Autonomously Semantifying Wikipedia

Fei Wu Daniel S. Weld
Computer Science & Engineering Department,
University of Washington, Seattle, WA, USA
{wufei,weld}@cs.washington.edu

ABSTRACT

Berners-Lee's compelling vision of a Semantic Web is hindered by a chicken-and-egg problem, which can be best solved by a bootstrapping method — creating enough structured data to motivate the development of applications. This paper argues that autonomously "Semantifying Wikipedia" is the best way to solve the problem. We choose Wikipedia as an initial data source, because it is comprehensive, not too large, high-quality, and contains enough manually-derived structure to bootstrap an autonomous, self-supervised process. We identify several types of structures which can be automatically enhanced in Wikipedia (e.g., link structure, taxonomic data, infoboxes, etc.), and we describe a prototype implementation of a self-supervised, machine learning system which realizes our vision. Preliminary experiments demonstrate the high precision of our system's extracted data — in one case equaling that of humans.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Management, Design

Keywords

Information Extraction, Wikipedia, Semantic Web

1. INTRODUCTION

While compelling in the long term, Berners-Lee's vision of the Semantic Web [5] is developing slowly. Researchers have argued that the relative difficulty of authoring structured data is a primary cause [17]. A chicken-and-egg problem results: if there was more structured data available, people would develop applications; but without compelling applications, it is not worth people's time to structure their data. In order to break this deadlock, a bootstrapping method is needed — some method of automatically structuring a large amount of existing data.

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

The ideal vision is a system which autonomously extracts information from the Web. Because of the wide range of information categories, supervised machine learning will require too much human effort to scale. Instead, such a system should use unsupervised or self-supervised techniques. Several systems of this form have been proposed, e.g. MULDER [18], AskMSR [7], and KNOWITALL [14], showing some signs of early success. The insight underlying these systems stems from the huge redundancy of knowledge on the Web — many things worth extracting are stated many times, in different ways and on disparate Web pages. As a result, complex linguistic processing is unnecessary, because one of the occurrences is likely written in a form which can be correctly extracted with simple methods. Furthermore, the Web's statistical properties, as calculated by a search engine, are a powerful tool for extraction [8, 13, 12]. Unfortunately, many of the things published on the Web are incorrect (e.g. "Elvis killed John Kennedy"), and the increasing linguistic sophistication of link spam poses a growing challenge to these methods.

Our paper proposes a very different approach to massive information extraction. Instead of using the whole Web, we focus on a single site: `en.wikipedia.org`.[1]

1.1 Why Wikipedia?

Focusing on Wikipedia largely solves the problem of inaccurate source data, but introduces new challenges. For example, redundancy is very greatly reduced — apparently increasing the need for deep syntactic analysis. On the other hand, Wikipedia has several attributes that make it ideal for extraction:

- Wikipedia gives all important concepts their own unique identifier — the URI of a definitional page. The first reference to such a concept typically includes a link which can be used for disambiguation. As a result, homonyms are much less of a problem than in unstructured text.
- Infoboxes, tabular summaries of an object's key attributes, may be used as a source of training data, allowing for self-supervised learning.
- Wikipedia *lists* and *categories* provide a simple type system and rudimentary taxonomic hierarchy, respectively.
- Redirection pages can be used to induce synonyms.
- Disambiguation pages can be used to generate a list of candidate targets for homonym resolution, e.g. to increase the number of links between Wikipedia pages.

[1]Actually, we view Wikipedia simply as a first bootstrapping step, which will enable subsequent extraction from the Web as a whole.

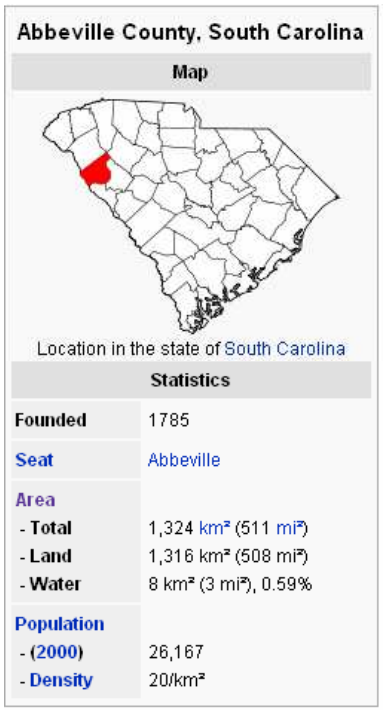

Figure 1: Sample Wikipedia infobox.

- With over 1.7 million articles, Wikipedia is appropriately sized — big enough to provide a sufficient dataset, yet enough smaller than the full Web that a hundred-node cluster is unnecessary for corpus processing.

1.2 Overview

Our grand vision is a combination of autonomous and collaborative techniques for semantically marking up Wikipedia. Such a system would create or complete infoboxes by extracting information from the page, rationalize tags, merge replicated data using microformats, disambiguate links and add additional links where needed, engage humans to verify information as needed, and perhaps add new topic pages (e.g., by looking for proper nouns with no corresponding primary page or perhaps by tracking news stories).

As a first step towards this vision we present KYLIN, a prototype which automates part of this vision. KYLIN looks for classes of pages with similar infoboxes, determines common attributes, creates training examples, learns CRF extractors, and runs them on each page — creating new infoboxes and completing others. KYLIN also automatically identifies missing links for proper nouns on each page, resolving each to a unique identifier. Experiments show that the performance of KYLIN is roughly comparative with manual labelling in terms of precision and recall. On one domain, it does even better.

2. GENERATING INFOBOXES

Many Wikipedia articles include *infoboxes*, a concise, tabular summary of the subject's attributes. For example, Figure 1 shows a sample infobox from the article on "Abbeville County," which was generated dynamically from the data shown in Figure 2.

Because of their relational nature, infoboxes may be easily converted to semantic form as shown by Auer and Lehmann's DBpedia [3]. Furthermore, for each class of objects, infoboxes and their

```
{{US County infobox|
county  = Abbeville County|
state = South Carolina |
seal =  |
map = Map of South Carolina highlighting Abbeville County.png |
map size = 200|
founded = 1785 |
seat = [[Abbeville, South Carolina|Abbeville]] |
area = 1,324 [[square kilometre|kmÂ²]] (511 [[square mile|miÂ²]]) |
area land = 1,316 kmÂ² (508 miÂ²) |
area water = 8 kmÂ² (3 miÂ²) |
area percentage = 0.59% |
census yr = 2000|
pop = 26,167 |
density = 20|
web =|
|}}
```

Figure 2: Attribute/value data generating the infobox in Fig. 1

templates implicitly define the most important and representative attributes; hence, infoboxes are valuable ontological resources. In this section we explain how KYLIN automatically constructs and completes infoboxes. The basic idea is to use existing infoboxes as a source of training data with which to learn extractors for gathering more data. As shown in Figure 3, KYLIN's infobox generation module has three main components: preprocessor, classifier, and extractor.

The *preprocessor* performs several functions. First, it selects and refines infobox schemata, choosing relevant attributes. Secondly, the preprocessor generates a dataset for training machine learners.

KYLIN trains two types of *classifiers*. The first type predicts whether a given Wikipedia article belongs to certain class. The second type of classifier predicts whether a given sentence contains the value of a given attribute. If there are C classes and A attributes per class, KYLIN automatically learns $C + AC$ different classifiers.

Extractors are learned routines which actually identify and clip out the necessary attribute values. KYLIN learns one extractor per attribute per class; each extractor is a conditional random fields (CRF) model. Training data are taken from existing infoboxes as dictated by the predictions of the classifiers.

We explain the operation and performance of these modules below. But first we discuss the nature of the existing Wikipedia infoboxes and why they are harder to use for training than might be expected.

2.1 Challenges for Infobox Completion

While infoboxes contain much valuable information, they suffer from several challenging problems:

Incompleteness: Since infobox and article text are kept separate in Wikipedia, existing infoboxes are manually created when human authors create or edit an article — a tedious and time-consuming process. As a result, many articles have no infoboxes and the majority of infoboxes which *do* exist are incomplete. For example, in the "U.S. County" class less than 50% of the articles have an infobox. Still there in many classes, there is plenty of data for training.

Inconsistency: The manual creation process is noisy, causing contradictions between the article text and the infobox summary. For example, when we manually checked a random sample of 50 infoboxes in the "U.S. County" class, we found that 16% contained one or more errors. We suspect that many of the errors are introduced when an author updates an article with a revised attribute

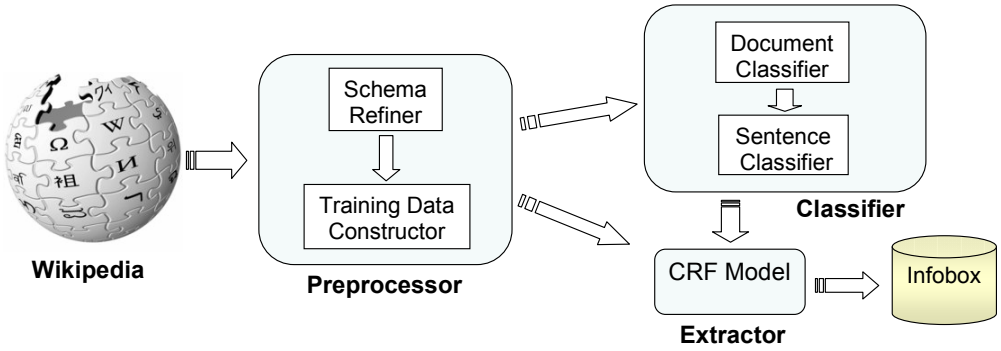

Figure 3: Architecture of KYLIN's infobox generator.

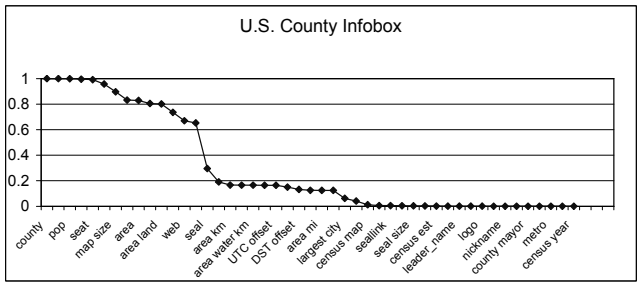

Figure 4: Usage percentage for attributes of the "U.S. County" infobox template.

value (e.g., population) and neglects to change both the text and the infobox — another effect of keeping infobox and text separate.

Schema Drift: Since users are free to create or modify infobox templates, and since they typically create an article by copying parts (e.g., the infobox template) from a similar article, the infobox schema for a class of articles tends to evolve during the course of authoring. This leads to several problems: schema duplication, attribute duplication, and sparseness. As an example of schema duplication, note that four different templates: "U.S. County" (1428), "US County" (574),"Counties"(50) and "County" (19) are used to describe the same type of object. Similarly, multiple tags denote the same semantic attribute. For example, "Census Yr", "Census Estimate Yr", "Census Est." and "Census Year" all mean the same thing. Furthermore, many attributes are used very rarely. Figure 4 shows the percent usage for the attributes of the "U.S. County" infobox template; only 29% of the attributes are used by 30% or more of the articles, and only 46% of the attributes are used by at least 15% of the articles.

Typefree System: The design of Wikipedia is deliberately low-tech, to facilitate human generation of content, and infoboxes are no exception. In particular, there is no type system for infobox attributes. For example, the infobox for "King County, Washington" has a tuple binding the attribute "land area" to equal "2126 square miles" and another tuple defining "land area km" to be "5506 square km" despite the fact that one can be easily derived from another. Clearly this simple approach bloats the schema and increases inconsistency; the similarity between these related attributes also increases the complexity of extraction.

Irregular Lists: List pages, which link to large numbers of similar articles, are a potential source of valuable type information. Unfortunately, because they are designed for human consumption, automated processing is difficult. For example, some list pages separate information in items, while others use tables with different schemas. Sometimes, lists are nested in an irregular, hierarchical manner, which greatly complicates extraction. For example, the "List of cities, towns, and villages in the United States" has an item called "Places in Florida", which in turn contains "List of counties in Florida."

Flattened Categories: While Wikipedia's category tag system seems promising as a source of ontological structure, it is so flat and quirky that utility is low. Furthermore, many tags are purely administrative, e.g. "Articles to be merged since March 2007."

2.2 Preprocessor

The preprocessor is responsible for creating a training suite that can be used to learn extraction code for creating infoboxes. We divide this work into two functions: schema refinement and the construction of training datasets.

Schema Refinement: The previous section explained how collaborative authoring leads to infobox schema drift, resulting in the problems of schema duplication, attribute duplication and sparsity. Thus, a necessary prerequisite for generating good infoboxes for a given class is determining a uniform target schema.

This can be viewed as an instance of the difficult problem of schema matching [11]. Clearly, many sophisticated techniques can be brought to bear, but we adopt a simple statistical approach for our prototype. KYLIN scans the Wikipedia corpus and selects all articles containing the exact name of the given infobox template name. Next, it catalogs all attributes mentioned and selects the most common. Our current implementation restricts attention to attributes used in at least 15% of the articles, which yields plenty of training data.

Constructing Training Datasets: Next, the preprocessor constructs training datasets for use when learning classifiers and extractors. KYLIN iterates through the articles. For each article with an infobox mentioning one or more target attributes, KYLIN segments the document into sentences, using the OpenNLP library [1]. Then, for each target attribute, KYLIN tries to find a unique, corresponding sentence in the article. The resulting labelled sentences form positive training examples for each attribute. Other sentences form negative training examples.

Our current implementation uses several heuristics to match sentences to attributes. If an attribute's value is composed of several sub-values (e.g., "hub cities"), KYLIN splits them and processes each sub-value as follows:

1. For each internal hyperlink in the article and the infobox at-

tributes, find its unique primary URI in Wikipedia (through a redirect page if necessary). For example, both "USA" and "United States of America" will be redirected to "United States". Replace the anchor text of the hyperlink with this identifier.

2. If the attribute value is mentioned by exactly one sentence in the article, use that sentence and the matching token as a training example.

3. If the value is mentioned by several sentences, KYLIN determines what percentage of the tokens in the attribute's *name* are in each sentence. If the sentence matching the highest percentage of tokens has at least 60% of these keywords, then it is selected as a positive training example. For example, if the current attribute is "TotalArea: 123", then KYLIN might select the sentence "It has a total area of 123 square kms", because "123" and the tokens "total" and "area" are all contained in the sentence.

Unfortunately, there are several difficulties preventing us from getting a perfect training dataset. First, OpenNLP's sentence detector is imperfect. Second, the article may not even have a sentence which corresponds to an infobox attribute value. Third, we require exact value-matching between attribute values in the sentence and infobox. While this strict heuristic ensures precision, it substantially lowers recall. The values given in many are incomplete or written differently than in the infobox. Together, these factors conspire to produce a rather incomplete dataset.[2] Fortunately, we are still able to train our learning algorithms effectively.

2.3 Classifying Documents & Sentences

KYLIN learns two types of classifiers. For each class of article being processed, a heuristic *document classifier* is used to recognize members of the class. For each target attribute within a class a *sentence classifier* is trained in order to predict whether a given sentence is likely to contain the attribute's value.

Document Classifier: To accomplish autonomous infobox generation, KYLIN must first locate candidate articles for a given class — a familiar document classification problem. Wikipedia's manually-generated list pages, which gather concepts with similar properties, and category tags are highly informative features for this task. For example, the "List of U.S. counties in alphabetical order" points to 3099 items; furthermore, 68% of those items have additionally been tagged as "county" or "counties." Eventually, we will use lists and tags as features in a Naive Bayes, Maximum Entropy or SVM classifier, but as an initial baseline we used a simple, heuristic approach. First, KYLIN locates all list pages whose titles contain infobox class keywords. Second, KYLIN iterates through each page, retrieving the corresponding articles but ignoring tables. If the category tags of the retrieved article also contains infobox class keywords, KYLIN classifies the article as a member of the class. As shown in Section 4, our baseline document classifier achieves very high precision (98.5%) and reasonable recall (68.8%).

Sentence Classifier: It proves useful for KYLIN to be able to predict which attribute values, if any, are contained in a given sentence. This can be seen as a multi-class, multi-label, text classification problem. To learn these classifiers, KYLIN uses the training set produced by the preprocessor (Section 2.2). For features, we seek a domain-independent set which is fast to compute; our current implementation uses the sentence's tokens and their part of speech (POS) tags as features.

[2]Alternatively, one can view our heuristics as explicitly preferring incompleteness over noise — a logical consequence of our choice of high precision extraction over high-recall.

Feature Description	Example
First token of sentence	*Hello* world
In first half of sentence	*Hello* world
In second half of sentence	Hello *world*
Start with capital	Hawaii
Start with capital, end with period	Mr.
Single capital	A
All capital, end with period	CORP.
Contains at least one digit	AB3
Made up of two digits	99
Made up of four digits	1999
Contains a dollar sign	20$
Contains an underline symbol	km_square
Contains an percentage symbol	20%
Stop word	the; a; of
Purely numeric	1929
Number type	1932; 1,234; 5.6
Part of Speech tag	
Token itself	
NP chunking tag	
String normalization: capital to "A", lowercase to "a", digit to "1", others to "0"	$TF-1 \Longrightarrow AA01$
Part of anchor text	Machine *Learning*
Beginning of anchor text	*Machine* Learning
Previous tokens (window size 5)	
Following tokens (window size 5)	
Previous token anchored	Machine *Learning*
Next token anchored	*Machine* Learning

Table 1: Feature sets used by the CRF extractor

For our classifier, we employed a Maximum Entropy model [24] as implemented in Mallet [21], which predicts attribute labels in a probabilistic way — suitable for multi-class and multi-label classifications.[3] To decrease the impact of a noisy and incomplete training dataset, we employed bagging [6] rather than boosting [22] as recommended by [25].

2.4 Learning Extractors

Extracting attribute values from a sentence may be viewed as a sequential data-labelling problem. We use the features shown in Table 1. Conditional random fields (CRFs) [19] are a natural choice given their leading performance on this task; we use the Mallet [21] implementation. We were confronted with two interesting choices in extractor design, and both concerned the role of the sentence classifier. We also discuss the issue of multiple extractions.

Training Methodology: Recall that when producing training data for extractor-learning, the preprocessor uses a strict pairing model. Since this may cause numerous sentences to be incorrectly labelled as negative examples, KYLIN uses the sentence classifier to *relabel* some of the training data as follows. All sentences which were assigned to be negative training examples by the preprocessor are sent through the sentence classifier; if the classifier disagrees with the preprocessor (i.e., it labels them positive), then they are

[3]We also experimented with a Naive Bayes model, but its performance was slightly worse.

eliminated from the training set for this attribute. Experiments in Section 4 show that this small adjustment greatly improves the performance of the learned CRF extractor.

KYLIN trains a different CRF extractor for each attribute, rather than training a single master extractor that clips all attributes. We chose this architecture largely for simplicity — by keeping each attribute's extractor independent, we ensure that the complexity does not multiply.

Classifier's Role in Extraction: We considered two different ways to combine the sentence classifier and extractor for infobox generation. The first is an intuitive pipeline mode where the sentence classifier selects the sentences which should be sent to the CRF extractor. We expected that this approach would decrease the number of false positives with a potential loss in recall. The second architecture treats the classifier's prediction as a CRF feature, but applies the extractor to all sentences. We expected better recall at the expense of speed. The experiments of Section 4 shows that our expectations were fulfilled, but the pipeline's boost to precision was higher than expected, creating a more effective architecture.

Multiple Extractions: Sometimes the extractor finds multiple values for a single attribute. This often happens as a mistake (e.g. because of an extractor error or redundant text in the article) but can also happen when the attribute is not functional (e.g. a band likely has several members). KYLIN distinguishes the cases by seeing if multiple values are found in the attribute's training set. If so, the set of extractions is returned as the final result. Otherwise, KYLIN returns the single value with the highest confidence.

3. AUTOMATIC LINK GENERATION

A second goal for KYLIN is autonomous link generation for Wikipedia articles. Two Wikipedia resources are useful for this task. Disambiguation pages, which list alternative definitions of a term along with a concise description, help distinguish the correct target for new links. Redirection pages, which redirect a pointer for one term to another article, can be used to identify sets of synonyms. KYLIN uses the following procedure to generate internal links.

First, KYLIN extracts each noun-phrase (NP) from the article, again using OpenNLP. Next, it converts the NPs to their normalized forms. For example, determinants like "a" and "the" are discarded. Only *proper* nouns (i.e., whose first word is capitalized) are retained as candidates for link generation.[4] Finally, for each candidate NP, KYLIN checks the following conditions in order; if one matches, it adds a link.

1. **MatchAnchor:** Exactly matches some existing anchor text in the article.
2. **MatchTitle:** Exactly matches the title of the article.
3. **MatchURI:** Matches a primary URI in Wikipedia without ambiguity.
4. **Disambiguation:** If there is a corresponding disambiguation page, define the *context* of the NP to be the title of the current article plus the sentence where the NP appears, minus all stop words. For each entry in the disambiguation page, check whether the overlap degree between the entry and the *context* is over threshold (e.g. 0.3 in our experiments); if so, choose the best match.
5. **InTitle:** NP is contained in the title.
6. **InAnchor:** NP is contained in some existing anchor text.

[4]A random sample of 50 Wikipedia articles yielded 1213 existing hyperlinks edited by users, and 70.2% had proper nouns as anchors. Clearly, KYLIN should add links defining things other than proper nouns, but selecting which are worthy is a difficult future topic.

	County	Airline	Actor	University
Predicted	3302	2764	6984	4309
Precision (%)	100	100	100	94

Table 2: Estimated precision of the document classifier.

This technique, while ad hoc, performs well. But one might question is whether our matching heuristics are applied in the correct order. We answer this question experimentally in Section 4.

KYLIN assumes that a given NP string denotes the same concept within an article, which is reasonable for Wikipedia. However, when a NP appears in a different sentence, its context changes and hence the disambiguation rule may predict a different target. Thus KYLIN identifies all potential targets for a NP and chooses the one with the highest confidence — increasing both precision and recall. To see why, consider the meaning of "Portland" in the following two sentences:

- *FBI agents from Portland handled the case, in which payment of a demanded ransom of $200,000 secured the release of the victim.*
- *The dominant intercity transportation link between Tacoma and other parts of the Puget Sound is Interstate 5, which links Tacoma with Seattle to the north and Portland, Oregon, to the south.*

Though it is hard to determine the correct target for "Portland" in the first sentence, the context is much more informative in the second.

4. EXPERIMENTS

To avoid overloading the Wikipedia server, we downloaded the 2007.02.06 data in order to test the performance of KYLIN's infobox generation and link creation algorithms.

4.1 Infobox Generation

For testing, we selected four popular classes: U.S. county, airline, actor, and university. Each was among the top 100 classes in terms of infobox usage. We address four questions:

- What are the precision and recall of the document classifier?
- What are the precision and recall of the infobox attribute extractor and how does it compare to the performance of human users?
- Is the precision of the extractor improved by pruning the set of training data with the sentence classifier? What is the cost in terms of recall?
- Should one use the sentence classifier as a serial pipeline filter preceding the extractor or simply make the classifier's output available as a feature for the extractor's use?

	County	Airline	Actor	University
Tagged	1245	791	3819	4025
Recall (%)	98.1	85.3	41.3	50.3

Table 3: Estimated recall of the document classifier

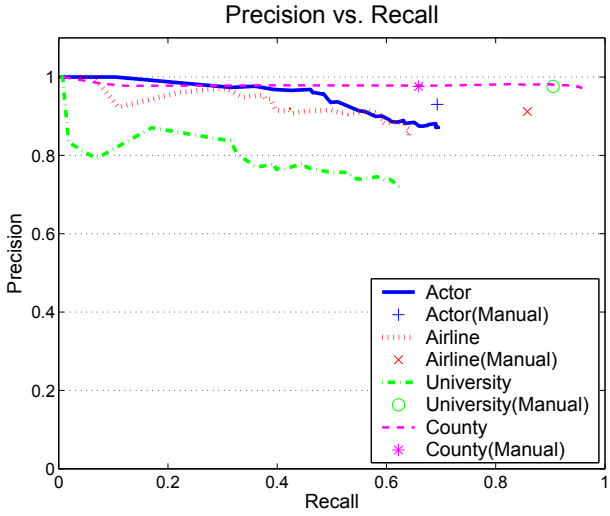

Figure 5: Precision vs. recall curves of infobox generation. The individual points correspond to the performance of Wikipedia users' manual edition.

Document Classification: We use sampling plus human labelling to estimate the precision and recall of the classifiers. We measure the precision of a class' classifier by taking a random sample of 50 pages which were predicted to be of that class and manually checking their accuracy. Table 2 lists the estimated precision for our four classes. On average, the classifiers achieve 98.5% precision.

To estimate recall, we introduce some notation, saying that an article is *tagged* with a class if it has had an infobox of that type manually created by a human author. We use the set of tagged pages as a sample from the universal set and count how many of them are identified by the classifier. Table 3 shows the detailed results, but averaging uniformly over the four classes yields an average recall of 68.8%. This is quite good for our baseline implementation, and it seems likely that a machine-learning approach could result in substantially higher recall.

Note that there are some potential biases which might potentially affect our estimates of precision and recall. First, as mentioned in Section 2.1, some list pages are challenging to exploit, and list page formatting varies a lot between different classes. Second, articles with user-added infobox classes tend to be on more popular topics, and these may have a greater chance to be included in list pages. This could lead to minor overestimation of KYLIN recall. But we believe that these factors are small, and our estimates of precision and recall are accurate.

Infobox Attribute Extractor: In order to be useful as an autonomous system, KYLIN must be able to extract attribute values with very high precision. High recall is also good, but of less importance. Since our CRF extractor outputs a confidence score for its extraction, we can modulate the confidence threshold to control the precision/recall tradeoff as shown in Figure 5.

Interestingly, the precision/recall curves are extremely flat, which means the precision is rather stable w.r.t the variation of recall. In practice, KYLIN is able to automatically tune the confidence threshold based on training data provided by the preprocessor for various precision/recall requirements. In order to reduce the need for human fact checking, one can set a high threshold (e.g., 0.99), boosting precision. A lower threshold (e.g. 0.3) extends recall substantially, at only a small cost in precision.

	People		System	
Class	Pre.(%)	Rec.(%)	Pre.(%)	Rec.(%)
County	97.6	65.9	97.3	95.9
Airline	92.3	86.7	87.2	63.7
Actor	94.2	70.1	88.0	68.2
University	97.6	90.5	73.9	60.5

Table 4: Relative performance of people and KYLIN on infobox attribute extraction.

In our next experiment, we use a fixed threshold of 0.6, which achieves both reasonable precision and recall for all classes. We now ask how KYLIN compares against strong competition: human authors. For each class, we randomly selected 50 articles with existing infobox templates. By manually extracting all attributes mentioned in the articles, we could check the performance of both the human authors and of KYLIN. The results are shown in Table 4. We were proud to see that KYLIN performs better on the "U.S. County" domain, mainly because its numeric attributes are relatively easy to extract. In this domain, KYLIN was able to successfully recover a number of values which had been neglected by humans. For the "Actor" and "Airline" domains, KYLIN performed slightly worse than people. And in the "University" domain, KYLIN performed rather badly, because of implicit references and the type of flexible language used in those articles. For example, KYLIN extracted "Dwight D. Eisenhower" as the president of "Columbia University" from the following sentence.

- *Former U.S. President Dwight D. Eisenhower served as President of the University.*

Unfortunately, this is incorrect, because Eisenhower was a *former* president (indicated somewhere else in the article) and thus the incorrect value for the current president.

Implicit expressions also lead to challenging extractions. For example, the article on "Binghamton University" individually describes the number of undergraduate and graduate students in each college and school. In order to correctly extract the total number of students, KYLIN would need to reason about disjoint sets and perform arithmetic, which is beyond the abilities of most textual entailment systems [9, 20], let alone one that scale to a Wikipedia-sized corpus.

Using the Sentence Classifier with the Attribute Extractor: Recall that KYLIN uses the sentence classifier to prune some of the negative training examples generated by the preprocessor before training the extractor. We also explored two ways of connecting the sentence classifier to the extractor: as a pipeline (where only sentences satisfying the classifier are sent to the extractor) or by feeding the extractor every sentence, but letting it use the classifier's output as a feature. In this experiment, we consider four possible configurations:

- Relabel, Pipeline — uses the classifier's results to relabel the training dataset for the extractor and uses a pipeline architecture.
- Relabel, #Pipeline — also uses the classifier's results to relabel the training dataset for the extractor, but doesn't use a

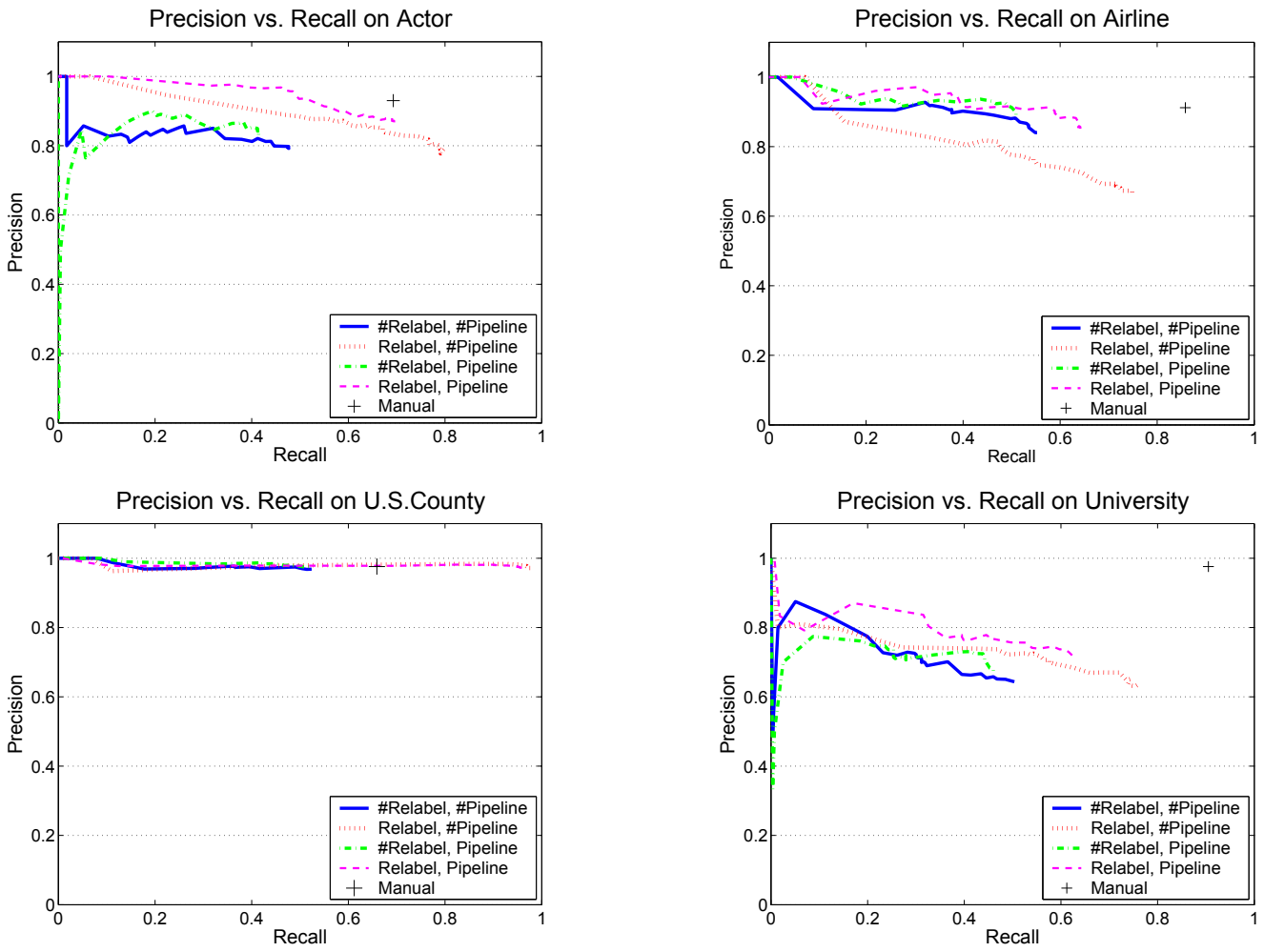

Figure 6: Precision and recall curves with different policies on combining sentence classifier and CRF extractor. The cross points correspond to Wikipedia users' manual edition.

pipeline (instead it provides the classifier's output to the CRF extractor).

- #Relabel, Pipeline — training examples are not relabelled, but the pipelined architecture is used.
- #Relabel, #Pipeline — training examples are not relabelled and a pipeline is eschewed (the classifier's output is fed directly to the extractor).

Figure 6 shows the detailed results. In most cases the "Relabel, Pipeline" policy achieves the best performance. We draw the following observations:

- Noise and incompleteness within the training dataset provided by the Preprocessor makes the CRF extractor unstable, and hampers its performance (especially recall) in most cases.
- By using classifier to refine the training dataset, many false negative training examples are pruned; this helps to enhance the CRF extractor's performance, especially in terms of robustness and recall.
- The pipeline architecture improves precision in most cases by reducing the risk of false positive extractions on irrelevant sentences. But since fewer sentences are even given to the extractor, recall suffers.

4.2 Internal Link Generation

In this section we address the following questions:

- What are the precision and recall of KYLIN's link-generation method?
- How do the heuristic rules effect this performance? Can KYLIN figure out the right order in which to apply the heuristics via self-supervised learning?

The first question is easy. Sampling over 50 randomly generated pages, we found 1213 unique human-generated hyperlinks, 852 of which were anchored by proper nouns. The set of pages contained additional 369 unique proper nouns that we judged deserving of a link.[5] Thus, we see that human authors display 69.8% recall,

[5]This judgment is a bit subjective and it is important to note that we are treating links as semantic structure whose objective is to uniquely specify the noun's meaning — not facilitate a human's reading pleasure.

Heuristic	Pre.(%)	Rec.(%)
MatchTitle	100	0.2
MatchAnchor	92.8	9.0
MatchUrl	85.0	49.2
Disambiguation	38.5	1.8
InTitle	18.8	0.3
InAnchor	14.3	0.1

Table 5: Performance of various link-generation heuristics on existing links.

Heuristic	Pre.(%)	Rec.(%)
MatchTitle	100	4.06
MatchAnchor	97.9	12.7
MatchUrl	90.8	42.5
Disambiguation	62.5	4.07
InTitle	75.4	11.6
InAnchor	57.8	17.1

Table 6: Performance of various link-generation heuristics on new links.

presumably with near 100% precision. When KYLIN was asked to find links for the proper nouns left unlinked by humans, it generated 291 links of which 261 were correct. This yields a link-generation precision of 89.7% and a human-level recall of 70.7%.

The six heuristics listed in Section 3 form the core of KYLIN's link generator. Each of the heuristics sounds plausible, but as one might suspect, the correct order matters considerably. Interestingly, we can use the set of human-authored internal links as another training set and use it to choose the right order for the heuristics — another form of self-supervised learning.

We first measure each heuristic's performance on the suite of user-added hyperlinks, and then order them in order of decreasing precision score. Table 5 shows each heuristic's precision on a random sample of 50 Wikipedia articles. The order determined by KYLIN matches well with our intuitions. To demonstrate that the ordering works well on as-yet unlinked noun phrases, we tested it on a manually-labelled sample of 50 articles (Table 6). Note that while the individual numbers have changed substantially, the relative order is stable.

The difference in quantitative precision/recall numbers is due to the distinct characteristics of the nouns in datasets corresponding to Tables 5 and 6. Table 5 is based on existing anchor texts edited by users. A quick check of Wikipedia articles reveals that users seldom add links pointing to the current article, which leads to a performance decrease of the "MatchTitle" and "InTitle" heuristics. If the same NP appears several times, users tend to only add a link at its first occurrence, which effects the "MatchAnchor" and "InAnchor" heuristics. When users add a link to help disambiguate concepts, usually they are "harder" than randomly picked noun-phrases, which explains the lower performance of the "Disambiguation" heuristic in Table 5.

If human linking behavior is so different from KYLIN's exhaustive approach, one might question the utility of automatic link generation itself. We agree that KYLIN's links might not help *human* readers as much as those added by human authors, but our objective is to help programs, not people! By disambiguating numerous noun phrases by linking to a unique identifier, we greatly simplify subsequent processing. Furthermore, we note that many are to pages which have not yet been linked in the article.

For the next experiment, we enumerated all $6! = 720$ heuristic orderings and measured the precision and recall of the collection as a whole. Table 7 lists the performance of KYLIN's ordering as well as the best and worst orders. We can see there is little difference between KYLIN and the optimal one, and both of them perform more than 10% better than the worst ordering.

Ordering	Pre.(%)	Rec.(%)
KYLIN	89.7	70.7
Best Order	90.0	71.0
Worst Order	78.4	61.8

Table 7: Effect of different heuristic orders on link generation performance.

5. RELATED WORK

We group related work into several categories: bootstrapping the semantic web, unsupervised information extraction, extraction from Wikipedia, and related Wikipedia-based systems.

Bootstrapping the Semantic Web: REVERE [17] aims to cross the chasm between structured and unstructured data by providing a platform to facilitate the authoring, querying and sharing of data. It relies on human effort to gain semantic data, while our KYLIN is fully autonomous. DeepMiner [30] bootstraps domain ontologies for semantic web services from source web sites. It extracts concepts and instances from semi-structured data over source interface and data pages, while KYLIN handles both semi-structured and unstructured data in Wikipedia. The SemTag and Seeker [10] systems perform automated semantic tagging of large corpora. They use the TAP knowledge base [27] as the standard ontology, and use it to match instances on the Web. In contrast, KYLIN doesn't assume any particular ontology, and tries to extract all desired semantic data within Wikipedia.

Unsupervised Information Extraction: Since the Web is large and highly heterogeneous, unsupervised and self-supervised learning is necessary for scaling. Several systems of this form have been proposed. MULDER [18] and AskMSR [7, 13] use the Web to answer questions, exploiting the fact that most important facts are stated multiple times in different ways, which licenses the use of simple syntactic processing. KNOWITALL [14] and TEXTRUNNER [4] use search engines to compute statistical properties enabling extraction. Each of these systems relies heavily on the Web's information redundancy. However, unlike the Web, Wikipedia has little redundancy — there is only one article for each unique concept in Wikipedia. Instead of utilizing redundancy, KYLIN exploits Wikipedia's unique structure and the presence of user-tagged data to train machine learners.

Information Extraction from Wikipedia: Several other systems have addressed information extraction from Wikipedia. Auer and Lehmann developed the DBpedia [3] system which extracts information from existing infoboxes within articles and encapsulate them in a semantic form for query. In contrast, KYLIN populates infoboxes with *new* attribute values. Suchanek et al. describe the YAGO system [28] which extends WordNet using facts extracted from Wikipedia's category tags. But in contrast to KYLIN, which can learn to extract values for *any* attribute, YAGO only extracts values for a limited number of predefined relations.

Nguyen et al. proposed to extract relations from Wikipedia by exploiting syntactic and semantic information [23]. Their work is the most similar with ours in the sense of stepping towards autonomously semantifying both semi-structured and unstructured data. However, there are still several obvious distinctions. First, their

system only classifies whether a sentence is related to some attribute, while KYLIN also *extracts* the particular attribute value within the sentences. Second, they only care about the relationship-typed attributes between concepts (i.e. objects having their own identifying pages), while KYLIN tries to extract *all* important attributes. Third, their system targets a limited number of predefined attributes, while KYLIN can dynamically refine infobox templates for different domains.

Other Wikipedia-Related Systems: Völkel et al. proposed an extension to be integrated with Wikipedia, which allows the typing of links between articles and the specification of typed data inside the articles in an easy-to-use manner [29]. Though a great step towards semantifying Wikipedia, it still relies on manual labelling by human users. Gabrilovich et al. used Wikipedia to enhance text categorization [15], and later proposed a semantic-relatedness metric using Wikipedia-based explicit semantic analysis [16]. Ponzetto et al. derived a large scale taxonomy containing subsumption relations based on the category system in Wikipedia [26]. Adafre et al. tried to discover missing links in Wikipedia by first computing a cluster of highly similar pages around a target page, then identifying candidate links from those similar pages [2]. In contrast, KYLIN searches all proper noun-phrases within the target page for link creations.

6. DISCUSSION

Although our objective is the automatic extraction of structured data from natural-language text on Wikipedia and eventually the whole Web, our investigation has uncovered some lessons that directly benefit Wikipedia and similar collaborative knowledge repositories. Specifically, Wikipedia could greatly improve consistency if it were augmented with a software robot (perhaps based on KYLIN) which functioned as an automatic fact-checker. When an article was created or edited, this agent could:

- Suggest new entries for an associated infobox
- Detect inconsistencies between the text and infobox attributes
- Note schema inconsistencies in infoboxes and suggest attribute names which have been used previously
- Detect incomplete disambiguation pages and add links to other matching articles
- Suggest additional internal links.

6.1 Conclusion

This paper described KYLIN, a prototype system which autonomously extracts structured data from Wikipedia and regularlizes its internal link structure. Since KYLIN uses self-supervised learning, which is bootstrapped on existing user-contributed data, it requires little or no human guidance. We make the following contributions:

- We propose bootstrapping the Semantic Web by mining Wikipedia and we identify some unique challenges (lack of redundancy) and opportunities (unique identifiers, user-supplied training data, lists, categories, etc.) of this approach. We also identify additional issues resulting from Wikipedia's growth through decentralized authoring (e.g., inconsistency, schema drift, etc.). This high-level analysis should benefit future work on Wikipedia and similar collaborative knowledge repositories.
- We describe a systems for automatically generating attribute/value pairs summarizing an article's properties. Based on self-supervised learning, KYLIN achieves performance which is roughly comparable with that of human editors. In one case, KYLIN does even better.
- By automatically identifying missing internal links for proper nouns, more semantic tags are added. Because these links resolve noun phrases to unique identifiers, they are useful for many purposes such as information retrieval, structural analysis, and further semantic processing. Meaning lies in the graph structure of concepts defined in terms of each other, and KYLIN helps complete that graph.
- Collaboratively authored data is rife with noise and incompleteness. We identify robust learning methods which can cope in this environment. Extensive experiments demonstrate the performance of our system and characterize some of the crucial architectural choices (e.g., the optimal ordering of heuristics, the utility of classifier-based training data refinement, a pipelined architecture for attribute extraction).

6.2 Future Work

For an initial prototype, KYLIN performs quite well. But there are numerous directions for improvement. Many of KYLIN's components are simple baseline implementations, because we wished an end-to-end system. We wish to apply learning to the problem of document classification, consider more sophisticated ways of combining heuristics (e.g., stacked metalearning), test on more cases, make the result public (e.g., as a Firefox extension), and other improvements. In the longer term, we will investigate the following directions:

- **Schema Matching:** Currently we took a simple statistical way to clean infobox templates. More sophisticated strategies for schema matching are desired.
- **Information Verification:** Besides automatic information extraction, KYLIN should also be able to verify the correctness of the extracted information so that inconsistency can be correctly resolved. An intuitive way is utilizing outside Web knowledge such as Google indices.
- **Taxonomy Construction:** Wikipedia contains many distinct types of information, which have very similar schemata. List and category information is rudimentary. Never-the-less, there appears to be great potential for automatic construction of a useful ontological resource.
- **Topic Discovery:** As the largest collaborative encyclopedia, Wikipedia should extend its coverage as much as possible. But as Wikipedia grows, it is getting harder and harder for users to identify missing or incomplete articles. KYLIN should be able to identify these topics, and facilitate users' editting (for example, pointing users to useful information sources outside Wikipedia).

7. ACKNOWLEDGMENTS

We thank Oren Etzioni, Alex Yates, Matt Broadhead, and Michele Banko for providing the code of their software and useful discussions. We also thank anonymous reviewers for valuable suggestions and comments. This work was supported by NSF grant IIS-0307906, ONR grant N00014-06-1-0147, SRI CALO grant 03-000225 and the WRF / TJ Cable Professorship.

8. REFERENCES

[1] http://opennlp.sourceforge.net/.

[2] S. F. Adafre and M. de Rijke. Discovering missing links in wikipedia. In *Proceedings of the 3rd International Workshop on Link Discovery at KDD05*, Chicago, USA, August 2005.

[3] S. Auer and J. Lehmann. What have Innsbruck and Leipzig in common? Extracting semantics from wiki content. In *ESWC*, 2007.

[4] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *Proceedings of EMNLP*, 2002.

[8] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference*, 2001.

[9] R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *National Conference on Artificial Intelligence (AAAI)*, pages 1678–1679, 2005.

[10] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Y. Zien. Semtag and Seeker: bootstrapping the Semantic Web via automated semantic annotation. In *Proceedings of 12th International World Wide Web Conference*, pages 178–186, 2003.

[11] A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine, Special Issue on Semantic Integration*, 2005.

[12] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *Procs. of IJCAI 2005*, 2005.

[13] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference*, 2002.

[14] 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.

[15] E. Gabrilovich and S. Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1301–1306, 2006.

[16] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, January 2007.

[17] A. Y. Halevy, O. Etzioni, A. Doan, Z. G. Ives, J. Madhavan, L. McDowell, and I. Tatarinov. Crossing the structure chasm. In *Proceedings of CIDR*, 2003.

[18] C. T. Kwok, O. Etzioni, and D. Weld. Scaling question answering to the Web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

[19] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 15th International Conference on World Wide Web*, Edinburgh, Scotland, May 2001.

[20] B. MacCartney and C. D. Manning. Natural logic for textual inference. In *Workshop on Textual Entailment and Paraphrasing, ACL 2007*, 2007.

[21] A. K. McCallum. Mallet: A machine learning for language toolkit. In *http://mallet.cs.umass.edu*, 2002.

[22] R. Meir and G. Rätsch. An introduction to boosting and leveraging. *Journal of Artificial Intelligence Research*, Advanced Lectures on Machine Learning:118–183, 2003.

[23] D. P. Nguyen, Y. Matsuo, and M. Ishizuka. Exploiting syntactic and semantic information for relation extraction from wikipedia. In *IJCAI07-TextLinkWS*, 2007.

[24] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[25] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, pages 169–198, 1999.

[26] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22st National Conference on Artificial Intelligence*, pages 1440–1445, 2007.

[27] E. Riloff and J. Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124, Providence, RI, 1997.

[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[29] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[30] W. Wu, A. Doan, C. Yu, and W. Meng. Bootstrapping domain ontology for Semantic Web services from source web sites. In *Proceedings of the VLDB-05 Workshop on Technologies for E-Services*, 2005.