Open Information Extraction from the Web

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{banko,mjc,soderlan,hastur,etzioni}@cs.washington.edu

Abstract

Traditionally, Information Extraction (IE) has focused on satisfying precise, narrow, pre-specified requests from small homogeneous corpora (e.g., extract the location and time of seminars from a set of announcements). Shifting to a new domain requires the user to name the target relations and to manually create new extraction rules or hand-tag new training examples. This manual labor scales linearly with the number of target relations.

This paper introduces *Open IE* (OIE), a new extraction paradigm where the system makes a single data-driven pass over its corpus and extracts a large set of relational tuples without requiring *any* human input. The paper also introduces TEXTRUNNER, a fully implemented, highly scalable OIE system where the tuples are assigned a probability and indexed to support efficient extraction and exploration via user queries.

We report on experiments over a 9,000,000 Web page corpus that compare TEXTRUNNER with KNOWITALL, a state-of-the-art Web IE system. TEXTRUNNER achieves an error reduction of 33% on a comparable set of extractions. Furthermore, in the amount of time it takes KNOWITALL to perform extraction for a handful of pre-specified relations, TEXTRUNNER extracts a far broader set of facts reflecting orders of magnitude more relations, discovered on the fly. We report statistics on TEXTRUNNER's 11,000,000 highest probability tuples, and show that they contain over 1,000,000 concrete facts and over 6,500,000 more abstract assertions.

1 Introduction and Motivation

This paper introduces *Open Information Extraction (OIE)*—a novel extraction paradigm that facilitates domain-independent discovery of relations extracted from text and readily scales to the diversity and size of the Web corpus. The sole input to an OIE system is a corpus, and its output is a set of extracted relations. An OIE system makes a single pass over its corpus guaranteeing scalability with the size of the corpus.

Information Extraction (IE) has traditionally relied on extensive human involvement in the form of hand-crafted extraction rules or hand-tagged training examples. Moreover, the user is required to explicitly pre-specify each relation of interest. While IE has become increasingly automated over time, enumerating all potential relations of interest for extraction by an IE system is highly problematic for corpora as large and varied as the Web. To make it possible for users to issue diverse queries over heterogeneous corpora, IE systems must move away from architectures that require relations to be specified prior to query time in favor of those that aim to discover all possible relations in the text.

In the past, IE has been used on small, homogeneous corpora such as newswire stories or seminar announcements. As a result, traditional IE systems are able to rely on "heavy" linguistic technologies tuned to the domain of interest, such as dependency parsers and Named-Entity Recognizers (NERs). These systems were not designed to scale relative to the size of the corpus or the number of relations extracted, as both parameters were fixed and small.

The problem of extracting information from the Web violates all of these assumptions. Corpora are massive and heterogeneous, the relations of interest are unanticipated, and their number can be large. Below, we consider these challenges in more detail.

Automation The first step in automating IE was moving from knowledge-based IE systems to trainable systems that took as input hand-tagged instances [Riloff, 1996] or document segments [Craven *et al.*, 1999] and automatically learned domain-specific extraction patterns. DIPRE [Brin, 1998], SNOWBALL [Agichtein and Gravano, 2000], and Web-based question answering systems [Ravichandran and Hovy, 2002] further reduced manual labor needed for relation-specific text extraction by requiring only a small set of tagged seed instances or a few hand-crafted extraction patterns, per relation, to launch the training process. Still, the creation of suitable training data required substantial expertise as well as non-trivial manual effort *for every relation extracted*, and the relations have to be specified in advance.

Corpus Heterogeneity Previous approaches to relation extraction have employed kernel-based methods [Bunescu

and Mooney, 2005], maximum-entropy models [Kambhatla, 2004], graphical models [Rosario and Hearst, 2004; Culotta *et al.*, 2006], and co-occurrence statistics [Lin and Pantel, 2001; Ciaramita *et al.*, 2005] over small, domain-specific corpora and limited sets of relations. The use of NERs as well as syntactic or dependency parsers is a common thread that unifies most previous work. But this rather "heavy" linguistic technology runs into problems when applied to the heterogeneous text found on the Web. While the parsers work well when trained and applied to a particular genre of text, such as financial news data in the Penn Treebank, they make many more parsing errors when confronted with the diversity of Web text. Moreover, the number and complexity of entity types on the Web means that existing NER systems are inapplicable [Downey *et al.*, 2007].

Efficiency KNOWITALL [Etzioni *et al.*, 2005] is a state-of-the-art Web extraction system that addresses the automation challenge by learning to label its own training examples using a small set of domain-independent extraction patterns. KNOWITALL also addresses corpus heterogeneity by relying on a part-of-speech tagger instead of a parser, and by not requiring a NER. However, KNOWITALL requires large numbers of search engine queries and Web page downloads. As a result, experiments using KNOWITALL can take weeks to complete. Finally, KNOWITALL takes *relation names as input*. Thus, the extraction process has to be run, and re-run, each time a relation of interest is identified. The OIE paradigm retains KNOWITALL's benefits but eliminates its inefficiencies.

The paper reports on TEXTRUNNER, the first scalable, domain-independent OIE system. TEXTRUNNER is a fully implemented system that extracts relational tuples from text. The tuples are assigned a probability and indexed to support efficient extraction and exploration via user queries.

The main contributions of this paper are to:

- Introduce Open Information Extraction (OIE)—a new extraction paradigm that obviates relation specificity by automatically discovering possible relations of interest while making only a single pass over its corpus.
- Introduce TEXTRUNNER, a fully implemented OIE system, and highlight the key elements of its novel architecture. The paper compares TEXTRUNNER experimentally with the state-of-the-art Web IE system, KNOWITALL, and show that TEXTRUNNER achieves a 33% relative error reduction for a comparable number of extractions.
- Report on statistics over TEXTRUNNER's 11,000,000 highest probability extractions, which demonstrates its scalability, helps to assess the quality of its extractions, and suggests directions for future work.

The remainder of the paper is organized as follows. Section 2 introduces TEXTRUNNER, focusing on the novel elements of its architecture. Section 3 reports on our experimental results. Section 4 considers related work, and the paper concludes with a discussion of future work.

2 Open IE in TEXTRUNNER

This section describes TEXTRUNNER's architecture focusing on its novel components, and then considers how TEXTRUNNER addresses each of the challenges outlined in Section 1. TEXTRUNNER's sole input is a corpus and its output is a set of extractions that are efficiently indexed to support exploration via user queries.

TEXTRUNNER consists of three key modules:

1. **Self-Supervised Learner:** Given a small corpus sample as input, the Learner outputs a classifier that labels candidate extractions as "trustworthy" or not. The Learner requires no hand-tagged data.
2. **Single-Pass Extractor:** The Extractor makes a single pass over the entire corpus to extract tuples for all possible relations. The Extractor does *not* utilize a parser. The Extractor generates one or more candidate tuples from each sentence, sends each candidate to the classifier, and retains the ones labeled as trustworthy.
3. **Redundancy-Based Assessor:** The Assessor assigns a probability to each retained tuple based on a probabilistic model of redundancy in text introduced in [Downey *et al.*, 2005].

Below, we describe each module in more detail, discuss TEXTRUNNER's ability to efficiently process queries over its extraction set, and analyze the system's time complexity and speed.

2.1 Self-Supervised Learner

The Learner operates in two steps. First, it automatically labels its own training data as positive or negative. Second, it uses this labeled data to train a Naive Bayes classifier, which is then used by the Extractor module.

While deploying a deep linguistic parser to extract relationships between objects is not practical at Web scale, we hypothesized that a parser can help to *train* an Extractor. Thus, prior to full-scale relation extraction, the Learner uses a parser [Klein and Manning, 2003] to automatically identify and label a set of trustworthy (and untrustworthy) extractions. These extractions are are used as positive (or negative) training examples to a Naive Bayes classifier.[1] Our use of a noise-tolerant learning algorithm helps the system recover from the errors made by the parser when applied to heterogeneous Web text.

Extractions take the form of a tuple $t = (e_i, r_{i,j}, e_j)$, where e_i and e_j are strings meant to denote entities, and $r_{i,j}$ is a string meant to denote a relationship between them. The trainer parses several thousand sentences to obtain their dependency graph representations. For each parsed sentence, the system finds all base noun phrase constituents e_i.[2] For each pair of noun phrases $(e_i, e_j), i < j$, the system traverses the parse structure connecting them to locate a sequence of words that becomes a potential relation $r_{i,j}$ in the tuple t. The Learner labels t as a positive example if certain constraints

[1] Since the Learner labels its own training data, we refer to it as *self supervised.*

[2] Base noun phrases do not contain nested noun phrases, or optional phrase modifiers, such as prepositional phrases.

on the syntactic structure shared by e_i and e_j are met. These constraints seek to extract relationships that are likely to be correct even when the parse tree contains some local errors; if any constraint fails, t is labeled as a negative instance. Some of the heuristics the system uses are:

- There exists a dependency chain between e_i and e_j that is no longer than a certain length.
- The path from e_i to e_j along the syntax tree does not cross a sentence-like boundary (e.g. relative clauses).
- Neither e_i nor e_j consist solely of a pronoun.

Once the Learner has found and labeled a set of tuples of the form $t = (e_i, r_{i,j}, e_j)$, it maps each tuple to a feature vector representation. All features are domain independent, and can be evaluated at extraction time without the use of a parser. Examples of features include the presence of part-of-speech tag sequences in the relation $r_{i,j}$, the number of tokens in $r_{i,j}$, the number of stopwords in $r_{i,j}$, whether or not an object e is found to be a proper noun, the part-of-speech tag to the left of e_i, the part-of-speech tag to the right of e_j. Following feature extraction, the Learner uses this set of automatically labeled feature vectors as input to a Naive Bayes classifier.

The classifier output by the Learner is language-specific but contains no relation-specific or lexical features. Thus, it can be used in a domain-independent manner.

Prior to using a learning approach, one of the authors invested several weeks in manually constructing a relation-independent extraction classifier. A first attempt at relation extraction took the entire string between two entities detected to be of interest. Not surprisingly, this permissive approach captured an excess of extraneous and incoherent information. At the other extreme, a strict approach that simply looks for verbs in relation to a pair of nouns resulted in a loss of other links of importance, such as those that specify noun or attribute-centric properties, for example, (*Oppenheimer, professor of, theoretical physics*) and (*trade schools, similar to, colleges*). A purely verb-centric method was prone to extracting incomplete relationships, for example, (*Berkeley, located, Bay Area*) instead of (*Berkeley, located in, Bay Area*). The heuristic-based approaches that were attempted exposed the difficulties involved in anticipating the form of a relation and its arguments in a general manner. At best, a final hand-built classifier, which is a natural baseline for the learned one, achieved a mere one third of the accuracy of that obtained by the Learner.

2.2 Single-Pass Extractor

The Extractor makes a single pass over its corpus, automatically tagging each word in each sentence with its most probable part-of-speech. Using these tags, entities are found by identifying noun phrases using a lightweight noun phrase chunker.[3] Relations are found by examining the text between the noun phrases and heuristically eliminating non-essential phrases, such as prepositional phrases that overspecify an entity (*e.g.*"*Scientists from many universities are studying ...*" is analyzed as "*Scientists are studying...*"), or individual tokens, such as adverbs (*e.g.*"*definitely developed*" is reduced to "*developed*").

For each noun phrase it finds, the chunker also provides the probability with which each word is believed to be part of the entity. These probabilities are subsequently used to discard tuples containing entities found with low levels of confidence. Finally, each candidate tuple t is presented to the classifier. If the classifier labels t as trustworthy, it is extracted and stored by TEXTRUNNER.

2.3 Redundancy-based Assessor

During the extraction process, TEXTRUNNER creates a normalized form of the relation that omits non-essential modifiers to verbs and nouns, e.g. *was developed by* as a normalized form of *was originally developed by*. After extraction has been performed over the entire corpus, TEXTRUNNER automatically merges tuples where both entities and normalized relation are identical and counts the number of distinct sentences from which each extraction was found.

Following extraction, the Assessor uses these counts to assign a probability to each tuple using the probabilistic model previously applied to unsupervised IE in the KNOWITALL system. Without hand-tagged data, the model efficiently estimates the probability that a tuple $t = (e_i, r_{i,j}, e_j)$ is a correct instance of the relation $r_{i,j}$ between e_i and e_j given that it was extracted from k different sentences. The model was shown to estimate far more accurate probabilities for IE than noisy-or and pointwise mutual information based methods [Downey *et al.*, 2005].

2.4 Query Processing

TEXTRUNNER is capable of responding to queries over millions of tuples at interactive speeds due to a inverted index distributed over a pool of machines. Each relation found during tuple extraction is assigned to a single machine in the pool. Every machine then computes an inverted index over the text of the locally-stored tuples, ensuring that each machine is guaranteed to store all of the tuples containing a reference to any relation assigned to that machine.[4]

The efficient indexing of tuples in TEXTRUNNER means that when a user (or application) wants to access a subset of tuples by naming one or more of its elements, the relevant subset can be retrieved in a manner of seconds, and irrelevant extractions remain unrevealed to the user. Since the relation names in TEXTRUNNER are drawn directly form the text, the intuitions that they implicitly use in formulating a search query are effective. Querying relational triples will be easier once TEXTRUNNER is able to know which relations are synonymous with others. However, asking the user to "guess the right word" is a problem that is shared by search engines, which suggests that it is manageable for naïve users.

[3] TEXTRUNNER performs this analysis using maximum-entropy models for part-of-speech tagging and noun-phrase chunking [Ratnaparkhi, 1998], as implemented in the OpenNLP toolkit. Both part-of-speech tags and noun phrases can be modeled with high accuracy across domains and languages [Brill and Ngai, 1999; Ngai and Florian, 2001]. This use of relatively "light" NLP techniques enables TEXTRUNNER to be more robust to the highly diverse corpus of text on the Web.

[4] The inverted index itself is built using the Lucene open source search engine.

Finally, TEXTRUNNER's relation-centric index enables complex relational queries that are not currently possible using a standard inverted index used by today's search engines. These include relationship queries, unnamed-item queries, and multiple-attribute queries, each of which is described in detail in [Cafarella *et al.*, 2006].

2.5 Analysis

Tuple extraction in TEXTRUNNER happens in $O(D)$ time, where D is the number of documents in the corpus. It subsequently takes $O(T \log T)$ time to sort, count and assess the set of T tuples found by the system. In contrast, each time a traditional IE system is asked to find instances of a new set of relations R it may be forced to examine a substantial fraction of the documents in the corpus, making system run-time $O(R \cdot D)$. Thus, when D and R are large, as is typically the case on the Web, TEXTRUNNER's ability to extract information for *all* relations at once, *without* having them named explicitly in its input, results in a significant scalability advantage over previous IE systems (including KNOWITALL).

TEXTRUNNER extracts facts at an average speed of 0.036 CPU seconds per sentence. Compared to dependency parsers which take an average of 3 seconds to process a single sentence, TEXTRUNNER runs more than 80 times faster on our corpus. On average, a Web page in our corpus contains 18 sentences, making TEXTRUNNER's average processing speed per document 0.65 CPU seconds and the total CPU time to extract tuples from our 9 million Web page corpus less than 68 CPU hours. Because the corpus is easily divided into separate chunks, the total time for the process on our 20 machine cluster was less than 4 hours. It takes an additional 5 hours for TEXTRUNNER to merge and sort the extracted tuples. We compare the performance of TEXTRUNNER relative to a state-of-the-art Web IE system in Section 3.1.

The key to TEXTRUNNER's scalability is processing time that is linear in D (and constant in R). But, as the above measurements show, TEXTRUNNER is not only scalable in theory, but also fast in practice.

3 Experimental Results

We first compare recall and error rate of TEXTRUNNER with that of a closed IE system on a set of relations in Section 3.1. We then turn to the fascinating challenge of characterizing the far broader set of facts and relations extracted by TEXTRUNNER in Section 3.2.

3.1 Comparison with Traditional IE

One means of evaluating Open IE is to compare its performance with a state-of-the-art Web IE system. For this comparison we used KNOWITALL [Etzioni *et al.*, 2005], a unsupervised IE system capable of performing large-scale extraction from the Web. To control the experiments, both TEXTRUNNER and KNOWITALL were tested on the task of extracting facts from our 9 million Web page corpus.

Since KNOWITALL is a closed IE system, we needed to select a set of relations in advance. We randomly selected the following 10 relations that could be found in at least 1,000 sentences in the corpus, manually filtering out relations that were overly vague (*e.g.*"includes"):

(<proper noun>, acquired, <proper noun>)
(<proper noun>, graduated from, <proper noun>)
(<proper noun>, is author of, <proper noun>)
(<proper noun>, is based in, <proper noun>)
(<proper noun>, studied, <noun phrase>)
(<proper noun>, studied at, <proper noun>)
(<proper noun>, was developed by, <proper noun>)
(<proper noun>, was formed in, <year>)
(<proper noun>, was founded by, <proper noun>)
(<proper noun>, worked with, <proper noun>)

Table 1 shows the average error rate over the ten relations and the total number of correct extractions for each of the two systems. TEXTRUNNER's average error rate is 33% lower than KNOWITALL's, but it finds an almost identical number of correct extractions. TEXTRUNNER's improvement over KNOWITALL can be largely attributed to its ability to better identify appropriate arguments to relations.

Still, a large proportion of the errors of both systems were from noun phrase analysis, where arguments were truncated or stray words added. It is difficult to find extraction boundaries accurately when the intended type of arguments such as company names, person names, or book titles is not specified to the system. This was particularly the case for the *authorOf* relation, where many arguments reflecting book titles were truncated and the error rate was was 32% for TEXTRUNNER and 47% for KNOWITALL. With this outlier excluded, the average error rate is 10% for TEXTRUNNER and 16% for KNOWITALL.

Even when extracting information for only ten relations, TEXTRUNNER's efficiency advantage is apparent. Even though they were run over the same 9 million page corpus, TEXTRUNNER's distributed extraction process took a total of 85 CPU hours, to perform extraction for all relations in the corpus at once, whereas KNOWITALL, which analyzed all sentences in the corpus that potentially matched its rules, took an average of 6.3 hours *per relation*. In the amount of time that KNOWITALL can extract data for 14 pre-specified relations, TEXTRUNNER discovers orders of magnitude more relations from the same corpus.

Beyond the ten relations sampled, there is a fundamental difference between the two systems. Standard IE systems can only operate on relations given to it *a priori* by the user, and are only practical for a relatively small number of relations. In contrast, Open IE operates without knowing the relations *a priori*, and extracts information from all relations at once. We consider statistics on TEXTRUNNER's extractions next.

3.2 Global Statistics on Facts Learned

Given a corpus of 9 million Web pages, containing 133 million sentences, TEXTRUNNER automatically extracted a set of 60.5 million tuples at an extraction rate of 2.2 tuples per sentence.

When analyzing the output of open IE system such as TEXTRUNNER, several question naturally arise: How many of the tuples found represent actual relationships with plausible arguments? What subset of these tuples is correct? How many of these tuples are distinct, as opposed to identical or synonymous? Answering these questions is challenging due to both the size and diversity of the tuple set. As explained

	Average Error rate	Correct Extractions
TEXTRUNNER	12%	11,476
KNOWITALL	18%	11,631

Table 1: **Over a set of ten relations, TEXTRUNNER achieved a 33% lower error rate than KNOWITALL, while finding approximately as many correct extractions.**

below, we made a series of estimates and approximations in order to address the questions.

As a first step, we restricted our analysis to the subset of tuples that TEXTRUNNER extracted with high probability. Specifically, the tuples we evaluated met the following criteria: 1) TEXTRUNNER assigned a probability of at least 0.8 to the tuple; 2) The tuple's relation is supported by at least 10 distinct sentences in the corpus; 3) The tuple's relation is not found to be in the top 0.1% of relations by number of supporting sentences. (These relations were so general as to be nearly vacuous, such as (NP1, *has*, NP2)). This filtered set consists of 11.3 million tuples containing 278,085 distinct relation strings. This filtered set is the one used in all the measurements described in this section.

Estimating the Correctness of Facts

We randomly selected four hundred tuples from the filtered set as our sample. The measurements below are extrapolated based on hand tagging the sample. Three authors of this paper inspected the tuples in order to characterize the data extracted by TEXTRUNNER. Each evaluator first judged whether the relation was *well-formed*. A relation r is considered to be well-formed if there is some pair of entities X and Y such that (X, r, Y) is a relation between X and Y. For example, (*FCI, specializes in, software development*) contains a well-formed relation, but (*demands, of securing, border*) does not. If a tuple was found to possess a well-formed relation, it was then judged to see if the arguments were reasonable for the relation. X and Y are *well-formed arguments* for the relation r if X and Y are of a "type" of entity that can form a relation (X, r, Y). An example of a tuple whose arguments are not well-formed is (*29, dropped, instruments*).

We further classified the tuples that met these criteria as either concrete or abstract. *Concrete* means that the truth of the tuple is grounded in particular entities, for example, (*Tesla, invented, coil transformer*). *Abstract* tuples are underspecified, such as (*Einstein, derived, theory*), or refer to entities specified elsewhere, but imply properties of general classes, such as (*executive, hired by, company*).

Finally, we judged each concrete or abstract tuple as true or false, based on whether it was consistent with the truth value of the sentence from which it was extracted. Figure 1 summarizes this analysis of the extracted tuples.

TEXTRUNNER finds 7.8 million facts having both a well-formed relation and arguments and probability at least 0.8. Of those facts, 80.4% were deemed to be correct according to human reviewers. Within a given relation, an average of 14% of the tuples are concrete facts of which 88.1% are correct, and 86% are abstract facts of which 77.2% are correct. Concrete facts are potentially useful for information extraction or question answering, while abstract assertions are useful for ontology learning and other applications. Of course, only a small subset of the universe of tuples would be of interest in any particular application (*e.g.*, the tuples corresponding to the relations in the experiment in Section 3.1).

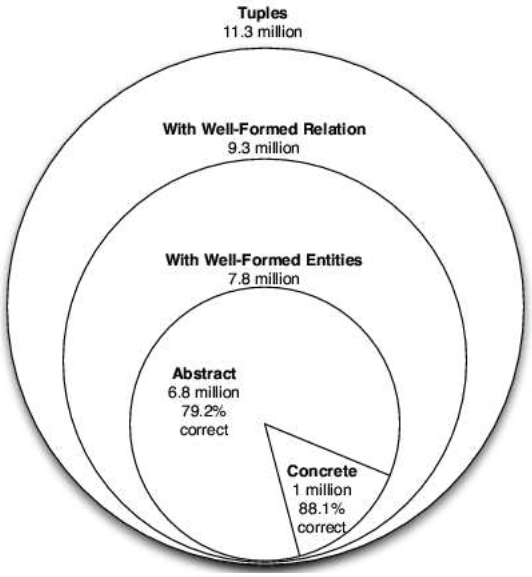

Figure 1: **Overview of the tuples extracted from 9 million Web page corpus. 7.8 million well-formed tuples are found having probability $\geq$ 0.8. Of those, TEXTRUNNER finds 1 million concrete tuples with arguments grounded in particular real-world entities, 88.1% of which are correct, and 6.8 million tuples reflecting abstract assertions, 79.2% of which are correct.**

Estimating the Number of Distinct Facts

Of the millions of tuples extracted by TEXTRUNNER, how many reflect distinct statements as opposed to reformulations of existing extractions? In order to answer this question, one needs to be able to detect when one relation is synonymous with another, as well as when an entity is referred to by multiple names. Both problems are very difficult in an unsupervised, domain-independent context with a very large number of relations and entities of widely varying types. In our measurements, we were only able to address relation synonymy, which means that the measurements reported below should be viewed as rough approximations.

In order to assess the number of distinct relations found by TEXTRUNNER, we further merged relations differing only in leading or trailing punctuation, auxiliary verbs, or in leading stopwords such as *that*, *who* and *which*. For example, *"are consistent with"* is merged with *", which is consistent with"*. We also merged relations differing only by their use of active and passive voice (*e.g.*, *invented* is merged with *was invented by*). This procedure reduced the number of distinct relations to 91% of the number before merging.

Even after the above merge, the question remains: how many of the relation strings are synonymous? This is exceedingly difficult to answer because many of the relations that TEXTRUNNER finds have multiple senses. The relation *developed*, for example, may be a relation between a person and

an invention but also between a person and a disease. It is rare to find two distinct relations that are truly synonymous in all senses of each phrase unless domain-specific type checking is performed on one or both arguments. If the first argument is the name of a scientist, then *developed* is synonymous with *invented* and *created*, and is closely related to *patented*. Without such argument type checking, these relations will pick out overlapping, but quite distinct sets of tuples.[5]

It is, however, easier for a human to assess similarity at the tuple level, where context in the form of entities grounding the relationship is available. In order to estimate the number of similar facts extracted by TEXTRUNNER, we began with our filtered set of 11.3 million tuples. For each tuple, we found clusters of concrete tuples of the form $(e_1, r, e_2), (e_1, q, e_2)$ where $r \neq q$, that is tuples where the entities match but the relation strings are distinct. We found that only one third of the tuples belonged to such "synonymy clusters".

Next, we randomly sampled 100 synonymy clusters and asked one author of this paper to determine how many distinct facts existed within each cluster. For example, the cluster of 4 tuples below describes 2 distinct relations R_1 and R_2 between *Bletchley Park* and *Station X* as delineated below:

R_1	(Bletchley Park,	*was location of*	, Station X)
R_2	(Bletchley Park,	*being called*	, Station X)
R_2	(Bletchley Park,	*, known as*	, Station X)
R_2	(Bletchley Park,	*, codenamed*	, Station X)

Overall, we found that roughly one quarter of the tuples in our sample were reformulations of other tuples contained somewhere in the filtered set of 11.3 million tuples. Given our previous measurement that two thirds of the concrete fact tuples do not belong to synonymy clusters, we can compute that $\frac{2}{3} + (\frac{1}{3} \times \frac{3}{4})$ or roughly 92% of the tuples found by TEXTRUNNER express distinct assertions. As pointed out earlier, this is an overestimate of the number of unique facts because we have not been able to factor in the impact of multiple entity names, which is a topic for future work.

4 Related Work

Traditional "closed" IE work was discussed in Section 1. Recent efforts [Pasca *et al.*, 2006] seeking to undertake large-scale extraction indicate a growing interest in the problem.

This year, Sekine [Sekine, 2006] proposed a paradigm for "on-demand information extraction," which aims to eliminate customization involved with adapting IE systems to new topics. Using unsupervised learning methods, the system automatically creates patterns and performs extraction based on a topic that has been specified by a user.

Also this year, Shinyama and Sekine [Shinyama and Sekine, 2006] described an approach to "unrestricted relation discovery" that was developed independently of our work, and tested on a collection of 28,000 newswire articles. This work contains the important idea of avoiding relation-specificity, but does not scale to the Web as explained below.

Given a collection of documents, their system first performs clustering of the entire set of articles, partitioning the corpus into sets of articles believed to discuss similar topics. Within each cluster, named-entity recognition, co-reference resolution and deep linguistic parse structures are computed and then used to automatically identify relations between sets of entities. This use of "heavy" linguistic machinery would be problematic if applied to the Web.

Shinyama and Sekine's system, which uses pairwise vector-space clustering, initially requires an $O(D^2)$ effort where D is the number of documents. Each document assigned to a cluster is then subject to linguistic processing, potentially resulting in another pass through the set of input documents. This is far more expensive for large document collections than TEXTRUNNER's $O(D + T \log T)$ runtime as presented earlier.

From a collection of 28,000 newswire articles, Shinyama and Sekine were able to discover 101 relations. While it is difficult to measure the exact number of relations found by TEXTRUNNER on its 9,000,000 Web page corpus, it is at least two or three orders of magnitude greater than 101.

5 Conclusions

This paper introduces Open IE from the Web, an unsupervised extraction paradigm that eschews relation-specific extraction in favor of a single extraction pass over the corpus during which relations of interest are automatically discovered and efficiently stored. Unlike traditional IE systems that repeatedly incur the cost of corpus analysis with the naming of each new relation, Open IE's one-time relation discovery procedure allows a user to name and explore relationships at interactive speeds.

The paper also introduces TEXTRUNNER, a fully implemented Open IE system, and demonstrates its ability to extract massive amounts of high-quality information from a nine million Web page corpus. We have shown that TEXTRUNNER is able to match the recall of the KNOWITALL state-of-the-art Web IE system, while achieving higher precision.

In the future, we plan to integrate scalable methods for detecting synonyms and resolving multiple mentions of entities in TEXTRUNNER. The system would also benefit from the ability to learn the types of entities commonly taken by relations. This would enable the system to make a distinction between different senses of a relation, as well as better locate entity boundaries. Finally we plan to unify tuples output by TEXTRUNNER into a graph-based structure, enabling complex relational queries.

Acknowledegments

We thank the following people for helpful comments on previous drafts: Dan Weld, Eytan Adar, and Doug Downey.

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324 as well as gifts from Google, and carried out at the University of

[5] We carried out several preliminary experiments using a data-driven approach to synonym discovery based on DIRT [Lin and Pantel, 2001] that confirmed this hypothesis.

Washington's Turing Center. The first author of this paper received additional support thanks to a Google Anita Borg Memorial Scholarship.

References

[Agichtein and Gravano, 2000] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.

[Brill and Ngai, 1999] E. Brill and G. Ngai. Man (and woman) vs. machine: a case study in base noun phrase learning. In *Proceedings of the ACL*, pages 65–72, 1999.

[Brin, 1998] S. Brin. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, pages 172–183, Valencia, Spain, 1998.

[Bunescu and Mooney, 2005] R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *Proc. of the HLT/EMLNP*, 2005.

[Cafarella *et al.*, 2006] Michael J. Cafarella, Michele Banko, and Oren Etzioni. Relational web search. Technical Report 06-04-02, University of Washington, 2006.

[Ciaramita *et al.*, 2005] M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric, and I. Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of IJCAI*, 2005.

[Craven *et al.*, 1999] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. In *Artificial Intelligence*, 1999.

[Culotta *et al.*, 2006] A. Culotta, A. McCallum, and J. Betz. Integrating probabilistic extraction models and relational data mining to discover relations and patterns in text. In *Proceedings of HLT-NAACL, New York, NY*, 2006.

[Downey *et al.*, 2005] D. Downey, O. Etzioni, and S. Soderland. A Probabilistic Model of Redundancy in Information Extraction. In *Proc. of IJCAI*, 2005.

[Downey *et al.*, 2007] D. Downey, M. Broadhead, and O. Etzioni. Locating Complex Named Entities in Web Text. In *Proc. of IJCAI*, 2007.

[Etzioni *et al.*, 2005] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[Kambhatla, 2004] N. Kambhatla. Combining lexical, syntactic and semantic features with maximum entropy models. In *Proceedings of ACL*, 2004.

[Klein and Manning, 2003] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the ACL*, 2003.

[Lin and Pantel, 2001] D. Lin and P. Pantel. Discovery of inference rules from text. In *Proceedings of KDD*, 2001.

[Ngai and Florian, 2001] G. Ngai and R. Florian. Transformation-based learning in the fast lane. In *Proceedings of the NAACL*, pages 40–47, 2001.

[Pasca *et al.*, 2006] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *(To appear) Proc. of ACL/COLING 2006*, 2006.

[Ratnaparkhi, 1998] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

[Ravichandran and Hovy, 2002] D. Ravichandran and D. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the ACL*, pages 41–47, Philadelphia, Pennsylvania, 2002.

[Riloff, 1996] E. Riloff. Automatically constructing extraction patterns from untagged text. In *Proc. of AAAI*, 1996.

[Rosario and Hearst, 2004] B. Rosario and M. Hearst. Classifying semantic relations in bioscience text. In *Proc. of ACL*, 2004.

[Sekine, 2006] S. Sekine. On-demand information extraction. In *Procs. of COLING*, 2006.

[Shinyama and Sekine, 2006] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proc. of the HLT-NAACL*, 2006.