Structured Search in Annotated Document Collections

Dhruv Gupta
Max Planck Institute for Informatics
Saarbrücken Graduate School of Computer Science
Saarland Informatics Campus, Germany
dhgupta@mpi-inf.mpg.de

Klaus Berberich
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
htw saar, Saarbrücken, Germany
kberberi@mpi-inf.mpg.de

ABSTRACT

In this work, we demonstrate structured search capabilities of the GYANI indexing infrastructure. GYANI allows linguists, journalists, and scholars in humanities to search large semantically annotated document collections in a structured manner by supporting queries with regular expressions between word sequences and annotations. In addition to this, we provide support for attaching semantics to words via annotations in the form of part-of-speech, named entities, temporal expressions, and numerical quantities. We demonstrate that by enabling such structured search capabilities we can quickly gather annotated text regions for various knowledge-centric tasks such as information extraction and question answering.

ACM Reference Format:
Dhruv Gupta and Klaus Berberich. 2019. Structured Search in Annotated Document Collections. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3289600.3290618

1 INTRODUCTION

Linguists, journalists, and scholars in humanities have often relied on off-the-shelf document indexers such as ElasticSearch [2], Lucene [6], or Solr [9] for document retrieval. Often, for them it also suffices to use commercial search engine APIs offered by Google [4] or Bing [1] to retrieve documents. Having obtained the documents, further analysis using natural language processing (NLP) tools is done to filter the sentences bearing the relevant information for the task at hand. With such a tedious process many relevant documents are missed. One of the primary reasons being: the user is only offered Boolean operators and keywords to model their queries.

With modern NLP toolkits (e.g., Stanford CoreNLP [18]) it is now possible to annotate large document collections with semantic annotations in the form of part-of-speech, named entities, temporal expressions, and numerical values. Our indexing infrastructure GYANI [14] is designed for analysis of large semantically annotated document collections with GREP-like search capabilities. GYANI allows the user to express queries using regular expressions between annotations and word sequences. Furthermore, it allows for attaching semantics to word sequences thus truly enabling semantic search. GYANI further distills text regions from the relevant documents that match the query, thereby alleviating the user from the need to further prune the document for relevant snippets.

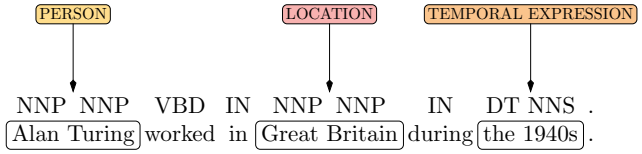

Figure 1: An example of a semantically annotated text region. With GYANI users have the capability to search for text regions by attaching semantics to word sequences as well as expressing regular expressions between word sequences and annotations.

GYANI is designed to support knowledge-centric tasks that are often performed by linguists, journalists, and scholars in humanities. In our demonstration, we showcase five knowledge-centric tasks: information extraction, question answering, relationship extraction, fact spotting, and semantic search. We outline the demonstration scenarios of these five knowledge-centric tasks below.

Information Extraction and Question Answering. Information extraction and question answering [13] tasks employ templates for which missing values are needed to be obtained from large document collections. As an example of this task, consider that we need to acquire the names of US presidents that were sworn in over time, we can issue the following query with GYANI:

$$\text{PERSON}\ \boxed{.*}\ \langle\ \texttt{sworn in as us president}\ \rangle\ \boxed{.*}\ \text{DATE}.$$

Relationship Extraction and Fact Spotting. Relationship extraction and fact spotting [20] task require us to identify sentences or text regions that contain mentions of entities or are witness to a fact. Since entities and predicates can be referred by many surface forms, we need to additionally convey these paraphrases in the query to increase recall. For instance, to identify all the sentences in which Alan Turing is recognized for his contribution to Computer Science, we can formulate the following query with GYANI:

$$[\ \texttt{alan turing}\ |\ \texttt{alan mathieson turing}\ ]\ \boxed{.*}$$
$$[\ \texttt{computer science}\ |\ \texttt{informatics}\ |\ \texttt{cs}\ ].$$

Semantic Search. To assist the user in formulating her information need precisely, we further allow them to attach semantics to words. For instance, to identify sentences in which the word "green" has been used to refer to environmental friendly technologies we can attach the part-of-speech tag "adjective" in order to restrict our search to only those sentences. With GYANI we can formulate such a query as follows:

$$\texttt{green} \oplus \text{ADJ}.$$

Organization of the article is as follows. First, we describe the indexing infrastructure underlying GYANI in Section 2. Second, we discuss the implementation details of GYANI in Section 3. Third, we discuss the demonstration of GYANI in Section 4. Finally, we cover the related work with respect our problem statement in Section 5.

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290618

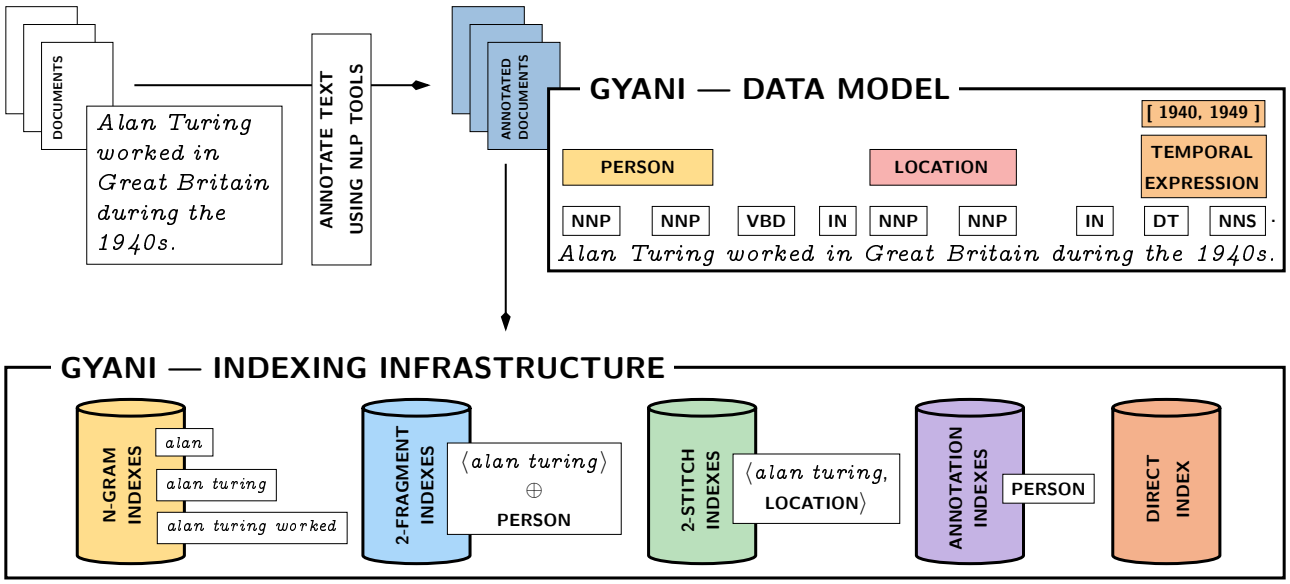

Figure 2: The data model and the indexing infrastructure underlying GYANI.

2 GYANI INDEXING INFRASTRUCTURE

To support structured search with regular expressions between word sequences and annotations, GYANI relies on a novel data model that accommodates the semantic annotations that a text document can be enriched with. Based on this novel data model we derive indexing units that help us retrieve the text regions corresponding to regular expression queries quickly. Technical details of GYANI's indexing infrastructure have been published in [14]. We briefly describe GYANI's data model and the key indexes that help us retrieve text regions for structured queries. A high-level overview of the data model and indexing infrastructure is given in Figure 2.

2.1 Data Model

Natural language processing (NLP) tools allows us to process text with different annotators. Each such annotator (e.g., named entity recognizer) marks a sequence of words with tags from its annotation vocabulary (e.g., PERSON, LOCATION, or ORGANIZATION). As semantic annotations, we process the document collections with annotations in the form of part-of-speech, named entity, temporal expressions, and numerical values. GYANI's data model consists of maintaining the positional spans of not only word sequences but also the annotations that have been marked by the NLP annotators. Thus, as can be seen in Figure 2, we keep each layer of annotation over the word sequences along with their positional spans.

2.2 Indexing Infrastructure

The infrastructure underlying GYANI relies on indexes built on different combinations of the word sequences and semantic annotations from different layers. This pair-wise choice was made as it is highly flexible and allows us to retrieve the text regions given a query involving regular expressions quickly. A detailed discussion of the complete design space is given in [14]. Next we briefly, give an overview of the five different kinds of indexes that are maintained in GYANI for structured search. Indexing units for the five index types are exemplified in Figure 2.

- **N-GRAM INDEXES** record unigrams, bigrams, and trigrams along with their positional spans. N-GRAM INDEXES allow us to quickly decompose phrases in queries into overlapping n-grams and computing the resulting text regions.
- **ANNOTATION INDEXES** record the annotations (e.g., part-of-speech, named entity, temporal expressions, and numerical values) along with their positional span. The ANNOTATION INDEXES thus allows us to retrieve results when only annotations are expressed in the query.
- **2-FRAGMENT INDEXES** record pair-wise combinations of word sequences and the accompanying annotations e.g., part-of-speech, named entity, temporal expressions, or numerical values. With the 2-FRAGMENT INDEXES we can efficiently retrieve results for those queries in which the semantics are accompanied with the word sequences.
- **2-STITCH INDEXES** record ordered co-occurrences of word sequences with annotations. The 2-STITCH INDEXES allows us to quickly retrieve text regions as results when regular expressions between word sequences and annotations are expressed. In such cases, computing the results using N-GRAM and ANNOTATION INDEXES may be too time consuming.
- **DIRECT INDEX** records the word sequences in a document along with all its accompanying annotation layers. In addition to this, the DIRECT INDEX records the sentence boundaries. They are used to restrict matches to within sentence boundaries when identifying the relevant text regions.

With a combination of the above five indexes we can answer all possible queries that can arise in the tasks of information extraction, question answering, relationship extraction, fact spotting, and semantic search.

Table 1: Collection and annotation statistics.

COLLECTION	SIZE (GB)	$n_{\text{documents}}$	n_{words}	$n_{\text{sentences}}$	$n_{\text{part-of-speech}}$	$n_{\text{named entity}}$	n_{time}	n_{numbers}
NEW YORK TIMES	49.7	1,855,623	1,058,949,098	54,024,146	1,058,949,098	107,745,696	15,411,681	21,720,437
WIKIPEDIA	156.0	5,327,767	2,807,776,276	192,925,710	2,807,776,276	444,301,507	97,064,344	82,591,612
GIGAWORD	193.6	9,870,655	3,988,683,648	181,386,746	3,988,683,648	517,420,195	72,247,124	102,299,554
GDELT	296.2	14,320,457	6,371,451,092	297,861,511	6,371,451,092	640,812,778	94,009,542	104,964,085

2.3 Query Operators

GYANI's query language supports regular expressions just like GREP to retrieve text regions that can assist in knowledge acquisition. We briefly describe the query operators that the user can express with GYANI.

Boolean Operators. We support three Boolean operators to be expressed between word sequences and annotations. These are AND, OR, and NEGATION.

- AND operator ($\boxed{\wedge}$). The AND operator is a Boolean operator that allows the user to retrieve documents that contain mentions of the both its operands. For instance, to retrieve documents that contain both mention of a location along with the word sequence "Alan Turing" we can formulate the following query: $\langle$ *alan turing* $\rangle$ $\boxed{\wedge}$ LOCATION.
- OR operator ($\boxed{\vee}$). The OR operator is a Boolean operator that allows the user to retrieve documents that contain mentions of either of its operands.
- NEGATION operator ($\boxed{\neg}$). The NEGATION operator is a unary operator that obtains those documents that do not contain presence of the word sequence or annotation as its operand. For instance, we can identify documents that contain either the mention of "Alan Turing" with either a PERSON or ORGANIZATION but not with LOCATION via the following query: $\big[$*alan turing* $\boxed{\wedge}$ [PERSON $\boxed{\vee}$ ORGANIZATION]$\big]$ $\boxed{\neg}$ LOCATION.

Regular Expression Operators. We instantiated five regular expression operators that can be expressed between word sequences and annotations. These operators are briefly explained below.

- DOT STAR operator ($\boxed{.*}$) is a regular expression operator that matches a text region that has zero or more words between its operands. For instance, the query $\langle$*alan turing studied in*$\rangle$ $\boxed{.*}$ LOCATION, will match those text regions in documents that have zero or more words between the word sequence "Alan Turing studied in" and any location.
- DOT PLUS operator ($\boxed{.+}$) is a regular expression operator that matches a text region that has one or more words between its operands.
- DOT QUES operator ($\boxed{.?}$) is a regular expression operator that matches a text region that has zero words or only a word between its operands.
- DOT operator ($\boxed{.}$) is a regular expression operator that matches a text region that has only a word between its operands.
- UNION operator (|) is a regular expression operator that helps us union results from multiple word sequences. This operator is useful when specifying multiple surface forms for the same entity. For instance, $\big[$ *united states of america* | *usa* $\big]$ groups together all the text regions that mention the surface forms for USA.

The $\boxed{.*}$ and $\boxed{.+}$ operators can further be turned lazy by additionally specifying the $\boxed{?}$ flag. Thus the variants $\boxed{.*?}$ and $\boxed{.+?}$ operators match only the shortest text region in the documents as results.

Projection Operators. We further instantiate variants of the regular expression operators that return only text regions that lie within sentences. These operators are either associated with words or a particular annotation type. These projection operators are: $\boxed{\ell}$, $\boxed{\ell?}$, $\boxed{\ell*}$, and $\boxed{\ell+}$. Where, ℓ denotes words or an annotation type.

3 GYANI ARCHITECTURE

We next describe the document collections that were annotated and indexed using GYANI. We also describe the technical and hardware details of GYANI's implementation.

3.1 Document Collections and Annotations

We indexed four document collections with GYANI. Three of the indexed document collections are news archives: the New York Times [7], the English Gigaword [8], and GDelt news articles [3]. As our final document collection, we indexed the complete English Wikipedia [11]. To annotate these document collections we used Stanford's CoreNLP natural language processing toolkit [18]. We enriched the document collections with part-of-speech and named entity annotations. Furthermore, for the named entity annotations of type temporal expressions and numerical values we normalized them to obtain the time interval values and canoncalized numerical values. The time intervals and numerical values are thus treated as separate layers of semantic annotations. All the four document collections present us with well-written text so that we are able to obtain high-precision annotations with Stanford's CoreNLP toolkit. Table 1 shows the collection statistics as well as the sizes of the annotated document collections.

3.2 Implementation Details

We have implemented the entire infrastructure from scratch using the Java programing language. We utilized HBase, a distributed extensible-record store, for storing our indexes. The key for a record in HBase stores the indexing unit of our indexes, while the value of the record stores the compressed payload of each posting list. The compression for the payloads was carried out using the PFORDelta compression technique [5]. The HBase service runs on our Hadoop cluster that consists of twenty machines. Each machine is equipped with a 24-core Intel Xeon CPU with processing speeds of upto 3.50 GHz, with upto 128 gigabytes of primary memory, and upto four terabytes of secondary storage. The front-end for the GYANI's indexing infrastructure was built using Java-FX. The annotation highlighting was done by leveraging the CSS library from [10].

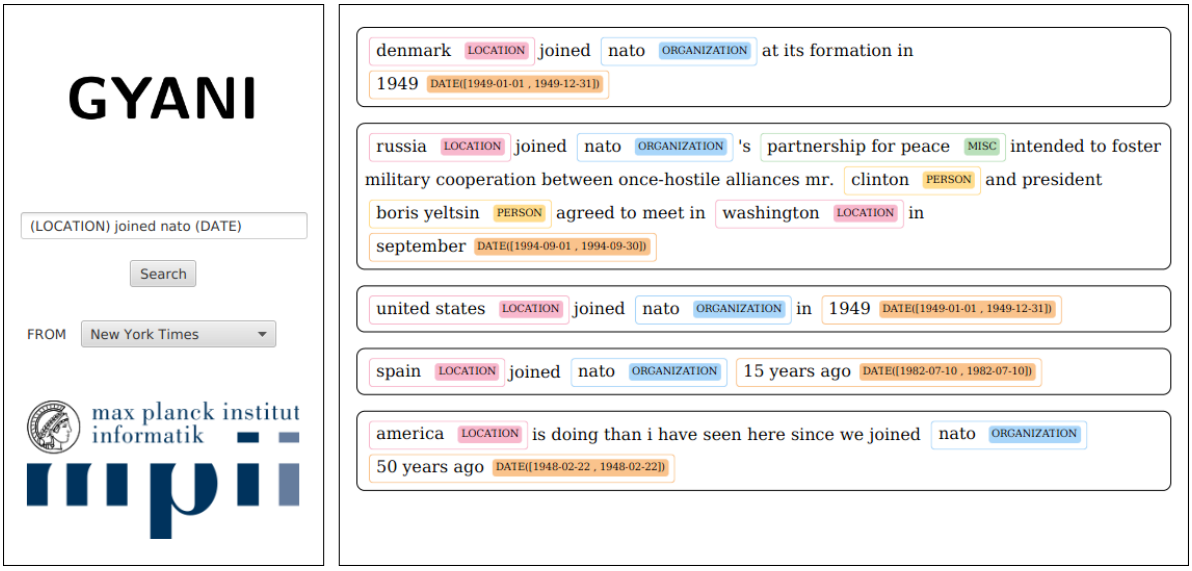

Figure 3: GYANI's GUI. The query, "LOCATION ⟨joined nato⟩ DATE", uses the projection operator ⟦ℓ⟧ to retrieve sentences as evidences.

4 GYANI DEMONSTRATION

The graphical user interface (GUI) for GYANI is shown in Figure 3. The GUI features a search bar in which the user can enter the query using the operators described in Section 2. The user can then select from the four different document collections (see Table 1) indexed with GYANI to retrieve the relevant text regions. The retrieved text regions are shown in the results page on the right-hand side of the GUI. For each text region we show the named entity annotation layer by highlighting the word sequences which have been annotated by the Stanford's CoreNLP named entity recognizer.

Target Audience for our demonstration are linguists, journalists, and scholars in humanities. Often for them structured search capabilities are not offered by the most accessible commercial search engines. Structured search in large document collections is crucial for them. For instance, when performing fact checking, journalists often need to verify claims involving named entities, temporal expressions, and numerical values. In such instances, the knowledge-centric tasks discussed in Section 1 naturally arise.

Demonstration at the conference will allow the audience to interactively formulate queries involving regular expressions, word sequences, and semantic annotations for the five knowledge-centric tasks and obtain text regions from the indexed document collections. As an example scenario, an attendee can query GYANI to retrieve all evidences that detail acquisitions by Google:

$$[\,google\,|\,search\ giant\,]\ .*\ [\,invested\ in\,|\,acquired\,]\ .*\ \text{ORG}.$$

Furthermore, to identify a chronology of important events mentioned in news regarding Silicon Valley (the location and not the television series) startups with monetary values, the attendee can formulate the following query:

$$\text{DATE}\ .*\ \langle silicon\ valley\rangle \oplus \text{LOCATION}\ .*\ \text{MONEY}.$$

We shall thus demonstrate at the conference, that knowledge acquisition using GYANI's structured search capabilities can indeed be done interactively and at scale across millions of documents.

5 RELATED WORK

Semantically annotated text has been leveraged for entity-driven search in several systems [12, 15–17, 19]. The BROCOLLI system [12] allows the user to specify entities, relations, and categories of entities as a way of navigating semantically annotated document collections. The EVELIN system [19] further allows the user to navigate Wikipedia via entity co-occurrence networks. The STICS system [16, 17] leverages disambiguated named entities and a background knowledge graph to perform analytics using named entity counts and publication dates of documents. The DIGITALHISTORIAN system [15] allows analytics over time intervals of interest computed using temporal expressions and co-occurring disambiguated named entities in document contents. However, the aforementioned systems do not support structured search using regular expressions.

REFERENCES

[1] Bing. https://www.bing.com/.
[2] ElasticSearch. https://www.elastic.co/.
[3] The GDELT Project. https://www.gdeltproject.org/.
[4] Google. https://www.google.com/.
[5] JavaFastPFOR. https://github.com/lemire/JavaFastPFOR.
[6] Lucene. https://lucene.apache.org/.
[7] The NYT Corpus. https://catalog.ldc.upenn.edu/LDC2008T19.
[8] English Gigaword. https://catalog.ldc.upenn.edu/LDC2011T07.
[9] Solr. https://lucene.apache.org/solr/.
[10] spaCy. https://spacy.io/usage/visualizers.
[11] Wikipedia: The Free Encyclopedia. https://www.wikipedia.org/.
[12] H. Bast et al. 2014. Semantic full-text search with broccoli. SIGIR'14.
[13] J. R. Frank et al. Evaluating Stream Filtering for Entity Profile Updates in TREC 2012, 2013, and 2014. TREC'14.
[14] D. Gupta and K. Berberich. GYANI: An Indexing Infrastructure for Knowledge-Centric Tasks. CIKM'18.
[15] D. Gupta et al. DIGITALHISTORIAN: Search & Analytics Using Annotations. HistoInformatics'16 at DH'16.
[16] J. Hoffart et al. AESTHETICS: Analytics with Strings, Things, and Cats. CIKM'14.
[17] J. Hoffart et al. STICS: searching with strings, things, and cats. SIGIR'14.
[18] C. D. Manning et al. The Stanford CoreNLP Natural Language Processing Toolkit. ACL'14.
[19] A. Spitz et al. EVELIN: Exploration of Event and Entity Links in Implicit Networks. WWW'17.
[20] S. Elbassuoni et al. ROXXI: Reviving witness dOcuments to eXplore eXtracted Information. *PVLDB* 3, 2 (2010), 1589–1592.