DOI:10.1145/1409360.1409378

Targeted IE methods are transforming into open-ended techniques.

BY OREN ETZIONI, MICHELE BANKO, STEPHEN SODERLAND, AND DANIEL S. WELD

Open Information Extraction from the Web

Say you want to select a quiet, centrally located Manhattan hotel. Google returns an overwhelming seven million results in response to the query "new york city hotels." Or, say you are trying to assemble a program committee for an annual conference composed of researchers who have published at the conference in previous years, and to balance it geographically. While today's Web search engines identify potentially relevant documents, you are forced to sift through a long list of URLs, scan each document to identify any pertinent bits of information, and assemble the extracted findings before you can solve your problem.

Over the coming decade, Web searching will increasingly transcend keyword queries in favor of systems that automate the tedious and error-prone task of sifting through documents. Moreover, we will build systems that fuse relevant pieces of information into a coherent overview, thus reducing from hours to minutes the time required to perform complex tasks.

Information extraction (IE)—a venerable technology that maps natural-language text into structured relational data—offers a promising avenue toward this goal. Although extracting data from text is inherently challenging, given the ambiguous and idiosyncratic nature of natural language, substantial progress has been made over the last few decades.

This article surveys a range of IE methods, but we highlight *Open* Information Extraction,[3, 4] wherein the identities of the relations to be extracted are unknown and the billions of documents found on the Web necessitate highly scalable processing.

Information Extraction

At the core of an IE system is an extractor, which processes text; it overlooks irrelevant words and phrases and attempts to home in on entities and the relationships between them. For example, an extractor might map the sentence "Paris is the stylish capital of France" to the relational tuple (Paris, CapitalOf, France), which might be represented in RDF or another formal language.

Considerable knowledge is necessary to accurately extract these tuples from a broad range of text. Existing techniques obtain it in ways ranging from direct knowledge-based encoding (a human enters regular expressions or rules) to supervised learning (a human provides labeled training examples) to self-supervised learning (the system automatically finds and labels its own examples). Here, we briefly survey these methods.

Knowledge-Based Methods. The first IE systems were domain-specific. A series of DARPA Message Understanding Conferences (MUCs) challenged the NLP community to build systems that handled robust extraction from naturally occurring text. The domain

ILLUSTRATION BY GLUEKIT

Figure 1: Training examples and learned patterns for the management-succession domain.

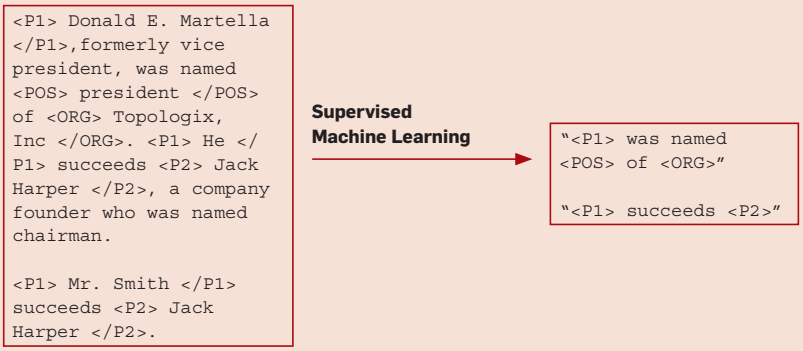


Figure 2: Sample Wikipedia infobox and the attribute/value data used to generate it.


```
{{Infobox Settlement
|official _ name = Běijīng
|other _ name = 北京
|native _ name =

|settlement _ type = [[Municipality of
         China|Municipality
|image _ skyline = SA Temple of Heaven.jpg
|image _ caption = The [[Temple of
         Heaven]], a symbol of Beijing
|citylogo _ size =
|image _ map = China-Beijing.png
|mapsize = 275px
|map _ caption = Location within China
|subdivision _ type = Country
|subdivision _ name = [[People's Republic of China]]
|subdivision _ type1 = [[Political divisions of
         China#County level|Countylevel&
         nbsp;divisions]]
|subdivision _ name1 = 18
|subdivision _ type2 = [[Political divisions of
         China#Township
         level|Township divisions]]
|subdivision _ name2 = 273
|leader _ title =[[Communist Party of
         China|CPC]] Beijing
|leader _ name =[[Liu Qi (Communist)|Liu Qi]]
         Committee Secretary
|leader _ title1 = [[Mayor]]
|leader _ name1 =[[Wang Qishan]]
|established _ title = Settled
|established _ date = ca. 473 BC
...
}}
```

of MUC-3 and MUC-4 was Latin-American Terrorism;[2] and the task was to fill templates with information about specific terrorist actions, with fields for the type of event, date, location, perpetrators, weapons, victims, and physical targets. Subsequent MUC conferences focused on domains such as joint ventures, microelectronics, or management succession.

The first IE systems relied on some form of pattern-matching rules that were manually crafted for each domain. Rules that assigned the semantic class *PhysicalTarget* space to the term bank in the terrorism domain, for example, needed to be altered to identify instances of the class *Corporation* in the joint-ventures domain. These systems were clearly not scalable or portable across domains.

Supervised Methods. Modern IE, beginning with the works of Soderland,[21, 22] Riloff,[17] and Kim and Moldovan,[11] automatically learns an extractor from a training set in which domain-specific examples have been tagged. With this machine-learning approach, an IE system uses a domain-independent architecture and sentence analyzer. When the examples are fed to machine-learning methods, domain-specific extraction patterns can be automatically learned and used to extract facts from text. Figure 1 shows an example of such extraction rules, learned to recognize persons moving into and out of top corporate-management positions.

The development of suitable training data for IE requires substantial effort and expertise. DIPRE,[5] Snowball,[1] and Meta-Bootstrapping[18] sought to address this problem by reducing the amount of manual labor necessary to perform relation-specific extraction. Rather than demand hand-tagged corpora, these systems required a user to specify relation-specific knowledge through either of the following: a small set of seed instances known to satisfy the relation of interest; or a set of hand-constructed extraction patterns to begin the training process. For instance, by specifying the set *Bolivia*, *city*, *Colombia*, *district*, *Nicaragua* over a corpus in the terrorism domain, these IE systems learned patterns (for example, *headquartered in <x>*, *to occupy <x>*, *and shot in <x>*) that identified additional names of locations. Recent advances include automatic induction of features when learning conditional random fields[13] and high-level specification of extraction frameworks using Markov logic networks.[14] Nevertheless, the amount of manual effort still scales linearly with the number of relations of interest, and these target relations must be specified in advance.

Self-Supervised Methods. The KnowItAll Web IE system[9] took the next step in automating IE by learning to label its own training examples using only a small set of domain-independent extraction patterns. KnowItAll was the first published system to carry out extraction from Web pages that was unsupervised, domain-independent, and large-scale.

For a given relation, the set of generic patterns was used to automatically instantiate relation-specific extraction rules, which were then used to learn domain-specific extraction

rules. The rules were applied to Web pages identified via search-engine queries, and the resulting extractions were assigned a probability using information-theoretic measures derived from search engine hit counts. For example, KnowItAll utilized generic extraction patterns like "<*X*> is a <*Y*>" to find a list of candidate members *X* of the class *Y*. When this pattern was used, say, for the class *Country*, it would match a sentence that included the phrase "*X* is a country."

Next, KnowItAll used frequency statistics computed by querying search engines to identify which instantiations were most likely to be bona fide members of the class. For example, in order to estimate the likelihood that "China" was the name of a country, KnowItAll used automatically generated phrases associated with the class *Country* to see if there was a high correlation between the numbers of documents containing the word "China" and those containing the phrase "countries such as." Thus KnowItAll was able to confidently label China, France, and India as members of the class *Country* while correctly knowing that the existence of the sentence, "Garth Brooks is a country singer" did not provide sufficient evidence that "Garth Brooks" is the name of a country.[7] Moreover, KnowItAll learned a set of relation-specific extraction patterns (for example, "capital of <country>") that led it to extract additional countries, and so on.

KnowItAll is self-supervised; instead of utilizing hand-tagged training data, the system selects and labels its own training examples and iteratively bootstraps its learning process. But while self-supervised systems are a species of unsupervised systems, unlike classic unsupervised systems they do utilize labeled examples and do form classifiers whose accuracy can be measured using standard metrics. Instead of relying on hand-tagged data, self-supervised systems autonomously "roll their own" labeled examples. (See Feldman[10] for discussion of an additional self-supervised IE system inspired by KnowItAll.)

While self-supervised, KnowItAll is relation-specific. It requires a laborious bootstrapping process for each relation of interest, and the set of relations has to be named by the human user in advance. This is a significant obstacle to open-ended extraction because unanticipated concepts and relations are often encountered while processing text.

The Intelligence in Wikipedia (IWP) project[23] uses a different form of self-supervised learning to train its extractors. IWP bootstraps from the Wikipedia corpus, exploiting the fact that each article corresponds to a primary object and that many articles contain infoboxes—tabular summaries of the most important attributes (and their values) of these objects. For example, Figure 2 shows the "Beijing" infobox for the class *Settlement* that was dynamically generated from the accompanying attribute/value data.

IWP is able to use Wikipedia pages with infoboxes as training data in order to learn classifiers for page type. By using the values of infobox attributes to match sentences in the article, IWP can train extractors for the various attributes. Further, IWP can autonomously learn a taxonomy over infobox classes, construct schema mappings between the attributes of parent/child classes, and thus use shrinkage to improve both recall and precision. Once extractors have been successfully learned, IWP can extract values from general Web pages in order to complement Wikipedia with additional content.

Open Information Extraction

While most IE work has focused on a small number of relations in specific preselected domains, certain corpora—encyclopedias, news stories, email, and the Web itself—are unlikely to be amenable to these methods. Under such circumstances, the relations of interest are both numerous and serendipitous—they are not known in advance. In addition, the Web corpus contains billions of documents, necessitating highly scalable extraction techniques.

The challenge of Web extraction led us to focus on Open Information Extraction (Open IE), a novel extraction paradigm that tackles an unbounded number of relations, eschews domain-specific training data, and scales linearly (with low constant factor) to handle Web-scale corpora.

For example, an Open IE system might operate in two phases. First, it would learn a general model of how relations are expressed in a particular language. Second, it could utilize this model as the basis of a relation-independent extractor whose sole input is a corpus and whose output is a set of extracted tuples that are instances of a potentially unbounded set of relations. Such an Open IE system would learn a general model of *how* relations are expressed (in a particular language), based on unlexicalized features such as part-of-speech tags (for example, the identification of a verb in the surrounding context) and domain-independent regular expressions (for example, the presence of capitalization and punctuation).

Is there a general model of relationships in English, though? To address this question we examined a sample of 500 sentences selected at random from the IE training corpus developed by Bunescu and Mooney.[6] We found that most relationships expressed in this sample could in fact be characterized by a compact set of relation-independent patterns. See Table 1 for these patterns and an estimate of their frequency.[a] In contrast, traditional IE methods learn lexical models of individual relations from hand-labeled examples of sentences that express these relations. Such an IE system might learn that the presence of the phrase "headquarters located in" indicates an instance of the headquarters relation. But lexical features are relation-specific. When using the Web as a corpus, the relations of interest are not known prior to extraction, and their number is immense. Thus an Open IE system cannot rely on hand-labeled examples of each relation. Table 2 summarizes the differences between traditional and Open IE.

Systems such as KnowItAll and IWP may be seen as steps in the direction of Open IE, but the former didn't scale as well as desired and the latter seems incapable of extracting more than 40,000 relations. Knext[19] appears to fit the Open IE paradigm,

a For simplicity, we restricted our study to binary relationships.

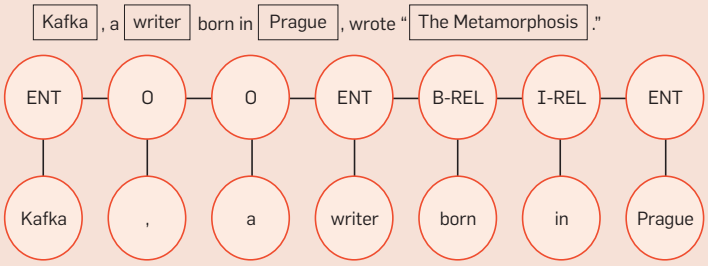
Table 1: Taxonomy of binary relationships. Nearly 95% of 500 randomly selected sentences belong to one of the eight categories noted here.

Relative Frequency	Category	Simplified Lexico-Syntactic Pattern
37.8	Verb	E_1 Verb E_2 *X established Y*
22.8	Noun + Prep	E_1 NP Prep E_2 *X settlement with Y*
16.0	Verb + Prep	E_1 Verb Prep E_2 *X moved to Y*
9.4	Infinitive	E_1 to Verb E_2 *X plans to acquire Y*
5.2	Modifier	E_1 Verb E_2 Noun *X is Y winner*
1.8	$Coordinate_n$	E_1 (and|,|-|:) E_2 NP *X-Y deal*
1.0	$Coordinate_v$	E_1 (and|,) E_2 Verb *X , Y merge*
0.8	Appositive	E_1 NP (:|,)? E_2 *X hometown : Y*

Table 2: The contrast between traditional and open IE.

	Traditional IE	Open IE
Input	Corpus + Labeled Data	Corpus + Domain-Independent Methods
Relations	Specified In Advance	Discovered Automatically
Complexity	*O* (*D* * *R*) *D* documents, *R* relations	*O* (*D*) *D* documents

Figure 3: Information extraction as sequence labeling. A CRF is used to identify the relationship, *born in*, between *Kafka and Prague*. Entities are labeled as ENT. The B-REL label indicates the start of a relation, with I-REL indicating the continuation of the sequence.



but its precision, recall, and scalability have yet to be demonstrated.

The TextRunner System

TextRunner[3, 4] is a fully implemented Open IE system that utilizes the two-phase architecture described here.[b] TextRunner extracts high-quality information from sentences in a scalable and general manner. Instead of requiring relations to be specified in its input, TextRunner learns the relations, classes, and entities from its corpus using its relation-independent extraction model.

TextRunner's first phase uses a general model of language. Specifically, it trains a graphical model called a conditional random field (CRF)[12] to maximize the conditional probability of a finite set of labels, given a set of input observations. By making a first-order Markov assumption about the dependencies among the output variables, and thus arranging variables sequentially in a linear chain, extraction can be treated as a sequence-labeling problem. Using a CRF, the extractor learns to assign labels to each of the words in a sentence denoting the beginning and end both of entity names and the relationship string.[c] See Figure 3 for an illustration.

In the second phase, TextRunner's extractor scans sentences linearly and rapidly extracts one or more textual triples that aim to capture (some of) the relationships in each sentence. For example, given the sentence "Kafka, a writer born in Prague, wrote *The Metamorphosis*," the extractor forms the triple (*Kafka*, *born in*, *Prague*). The triple consists of three strings, in which the first and third are meant to denote entities and the second to denote the relationship between them.

Of course, there are many subtleties to successful extraction from a corpus as large and heterogeneous as the Web. First, the same entities may be referred to by a variety of names (for example, *Edison*, *Thomas Edison*, *Thomas Alva Edison*, and so on). Second, the same string (say, *John Smith*) may refer to different entities. Third, vagaries of natural language (such as pronoun resolution, metaphor, anaphora, complex or ungrammatical sentences) have to be unraveled to correctly extract information. Fourth, the Web is rife with incorrect information (for example, *Elvis killed JFK*). In fact, there are many more challenges that we do not have room to discuss here, though we have addressed some of them in our research. For instance, the Resolver system[24] computes the probability that two strings are synonymous based on a highly scalable and unsupervised analysis of TextRunner tuples. Numerous other issues remain

b See www.cs.washington.edu/research/textrunner

c Although TextRunner has initially focused on extracting binary relationships, its model structure can be extended to identify relationships with greater arity.

as open problems for future work.

Post-extraction, TextRunner's collection of triples is made efficiently searchable by using Lucene, a high-performance indexing and search engine.[d] Thus TextRunner can be queried for tuples containing particular entities (for example, *Edison*), relationships (*invented*), or relationships between two entities (such as *Microsoft* and *IBM*). The different triples returned in response to a query are ranked by a fairly complex formula, but a key parameter that boosts ranking is the number of times a tuple has been extracted from the Web. Because the Web corpus is highly redundant, we have found that repeated extractions are strongly correlated with increased likelihood that an extraction is correct.

We have run TextRunner on a collection of over 120 million Web pages and extracted over 500 million tuples. By analyzing random samples of the output, we have determined that the precision of the extraction process exceeds 75% on average.[4] In collaboration with Google, we have also run a version of TextRunner on over one billion pages of public Web pages and have found that the use of an order-of-magnitude larger corpus boosts both precision and recall. Other researchers have investigated techniques closely related to Open IE, but at a substantially smaller scale.[20, 23]

Applications of Open IE

IE has numerous applications, but some tasks require the full power of Open IE because of the scope and diversity of information to be extracted. This diversity is often referred to as the "long tail" to reflect the distribution of information requests—some are very common but most are issued infrequently.

We consider three such tasks here. First and foremost is "question answering," the task of succinctly providing an answer to a user's factual question. In Figure 4, for example, the question is "What kills bacteria?" It turns out that the most comprehensive answer to that question is produced by collecting information across thousands of Web sites that address this topic. Using Open IE, the range of questions TextRunner can address mirrors the unbounded scope and diversity of its Web corpus.

The two additional tasks are:

- "Opinion mining," in which Open IE can extract opinion information about particular objects (including products, political candidates, and more) that are contained in blog posts, reviews, and other texts.
- "Fact checking," in which Open IE can identify assertions that directly or indirectly conflict with the body of knowledge extracted from the Web and various other knowledge bases.

Opinion Mining is the process of taking a corpus of text expressing multiple opinions about a particular set of entities and creating a coherent overview of those of opinions. Through this process, opinions are labeled as positive or negative, salient attributes of the entities are identified, and specific sentiments about each attribute are extracted and compared.

In the special case of mining product reviews, opinion mining can be decomposed into the following main subtasks, originally described in Popescu:[15]

1. *Identify product features.* In a given review, features can be explicit (for example, "the size is too big") or implicit ("the scanner is slow).

2. *Identify opinions regarding product features.* For example, "the size is too big" contains the opinion phrase "too big," which corresponds to the "size" feature.

3. *Determine the polarity of opinions.* Opinions can be positive (for example, "this scanner is so great") or negative ("this scanner is a complete disappointment").

4. *Rank opinions based on their strength.* "Horrible," say, is a stronger adjective than "bad."

Opine[16] is an unsupervised information-extraction system that embodies solutions to all of the mentioned subtasks. It relies on Open IE techniques to address the broad and diverse range of products without requiring hand-tagged examples of each type of product. Opine was the first to report its precision and recall on the tasks of opinion-phrase extraction and opinion-polarity determination in the context of known product features and sentences. When tested on hotels and consumer electronics, Opine was found to extract opinions with a precision of 79% and a recall of 76%. The polarity of opinions could be identified by Opine with a precision of 86% and a recall of 89%.

Fact Checking. Spell checkers and grammar checkers are word-processing utilities that we have come to take

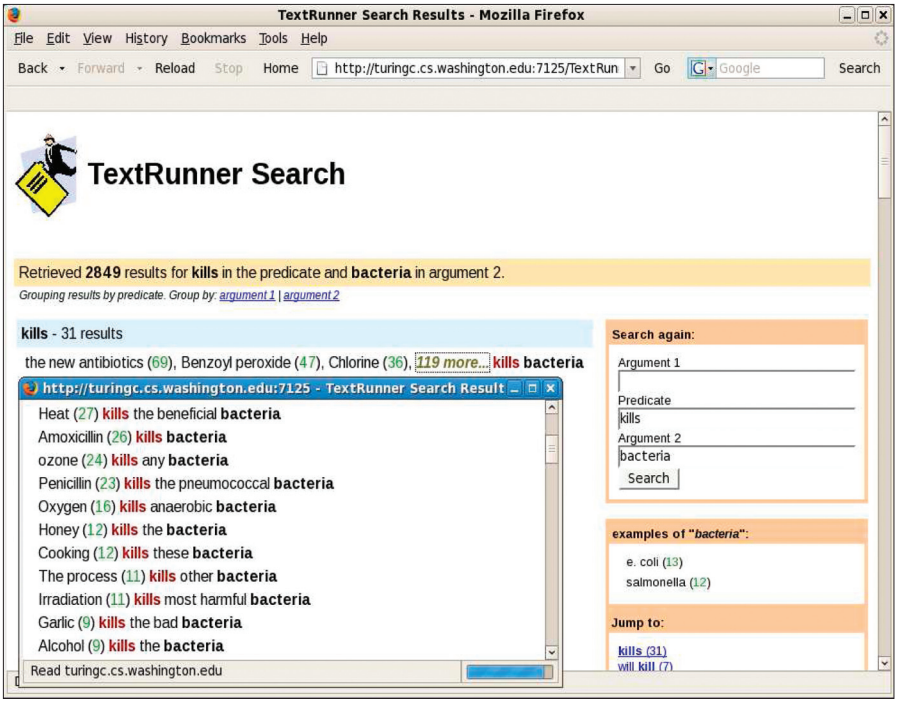


Figure 4: TextRunner aggregates answers to the query "What kills bacteria?"

d http://lucene.apache.org/

for granted. A fact checker based on Open IE seems like a natural next step.[e]

Consider a schoolchild incorrectly identifying the capital of North Dakota, or the date of India's independence, in her homework. The fact checker could automatically detect the error and underline the erroneous sentence in blue.[f] Right-clicking on the underlined sentence would bring up the conflicting facts that led the checker to its conclusion.

Where would the fact checker's knowledge base originate? While resources such as WordNet and the CIA World Fact book are of high quality, they are inherently limited in scope because of the labor-intensive process by which they are compiled. Even Wikipedia, which is put together by a large number of volunteers, only had about two million articles at last count—and they were not guaranteed to contain accurate information. To provide a checker with broad scope, it is natural to use all of the above but also include information extracted from the Web via Open IE.

Of course, the use of information extracted from the Web increases the chance that a correct fact will be flagged as erroneous. Again, this is similar to utilities such as the spell checker and grammar checker, which also periodically misidentify words or sentences as incorrect. Our goal, of course, is to build fact checkers with high precision and recall. In addition, when a fact is flagged as potentially incorrect, the checker provides an easy means of accessing the source of the information that led it to this determination.

Conclusion and Directions for Future Work

This article sketched the transformation of information extraction (IE) from a targeted method, appropriate for finding instances of a particular relationship in text, to an open-ended method (which we call "Open IE") that scales to the entire Web and can support a broad range of unanticipated questions over arbitrary relations. Open IE also supports aggregating, or "fusing," information across a large number of Web pages in order to provide comprehensive answers to questions such as "What do people think about the Thinkpad laptops?" in the Opine system[15] or "What kills bacteria?" in Figure 4.

We expect future work to improve both the precision and recall of Open IE (for example, see Downey[8] and Yates[24]). We have begun to integrate Open IE with inference, which would enable an Open IE system to reason based on the facts and generalizations it extracts from text. The challenge, of course, is to make this reasoning process tractable in the face of billions of facts and rules. We foresee opportunities to unify Open IE with information provided by ontologies such as WordNet and Cyc, as well as with human-contributed knowledge in OpenMind and FreeBase, in order to improve the quality of extracted information and facilitate reasoning. Finally, we foresee the application of Open IE to other languages besides English.

Acknowledgments

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, ONR grant N00014-08-1-0431, SRI CALO grant 03-000225, and the WRF/TJ Cable Professorship, as well as by gifts from Google. It was carried out at the University of Washington's Turing Center. We wish to thank the members of AI and KnowItAll groups for many fruitful discussions.

e This idea comes from Krzysztof Gajos.

f Blue is used to distinguish its findings from the red underline for misspellings and the green underline for grammatical errors.

References

1. Agichtein, E. and Gravano, L. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries* (2000).
2. ARPA. *Proceedings of the 3rd Message Understanding Conference* (1991).
3. Banko, M., Cafarella, M., Soderland, S., Broadhead, M. and Etzioni, O. Open information extraction from the Web. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2007).
4. Banko, M. and Etzioni, O. The tradeoffs between traditional and open relation extraction. In *Proceedings of the Association of Computational Linguistics* (2008).
5. Brin, S. Extracting patterns and relations from the World Wide Web. In *Proceedings of the Workshop at the 6th International Conference on Extending Database Technology*, (Valencia, Spain, 1998), 172–183.
6. Bunescu, R. and Mooney, R. Learning to extract relations from the Web using minimal supervision. In *Proceedings of the Association of Computational Linguistics* (2007).
7. Downey, D., Etzioni, O. and Soderland, S. A probabilistic model of redundancy in information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2005).
8. Downey, D., Schoenmackers, S. and Etzioni, O. Sparse information extraction: Unsupervised language models to the rescue. In *Proceedings of the Association of Computational Linguistics* (2007).
9. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. and Yates, A. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence 165*, 1 (2005), 91–134.
10. Feldman, R., Rosenfeld, B., Soderland, S. and Etzioni, O. Self-supervised relation extraction from the Web. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems* (2006), 755–764.
11. Kim, J. and Moldovan, D. Acquisition of semantic patterns for information extraction from corpora. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications* (1993), 171–176.
12. Lafferty, J., McCallum, A. and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 2001 International Conference on Machine Learning.*
13. McCallum, A. Efficiently inducing features of conditional random fields. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence* (Acapulco, 2003), 403–410.
14. Poon, H. and Domingos, P. Joint inference in information extraction. In *Proceedings of the 22nd National Conference on Artificial Intelligence* (2007), 913–918.
15. Popescu, A. and Etzioni, O. Extracting product features and opinions from reviews. In *Proceedings of the Empirical Methods on Natural Language Processing Conference* (2005).
16. Popescu. A-M. Information extraction from unstructured Web text. Ph.D. thesis, University of Washington (2007).
17. Riloff, E. Automatically constructing extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence* (1996), 1044–1049.
18. Riloff, E. and Jones, R. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the AAAI-99 Conference* (1999), 1044–1049.
19. Schubert, L. Can we derive general world knowledge from texts? In *Proceedings of the Human Language Technology Conference* (2002).
20. Shinyama, Y. and Sekine, S. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology/ NAACL Conference* (2006).
21. Soderland, S. Learning information extraction rules for semi-structured and free text. *Machine Learning 34*, 1–3 (1999), 233–272.
22. Soderland, S., Fisher, D., Aseltine, J. and Lehnert, W. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (1995), 1314–1321.
23. Weld, D., Wu, F., Adar, E., Amershi, S., Fogarty, J., Hoffmann, R., Patel, K. and Skinner, M. Intelligence in Wikipedia. In *Proceedings of the 23rd Conference on Artificial Intelligence* (2008).
24. Yates, A. and Etzioni, O. Unsupervised resolution of objects and relations on the Web. In *Proceedings of the Human-Language Technology Conference* (2007).

Oren Etzioni (etzioni@cs.washington.edu) is a professor of computer science and the founder and director of the Turing Center at the University of Washington, Seattle.

Michele Banko (banko@cs.washington.edu) is a Ph.D. candidate at the University of Washington, Seattle.

Stephen Soderland (soderland@cs.washington.edu) is a research scientist in the Department of Computer Science and Engineering at the University of Washington, Seattle.

Daniel S. Weld (weld@cs.washington.edu) is the Thomas J. Cable/WRF Professor of Computer Science and Engineering at the University of Washington, Seattle.