data, back in 2006⁵. In the last five years, 71% of all papers published on Web search query analysis in the top-tier IR conferences – *WSDM*, *WWW*, *SIGIR*, *CIKM* and *ECIR* had at least one author from the search industry. Fig. 6.8 shows a break-up across years 2007 to 2011. The papers which did not have any author from the industry either used TREC data, AOL query logs (still available on the Internet, unofficially), the Microsoft 2006 RFP dataset, or proposed theoretic models that did not need not a real log for evaluation. However, TREC Web Track ad hoc task queries⁶ (topics) are typically around 50 in number, and is not meant for query *log* analysis. Use of the AOL queries is no longer legally permitted and access to Microsoft’s RFP dataset⁷ was granted only to participants of the WSDM 2009 workshop on Web Search and Click Data (WSCD ’09) and is thus not publicly available.

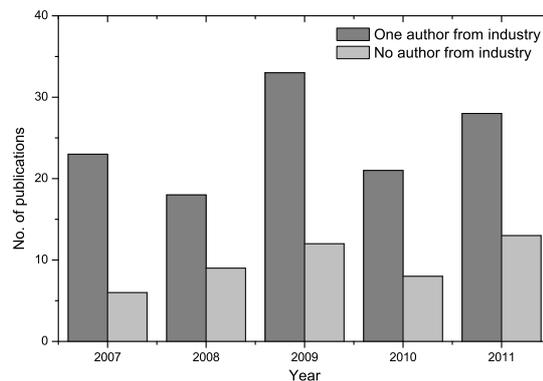


Figure 6.8: Industry authors in papers on query log analysis.

Creation of large synthetic query logs is one of the best possible ways to solve this data scarcity problem, because synthetic logs can be publicly shared without any breach of privacy. However, the main challenge involving synthetic query logs is not the generation, but rather the evaluation of the quality of the generated queries. The simplest method may be to get it checked and cleaned by human annotators. However, it is practically infeasible

⁵http://en.wikipedia.org/wiki/AOL_search_data_leak

⁶<http://goo.gl/9o1sD>

⁷<http://goo.gl/5b7sQ>

to get a query log consisting of millions of queries checked manually. In fact, there is an even more subtle and basic problem with this approach. Given the diverse information needs and idiosyncrasies in query formulations of Web users, it is extremely hard for a single or a handful of humans to say whether any given sequence of words is likely to be a real Web query. An alternative automatic approach to evaluation of synthetic query logs could be to look at the number of search results returned by a query, presuming that more “real” queries will return more search results. However, it also turns out to be a bad indicator; bizarre queries like `world cup football lambda calculus` can return similar or even many more search results than perfectly valid ones like `summer internship mpi 2015` (the former has 20.5M hits and the latter around 13.4M hits on Bing US, 09 November 2014).

Furthermore, beyond ensuring that individual queries are realistic, a synthetic log also needs to faithfully replicate the aggregate properties of the real log. Real logs always possess holistic properties which are typical of the information needs of the people of the time and geographical region from which the queries are issued. Hence, automatic evaluation of large synthetic query logs is an important and difficult problem that has hardly received any attention till date. Even though there have been related attempts at question generation in natural language [89, 150], as far as we know, there has only been one major contribution in the area of synthetic Web queries – the research presented by Li et al. [134]. Their synthetic query log⁸ (named *QRU-1*) contains 2,030 queries that were generated by applying a string transformation method on 100 base queries. The base queries were sampled from the query set used for the TREC 2009 and 2010 Web Tracks. As a guarantee of goodness, they report that 70% of these queries were actually found in a separate log from Bing. Even though the quality of the generated queries were good, this is too small a log to be useful for practical applications like attribute extraction [159].

Through the concepts presented in this chapter, we are able to make the first attempt towards complete automatic evaluation of synthetic query logs using concepts from *complex network theory*, and independently verify our evaluation scheme through crowdsourced manual labeling (intrinsic evaluation). In light of the present state-of-the-art in artificial query log evaluation and generation, the contributions of this work are: (a) formulating the

⁸<http://goo.gl/v301S>

problem of evaluation of synthetic query logs and proposing a framework for the same; (b) application of complex network theory to modeling and understanding the structure of real and synthetic query logs; and (c) proposing a novel crowdsourcing-based approach to manual evaluation of synthetic queries. However, as discussed in our results, since the various network statistics predict slightly different orderings of the synthetic models in terms of their closeness to the real network, it is far from obvious which of these statistics are better predictors of the quality and how they can be combined. Moreover, the similarity values for the network statistics as well as the AMT and segmentation results have very different scales, and therefore, cannot be directly compared or combined. Moreover, generating synthetic queries is practical only when we have certain application(s) in mind, and different applications may impose very different requirements and restrictions on the properties of the synthetic logs.

We believe that there are several other such interesting questions that can be investigated and new generative models can be developed by systematically optimizing the match of the network statistics to that of the real WCNs. We believe that our work on synthetic queries can act as an ideal foundation for a potent area of research in query analysis.

6.7 Conclusions

In this chapter, we have tried to understand the syntactic complexity of Web search queries, a distinct mode of interaction between man and man-made systems. We have adopted a three-pronged approach: applying statistical language models (using n -grams and n -terms), asking native speakers (Web search users) and using complex network modeling (with WCNs). Our results underline the necessity of using multiple independent perspectives. Having entropy or perplexity similar to or lower than NL need not, by itself, be indicative of an underlying language system [210]. Network analysis shows bigrams to be within striking distance of replicating real log syntax at a corpus-level. However, when native speakers are consulted, individual queries generated by trigrams are found to be much more acceptable than those by bigrams. Only a combined approach is successful in bringing out the complete picture of n -gram-based statistics being inadequate, and the need for a language model that imbibes syntactic constraints specific to Web search queries. More

interestingly, in both network and user experiments, a common behavior emerges: the results are distinct both from scenarios that assume queries being random word sequences or following the syntactic constraints of the parent natural language.

Chapter 7

Conclusions and Future Work

In this final chapter, we summarize the main contributions of the thesis (Section 7.1) and take a stock of our achievements vis-à-vis the objectives set up in the introductory chapter. Finally, we wrap up by pointing out some of the possible future directions of research that have been opened up by this thesis (Section 7.2).

7.1 Summary of the contributions

In this thesis, we have made the following contributions: (i) we have developed and evaluated an unsupervised flat query segmentation algorithm that uses query logs and Wikipedia titles; (ii) we have developed and evaluated an unsupervised nested query segmentation algorithm that uses only query logs; (iii) we have developed an unsupervised technique for labeling content and intent units in queries; and (iv) we have proposed a framework for the quantification of the syntactic complexity of search queries. Thus we find that the objectives that we had set out in the Introduction (Section 1.6) have been largely achieved. We recapitulate the contributions in the rest of this section.

Development and evaluation of unsupervised flat query segmentation algorithm

In this work, we have developed a query segmentation algorithm that uses only query logs,

which has been subsequently enhanced with Wikipedia titles. We have proposed an IR-based framework to evaluate our algorithm and to compare its performance with the state-of-the-art. We have also presented an idea for using POS patterns in query logs for enriching query segmentation. The contributions of this study can be summarized as follows:

1. Earlier measures of word association generally rely on some form of unigram statistics, which can often be misleading. We have proposed a novel method for scoring word associations based only on the queries that contain all the words of the candidate multiword expression, and assign the significance score by comparing occurrences in the *correct* order vis-à-vis *any* order.

2. While traditional query segmentation algorithms have used resources like Web page content, search result snippets and clickthrough information, our algorithm relies primarily on query logs, thus discovering syntactic structure unique to queries. Wikipedia titles have been used only to detect evidence of rare named entities.

3. Past approaches to query segmentation have been evaluated against manual annotations, which is based on the flawed assumption that humans “know” the “correct” segmentation for a query. We have challenged this assumption on the grounds that the end-user of query segmentation is the search engine. In this regard, we have developed the first IR-based evaluation framework for query segmentation that uses only the standard resources required for any IR-system evaluation – test queries, a document pool and corresponding human relevance judgments. The proposed algorithm has been shown to outperform the state-of-the-art in the IR evaluation setup.

4. We have shown promising initial results with an idea for enhancing query segmentation using POS patterns. Segments vital from an IR perspective may have low statistical evidence and hence may be missed by traditional algorithms during the lexicon building phase. Our novel idea learns POS sequences from frequent word patterns and uses them to pull segments with such patterns into the lexicon, even if they have low occurrence probabilities in the log. Such a method has been shown to improve IR performance. Interestingly, it bridges the conceptual gap between the analogous processes of query segmentation and NL chunking.

Development and evaluation of unsupervised nested query segmentation algorithm

We have emphasized the importance of using nested or hierarchical query segmentation, which provides a richer syntactic representation of a query. We have developed an unsupervised algorithm using only query logs for inferring the nested representation of a query. We have shown how such a representation can be vital in ranking documents in response to slightly longer queries. The contributions of this study can be summarized as follows:

1. We have proposed an unsupervised algorithm for nested query segmentation that uses only query logs. Our algorithm uses simple low-order n -gram statistics to arrive at the complete hierarchical partitioning.
2. We have proposed the first deterministic approach to leverage nested query segmentation for improving document ranking. To this end, we have shown how the distances in the nested segmentation tree for a query can be used to normalize document distances for matched query terms, and to subsequently improve the ranking produced by issuing an unsegmented query to the search engine.

Proposal of framework for unsupervised role induction for query segments

In this study, we have proposed that all query segments can be broadly classified as content or intent, where content units act as topics for queries while intent units are markers of explicit user intent. Content and intent units can be effectively labeled within queries using corpus distributional properties of each class. The contributions of this study can be summarized as follows:

1. An unsupervised framework for labeling content and intent units in the context of individual queries has been shown to achieve reasonable levels of precision and recall. As with other parts of this thesis, the algorithm uses only query logs to perform the labeling. The novelty of the simple labeling algorithm can be attributed to the use of co-occurrence entropy, which is the entropy of the co-occurrence distribution associated with each unit. Co-occurrence statistics are shown to be effective discriminators of content and intent units in queries, as well as of content and function words in NL.
2. We have proposed a new formulation for modeling overlaps in clicked URL sets.

Using this formulation, we have shown how to evaluate content-intent labeling without relying on expensive manual annotations. The two evaluation strategies are shown to produce highly similar results, emphasizing the feasibility of using click data as an alternative resource of labeling evaluation.

3. We have shown how content-intent labeling can be used for improving IR performance through simple experiments based on matching query segments in documents. Previous works that try to tag units along these lines do not provide a means of applying such labeling to improve IR. Our experiments are based on our operational definitions that content segments have to be matched exactly for documents to be relevant, and intent units need not be present in the document text. While such experiments do show direct IR benefits of our labeling, we believe that knowledge of intent units can be leveraged in much more intelligent ways to improve result quality than simply relaxing the exact matching constraint.

4. We have provided a principled taxonomy of intent units, based on the likely relationships between content and intent units. We have proposed that intent units in queries broadly serve to restrict or rank result pages containing content units. We believe that the proposed taxonomy can have important use cases in semantic search.

5. Finally, the concept of content and intent units, as proposed by us, are not restricted to specific domains or categories of queries, and provides an overarching framework for consolidating several allied lines of research in this area.

Proposal of framework for measuring the syntactic complexity of search queries

In this study, we have proposed a holistic framework for measuring the syntactic complexity of search queries. We have used word co-occurrence networks and human judgments collected through crowdsourcing as independent evaluations of the quality of model-generated query logs, and have shown that queries are more complex than the commonly assumed bag-of-words model, but is beyond what simple n -grams can capture. The contributions of this study can be summarized as follows:

1. While it may seem to be a common perception that queries are growing in complexity, a precise quantification of this complexity had been lacking. Our framework provides

a way to objectively measure the syntactic complexity of a query generative model, and to meaningfully compare such complexities of real and artificial query logs.

2. Our framework is holistic in the sense that it provides both macroscopic and microscopic perspectives on the quality of generated query logs. Specifically, while advanced properties of word co-occurrence networks like motif signatures represent corpus-level aggregates, judgments of average Web users on individual queries represent “native speaker”-intuition for the query language.

3. Our cleverly designed experiment using query triplets for assessing the goodness of a generative model as perceived by humans, bypasses the difficulty of directly asking a user whether a generated sequence of words is a meaningful query. As our final results show, average Web users are able to identify the real query hidden among two generated ones about 60% of the time, implying that searchers do have a cognitive model of the acceptability of a query.

Datasets developed as part of this research

The following datasets have been developed as part of the present research and made publicly available:

1. Our dataset for evaluating flat query segmentation comprises of 500 Bing Australia queries (relatively rarer queries with query frequency between five and fifteen in the original Bing Australia log of May 2010), a list of numbered Web URLs with text content, relevance judgment sets (*qrels*) for each query, query segmentations according to four algorithms and three human annotators, and the best quoted query versions (as explored through brute force). This dataset is available at <http://cse.iitkgp.ac.in/resgrp/cnerg/qa/querysegmentation.html>.

2. Our dataset for evaluating nested query segmentation includes the query sets of SGCL12 and TREC-WT (Chapter 4), and the 16 nested segmentation variants for each of these two query sets. The code and executables for generating these nested segmentations, as well as evaluating them in our IR-based setup, are also being shared. This dataset is available at <http://cse.iitkgp.ac.in/resgrp/cnerg/qa/nestedsegmentation.html>.

7.2 Directions of future work

In this final section, we discuss few of the many possible directions of future work that have been opened up by this thesis.

1. In Chapter 3, a hypothetical oracle has been shown to be quite useful. But we realize that it will be a much bigger contribution to the community if we could implement a context-aware oracle that can actually tell the search engine which quoted version of a segmented query should be chosen at runtime. Also, the best strategy for combining word association scores with POS pattern counts needs further exploration. It seems that PTB or UTS tagsets are complementary to the Bie-S tagset, and it would therefore be useful to develop techniques that combine these two approaches and exploit the benefits of both. It would be also interesting to study which kind of queries benefit maximally from POS-enhanced segmentation and whether they can be automatically identified. We believe that there is a huge potential for unsupervised POS induction in query understanding and representation that has not yet been leveraged.

3. In Chapter 4, it could be useful from an IR perspective to assign weights to term pairs before their tree or query distance is considered. The nesting algorithm and the allied re-ranking strategy can be improved through more sophisticated data-driven approaches and NLP techniques. In fact, nested query segmentation can be viewed as the first step towards query parsing, and can lead to a generalized query grammar. We believe that our findings can make a major impact on query understanding if effort is appropriately channelized along these avenues.

4. In Chapter 5, a more principled way of combining our different features for computing the intent-ness score is an important issue to be addressed. But more generally, the research in this chapter opens up the following broad areas for future work: (a) Seamless integration of intelligent techniques into search systems that allow for special treatment of intent units to serve better pages; and (b) Development of automatic classifiers for assigning detected intent units to their respective categories. Like all aspects of semantic search, problems of vagueness and evaluation pose stiff challenges in these directions. Our research attempts to be a stepping stone in pinning down such difficulties to focused areas

and making them addressable by concerted efforts from the community.

5. In Chapter 6, the crowdsourcing experiments throw up several open questions that can be verified with more detailed study. For example, it would be interesting to know whether searchers can identify cases when there are multiple real queries in the triplets, or when there are none. But more generally, through this research, we highlight the scope for a more realistic query generation model that imbibes more information than mere n -gram probabilities. One of the ways to approach an improved model would be to incorporate constraints based on content and intent units into the generative process.

7.3 Final words

Existence of both shallow and deep syntactic constraints as detected through flat and nested query segmentation, presence of broad syntactic categories of content and intent, and evidence of syntactic complexity beyond simple n -grams – all provide support in favor of our original hypothesis of queries evolving into a language of their own. However, we do not claim that this research is complete by itself. Rather, it is only the first step towards a more holistic goal – when queries, communicating information needs of millions of users, can be established to be an independent language system from all the three aspects – structure, function and dynamics. Our research paves the way for answering a bigger question: are queries evolving to resemble their parent natural language or diverging away from it?

At the time of writing this thesis, the landscape of search is undergoing a great shift in paradigm. The traditional presentation style of the “ten blue links” as results is increasingly being enriched with direct answers, structured knowledge representations on entities, and specialized integrations of non-textual content types like maps, images and videos. The process of harnessing the wealth of relevant content on social media has just begun. While newer and more complex information needs are being created every moment, increased use of the query auto-completion feature potentially tries to decrease the number of distinct queries. All these factors will influence the dynamics of Web search queries, and will play a big role in shaping their future.

Bibliography

- [1] S. Abney. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-Based Parsing*, volume 44 of *Studies in Linguistics and Philosophy*, pages 257–278. Springer Netherlands, 1992.
- [2] S. Abney. Prosodic structure, performance structure and phrase structure, 1992.
- [3] S. P. Abney. *Parsing by chunks*. Springer, 1992.
- [4] G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards rich query interpretation: Walking back and forth for mining query templates. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 1–10, New York, NY, USA, 2010. ACM.
- [5] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06*, New York, NY, USA, 2006. ACM.
- [6] E. Agichtein, R. W. White, S. T. Dumais, and P. N. Bennet. Search, interrupted: Understanding and predicting search task continuation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, SIGIR '12*, pages 315–324, New York, NY, USA, 2012. ACM.
- [7] A. Aizawa. An information-theoretic perspective of TF-IDF measures. *Information Processing and Management*, 39(1):45 – 65, 2003.
- [8] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
- [9] E. Alfonseca, M. Paşca, and E. Robledo-Arnuncio. Acquisition of instance attributes via labeled and related instances. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, New York, NY, USA, 2010. ACM.

- [10] J. Allan and H. Raghavan. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 307–314, New York, NY, USA, 2002. ACM.
- [11] O. Alonso and R. Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *ECIR '11*, 2011.
- [12] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In S. McDonald and J. Tait, editors, *Advances in Information Retrieval*, volume 2997 of *Lecture Notes in Computer Science*, pages 127–137. Springer Berlin Heidelberg, 2004.
- [13] A. Ashkan, C. L. A. Clarke, E. Agichtein, and Q. Guo. Classifying and characterizing query intent. In M. Boughanem, C. Berrut, J. Mothe, and C. Soule-Dupuy, editors, *Advances in Information Retrieval*, volume 5478 of *Lecture Notes in Computer Science*, pages 578–586. Springer, Berlin, Heidelberg, 2009.
- [14] L. Azzopardi, M. de Rijke, and K. Balog. Building Simulated Queries for Known-item Topics: An Analysis Using Six European Languages. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 455–462, New York, NY, USA, 2007. ACM.
- [15] R. Baeza-Yates, L. Calderón-Benavides, and C. González-Caro. The intention behind web queries. In F. Crestani, P. Ferragina, and M. Sanderson, editors, *String Processing and Information Retrieval*, volume 4209 of *Lecture Notes in Computer Science*, pages 98–109. Springer Berlin Heidelberg, 2006.
- [16] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology, EDBT 2004 Workshops*, pages 588–596. Springer, 2004.
- [17] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [18] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 79–85, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.
- [19] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 5(2), 1983.

- [20] J. Bai, Y. Chang, H. Cui, Z. Zheng, G. Sun, and X. Li. Investigation of partial query proximity in web search. In *Proceedings of the 17th international conference on World Wide Web, WWW '08*, New York, NY, USA, 2008. ACM.
- [21] P. Bailey, N. Craswell, I. Soboroff, P. Thomas, A. P. de Vries, and E. Yilmaz. Relevance assessment: are judges exchangeable and does it matter. In *SIGIR '08*. ACM, 2008.
- [22] C. Barr, R. Jones, and M. Regelson. The linguistic structure of English web-search queries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08*, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
- [23] V. Batagelj and A. Mrvar. Pajek: Analysis and visualization of large networks. In *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002.
- [24] J. Bellegarda. Statistical language model adaptation: Review and perspectives. *Speech communication*, 42(1), 2004.
- [25] M. Bendersky, W. Croft, and D. Smith. Structural annotation of search queries using pseudo-relevance feedback. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010.
- [26] M. Bendersky, W. Croft, and D. Smith. Joint annotation of search queries. In *Proc. of ACL*, 2011.
- [27] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08*, pages 491–498, New York, NY, USA, 2008. ACM.
- [28] M. Bendersky, W. B. Croft, and D. A. Smith. Two-stage query segmentation for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09*, New York, NY, USA, 2009. ACM.
- [29] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *EMNLP-CoNLL*, volume 7, Stroudsburg, PA, USA, 2007. ACL.
- [30] D. Bickerton. *Language and Human Behavior*. University College London Press, 1995.
- [31] C. Biemann. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics:*

- Student Research Workshop*, COLING ACL '06, pages 7–12, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [32] C. Biemann. *Unsupervised and Knowledge-Free Natural Language Processing in the Structure Discovery Paradigm*. Thesis, University of Leipzig, July 2007.
- [33] C. Biemann, S. Roos, and K. Weihe. Quantifying semantics using complex network analysis. In *Proceedings of COLING 2012*, pages 263–278, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.
- [34] R. Blanco and C. Lioma. Graph-based term weighting for information retrieval. *Information Retrieval*, 15, 2012.
- [35] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.
- [36] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.
- [37] D. J. Brenes, D. Gayo-Avello, and R. Garcia. On the fly query segmentation using snippets. In *Proceedings of the First Spanish Conference on Information Retrieval*, CERI '10, Madrid, Spain, 2010. The First Spanish Conference on Information Retrieval.
- [38] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.
- [39] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, Apr. 1998.
- [40] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36, September 2002.
- [41] P. Brown, P. Desouza, R. Mercer, V. Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4), 1992.
- [42] P. Brown, V. Pietra, R. Mercer, S. Pietra, and J. Lai. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.
- [43] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 875–883, New York, NY, USA, 2008. ACM.
- [44] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there: preference judgments for relevance. In *ECIR '08*, 2008.

- [45] V. R. Carvalho, M. Lease, and E. Yilmaz. Crowdsourcing for search evaluation. *SIGIR Forum*, 44(2):17–22, Jan. 2011.
- [46] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, pages 621–630, New York, NY, USA, 2009. ACM.
- [47] G. Chierchia and S. McConnell-Ginet. *Meaning and grammar: An introduction to semantics*. MIT press, 2000.
- [48] N. Chomsky. *Barriers*. MIT Press, 1986.
- [49] N. Chomsky. *Syntactic structures*. de Gruyter Mouton, Berlin, 2002.
- [50] M. Choudhury, D. Chatterjee, and A. Mukherjee. Global topology of word co-occurrence networks: Beyond the two-regime power-law. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 162–170, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [51] M. Choudhury and A. Mukherjee. The structure and dynamics of linguistic networks. In *Dynamics On and Of Complex Networks*. Birkhäuser Boston, 2009.
- [52] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, Mar. 1990.
- [53] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1), Apr. 1960.
- [54] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08*, pages 87–94, New York, NY, USA, 2008. ACM.
- [55] W. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley, 2010.
- [56] R. Cummins and C. O’Riordan. Learning in a pairwise term-term proximity framework for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09*, New York, NY, USA, 2009. ACM.
- [57] A. L. da Costa Carvalho, E. S. de Moura, and P. Calado. Using statistical features to find phrasal terms in text collections. *JIDM*, 1(3), 2010.

- [58] E. F. de Lima and J. O. Pedersen. Phrase recognition and expansion for short, precision-biased queries based on a query log. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 145–152, New York, NY, USA, 1999. ACM.
- [59] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA, 2008. ACM.
- [60] J.-L. Dessalles. Du protolangage au langage : modèle d'une transition. *Marges linguistiques*, 11, June 2006.
- [61] S. N. Dorogovtsev and J. F. F. Mendes. Language as an evolving word web. *Proceedings of the Royal Society of London B*, 268(1485), December 2001.
- [62] J. Downie. *Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text*. PhD thesis, The University of Western Ontario, 1999.
- [63] J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. Using search session context for named entity recognition in query. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 765–766, New York, NY, USA, 2010. ACM.
- [64] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, WWW '11, New York, NY, USA, 2011. ACM.
- [65] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, Mar. 1993.
- [66] T. Dunning. *Statistical identification of language*. Computing Research Laboratory, New Mexico State University, 1994.
- [67] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.
- [68] P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 5, 1959.
- [69] R. Ferrer-i-Cancho and R. V. Solé. The small world of human language. *Proceedings of the Royal Society of London B*, 268(1482), 2001.
- [70] E. Fox and J. Shaw. Combination of multiple searches. Technical report, National Institute of Standards and Technology, 1994.

- [71] N. Fukui and M. Speas. Specifiers and projection. *MIT Working Papers in Linguistics*, 8, 1986.
- [72] K. Ganchev, K. Hall, R. McDonald, and S. Petrov. Using search-logs to improve query tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [73] D. Ganguly, J. Leveling, and G. J. Jones. United We Fall, Divided We Stand: A Study of Query Segmentation and PRF for Patent Prior Art Search. In *Proceedings of the 4th Workshop on Patent Information Retrieval*, PaIR '11, pages 13–18, New York, NY, USA, 2011. ACM.
- [74] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, New York, NY, USA, 2004. ACM.
- [75] J. Gauvain, A. Messaoudi, and H. Schwenk. Language recognition using phone lattices. In *Proc. ICSLP*, volume 4, 2004.
- [76] C. González-Caro and R. Baeza-Yates. A multi-faceted approach to query intent classification. In R. Grossi, F. Sebastiani, and F. Silvestri, editors, *SPIRE '11*, volume 7024 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2011.
- [77] N. Gövert and G. Kazai. Overview of the initiative for the evaluation of xml retrieval (inex) 2002. In *INEX Workshop*, pages 1–17. Citeseer, 2002.
- [78] S. Green, M.-C. de Marneffe, and C. D. Manning. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227, 2013.
- [79] W. R. Greiff. A theory of term weighting based on exploratory data analysis. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 11–19, New York, NY, USA, 1998. ACM.
- [80] E. Guichard. *L'internet: Mesures des appropriations d'une technique intellectuelle*. These, Ecole des hautes études en sciences sociales, Oct 2002.
- [81] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 267–274, New York, NY, USA, 2009. ACM.
- [82] M. Hagen, M. Potthast, A. Beyer, and B. Stein. Towards optimum query segmentation: In doubt without. In *Proceedings of the 21st ACM international conference on*

- Information and knowledge management*, CIKM '12, New York, NY, USA, 2012. ACM.
- [83] M. Hagen, M. Potthast, B. Stein, and C. Braeutigam. The power of naive query segmentation. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, New York, NY, USA, 2010. ACM.
- [84] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. Query segmentation revisited. In *Proceedings of the 20th international conference on World wide web*, WWW '11, New York, NY, USA, 2011. ACM.
- [85] D. Hakkani-Tür, A. Çelikyılmaz, L. P. Heck, and G. Tür. A weakly-supervised approach for discovering new user intents from search query logs. In *INTERSPEECH*, pages 3780–3784, 2013.
- [86] C. Hauff, V. Murdock, and R. Baeza-Yates. Improved query difficulty prediction for the web. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 439–448, New York, NY, USA, 2008. ACM.
- [87] M. D. Hauser, N. Chomsky, and W. T. Fitch. The Faculty of Language: What Is It, Who Has It, and How Did It Evolve? *Science*, 298(5598):1569–1579, 2002.
- [88] B. He, J. X. Huang, and X. Zhou. Modeling term proximity for probabilistic information retrieval models. *Inf. Sci.*, 181(14), July 2011.
- [89] M. Heilman and N. A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 609–617, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [90] K. M. Hermann and P. Blunsom. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
- [91] D. Hiemstra. A probabilistic justification for using tfidf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.
- [92] N. Hochstotter and M. Koch. Standard parameters for searching behaviour in search engines and their empirical evaluation. *Journal of Information Science*, 35(1):45–65, 2009.
- [93] C. F. Hockett. The origin of speech. *Scientific American*, 203, 1960.
- [94] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 1963.

- [95] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.
- [96] T.-P. Hong, C.-W. Lin, K.-T. Yang, and S.-L. Wang. Using tf-idf to hide sensitive itemsets. *Applied Intelligence*, 38(4):502–510, 2013.
- [97] Y. Hou, X. Zhao, D. Song, and W. Li. Mining pure high-order word associations via information geometry for information retrieval. *ACM Trans. Inf. Syst.*, 31(3):12:1–12:32, Aug. 2013.
- [98] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 471–480, New York, NY, USA, 2009. ACM.
- [99] J. Huang, J. Gao, J. Miao, X. Li, K. Wang, F. Behr, and C. L. Giles. Exploring web scale language models for search query processing. In *Proceedings of the 19th international conference on World wide web*, WWW '10, New York, NY, USA, 2010. ACM.
- [100] J. J. Hull and S. N. Srihari. Experiments in text recognition with binary n-gram and viterbi algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(5), May 1982.
- [101] R. Jackendoff. *X-bar syntax*. The MIT Press, Cambridge, MA, 1977.
- [102] A. Jain and M. Pennacchiotti. Open entity extraction from web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 510–518, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [103] B. Jansen, A. Spink, and A. Pfaff. Web query structure: Implications for IR system design. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics*, SCI '00, pages 169–176, 2000.
- [104] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing and Management*, 44(3):1251 – 1266, 2008.
- [105] B. J. Jansen and A. Spink. How are we searching the world wide web? a comparison of nine search engine transaction logs. *Information Processing and Management*, 42(1):248 – 263, 2006. Formal Methods for Information Retrieval.
- [106] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, Apr. 1998.
- [107] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36(2), Jan. 2000.

- [108] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20, October 2002.
- [109] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.
- [110] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 25(2), Apr. 2007.
- [111] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: Development and comparative experiments: Part 1. *Information Processing and Management*, 36(6):779 – 808, 2000.
- [112] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: Development and comparative experiments: Part 2. *Information Processing and Management*, 36(6):809 – 840, 2000.
- [113] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 699–708, New York, NY, USA, 2008. ACM.
- [114] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, New York, NY, USA, 2006. ACM.
- [115] Z. R. M. Kashani, H. Ahrabian, E. Elahi, A. Nowzari-Dalini, E. S. Ansari, S. Asadi, S. Mohammadi, F. Schreiber, and A. Masoudi-Nejad. Kavosh: a new algorithm for finding network motifs. *BMC Bioinformatics*, 10, 2009.
- [116] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11), Jul 2004.
- [117] E. M. Keen. The use of term position devices in ranked output experiments. *J. Doc.*, 47(1), Mar. 1991.
- [118] M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1/2), June 1938.
- [119] R. Kindermann and J. L. Snell. *Markov random fields and their applications*. American Mathematical Society, Providence, RI, 1980.

- [120] J. Kiseleva, Q. Guo, E. Agichtein, D. Billsus, and W. Chai. Unsupervised query segmentation using click data: Preliminary results. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 1131–1132, New York, NY, USA, 2010. ACM.
- [121] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.
- [122] P. Koehn. *Statistical machine translation*, volume 11. Cambridge University Press, 2010.
- [123] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, NAACL '01*, pages 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.
- [124] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 1951.
- [125] G. Kumaran and J. Allan. Effective and efficient user interaction for long queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08*, pages 11–18, New York, NY, USA, 2008. ACM.
- [126] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09*, pages 564–571, New York, NY, USA, 2009. ACM.
- [127] K. L. Kwok. A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '96*, pages 187–195, New York, NY, USA, 1996. ACM.
- [128] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '01*, New York, NY, USA, 2001. ACM.
- [129] E. Law, A. Mityagin, and M. Chickering. Intentions: A game for classifying search query intent. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09*, pages 3805–3810, New York, NY, USA, 2009. ACM.
- [130] M. Lease. Natural Language Processing for Information Retrieval: The Time is Ripe (Again). In *Proceedings of the ACM First Ph.D. Workshop in CIKM, PIKM '07*, pages 1–8, New York, NY, USA, 2007. ACM.

- [131] M. Lease, J. Allan, and W. B. Croft. Regression Rank: Learning to Meet the Opportunity of Descriptive Queries. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, Berlin, Heidelberg, 2009. Springer-Verlag.
- [132] D. Lee, H. Chuang, and K. Seamons. Document ranking and the vector-space model. *Software, IEEE*, 14(2):67–75, Mar 1997.
- [133] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 391–400. ACM, 2005.
- [134] H. Li, G. Xu, W. B. Croft, M. Bendersky, Z. Wang, and E. Viegas. QRU-1: A Public Dataset for Promoting Query Representation and Understanding Research. In *WSCD '12*, 2012.
- [135] X. Li. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [136] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 339–346, New York, NY, USA, 2008. ACM.
- [137] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, New York, NY, USA, 2011. ACM.
- [138] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active Objects: Actions for Entity-centric Search. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 589–598, New York, NY, USA, 2012. ACM.
- [139] X. Liu, W. B. Croft, P. Oh, and D. Hart. Automatic recognition of reading levels from user queries. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 548–549, New York, NY, USA, 2004. ACM.
- [140] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [141] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT press, Cambridge, MA, 1999.

- [142] M. Manshadi and X. Li. Semantic tagging of web search queries. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [143] R. Mantegna, S. Buldyrev, A. Goldberger, S. Havlin, C. Peng, M. Simons, and H. Stanley. Systematic analysis of coding and noncoding dna sequences using methods of statistical linguistics. *Physical Review E*, 52(3), 1995.
- [144] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.
- [145] O. A. McBryan. GENVL and WWW: Tools for Taming the Web. In *Proceedings of the First International World Wide Web Conference*, pages 79–90, 1994.
- [146] D. Medeiros Eler and R. Garcia. Using otsu's threshold selection method for eliminating terms in vector space model computation. In *Information Visualisation (IV), 2013 17th International Conference*, pages 220–226, July 2013.
- [147] A. Mehler. Large text networks as an object of corpus linguistic studies. In *Corpus Linguistics. An International Handbook of the Science of Language and Society*, pages 328–382. De Gruyter, Berlin, Germany, 2008.
- [148] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, New York, NY, USA, 2005. ACM.
- [149] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594), 2002.
- [150] R. Mitkov, L. An Ha, and N. Karamanis. A computer-aided environment for generating multiple-choice test items. *Nat. Lang. Eng.*, 12(2), June 2006.
- [151] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 538–548, New York, NY, USA, 2002. ACM.
- [152] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *ACM Conference on research and Development in Information Retrieval, SIGIR, Predicting query difficulty-methods and applications workshop*, pages 7–10, 2005.
- [153] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45, March 2003.

- [154] V. B. Nguyen and M. Y. Kan. Functional faceted web query analysis. In *Proceedings of the Workshop on Query Log Analysis: Social And Technological Challenges*, WWW '07, 2007.
- [155] M. Paşca. Acquisition of categorized named entities for web search. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 137–145, New York, NY, USA, 2004. ACM.
- [156] M. Paşca. Finding Instance Names and Alternative Glosses on the Web: WordNet Reloaded. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 280–292. Springer Berlin Heidelberg, 2005.
- [157] M. Paşca. Turning web text and search queries into factual knowledge: Hierarchical class attribute extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAI'08, pages 1225–1230. AAAI Press, 2008.
- [158] M. Paşca. Outclassing Wikipedia in Open-domain Information Extraction: Weakly-supervised Acquisition of Attributes over Conceptual Hierarchies. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 639–647, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [159] M. Paşca. Attribute extraction from synthetic web search queries. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 401–409, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.
- [160] M. Paşca and B. Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *Proceedings of the 20th international joint conference on Artificial intelligence*, IJCAI'07, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.
- [161] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June 2008. Association for Computational Linguistics.
- [162] M. Paşca, B. Van Durme, and N. Garera. The role of documents vs. queries in extracting class attributes from text. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 485–494, New York, NY, USA, 2007. ACM.
- [163] J. H. Paik. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 343–352, New York, NY, USA, 2013. ACM.

- [164] T. Paikeday. *The native speaker is dead!: An informal discussion of a linguistic myth with Noam Chomsky and other linguists, philosophers, psychologists, and lexicographers*. Paikeday Pub., 1985.
- [165] S. Pandey and K. Punera. Unsupervised extraction of template structure in web search queries. In *Proceedings of the 21st International Conference on World Wide Web, WWW '12*, pages 409–418, New York, NY, USA, 2012. ACM.
- [166] B. Pang and R. Kumar. Search in the lost sense of “query”: Question formulation in web search queries and its temporal changes. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2011.
- [167] A. Parameswaran, R. Kaushik, and A. Arasu. Efficient parsing-based search over structured data. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management, CIKM '13*, pages 49–58, New York, NY, USA, 2013. ACM.
- [168] N. Parikh, P. Sriram, and M. Al Hasan. On segmentation of ecommerce queries. In *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management, CIKM '13*, pages 1137–1146, New York, NY, USA, 2013. ACM.
- [169] J.-H. Park and W. B. Croft. Query term ranking based on dependency parsing of verbose queries. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 829–830, New York, NY, USA, 2010. ACM.
- [170] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems, InfoScale '06*, New York, NY, USA, 2006. ACM.
- [171] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the DFR framework. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07*, New York, NY, USA, 2007. ACM.
- [172] M. Pennacchiotti and P. Pantel. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP '09*, pages 238–247, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [173] S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. In N. C. C. Chair, K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

- [174] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007.
- [175] E. Pitler and K. Church. Using word-sense disambiguation methods to classify web queries by intent. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1428–1436, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [176] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, New York, NY, USA, 1998. ACM.
- [177] M. Porter. An algorithm for suffix stripping. *Program*, 14:130, 1980.
- [178] M. F. Porter. An algorithm for suffix stripping. In K. Sparck Jones and P. Willett, editors, *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
- [179] K. Radinsky and N. Ailon. Ranking from pairs and triplets: Information quality, evaluation methods and query complexity. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 105–114, New York, NY, USA, 2011. ACM.
- [180] V. V. Raghavan and S. K. M. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287, 1986.
- [181] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 157–176. Springer Netherlands, 1999.
- [182] R. Rao, N. Yadav, M. Vahia, H. Joglekar, R. Adhikari, and I. Mahadevan. Entropic evidence for linguistic structure in the indus script. *Science*, 324(5931), 2009.
- [183] E. J. Ray, D. S. Ray, and R. Seltzer. *AltaVista Search Revolution*. Osborne Publishing, 1998.
- [184] J. Reisinger and M. Paşca. Low-cost supervision for multiple-source attribute extraction. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '09, Berlin, Heidelberg, 2009. Springer-Verlag.

- [185] K. M. Risvik, T. Mikolajewski, and P. Boros. Query segmentation for web search. In *Proceedings of the 12th international conference companion on World wide web, WWW '03*, New York, NY, USA, 2003. ACM.
- [186] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04*, pages 42–49, New York, NY, USA, 2004. ACM.
- [187] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.
- [188] S. E. Robertson, S. Walker, and M. Beaulieu. Experimentation as a way of life: Okapi at TREC. *Information Processing and Management*, 36(1):95 – 108, 2000.
- [189] S. E. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne. Okapi at trec-4. In *Proceedings of the fourth text retrieval conference, TREC-4*, pages 73–97. NIST Special Publication, 1996.
- [190] D. E. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of the 13th International Conference on World Wide Web, WWW '04*, pages 13–19, New York, NY, USA, 2004. ACM.
- [191] T. Sakai. Evaluation with informational and navigational intents. In *Proceedings of the 21st International Conference on World Wide Web, WWW '12*, pages 499–508, New York, NY, USA, 2012. ACM.
- [192] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11*, pages 1043–1052, New York, NY, USA, 2011. ACM.
- [193] G. Salton. *Automatic Information Organization and Retrieval*. McGraw Hill Text, 1968.
- [194] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.
- [195] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, Aug. 1988.
- [196] G. Salton, C. Buckley, and C. Yu. An evaluation of term dependence models in information retrieval. In G. Salton and H.-J. Schneider, editors, *Research and Development in Information Retrieval*, volume 146 of *Lecture Notes in Computer Science*, pages 151–173. Springer Berlin Heidelberg, 1983.

- [197] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.
- [198] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.
- [199] N. Sarkas, S. Pappas, and P. Tsaparas. Structured annotations of web queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, New York, NY, USA, 2010. ACM.
- [200] F. Schreiber and H. Schwöbbermeyer. Mavisto: a tool for the exploration of network motifs. *Bioinformatics*, 21(17), Sept. 2005.
- [201] E. Selkirk. The Prosodic Structure of Function Words. In J. L. Morgan and K. Demuth, editors, *Signal to Syntax: Bootstrapping from Speech to Grammar in Early Acquisition*. Routledge, 1996.
- [202] C. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30(1), 1951.
- [203] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, October 1948.
- [204] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.
- [205] G. Singer, U. Norbistrath, and D. Lewandowski. Ordinary search engine users assessing difficulty, effort, and outcome for simple and complex search tasks. In *Proceedings of the 4th Information Interaction in Context Symposium, IIX '12*, pages 110–119, New York, NY, USA, 2012. ACM.
- [206] K. Smith, H. Brighton, and S. Kirby. Complex Systems In Language Evolution: The Cultural Emergence Of Compositional Structure. *Advances in Complex Systems (ACS)*, 6(04):537–558, 2003.
- [207] F. Song and W. B. Croft. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management, CIKM '99*, pages 316–321, New York, NY, USA, 1999. ACM.
- [208] R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval, ECIR'08*, Berlin, Heidelberg, 2008. Springer-Verlag.
- [209] A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52, February 2001.

- [210] R. Sproat. Last words: Ancient symbols, computational linguistics, and the reviewing practices of the general science journals. *Computational Linguistics*, 36(3), 2010.
- [211] M. Srikanth and R. Srihari. Biterm language models for document retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 425–426, New York, NY, USA, 2002. ACM.
- [212] J. Steele, P. Jordan, and E. Cochrane. Evolutionary approaches to cultural and linguistic diversity. *Philos Trans R Soc Lond B Biol Sci*, 365(1559):3781–5, 2010.
- [213] S. H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.
- [214] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of tf-idf schemes for web pages using their hyperlinked neighboring pages. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '03, pages 198–207, New York, NY, USA, 2003. ACM.
- [215] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- [216] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, New York, NY, USA, 2008. ACM.
- [217] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, New York, NY, USA, 2007. ACM.
- [218] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2, March 2002.
- [219] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.
- [220] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics, 2000.

- [221] B. Van Durme and M. Paşca. Finding Cars, Goddesses and Enzymes: Parametrizable Acquisition of Labeled Instances for Open-Domain Information Extraction. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, AAAI '08, pages 1243–1248, 2008.
- [222] E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Inf. Process. Manage.*, 36, September 2000.
- [223] E. M. Voorhees. The trec question answering track. *Nat. Lang. Eng.*, 7(4), Dec. 2001.
- [224] J. B. Vuurens and A. P. de Vries. Distance matters! Cumulative proximity expansions for ranking documents. *Information Retrieval*, 17(4):380–406, 2014.
- [225] X. Wang, D. Chakrabarti, and K. Punera. Mining broad latent query aspects from search sessions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 867–876, New York, NY, USA, 2009. ACM.
- [226] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684), 1998.
- [227] S. Wernicke. A faster algorithm for detecting network motifs. In R. Casadio and G. Myers, editors, *Algorithms in Bioinformatics*, volume 3692 of *Lecture Notes in Computer Science*, pages 165–177. Springer Berlin Heidelberg, 2005.
- [228] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 4–11, New York, NY, USA, 1996. ACM.
- [229] G. R. Xue, H. J. Zeng, Z. Chen, Y. Yu, W. Y. Ma, W. Xi, and W. Fan. Optimizing Web search using Web click-through data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126, New York, NY, USA, 2004. ACM.
- [230] X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1445–1456, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.
- [231] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*, WWW '10, New York, NY, USA, 2010. ACM.

- [232] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web, WWW '10*, New York, NY, USA, 2010. ACM.
- [233] X. Yin, W. Tan, X. Li, and Y.-C. Tu. Automatic extraction of clickable structured web contents for name entity queries. In *Proceedings of the 19th international conference on World wide web, WWW '10*, New York, NY, USA, 2010. ACM.
- [234] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '05*, pages 512–519, New York, NY, USA, 2005. ACM.
- [235] C. T. Yu, C. Buckley, K. Lam, and G. Salton. A generalized term dependence model in information retrieval. Technical report, Cornell University, 1983.
- [236] H. Yu and F. Ren. Role-explicit query identification and intent role annotation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 1163–1172, New York, NY, USA, 2012. ACM.
- [237] S. Yu, D. Cai, J. rong Wen, and W. ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *In Intl. World Wide Web Conf. (WWW)*, pages 11–18, 2003.
- [238] X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proceedings of the First International Workshop on Keyword Search on Structured Data, KEYS '09*, New York, NY, USA, 2009. ACM.
- [239] C. Zhai. Fast statistical parsing of noun phrases for document indexing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing, ANLC '97*, pages 312–319, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.
- [240] C. Zhang, N. Sun, X. Hu, T. Huang, and T.-S. Chua. Query segmentation based on eigenspace similarity. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, ACLShort '09*, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [241] W. Zhang, Y. Cao, C.-Y. Lin, J. Su, and C.-L. Tan. An error driven approach to query segmentation. In *Proceedings of the 22nd international conference on World Wide Web companion, WWW '13 Companion*, pages 127–128, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

- [242] W. Zhang, Y. Cao, C.-Y. Lin, J. Su, and C.-L. Tan. Learning a replacement model for query segmentation with consistency in search logs. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 28–36, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.
- [243] W. Zhang, S. Liu, C. Yu, C. Sun, F. Liu, and W. Meng. Recognition and classification of noun phrases in queries for effective retrieval. In *CIKM '07*. ACM, 2007.
- [244] W. V. Zhang, X. He, B. Rey, and R. Jones. Query rewriting using active learning for sponsored search. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07*, pages 853–854, New York, NY, USA, 2007. ACM.
- [245] B. Zhao, M. Eck, and S. Vogel. Language model adaptation for statistical machine translation with structured query models. In *ACL*, 2004.
- [246] J. Zhao, J. X. Huang, and Z. Ye. Modeling term associations for probabilistic information retrieval. *ACM Trans. Inf. Syst.*, 32(2):7:1–7:47, Apr. 2014.
- [247] M. Zissman and E. Singer. Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1. IEEE, 1994.