Using Wikipedia to Bootstrap Open Information Extraction

Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA-98195, USA
weld@cs.washington.edu

Raphael Hoffmann
Computer Science & Engineering
University of Washington
Seattle, WA-98195, USA
raphaelh@cs.washington.edu

Fei Wu
Computer Science & Engineering
University of Washington
Seattle, WA-98195, USA
wufei@cs.washington.edu

1. INTRODUCTION

We often use 'Data Management' to refer to the manipulation of relational or semi-structured information, but much of the world's data is unstructured, for example the vast amount of natural-language text on the Web. The ability to manage the information underlying this unstructured text is therefore increasingly important. While information retrieval techniques, as embodied in today's sophisticated search engines, offer important capabilities, they lack the most important faculties found in relational databases: 1) queries comprising aggregation, sorting and joins, and 2) structured visualization such as faceted browsing [29].

Information extraction (IE), the process of generating structured data from unstructured text, has the potential to convert much of the Web to relational form — enabling these powerful querying and visualization methods. Implemented systems have used manually-generated extractors (e.g., regular expressions) to "screen scrape" for decades, but in recent years machine learning methods have transformed IE, speeding the development of relation-specific extractors and greatly improving precision and recall. While the technology has led to many commercial applications, it requires identifying target relations ahead of time and the laborious construction of a labeled training set. As a result, supervised learning techniques can't scale to the vast number of relations discussed on the Web.

1.1 Open Information Extraction

In order to extract the widely-varying type of information on the Web, attention has recently turned to the broader goal of what Etzioni *et al.* call *open* information extraction [2, 11] — the task of scalably extracting information to fill an unbounded number of relational schemata, whose structure is unknown in advance. Open IE is distinguished from traditional methods on three dimensions [11]:

Input: Traditional, supervised approaches require a set of labeled training data in addition to the corpus for extraction; open IE uses domain-independent methods instead.

Target Schema: In traditional IE, the target relation is specified in advance; open IE automatically discovers the relations of interest.

Computational Complexity: The runtime of traditional methods is $O(D * R)$, where D denotes the number of documents in the corpus and R denotes the number of relations; in contrast, scalability to the Web demands that open IE scale linearly in D.

1.2 The Challenge

IE techniques are typically measured in terms of *precision*, the percentage of extracted items which are correct, and *recall*, the percentage of potentially correct tuples which are actually extracted. In most cases, there is a tradeoff between precision and recall (at 0% recall one has 100% precision), so researchers often look at the recall at some fixed precision (determined by task needs) or the *area under the precision / recall (P/R) curve*.

While several successful systems have tackled open IE [2, 27, 1, 5, 23], demonstrating respectable performance, they face a harder problem than traditional supervised methods. Specifically, the tradeoff between precision and recall is further opposed by the need for generality in the type of text accepted and the range of relations considered. The goal of fully open IE, eschewing manual tuning or the tedious construction of training sets, represents the extreme point on the generality spectrum. The challenge for the future, then, is to improve all three dimensions: precision, recall and generality. How far is it possible to go?

The remainder of this paper presents Kylin as a case study of open IE. We start by describing Kylin's use of Wikipedia to power the self-supervised training of information extractors. Then, in Section 3 we show how Wikipedia training can be seen as a *bootstrapping method* enabling extraction from the wider set of general Web pages. Not even the best machine-learning algorithms have production-level precision; Section 4 discusses how users — engaged in some other primary task — may be unobtrusively enticed to validate extractions, enlarging the training set, and thus improving accuracy over time. Section 5 concludes with observations gleaned from our experience with Kylin. Finally, Section 6 concludes by revisiting the challenges listed above.

2. SELF-SUPERVISED LEARNING

As we have said, information extraction is a process which generates a set of relational tuples by "reading" unstructured text. For example, the simplest case of IE, called named-entity recognition, extracts tuples of a unary relation, *e.g.*, `company(IBM)` from natural-language sentences. Supervised learning of extractors operates in two phases:

Training: Typically a large training corpus is cleaned and

Motto:	*Lux sit* (Latin)
Motto in English:	Let there be light[1]
Established:	1861
Type:	Public flagship Sea grant Space grant
Endowment:	$3.18 billion[2]
President:	Mark Emmert
Provost:	Phyllis Wise
Staff:	3,623

Figure 1: Rendered form of a portion (8 of 18 attributes) of the infobox for the University of Washington.

converted to a sequence of words, $\vec{\mathbf{w}}$. Each word is marked (*e.g.*, by a human) to show which words correspond to companies, thus providing a training set. One basic strategy is then to train a sequential probabilistic model, such as an HMM, deriving the parameters θ which maximize $P[\vec{\mathbf{w}}|\theta]$ by simply counting inter-state transitions and word-emissions per state. It is also possible to make use of unlabeled data by using the greedy *Baum-Welch* algorithm.

Extraction: Given a trained HMM with parameters, θ, and a sequence of test words, *e.g.*, $\vec{\mathbf{w}}$= "Today Amazon released its quarterly earnings," IE uses the dynamic-programming *Viterbi* algorithm to find the most likely state sequence π, i.e. the π that maximizes $P[\vec{\mathbf{w}}, \pi|\theta]$. Any words predicted to have been emitted from the HMM's "company" state are returned as tuples, *e.g.*, `company(Amazon)`.

But how can machine learning be used in the case of *open* IE? One method, incorporated into the TextRunner system [2, 1], learns patterns which indicate *general relationships* and extracts the relation as well as the arguments. We advocate an alternative approach: using Wikipedia to generate *relation-specific* training data for a broad set of thousands of relations.

2.1 Heuristic Generation of Training Data

There are many reasons why Wikipedia is an excellent corpus for training an open IE system. First, it is a comprehensive source of high-quality text about a very wide variety of topics. Secondly, it has a great deal of internal structure which can be exploited for learning [27]. In this paper, we restrict our attention to *infoboxes*, tabular summaries of an article's salient details which are included on a number of Wikipedia pages. For example, Figure 1 shows the infobox from the page on the University of Washington. Since this table is generated via stylesheet from XML source, it is relatively easy to process. When a page has such an infobox one can determine the *class* of the infobox (e.g., the `University` class). By aggregating the source code for a number of instances of an infobox class, one may deduce the most important *attributes* of the class; these correspond to relations whose first argument is an instance of the class. In our example, we have the year `established`, the `provost` and sixteen others.

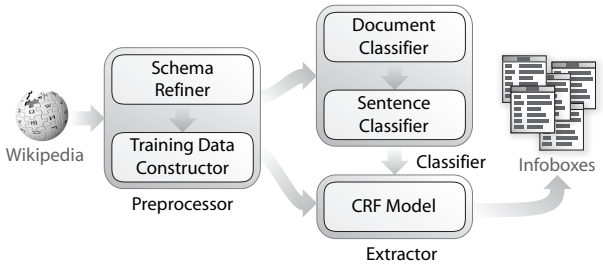

Figure 2: Architecture of Kylin's basic, self-supervised open information extraction system, before shrinkage is applied.

Kylin uses infoboxes to capture the schemata of the most important relations for summarizing the contents of Wikipedia articles. The main problem we address is how to populate those schemata from data extracted from the natural language text of articles *without* infoboxes. Kylin uses existing infoboxes for this second task as well, creating training sets for learning extractors for those relations [27]. The overall architecture of Kylin is shown in Figure 2; we discuss the main components below.

Preprocessor: The preprocessor selects and refines infobox-class schemata, choosing relevant relations; it then generates machine-learning datasets for training sentence classifiers and extractors. Refinement is necessary for several reasons. For example, *schema drift* occurs when authors create an infobox by copying one from a similar article and changing attribute values. If a new attribute is needed, they just make up a name, leading to schema and attribute duplication. Since this is a widespread problem, schema refinement clusters infobox classes which have similar schemata and names with low edit distance. Rare attributes are ignored.

Next, the preprocessor heuristically constructs two types of training datasets — those for sentence classifiers, and for CRF relation extractors. For each article with an infobox mentioning one or more target relations, Kylin tries to find a unique sentence in the article that mentions that attribute's value. The resulting labeled sentences form positive training examples for each relation; other sentences form negative training examples. If the attribute value is mentioned in several sentences, then one is selected heuristically.

Generating Classifiers: Kylin learns two types of classifiers. For each class of article being processed, a heuristic *document classifier* is used to recognize members of the infobox class. For each target relation within a class a *sentence classifier* is trained in order to predict whether a given sentence is likely to contain the attribute's value. For this, Kylin uses a maximum entropy model [17] with bagging. Features include a bag of words, augmented with part of speech tags.

Learning Extractors: Extracting attribute values from a sentence is best viewed as a sequential data-labeling problem. Kylin uses conditional random fields (CRFs) [15] with a wide variety of features (e.g., POS tags, position in the sentence, capitalization, presence of digits or special characters, relation to anchor text, etc.). Instead of training a single

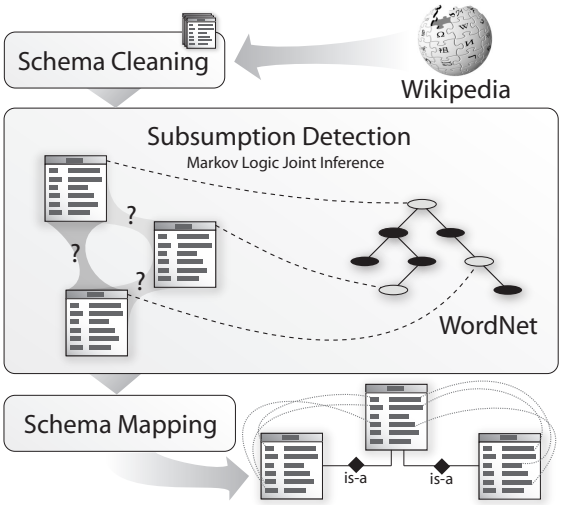

Figure 3: Architecture of Kylin Ontology Generator.

master extractor to clip all relations, Kylin trains a different CRF extractor for each relation, ensuring simplicity and fast retraining. This decomposition also enables easy parallelization for computational scalability.

2.2 Improving Recall with Shrinkage

Although Kylin performs well when it can find enough training data, it flounders on sparsely populated infobox classes — the majority of cases. Fortunately, there is a way to improve Kylin's performance through the use of shrinkage, a general statistical technique for improving estimators in the case of limited training data. McCallum et al. applied this technique for text classification in a hierarchy of classes by smoothing parameter estimates of a data-sparse child with its parent to get more robust estimates [16].

Similarly, we use shrinkage when training an extractor of an instance-sparse infobox class by aggregating data from its parent and child classes. For example, knowing that `Performer IS-A Person`, and `Performer.loc=Person.birth_plc`, we can use values from `Person.birth_plc` to help train an extractor for `Performer.loc`. The trick is finding a good subsumption hierarchy which relates attributes between parent and child classes. Unfortunately, Wikipedia does not contain such a taxonomy[1] Furthermore, previously-created taxonomies for Wikipedia, e.g. [18], don't contain the required relation-relation mappings between parent-child classes. Thus, we were led to devise our own taxonomy, which we did using a novel, autonomous process described below. After explaining the construction of this taxonomy, we describe our approach to shrinkage.

The Kylin Ontology Generator: The Kylin Ontology Generator (KOG) is an autonomous system that builds a rich taxonomy by combining Wikipedia infobox classes with WordNet using statistical-relational machine learning [28]. At the highest level KOG computes six different kinds of features, some metric and some Boolean: *similarity measures*, *edit history patterns*, *class-name string inclusion*, *category tags*, *Hearst patterns*, *search-engine statistics*, and *WordNet* mappings. These features are combined using statistical-relational machine learning, specifically joint inference over Markov logic networks [20], extending [21].

Figure 3 shows the architecture of KOG. First, its *schema cleaner* scans the infobox system to merge duplicate classes and relations, and infers the type signature of each relation. Then, the *subsumption detector* identifies the subsumption relations between infobox classes, and maps the classes to WordNet nodes. Finally, the *schema mapper* builds relation mappings between related classes, especially between parent-child pairs in the subsumption hierarchy [28].

KOG's taxonomy provides an ideal base for the shrinkage technique, as described below.

Shrinkage Using the KOG Ontology: Given a sparse target infobox class C, Kylin's shrinkage module searches upwards and downwards through the KOG taxonomy to aggregate training data from parent and children classes. The overall shrinkage procedure is as follows:

1. Given a class C, query KOG to collect the related class set: $S_C = \{C_i | path(C, C_i) \leq l\}$, where l is the preset threshold for path length. Currently Kylin only searches strict parent/child paths without considering siblings. Take the `Performer` class as an example: its parent `Person` and children `Actor` and `Comedian` could be included in S_C.
2. For each relation $C.r$ (e.g., `Performer.loc`) of C:
   (a) Query KOG for the mapped relation $C_i.r_j$ (e.g., `Person.birth_plc`) for each C_i.
   (b) Assign weight w_{ij} to the training examples from $C_i.r_j$ and add them to the training dataset for $C.r$. Note that w_{ij} may be a function both of the target relation $C.r$, the related class C_i, and C_i's mapped relation $C_i.r_j$.
3. Train the CRF extractors for C on the new training set.

With shrinkage, Kylin learns much better extractors, especially in classes with only a few instances containing infoboxes. For example, the area under the precision and recall curve for the `performer` class (which had 44 instances) improved by 57% after applying shrinkage from `person` (1201 examples), `actor` (8738 examples) and `comedian` (106 examples) [26]. Most of this improvement comes from increased recall, but precision gets a small boost as well.

3. BOOTSTRAPPING TO THE WEB

Even when Kylin does learn an effective extractor there are numerous cases where Wikipedia has an article on a topic, but the article simply doesn't have much information to be extracted. Indeed, a long-tailed distribution governs the length of articles in Wikipedia — around 44% of articles are marked as stub pages — indicating that much-needed

[1]Wikipedia's category system is not directly useful in this regard, for several reasons. First, the category system is too flat. Second, there are many administrative tags mixed into the categories. Third, the "content-bearing" tags are typically conjunctive, e.g. `Jewish physicist`, which require significant processing to decompose into an orthogonal taxonomy.

information is missing. Additionally, facts that are stated using uncommon or ambiguous sentence structures also hide from the extractors. This section shows how to extend the previously-described methods to support extraction from the broader Web.

The challenge for this approach — as one might expect — is maintaining high precision and recall. Since Kylin's extractors have been trained on the somewhat idiosyncratic Wikipedia corpus, they may not extend smoothly to the Web. After considering the roots of the problem, we discuss the extensions necessary to successfully bootstrap.

- **Reduced Recall When Extracting from the Web:** In many cases, the language used on Wikipedia is stylized and looks very different from pages on the general Web. For example, on Wikipedia a person's date of birth and his professions are often expressed in the first sentence, which begins with the person's name, then contains the birth date in parentheses, then the verb *is*, and finally a list of professions. When trained on such data, an extractor fails to extract from sentences of different style. For example, the extractor might not be able to extract a person's birthdate from the sentence "Obama was born on August 4, 1961."
- **Reduced Precision When Extracting from the Web:** Extractors trained on Wikipedia pages are often unable to discriminate irrelevant information. For example, a Kylin extractor for a person's birthdate is trained on a set of pages which all have that person as their primary subject. Such extractors become inaccurate when applied to a page which compares the lives of *several* people — even if the person in question is one of those mentioned.

While Kylin's self-supervised approach allows it to learn extractors for a comprehensive range of relations, using only Wikipedia as a training source also limits its ability to extract from the broader Web. In response, we developed two techniques which dramatically improved its precision and recall when extracting from the general Web.

3.1 Generalizing Extractors with Retraining

Our first idea is to harvest additional training data from the *outside* Web for improved training, which we call *retraining*. The challenge is automatically identifying relevant sentences given the sea of Web data. For this purpose, Kylin utilizes TextRunner, an open IE system [1], which extracts semantic relations $\{r|r = \langle obj_1, predicate, obj_2 \rangle\}$ from a crawl of about 100 million Web pages. Importantly for our purposes, TextRunner's crawl includes the top ten pages returned by Google when queried on the title of every Wikipedia article.

For each target relation within an infobox, Kylin queries to identify and verify relevant sentences that mention the attribute's value. Sentences passing the verification are labeled as extra positive training examples. Kylin also identifies the phrases (predicates) which are harbingers of each target relation. For example, "was married to" and "married" are identified for the "person.spouse" relation. These harbingers are used to eliminate potential false negative training examples generated from the Wikipedia data.

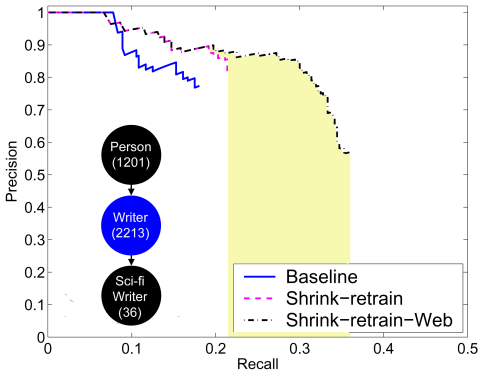

Figure 4: Using Kylin's retrained extractors to extract from the Web results in a substantial improvement to the area under the P/R curve — even in infobox classes, like `writer`, which have thousands of instances.

By adding new positive examples and excluding potential false negative sentences, retraining generates a cleaned and augmented training dataset which improves Kylin's performance. When used together with shrinkage, the improvement in recall is enormous. For example, on the `performer` class, recall improved by 73% on Wikipedia data, and it doubled on Web data. In other, more instance-sparse classes, the improvement in recall was even higher, as much as 1755% [26].

3.2 Selecting Quality Sources

Our second method, targeted towards increasing precision, is about deciding if an extraction from a Web page is indeed relevant to the subject of a Wikipedia article. We view this as an information-retrieval problem which we solve by carefully selecting and weighting extractions from Web pages. This requires a number of steps: First, Kylin generates a set of relevant queries and utilizes a general Web search engine, namely Google, to identify a set of pages which are likely to contain the desired information. These pages are fetched, and Kylin's extractors are applied to all content. Each extraction is then weighted using a combination of factors, including the rank of the page, the extraction confidence (score computed by the CRF), and the distance between the current sentence and the closest sentence containing the name of the Wikipedia article.

Our experiments revealed that the CRF's confidence is a poor choice for scoring different extractions of the same relations, but that the rank of the source page in response to our queries, and especially the distance between the extracted text and the closest sentence containing the name of the article are extremely useful. A weighted combination performed best, and roughly doubled precision of Web extractions for a variety of classes [26].

Extraction results from the Web are later combined with extraction results from Wikipedia. Higher weight is given to extractions from Wikipedia, because it is still likely that extractions from Wikipedia will be more precise. That is, in Wikipedia we can be more certain that a given page is highly relevant, is of higher quality, has a more consistent structure, for which Kylin's extractors have been particularly trained.

Integrating evidence from both Wikipedia and the greater Web further helps Kylin's performance. For example, on the

performer class, the area under the P/R curve improved 102%; for classes which are even more sparse, the improvement can be ten times as high [26]. Recall can even be improved on classes with many training instances; for example, Figure 4 shows the improvement on the writer class which has 2213 instances with infoboxes.

4. COMMUNAL CORRECTION

While some of the CRF extractors learned by Kylin have extremely high precision, in most cases precision is well below that of humans. Thus, a *fully* automated process for producing high-reliability, structured data (*e.g.*, as would be required in order to to add extracted data back into Wikipedia infoboxes) may be untenable. Instead, Kylin aims to amplify *human* effort towards this task. Figure 5 shows our first mockup design of this interface. In a series of interviews with members of the Wikipedia community, informal design reviews, and a final online user study we refined, explored and evaluated the space of interfaces, focusing on the design dimensions listed below.

4.1 Key Issues for Correction Interfaces

Contribution as a Non-Primary Task: Although tools already exist to help expert Wikipedia editors quickly make large numbers of edits [7], we instead want to enable contributions by the long tail of users *not yet contributing* [24]. In other words, we believe that pairing IE with *communal content creation* will be most effective if it encourages contributions by people who had not otherwise planned to contribute. This means that we must treat contributing as a *non-primary task* — encouraging contributions from people engaged in some other primary activity.

Inviting Contributions: Any system based in community content creation must provide an incentive for people to contribute. Bryant et al. report that newcomers become members of the Wikipedia community by participating in peripheral, yet productive, tasks that contribute to the overall goal of the community [4]. Given this behavior, our goal is to make the opportunity to contribute sufficiently salient that people will try it, but not so visible as to make the interface obtrusive or coercive. We designed three new interfaces to explore this tradeoff.

Presenting Ambiguity Resolution in Context: As shown in Figure 5 there are two plausible locations in an article for presenting each potential extraction: near the article text from which the value was extracted or proximal to the infobox where the data is needed. Presenting information at the latter location can be tricky because the contributor cannot verify information without knowing the context. Furthermore, varying the way that context is presented can dramatically affect user participation.

4.2 Preliminary User Study

We evaluated the effectiveness of our interfaces in stimulating edits as well as users' perception of interface intrusiveness in a novel study, deployed via Google Adwords and Yahoo Keyword Advertising [14]. While we developed

Figure 5: User interface mockup. Casual users are presented with a standard Wikipedia page highlighting a single attribute value; an ignorable popup window allows the user to verify the extraction if she wishes.

some interface designs which yielded even higher participation rates, the "winner" of our study was an icon design that was deemed relatively unobtrusive and yet which led people to voluntarily contribute an average of one fact validation for every 14 page visits. Validations had a precision of 90%. By validating facts multiple times from different visitors, we believe we can achieve very high precision on extracted tuples. Preliminary experiments also show that by adding these new tuples as additional training examples, Kylin will keep increasing extractor performance.

5. LESSONS LEARNED

Our experience in building the Kylin system and running it over the constantly-changing Wikipedia yields several lessons.

5.1 Approaches to Open IE

In traditionally, relational databases, rigid schemata are defined before any data is added; indeed, a well-defined schema facilitates the expression of queries. But if one wishes to extract a wide variety of information from the Web, for example a large fraction of data from Wikipedia, then human preengineering of such schemata (and associated construction of labeled training data for each relation) is impossible.

In response, we focus on what Etzioni *et al.* term *open* information extraction — algorithms which can handle an unbounded number of relations, eschew domain-specific training data, and scale linearly to handle Web-scale corpora. Besides Kylin, only a few open IE systems have yet been built, with the first being TextRunner [2, 1]. We believe that all open IE systems may be placed into two groups, which we term *relational-targeting* and *structural-targeting* methods. Kylin uses a relational approach, but to our knowledge all other open IE systems use structural targeting.

Relational Targeting: Learning a relation-specific extractor is the most common technique in traditional IE; indeed, the example in the beginning of Section 2 trained an HMM where one state corresponded to words naming an instance of the company relation. The first challenge for this method is acquiring enough training data for a comprehensive set of relations. This paper has shown how Kylin's self-supervised

approach uses Wikipedia infoboxes to heuristically assemble training sets. We estimate that Wikipedia contains 5000 infobox classes, each of which has approximately 10 attributes. Thus, while Kylin can't truly handle an *unbounded* number of relations, it seems capable of learning 50,000, which may be sufficient.

We contrast our approach with systems such as Yago [22], which also use Wikipedia as a corpus, but learn a fixed set of a dozen relations using substantial manual effort. While systems such as DBLIFE [8] scale relatively well, they aren't "open" either, requiring manually-crafted rules for extraction.

Structural Targeting: An alternative approach is to build a *general* extraction-engine which looks for some form of relation-independent structure on Web pages and uses this to extract tuples. A postprocessing step is often used to normalize the extractions, determining the precise relation and entities which have been extracted. There are many different forms of structure which may be used in this process. For example, TextRunner [2, 1] and Knext [23] exploit grammatical structure over sentences. Hearst patterns operate on phrases within sentences [13, 10]. Other work uses HTML structure, such as the DOM tree, to extract lists and tables [9, 10, 12, 5]

While open IE systems using structural targeting can truly handle an unbounded number of relations, such generality often comes at the cost of lower precision when compared to relationally-targeted extractors. This brings us back to the challenge described in Section 1.2: can open IE methods really provide generality along with high precision and recall. We think the answer is "yes." Both structural and self-supervised relational approaches have made impressive progress. And importantly, the different approaches are complementary, as we demonstrate with Kylin's retraining methodology. Future work may focus on other ways to combine these approaches.

5.2 Scalability of Relation-Specific Open IE

To scale extraction to a large number of relations from a large number of pages, parallelization is important and already we see information extraction algorithms leveraging map-reduce-style data processing frameworks [6]. Yet, parallelization alone is not sufficient. For example, Kylin learns a relation-specific extractor for each attribute of each infobox class in Wikipedia — potentially more than 50,000 extractors in total. It would be impractical to run every extractor on each Wikipedia page, let alone each page on the Web! Fortunately, Kylin's hierarchical approach for classifying Wikipedia pages and sentences, alleviates the problem when the extraction corpus is Wikipedia itself. However, it's unclear that this approach can be extended to the general Web.

5.3 Integrating Evidence

High precision and recall requires collecting information from many sources. In fact, leveraging the redundancy of content on the Web, utilizing structural properties of Web sites or their contents, and leveraging relevant databases [3] is often crucial for resolving extraction ambiguities. One method accomplishing this integration is joint inference [19, 25]. Additionally, the need to decompose extractors for reasons of efficiency further increases the amount of evidence being collected. The biggest challenge for this task is likely managing the tradeoff between computational blowup (if exact approaches are attempted) and greatly reduced accuracy (if approximations such as probabilistic independence are made).

Kylin learns a myriad of extractors independently, but what we consider as one of its main contributions, is the realization that a careful integration of extractors exploiting various structural properties of content and sites, can lead to dramatic improvements in precision and recall. Kylin automatically infers a taxonomy underlying Wikipedia data and utilizes this taxonomy to help extraction, even though the taxonomy itself is not its goal. When combined, techniques such as shrinkage, retraining, and Web-source selection enable Kylin to perform well, even when targeting the long tail of rarely occurring relations. We are also looking to improve Kylin's integration abilities through the use of probabilistic CFGs. This may provide a cleaner way to integrate page and sentence classification with extraction; this method also enables joint inference. As always, scalability is likely to be the defining challenge.

5.4 The Role of Humans

In the near future, we are unlikely to achieve extremely high precision extraction without human involvement. Researchers have explored four main ways for involving humans in the process: 1) humans write rule-based extraction procedures, 2) humans label training data for supervised learning of extractors, 3) humans validate candiate extractions, regardless of how they are produced, and 4) humans manually aggregate structured information (*e.g.*, as in Freebase). We believe, that in many cases, one can best utilize the rare human resource by combining self-supervised learning with crowd-sourced validation. The study described in Section 4.2 shows that high participation rates are possible.

6. CONCLUSION

By converting unstructured, natural-language text to relational form, information extraction enables many powerful Data Management techniques. However, in order to scale IE to the Web, we must focus on *open* IE — a paradigm that tackles an unbounded number of relations, eschews domain-specific training data, and scales computationally [2, 11]. This paper describes Kylin, which uses self-supervised learning to train relationally-targeted extractors from Wikipedia infoboxes. We explained how shrinkage and retraining allow Kylin to improve extractor robustness, and we demonstrate that these extractors can successfully mine tuples from a broader set of Web pages. Finally, we argued that the best way to utilize human efforts is by inviting humans to quickly validate the correctness of machine-generated extractions.

We distill several lessons from our experience. Perhaps our most important observation contrasts two approaches to open IE: relational *vs.* structural targeting. While Kylin primarily uses the relational approach, we argue that the best chance for jointly optimizing extraction generality, precision, and recall will be to further combine relational and structural approaches.

Acknowledgments: We thank Eytan Adar, Saleema Amershi, Mike Cafarella, AnHai Doan, Oren Etzioni, Krzysztof Gajos, James Fogarty, Chloé Kiddon, Shawn Ling, Kayur Patel, and Stefan Schoenmackers for valuable discussions. Evgeniy Gabrilovitch and Ken Schmidt greatly facilitated our user study. This work was supported by NSF grant IIS-0307906, ONR grant N00014-06-1-0147, SRI CALO grant 03-000225, the WRF / TJ Cable Professorship and a grant from Yahoo! Inc.

7. REFERENCES

[1] M. Banko and O. Etzioni. The tradeoffs between traditional and open relation extraction. *Proceedings of ACL08*, 2008.

[2] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the Web. *Proceedings of IJCAI07*, 2007.

[3] Kedar Bellare and Andrew McCallum. Learning extractors from unlabeled text using relevant databases. *Proceedings of IIWeb08 Workshop*, 2007.

[4] S. Bryant, A. Forte, and A. Bruckman. Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia. *Proceedings of GROUP05*, 2005.

[5] Michael J. Cafarella, Alon Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Webtables: Exploring the power of tables on the web. *Proceedings of VLDB08*, 2008.

[6] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng, and K Olukotun. Map-reduce for machine learning on multicore. *Proceedings of NIPS06*, Vancouver, Canada, 2006.

[7] D. Cosley, D. Frankowski, L. Terveen, and J. Riedl. Suggestbot: Using intelligent task routing to help people find work in wikipedia. *Proceedings of IUI07*, January 2007.

[8] P. DeRose, X. Chai, B. Gao, W. Shen, A. Doan, P. Bohannon, and J. Zhu. Building community wikipedias: A human-machine approach. *Proceedings of ICDE08*, 2008.

[9] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the World-Wide Web. *Proceedings of AGENTS97*, pages 39–48, Marina del Rey, California, 1997.

[10] 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.

[11] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the Web. *Communications of the ACM*, **51**(12) 2008.

[12] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. *Proceedings of WWW07*, 2007.

[13] M. Hearst. Automatic acquisition of hyponyms from large text corpora. *Proceedings of COLING92*, 1992.

[14] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, and D. S. Weld. Amplifying community content creation with mixed-initiative information extraction. *Proceedings of CHI09*, 2009.

[15] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML01*, Edinburgh, Scotland, May 2001.

[16] Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. Jude W. Shavlik, editor, *Proceedings of ICML98*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[17] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. *Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering*, 1999.

[18] S. Ponzetto and M. Strube. Deriving a large scale taxonomy from Wikipedia. *Proceedings of AAAI07*, pages 1440–1445, 2007.

[19] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. *Proceedings of AAAI08*, pages 913–918, 2007.

[20] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 2006.

[21] Rion Snow, Daniel Jurafsky, and A. Ng. Semantic taxonomy induction from heterogenous evidence. *Proceedings of ACL06*, 2006.

[22] F. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. *Proceedings of WWW07*, 2007.

[23] B. Van Durme and L.K. Schubert. Open knowledge extraction through compositional language processing. *Symposium on Semantics in Systems for Text Processing*, 2008.

[24] J. Voss. Measuring wikipedia. *International Conference of the International Society for Scientometrics and Informetrics*, 2005.

[25] Michael Wick, Khashayar Rohanimanesh, Karl Schultz, and Andrew McCallum. A unified approach for schema matching, coreference and canonicalization. *Proceedings of KDD08*, 2008.

[26] Fei Wu, Raphael Hoffmann, and Danel S. Weld. Information extraction from Wikipedia: Moving down the long tail. *Proceedings of KDD08*, 2008.

[27] Fei Wu and Daniel Weld. Autonomouslly semantifying Wikipedia. *Proceedings of CIKM07*, Lisbon, Porgugal, 2007.

[28] Fei Wu and Daniel Weld. Automatically refining the Wikipedia infobox ontology. *Proceedings of WWW08*, 2008.

[29] K. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *Proceedings of SIGCHI03*, 2003.