# Information Extraction from Web-Scale N-Gram Data

Niket Tandon
Max Planck Institute for Informatics
Saarbrücken, Germany
ntandon@mpi-inf.mpg.de

Gerard de Melo
Max Planck Institute for Informatics
Saarbrücken, Germany
gdemelo@mpi-inf.mpg.de

## ABSTRACT

Search engines are increasingly relying on structured data to provide direct answers to certain types of queries. However, extracting such structured data from text is challenging, especially due to the scarcity of explicitly expressed knowledge. Even when relying on large document collections, pattern-based information extraction approaches typically expose only insufficient amounts of information. This paper evaluates to what extent n-gram statistics, derived from volumes of texts several orders of magnitude larger than typical corpora, can allow us to overcome this bottleneck. An extensive experimental evaluation is provided for three different binary relations, comparing different sources of n-gram data as well as different learning algorithms.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.6 [**Artificial Intelligence**]: Learning

## General Terms

Algorithms

## 1. INTRODUCTION

Search engine users generally search for information, not for documents. In recent years, there has been a growing trend to go beyond standard keyword search over document collections by recognizing words and entity names and offering additional structured results when available [26]. For example, for a query like "*Pablo Picasso artwork*", the Bing search engine displays a list of Picasso paintings delivered by Freebase, a large database of structured data. Google has experimented with explicit answers to queries like "*capital of Oregon*" or "*prime minister of india*". Additionally, structured data is useful for query expansion [14], semantic search [23], and faceted search [3], among other things.


*SIGIR 2010 Web N-Gram Workshop*


Since much of the Web consists of unstructured text, *information extraction* techniques have been developed to harvest machine-readable structured information from textual data. Often, these techniques rely on pattern matching with textual templates. As an example, the pattern "$<X>$ *such as* $<Y>$" has been found to work well for the `isA` relation: A matching word sequence like "*... cities such as Paris ...*" allows us to induce knowledge of the form `isA(Paris,City)`. Previous work has shown that textual patterns can be surprisingly reliable [17]. One major problem, however, is the fact that occurrences of such patterns tend to be very rare. For instance, in a 20 million word New York Times article collection, Hearst found only 46 facts [17]. This paucity has been an important bottleneck with respect to the goal of establishing large-scale repositories of structured data.

Most information extraction systems have to date only been evaluated on small corpora, typically consisting of a couple of thousand [31] or perhaps a couple of hundred thousand documents [2]. Previous studies [7] have found that much greater precision and recall is possible by turning to search engines to benefit from Web-scale document collections. This, however, is shown to slow down the extraction process by several orders of magnitude. In this paper, we consider an alternative strategy. Instead of operating on specific document collections, we turn to n-gram statistics computed from large fractions of the entire Web, consisting of giga-scale numbers of documents. An n-gram is a sequence of $n$ consecutive items in text. While character n-grams, i.e. sequences of consecutive characters, are useful for tasks like language identification and data compression, word n-grams, i.e. sequences of consecutive word tokens, are useful for a wide range of tasks as listed in Section 2. An n-gram dataset is a resource that, for a given sequence of $n$ words, lists the corresponding occurrence frequency or probability of that string in a large document collection. Some of the available n-gram datasets [32] are computed from petabytes of the documents, yet such n-gram statistics have not been used in information extraction systems in previous work.

In this paper we a) discuss in which cases n-gram statistics may be used for information extraction b) present a framework to extract facts from n-grams and c) provide experimental results comparing different relations, n-gram datasets, and learning algorithms. The remainder of this paper is organized as follows. Section 2 discusses related work in this area. Section 3 lays out our approach to information extraction from n-gram data. Extensive experimental results are presented in Section 4. Finally, Section 5 provides concluding remarks.

## 2. RELATED WORK

The idea of searching for occurrences of specific textual patterns in text to extract information has a long history. Patterns for finding `isA` relationships were first discussed by Lyons [22] and Cruse [9]. Hearst [17] empirically evaluated the reliability of such patterns on corpora, and proposed a method for finding them automatically using seed instances. Since then, a large range of approaches have built upon these ideas, extending them to other relationships like `partOf` [13] as well as factual knowledge like birth dates of people and capital cities of countries [7].

Several projects studied using much more sophisticated techniques to extract very specific information from within single web pages or even smaller units of text. The goal was usually to extract information from the text as completely as possible. Some used deeper NLP techniques to discover more complex extraction rules [28]. Other systems investigated segmentation algorithms like CRFs to partition citation references into individual parts (author names, title, year, etc.) [29]. Wrapper induction systems [20] attempted to discover rules for template-based web sites, e.g. to extract pricing information from vendor product pages, as used by Google's Product Search.

A different line of research focussed instead on obtaining better results by scaling up the size of the document collection. The idea is to avoid making risky decisions for individual pieces of information, but instead exploit the redundancy of information in large document collections. Agichtein [1] and Pantel et al. [25] considered techniques to scale pattern-based approaches to larger corpora, however even these corpora represent only a very small fraction of the Web. Since Web-scale document collections are not easily obtainable, other approaches relied on web search engines to find instances [11, 24]. Approaches that directly rely on search engines interfaces face major challenges. Those that use search engines to discover new tuples will generally only retrieve top-$k$ results for a limited $k$ without being able to exploit the large amounts of facts that are in the long tail. For example, a query like "*such as*" can not be used on its own to retrieve very large numbers of `isA` pairs. Those approaches that use search engines to derive more information for specific output tuples first need to obtain the set of candidates from some other source, usually a much smaller document collection or perhaps pre-existing lists of named entities or words, which is likely to be limited in coverage. Cafarella et al. [7] investigated the costs of relying on slow commercial search engine interfaces in great detail. Their study shows how using a local document collection can speed up the time for extracting information from days to minutes. However, it also shows that this entails a significant loss of precision and recall.

This is why n-gram data is often a better alternative. Downey et al. [10] presented a post-processor that double-checks output tuples delivered by an information extraction system using n-gram language models. Their system checks if the arguments have similar contexts as seed arguments to ensure that they are of the correct type (e.g. city names). Additionally, they use general similarity of contexts to aid in assessing the overall validity of output pairs.

N-gram frequencies have also been used for other tasks. Previous work used online search engines to obtain string frequencies on-the-fly [21] for tasks like context-sensitive spelling correction, ordering of prenominal adjectives, compound bracketing, compound interpretation, and countability detection. Pre-computed n-gram frequencies have several advantages, however, including much greater scalability and reproducibility of experimental results. N-gram datasets have been used for statistical machine translation [5], spelling correction [19], word breaking to split "*baseratesoughtto*" into the most likely "*base rates ought to*" [32], and search query segmentation to segment web search engine queries like "*mike siwek laywer mi*" into "*mike siwek*", "*lawyer*", "*mi*" [18]. In our study, the n-gram data is used to find patterns and extract structured information.

## 3. RELYING ON N-GRAM STATISTICS

An n-gram dataset $f$ is a resource that accepts n-gram query strings $s = s_1 \cdots s_n$ consisting of $n$ consecutive tokens, and returns scores $f(s)$ based on the occurrence frequency of that particular string of tokens in a large document collection. In the simplest case, $f(s)$ values will just be raw frequency counts, e.g. $f$("*major cities like London*") would yield the number of times the string "*major cities like London*" occurs in the document collection. Some n-gram resources instead provide probabilities based on smoothed language models [32]. Additionally, we also consider n-gram datasets supporting wildcards "*?*" or regular expressions like "***", in which case $f$ returns a set of string-score tuples.

Some of the dangers of relying on Web n-grams include:

- Influence of spam and boilerplate text: Large portions of the Web consist of automatically generated text. This can be due to simple boilerplate text in templates, which is repeated on many pages, or due to malicious (spam) web sites, consisting of non-sensical text, often replicated millions of times on the same server or an entire network of servers.

- Less control over the selection of input documents: With conventional extraction systems, one typically chooses a document collection likely to contain relevant information. N-gram data is usually not limited to a specific domain. Among other things, this may mean that issues like polysemy become more of a problem.

- Less context information: Word sense disambiguation and part-of-speech tagging are more difficult when considered out-of-context.

- Less linguistically deep information: Approaches that analyse entire documents can rely on sophisticated NLP techniques like pronoun resolution and dependency parsing to analyse sentences.

We focus on binary relationships between entities. Since n-gram statistics are usually limited to rather small $n$, more complex $m$-ary relations between more than $m > 2$ entities usually cannot be extracted very well, unless they can be broken down into multiple independent binary relationships. For example, the founding year and founding location of a company can be treated independently, but the fact that country "*<V> imported <W> million dollars worth of <X> from country <Y> in year <Z>*" could not be broken down in a similar manner.

The following additional conditions apply:

- Very short items of interest: The words or entities of interest that should fill the placeholders in patterns

should be very short units, preferably just unigrams. For instance, it is possible to search for a relationship between "*Kyoto*" and "*Japan*" or between "*Mozart*" and "*1756*". However, it may not be possible to search for relationships between multiple people with their full names, or to find instances of the `hasMotto` relationship from YAGO [30], which connects an entity like `Connecticut` to a slogan like "*Full of Surprises*". Moving to such longer units requires long n-grams that are not captured in a normal n-gram dataset.

- Knowledge that is typically expressed using very short patterns: This approach will work only for relations that are often expressed using very short expressions. It will not work well when such knowledge is usually spread out over several sentences or expressed using longer patterns, e.g. "*<X> has an inflation rate of <Y>*". Even if only a small fraction of all theoretically possible n-grams actually occur in document collection, scaling n-gram datasets up to large $n$ becomes unpractical very quickly due to the combinatorial explosion. There may be work-arounds like using a preclassifier that acts as a filter and selects only those sentences for further indexing that are likely to contain useful knowledge, based on document topic, sentence interestingness (e.g. the presence of certain keywords in the sentence), etc.

The primary motivation for working with n-gram data is to provide information extraction systems with more input in order to obtain a better coverage as well as accuracy. A larger document collection as input not only means more output, but also output of higher quality. Due to the greater amount of redundancy, the system has more information to base its assessment on. This may also have implications on the model that the system is based on. Pantel et al. [25] showed that scaling to larger text collections alone can allow a rather simple technique to outperform much more sophisticated algorithms.

Certainly, one could obtain similar effects by running traditional information extraction systems on corpora as large as the document collections used to compute the n-gram statistics. However, there are several reasons why one may not choose that option:

- Availability: The carefully edited, balanced corpora traditionally used in NLP are comparatively small, e.g. the British National Corpus (BNC) consists of 100M tokens, and even Web document collections like the GigaWord Corpus [15] are several orders of magnitude smaller than the web-scale document collections that some of the available n-gram datasets are based on. For instance, Google's Web1T dataset is computed from a tera-word document collection. For most researchers, crawling the Web to obtain similar web-scale collections would be a slow and expensive process requiring sophisticated infrastructure (e.g. web link farm detection).

- Re-usability of N-Gram Data: N-gram statistics can be used for a variety of different purposes, apart from information extraction. In Section 2, we gave examples such as machine translation and spelling correction. Hence, there are many incentives to create such highly reusable n-gram data.

Our approach will be to first gather a set of patterns that allow us to identify candidate tuples. The n-gram frequency statistics for the candidate tuples as occurring with specific patterns are then used to derive a vector representation for each candidate tuple. Based on a training set of labelled tuples, a learning algorithm finally determines which candidate tuples should be accepted.

## 3.1 Pattern Induction

Our system begins with the pattern induction step, where it attempts to bootstrap the extraction starting out with just a set of correct seed instances of each relation under consideration. For instance, for the `partOf` relation, it could use a list including `(finger,hand)`, `(leaves,trees)`, and `(windows,houses)`. For the `isA` relation, seed patterns can include `(dogs,animals)` and `(gold,metal)`. The goal of this step is to obtain a list of simple textual patterns that can then be used to harvest further knowledge from the corpora.

The textual patterns can be obtained either from any conventional corpus, or from an n-gram dataset that supports queries with regular expressions or at least wildcards. For example, given a seed pair `(dogs,animals)` for the `isA` relation, the queries "*dogs * animals*" and "*animals * dogs*" may return an n-gram like "*dogs and other animals*". By replacing the seed words with wild cards, we obtain the pattern "*<X> and other <Y>*". For n-gram resources that do not support regular expression queries, one can use wildcard queries "*dogs ? animals*", "*dogs ? ? animals*", and so on. If wildcard queries are not supported, one can use a separate conventional text document collection, and look for the two seed words co-occurring within a small window.

For efficiency reasons and also to remove noise, one can remove some of the least useful patterns as determined by their Pointwise Mutual Information (PMI) score. Many information extraction systems have relied on PMI scores alone to produce the final scoring [24]. In our setup, we only make use of extremely low PMI scores in order to prune the pattern list.

## 3.2 Extraction and Assessment

After the first step, we have a set $P$ of patterns that are likely to be at least somewhat relevant for a given relation. A given pattern $p \in P$ can be instantiated for specific candidate tuples $(x, y)$ as $p(x, y)$ to yield an n-gram string. For instance, a pattern like "*<X> is located in <Y>*" can be instantiated with a tuple `(Paris,France)` to yield an n-gram "*Paris is located in France*". For such n-grams, we can then consult an n-gram dataset $f$ to obtain (raw or manipulated) frequency information $f(p(x, y))$ that reveals how frequent the n-gram is on the Web.

Ideally, in a large corpus like the Web, a given tuple will occur frequently with more than one pattern. Not all patterns are equally reliable, however, and PMI scores alone are not the best indicators of reliability. Hence, we opted for a supervised approach. This requires a set of examples that are manually labelled as positive (the two items do stand in the respective relation to each other) or negative (they do not stand in the given relationship to each other). For example, for the `hasProperty` relationship, the pair `(apple,edible)` would be considered positive and `(apple,new)` would be a negative pair.

For a set of $m$ patterns $P = \{p_1, \ldots, p_m\}$ , $k$ n-gram data sources $f_j$ $(j = 1 \ldots k)$, and a given pair of words $(x, y)$, we

**Algorithm 1** N-Gram Information Extraction

**procedure** HARVEST(n-gram datasets $\{f_i\}$, seeds $S$, labeled set $T$ with labels $l_{(x,y)} \in \{-1, +1\}$ for $(x, y) \in T$)
  $P \leftarrow$ INDUCE_PATTERNS$(f_1, S)$ ▷ **collect** large numbers of **patterns**, assuming $f_1$ supports wildcard queries
  **for all** $(x, y) \in T$ **do** ▷ create **training vectors** for pairs in $T$
    create training vector $\mathbf{v}_{(x,y)}$ using patterns in $P$ and n-gram datasets $f_i$ ▷ see Equation 1
  learn model $M$ from $\{(\mathbf{v}_{(x,y)}, l_{(x,y)}) \mid (x, y) \in T\}$ ▷ use **learning algorithm**
  $K \leftarrow$ EXTRACT$(\{f_i\}, P, M)$ ▷ **collect** and assess large numbers of tuples
  **return** accepted tuples $K$ ▷ accepted tuples as final **output**
**procedure** INDUCE_PATTERNS(n-gram dataset $f$, seeds $S$)
  $P \leftarrow \emptyset$ ▷ set of patterns
  **for all** $(x, y) \in S$ **do** ▷ for all seed pairs $(x, y)$
    **for all** $s$ from $f(x * y) \cup f(y * x)$ **do** ▷ find n-grams $s$ containing seed words $x$ and $y$
      $P \leftarrow P \cup \{$CREATEPATTERN$(s, x, y)\}$ ▷ replace $x$ and $y$ in $s$ with wildcards (or empty set if pattern too unreliable)
  **return** patterns $P$
**procedure** EXTRACT(n-gram datasets $\{f_i\}$, patterns $P$, model $M$)
  $T \leftarrow \emptyset$ ▷ candidate tuples
  **for all** $p \in P$ **do** ▷ for all patterns
    **for all** $s$ from $f_1(p)$ **do** ▷ retrieve n-grams $s$ matching pattern
      $T \leftarrow T \cup \{(\text{ARGX}(p, s), \text{ARGY}(p, s))\}$ ▷ strings within $s$ matching wildcards in $p$ form new candidate tuples
  **return** $\{(x, y) \in T \mid M(\mathbf{v}_{(x,y)}) > 0.5\}$ ▷ vectors for candidate tuples are created using $P$ and $f_i$ and assessed using $M$

produce a $k(m + 1)$-dimensional vector $\mathbf{v}_{(x,y)}$ to represent the corresponding pattern statistics.

$$v_{(x,y),i} = \begin{cases} \sum_{p \in P} f_{\phi(i)}(p(x, y)) & \sigma(i) = 0 \\ f_{\phi(i)}\left(p_{\sigma(i)}(x, y)\right) & \text{otherwise} \end{cases} \quad (1)$$

where $\phi(i) = \left\lfloor \frac{i}{m+1} \right\rfloor$ and $\sigma(i) = i \mod (m + 1)$. The idea is that for each n-gram dataset $f_j$, we have $|P|$ individual features $f_j(p(x, y))$ that capture pattern-specific frequency scores, as well as a single feature that captures the total frequency score $\sum_{p \in P} f_j(p(x, y))$.

Algorithm 1 gives the overall procedure for extracting information. Given the set of vectors $\mathbf{v}_{(x,y)}$ for the training examples $(x, y)$ together with their respective labels $l_{(x,y)}$, a learning algorithm like support vector machines is used to derive a prediction model $M$ for new pairs $(x, y)$. The prediction model $M$ provides values $M(\mathbf{v}_{(x,y)}) \in [0, 1]$ for the vectors of new pairs $(x, y)$, where values over 0.5 mean that the pair is accepted. The specific learning algorithms used to derive the prediction model are listed in Section 4. It is assumed that the first n-gram dataset $f_1$ supports wildcards, so for a pattern like "*<X> is located in <Y>*" we can simply query for "*? is located in ?*". The union $T$ of all tuples matching patterns in $P$ is the set of candidate tuples. Note that for the Google Web 1T dataset, we could simply iterate over all n-grams in the dataset, and see which n-grams match any of the patterns. Those candidate tuples predicted to be true by the classification model are the ones that constitute the final output.

# 4. EXPERIMENTAL RESULTS

## 4.1 N-Gram Datasets

We used three different sources of n-gram statistics.

- The Google Web 1T N-Gram Dataset (Version 1) [4]: Google has published a dataset of raw frequencies for n-grams ($n = 1, \ldots, 5$) computed from over 1,024G word tokens of English text, taken from Google's web page search index. In compressed form, the distributed data amounts to 24GB. While the dataset does not include n-grams with a frequency of less than 40, the fact that it is distributed as a complete dataset means that additional post-processing and indexing can be applied to support a more sophisticated query syntax.

- The Microsoft Web N-gram Corpus [32]: Microsoft's Web N-gram Corpus is based on the complete collection of documents indexed for the English US version of the Bing search engine. In addition to offering n-gram language models computed from the document bodies (1.4T tokens), it also offers separate n-gram statistics computed from document titles (12.5G tokens) and document anchor texts (357G tokens). Experiments have shown that the latter two have rather different properties [32]. The dataset is not distributed as such but made accessible by means of a Web service described using the WSDL standard. The service provides smoothed n-gram language models rather than raw frequencies, generated using an algorithm that dynamically incorporates new n-grams as they are discovered [18].

- The ClueWeb09 collection[1]: We also evaluated our approach on raw 5-gram frequencies (roughly 700M 5-grams) computed from the complete English subset of the ClueWeb collection, i.e. from over 500 million web pages. The ClueWeb09 collection is also used in several TREC tasks.

## 4.2 Patterns

We chose 3 relations that we believe fulfill the conditions mentioned earlier in Section 3. These are listed in Table 1. We used ConceptNet [16] as a source of 100 tuples per relation, selecting the top-ranked 100 as seeds. Upon further inspection, we noticed that some of the seeds are not correct, despite having high scores in ConceptNet, e.g. we found tuples like `partOf(children,parents)` and `isA(winning,everything)` in ConceptNet. Such incorrect tuples stem from the crowdsourcing approach that ConceptNet has adopted for gathering knowledge, where just a few input sentences like "*winning is everything*" suffice to give the `isA`-tuple a

[1] `http://boston.lti.cs.cmu.edu/Data/clueweb09/`

**Table 1: Relations**

| Relation | Seeds | Patterns discovered | Labelled examples All | Labelled examples Positive |
|---|---|---|---|---|
| `isA` | 100 | 2991 | 530 | 99 (19%) |
| `partOf` | 100 | 3883 | 516 | 208 (40%) |
| `hasProperty` | 100 | 3175 | 400 | 297 (74%) |

**Table 2: Patterns for `isA`**

| Pattern | PMI range |
|---|---|
| <X> *and almost any* <Y> | high |
| <X> *and some other* <Y> | high |
| <X> *betting basketball betting* <Y> | high |
| <X> *online* <Y> | high |
| <X> *is my favorite* <Y> | high |
| <X> *shoes online shoes* <Y> | high |
| <X> *wager online* <Y> | high |
| <X> *is a* <Y> | medium |
| <X> *is a precious* <Y> | medium |
| <X> *is the best* <Y> | medium |
| <X> *or any other* <Y> | medium |
| <X> *, and* <Y> | medium |
| <X> *and other smart* <Y> | medium |
| <X> *and small* <Y> | medium |
| <X> *and grammar* <Y> | low |
| <X> *content of the* <Y> | low |
| <X> *or storage* <Y> | low |
| <X> *when it changes* <Y> | low |

high score. Fortunately, our approach is robust with respect to inaccurate seed tuples for pattern induction. Our approach later relies on the pattern statistics for hundreds or thousands of induced patterns (rather than just "<X> *is* <Y>") to assess the validity of tuples. The learning algorithm should automatically learn that a pattern like "<X> *is* <Y>" alone is too unreliable.

Table 2 shows examples of top, middle, and bottom-ranked patterns for the `isA` relation, based on the Google Web 1T dataset. Tables 3 provides similar information for the `partOf` relation. We see that PMI alone is not a reliable estimator of pattern goodness, as some top-ranked patterns are wrong. In particular, we notice the effects of web spam. Many of the best patterns actually have mid-range PMI scores. Patterns with very low PMI scores, are indeed either incorrect or useless (too rare). This shows why it makes sense to mistrust the PMI values when working with large-scale web data. Supervised approaches have a much better chance of determining which patterns are most reliable. Overall, it is apparent that the patterns are very diverse. Using web-scale data, even fairly rare patterns can give us significant amounts of correct output tuples.

## 4.3 Labelled Data

For training and testing, we randomly chose tuples for the relations among all word pairs matching any of the patterns for the respective relation in the Google Web1T corpus. These tuples were then labelled manually as either correct or incorrect. The number of labelled examples is again given in Table 1. Note that these tuples are distinct from the initial seed tuples.

**Table 3: Patterns for `partOf`**

| Pattern | PMI range |
|---|---|
| <X> *with the other* <Y> | high |
| <X> *on your right* <Y> | high |
| <X> *of the top* <Y> | high |
| <X> *online* <Y> | high |
| <X> *and whole* <Y> | high |
| <X> *shoes online shoes* <Y> | high |
| <X> *wager online* <Y> | high |
| <X> *from the* <Y> | medium |
| <X> *or even entire* <Y> | medium |
| <X> *of host* <Y> | medium |
| <X> *from* <Y> | medium |
| <X> *on the* <Y> | medium |
| <X> *appearing on* <Y> | medium |
| <X> *of a different* <Y> | medium |
| <X> *entertainment and* <Y> | low |
| <X> *Download for thou* <Y> | low |
| <X> *affects your daily* <Y> | low |
| <X> *company home in* <Y> | low |

**Table 4: Overall Results**

| Relation | Precision | Recall | $F_1$ | Output per million n-grams[1] |
|---|---|---|---|---|
| `isA` | 88.9% | 8.1% | 14.8% | 983 |
| `partOf` | 80.5% | 34.0% | 47.8% | 7897 |
| `hasProperty` | 75.3% | 99.3% | 85.6% | 26180 |

1: the expected number of distinct accepted tuples per million input n-grams (the total number of 5-grams in the Google Web 1T dataset is ∼1,176 million)

As the main learning algorithm, we used RBF-kernel support vector machines as implemented in LIBSVM [8]. In Section 4.5, we additionally report and evaluate results obtained with alternative algorithms.

Given a set of positive and negative examples, precision, recall, and $F_1$ scores can be computed in the standard way with respect to the set. This means that the recall scores will be estimates based on the union of all results for all patterns in the Web 1T collection, which is equivalent to the pooling strategy adopted in TREC evaluations. We rely on 10-fold leave-one-out cross-validation, where the set of labelled examples is randomly partitioned into 10 equal-size parts, and an average score is computed over 10 runs. In each run, a different part is reserved for testing, and the remaining 9 parts are used as the training set.

## 4.4 Accuracy and Coverage

Table 4 gives the overall results for using RBF-kernel support vector machines simultaneously relying on all available n-gram datasets. The table lists the average precision, average recall, and average $F_1$ score, i.e. the harmonic mean of precision and recall. Since the recall was computed with respect to the union of all tuples matching any of the patterns in the Web-scale Goole Web 1T dataset, even very

low recall values mean that we can get hundreds of thousands of high-quality output tuples if we run over the entire dataset. We ran an additional experiment on a sample of 1 million input 5-grams to find the number of unique candidate tuples. The final column of Table 4 lists the expected number of distinct tuples accepted when running over a million n-grams. These figures were computed by counting the number of unique candidate tuples in the sample of one million n-grams, extrapolating the number of *true* tuples among those candidate tuples using the positive to negative ratio in the training set distributions, and then figuring in the recall values obtained by the classification model. The total number of 5-grams in the Google Web 1T dataset is over 1,176 million and additionally there are also 1,313 million 4-grams and 977 million trigrams, but a linear extrapolation to the entire dataset is not possible due to duplicates.

The overall results show that large numbers of high-accuracy output tuples can be obtained using n-gram data. Even greater numbers of tuples can be obtained by extending the range of patterns considered, as laid out in Section 5. In Section 3 we mentioned that an n-gram approach has less access to linguistic processing. For example, Hearst's study of extracting `isA` tuples from text used NLP techniques to filter the results. Using n-gram datasets, we can obtain similar results simply by relying on a greater range of patterns. Many linguistic features are implicitly captured via combinations of patterns.

Studying the results in further detail, we observe that there are great differences between the relations considered. The `isA` relation seems to be the hardest relation to extract with high coverage. It seems that many tuples are found that match a rather generic pattern like "*<X> , and <Y>*" and happen to be true `isA` pairs, but which do not occur frequently enough using more reliable patterns like the Hearst patterns [17]. The `partOf` relation is somewhat easier, and finally for the `hasProperty` relation, both precision and recall are very high.

## 4.5 Detailed Analysis

We conducted more specific experiments to evaluate the role of the size of the n-grams and differences between datasets. Table 5 provides results for the `partOf` relation. The results are reported for the random forest algorithm [6], as it balanced precision and recall more and thus produced results that are easier to compare than RBF-kernel SVMs. Overall, SVMs showed similar results and trends, however they sometimes preferred obtaining a high accuracy by classifying all examples as negative examples, leading to a recall of 0%.

Several interesting observations can be made. First of all, the Microsoft n-grams and Google n-grams show similar results on the document body, but by combining information from document bodies, titles, and anchor texts, Microsoft outperforms both its own individual 3-gram datasets as well as Google's 3-grams. This shows that the titles and anchor texts offer complementary information that may differ from what is found in the document body for certain patterns. Essentially, this means that in the future we may see a more diverse range of n-gram datasets being used in different applications rather than a simple one-size-fits-all solution. For example, we may find specific n-gram datasets being offered, based only on news text, only on blogs, or only on documents found in the `.edu` domain. We may also see n-gram datasets derived from medical journals or even other modalities like speech transcripts. Our findings indicate that even for a single application it can make sense to combine multiple datasets using supervised approaches or mixture models.

While Microsoft's diversity is beneficial for the model accuracy, distributable datasets like the Google Web 1T dataset also have a number of significant advantages for information extraction, as they enable iterating over the data to induce patterns and again to determine the large set of candidate tuples. While equivalent results could also be obtained by using service-based n-gram datasets that support regular expression or wild card searches, the API for the Microsoft n-gram dataset does not currently have these features. One advantage of web services is that they can be easier to use than distributable datasets, as not much infrastructure is required to access a web service, while it is non-trivial to handle the large volumes of data provided in distributable n-gram datasets.

In Figure 1, we plotted the $\sum_{p \in P} f_{\phi(i)}(p(x,y))$ score from Equation 1 for the labelled `isA` examples for Microsoft document body 3-grams against the corresponding scores with Microsoft anchor text 3-grams, title 3-grams, and Google Web 1T document body 3-grams. We see that the scores from the Microsoft body text are quite different than the ones from the anchor texts, while the correlation with titles and with the Google n-gram dataset is higher.

We further observe in Table 5 that different n-gram sizes show somewhat similar results, but that by combining different n-gram sizes improved results are obtained. Nevertheless, it should be pointed that longer n-grams still are generally preferred when wildcard queries are possible, because an $n$-gram dataset can then also be used to find patterns with lengths $n' < n$. For instance, for a pattern of length 3 like "*<X> are <Y>*" we could use 5-gram queries like "*apples are green ? ?*" and "*? ? apples are green*". In our experiments, to enable a fair comparison with datasets that do not offer wildcard support, for each $n$-gram dataset we only considered patterns with a respective corresponding length of exactly $n$. Additionally, if there is a minimal frequency threshold, as for the Web 1T dataset, then longer n-grams may also exhibit more sparsity than shorter ones.

Finally, these results suggest that even at a very large scale more data is better data. We can obtain more precise results by moving towards even larger document collections. The ClueWeb collection, which is already based on over 500 million documents, does not give the same results as the larger Google Web 1T collection. The Microsoft dataset gives even better results. In related experiments, we noted that the cut-off in the Google n-gram corpus was limiting. For example, we investigated obtaining patterns for the `imports` and `exports` relations from YAGO [30], e.g. `imports(China,Coal)`. However, due to the cut-off we found only very few patterns. Another reason, of course, is that such relations are typically expressed using longer patterns, e.g. "*<X> is the main exporter of <Y>*".

In an additional experiment, we evaluated different learning algorithms. Table 6 compares RBF-kernel SVMs [8], random forests [6], Adaptive Boosting M1 [12], and C4.5 Decision Trees [27] using all n-gram datasets together. The algorithms perform similarly for `hasProperty`, while for the other two relations LibSVM delivers a higher precision at the

**Table 5: Differences between n-gram datasets (`partOf` relation)**

| Dataset | N-gram length | Source | Precision | Recall | $F_1$ |
|---|---|---|---|---|---|
| Microsoft | 3-grams | Document Body | 58.5% | 33.2% | 42.3% |
| Microsoft | 3-grams | Document Title | 51.7% | 29.8% | 37.8% |
| Microsoft | 3-grams | Anchor Text | 57.3% | 36.1% | 44.2% |
| Microsoft | 3-grams | Body / Title / Anchor | 40.4% | 100.0% | 57.5% |
| Google Web 1T | 3-grams | Document Body | 55.9% | 38.5% | 45.6% |
| Microsoft | 4-grams | Document Title | 55.8% | 34.6% | 42.7% |
| Microsoft | 4-grams | Anchor Text | 49.6% | 27.4% | 35.3% |
| Microsoft | 4-grams | Title / Anchor | 40.3% | 100.0% | 57.3% |
| Google Web 1T | 4-grams | Document Body | 52.6% | 43.3% | 47.5% |
| Google Web 1T | 5-grams | Document Body | 48.1% | 42.8% | 45.3% |
| ClueWeb | 5-grams | Document Body | 51.7% | 35.6% | 42.2% |
| Microsoft | 3-/4-grams | Body (3-grams only) / Title / Anchor | 40.5% | 98.1% | 57.3% |
| Google Web 1T | 3-/4-grams | Document Body | 53.9% | 42.8% | 47.7% |
| Google Web 1T | 3-/4-/5-grams | Document Body | 58.7% | 43.8% | 50.1% |

**Table 6: Alternative learning algorithms**

| | SVM | Random Forests | AdaBoost | C4.5 |
|---|---|---|---|---|
| `isA` | | | | |
| Precision | 88.9% | 25.5% | 18.4% | 26.8% |
| Recall | 8.1% | 49.5% | 94.9% | 49.5% |
| $F_1$ | 14.8% | 33.7% | 30.8% | 34.8% |
| `partOf` | | | | |
| Precision | 80.5% | 45.0% | 39.4% | 48.0% |
| Recall | 34.0% | 67.8% | 80.3% | 46.2% |
| $F_1$ | 47.8% | 54.1% | 52.8% | 47.1% |
| `hasProperty` | | | | |
| Precision | 75.3% | 74.4% | 74.3% | 74.6% |
| Recall | 99.3% | 98.7% | 100.0% | 100.0% |
| $F_1$ | 85.6% | 84.8% | 85.2% | 85.5% |

expense of a lower recall. Overall, there is no clear winner.

# 5. CONCLUSION AND FUTURE WORK

We have introduced a framework for extracting structured data from Web-scale n-gram datasets derived from petabytes of original data. Our study highlights the enormous possibilities as well as the limitations of such an approach. Not all types of information can be harvested from n-gram data, and less contextual information is available. However, for semantic relations between words and short named entities, n-gram datasets allow us to go beyond the small corpora used in current information extraction systems. Although the recall scores are low relative to what is theoretically available on the Web, we are able extract very large quantities of data and tune the accuracy.

In future work, we will study to what extent patterns with additional context can improve the results, e.g. patterns like "*<X> are <Y>*" and "*<X> is a <Y>*" are rather unreliable for the `isA` relation. However, if we consider variants that are augmented with context words, we may obtain much more reliable patterns, e.g. "*all <X> are <Y>*" or "*every <X> is a <Y>*". Also, so far, we have only considered single tokens as names, e.g. "*Paris*", "*IBM*", "*gold*", "*grass*". In future work, we would like to extend our work to short multi-token units like "*New York*" and "*mobile phone*". While simple heuristics may work, a proper treatment might require more powerful structured prediction models [29]. Multi-stage approaches are also being considered, where one first uses the n-grams to assess whether a word is eligible for further consideration (e.g. the second argument of `isA` should preferably be a countable noun), and then judges the validity. Finally, we would like to put our work into practice to generate a freely available large-scale database of structured knowledge.

# 6. REFERENCES


[1] E. Agichtein. Scaling information extraction to large document collections. *IEEE Data Engineering Bulletin*, 28, 2005.

[2] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proc. 5th ACM Conference on Digital Libraries (DL '00)*, New York, NY, USA, 2000. ACM.

[3] H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: Efficient search in text, entities, and relations. In *Proc. SIGIR*, Amsterdam, Netherlands, 2007. ACM.

[4] T. Brants and A. Franz. Web 1T 5-gram Version 1. *Linguistic Data Consortium, Philadelphia*, 2006.

[5] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proc. EMNLP-CoNLL 2007*, 2007.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP*. Association for Computational Linguistics, 2005.

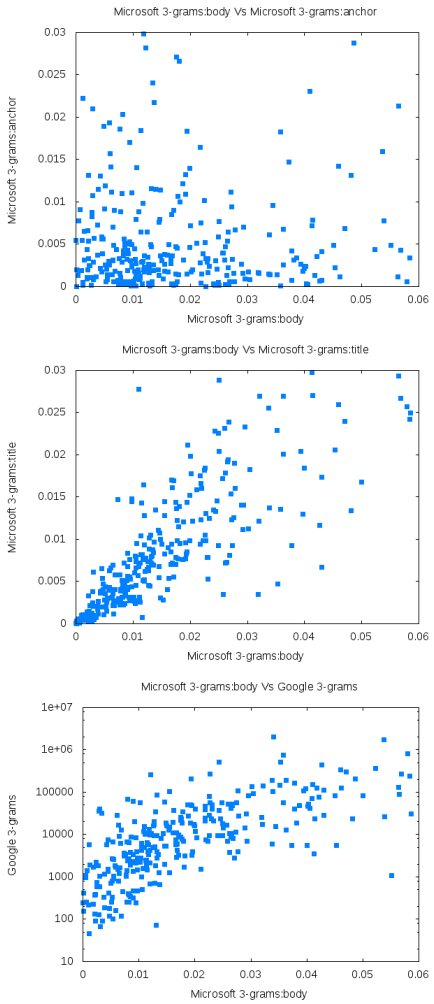


**Figure 1: Plots comparing Microsoft Document Body 3-grams with Anchor 3-grams (above), Title 3-grams (middle), and Google Body 3-grams (bottom)**


[8] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[9] D. A. Cruse. *Lexical Semantics*. Cambridge University Press, Cambridge, UK, 1986.

[10] D. Downey, S. Schoenmackers, and O. Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *Proc. ACL*, 2007.

[11] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in KnowItAll: Preliminary results. In *Proc. WWW 2004*, 2004.

[12] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. ICML 1996*, San Francisco, 1996. Morgan Kaufmann.

[13] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Comput. Linguist.*, 32(1):83–135, 2006.

[14] Z. Gong, C. W. Cheang, and L. H. U. Web query expansion by WordNet. In *Proc. DEXA 2005*, volume 3588 of *LNCS*. Springer, 2005.

[15] D. Graff and C. Cieri. English Gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

[16] C. Havasi, R. Speer, and J. Alonso. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Proc. RANLP 2007*, Borovets, Bulgaria, 2007.

[17] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. COLING*, 1992.

[18] 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 *Proc. WWW 2010*, New York, NY, USA, 2010. ACM.

[19] A. Islam and D. Inkpen. Real-word spelling correction using google web 1tn-gram data set. In *Proc. CIKM 2009*, New York, NY, USA, 2009. ACM.

[20] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *Proc. IJCAI Volume 1*, 1997.

[21] M. Lapata and F. Keller. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *Proc. HLT-NAACL*, 2004.

[22] J. Lyons. *Semantics, vol. 1*. Cambridge University Press, Cambridge, UK, 1977.

[23] D. N. Milne, I. H. Witten, and D. M. Nichols. A knowledge-based search engine powered by Wikipedia. In *Proc. CIKM 2007*, New York, NY, USA, 2007.

[24] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proc. ACL 2006*. ACL, 2006.

[25] P. Pantel, D. Ravichandran, and E. Hovy. Towards terascale knowledge acquisition. In *Proc. COLING 2004*, Morristown, NJ, USA, 2004.

[26] S. Paparizos, A. Ntoulas, J. Shafer, and R. Agrawal. Answering web queries using structured data sources. In *Proc. SIGMOD 2009*, New York, NY, USA, 2009.

[27] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[28] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, 1993.

[29] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *Proc. NIPS*, 2004.

[30] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proc. WWW 2007*, New York, NY, USA, 2007. ACM Press.

[31] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: a self-organizing framework for information extraction. In *Proc. WWW 2009*, New York, NY, USA, 2009. ACM.

[32] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. P. Hsu. An overview of Microsoft Web N-gram corpus and applications. In *Proc. NAACL HLT 2010 Demonstration Session*, Los Angeles, CA, USA, June 2010.