Augmenting Wikipedia-Extraction with Results from the Web

Fei Wu and **Raphael Hoffmann** and **Daniel S. Weld**
CSE Department
University of Washington
Seattle, USA
{wufei,raphaelh,weld}@cs.washington.edu

Abstract

Not only is Wikipedia a comprehensive source of quality information, it has several kinds of internal structure (e.g., relational summaries known as *infoboxes*), which enable self-supervised information extraction. While previous efforts at extraction from Wikipedia achieve high precision and recall on well-populated classes of articles, they fail in a larger number of cases, largely because incomplete articles and infrequent use of infoboxes lead to insufficient training data. This paper explains and evaluates a method for improving recall by extracting from the broader Web. There are two key advances necessary to make Web supplementation effective: 1) a method to filter promising sentences from Web pages, and 2) a novel *retraining* technique to broaden extractor recall. Experiments show that, used in concert with shrinkage, our techniques increase recall by a factor of up to 8 while maintaining or increasing precision.

Introduction

Like many others at the workshop, we wish to convert as many facts in Wikipedia as possible into semantic form. Such a system could be useful for next-generation search, question answering and much more. Performing this process autonomously is crucial, since the scale of available knowledge is vast. In many ways our vision is shared with those working on general-purpose information extraction, such as Snowball (Agichtein & Gravano 2000), KnowItAll (Etzioni *et al.* 2005) and Textrunner (Banko *et al.* 2007), but in contrast to systems which seek to extract from arbitrary Web text, we focus on Wikipedia and hope to expand from that base.

The Long-Tailed Challenge: While focusing on Wikipedia helps solve the problem of inaccurate and unreliable source data (Giles 2005), it introduces new challenges. For example, many previous systems (e.g., Mulder (Kwok, Etzioni, & Weld 2001), AskMSR (Brill, Dumais, & Banko 2002), and KnowItAll) exploit the presence of redundant information on the Web, enabling powerful statistical techniques. The Wikipedia corpus, however, has greatly reduced duplication. Fortunately, Wikipedia has several attributes that significantly facilitate extraction: 1) Infoboxes, tabular summaries of an object's key attributes, may be used as a source of training data, allowing for self-supervised learning. 2) Wikipedia gives important concepts their own unique identifier — the URI of a definitional page. The first reference to such a concept often includes a link which can be used for disambiguation. As a result, homonyms are much less of a problem than in unstructured text. 3) Wikipedia *lists* and *categories* provide valuable features for classifying pages.

This paper reports on K2, which extends Wu and Weld's available Kylin system — a self-supervised Wikipedia information extractor (Wu & Weld 2007). Like Kylin, K2 looks for sets of pages with similar infoboxes, determines common attributes for each class, creates training examples, learns extractors, and runs them on each page — creating new infoboxes and completing others. Kylin, itself, works quite well for *popular* infobox classes where users have previously created enough infoboxes to train an effective extractor model. For example, in the "U.S. County" class Kylin has 97.3% precision with 95.9% recall. Unfortunately, most classes contain only a *small number* of infobox-containing articles. Specifically, 1442 of 1756 (82%) classes have fewer than 100 articles, and 42% have 10 or fewer. For classes sitting on this long tail, Kylin can't get enough training data and its extraction performance may be unsatisfactory.

Furthermore, even when Kylin does learn an effective extractor there are numerous cases where a Wikipedia article simply doesn't have much information to be extracted. Indeed, another long-tailed distribution governs the *length* of articles; among the 1.8 million pages,[1] many are short articles and almost 800,000 (44.2%) are marked as *stub* pages, indicating that much-needed information is missing.

Contributions: Thus, in order to create a comprehensive semantic knowledge base summarizing Wikipedia topics, we must confront the problems of these long-tailed distributions. This paper presents K2, which extends Kylin with novel techniques for increasing recall.

- By mapping the contents of known Wikipedia infobox data to TextRunner, a state-of-the-art open information extraction system (Banko *et al.* 2007), K2 creates a larger and cleaner training dataset for learning more robust extractors.
- When it is unable to extract necessary information from a Wikipedia page, K2 retrieves relevant sentences from

[1] Unless noted otherwise, all statistics are taken from the 07/16/2007 snapshot of Wikipedia's English language version.

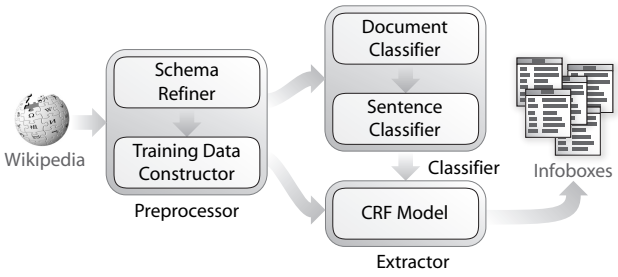

Figure 1: Kylin performs self-supervised information extraction, using Wikipedia inforboxes for training data.

the greater Web. The key to this method is a process for tightly filtering which non-Wikipedia sentences are given to the K2 extractors.

Our techniques work best in concert. Together with shrinkage, they improve the area under the P/R curve by as much as 8, compared with baseline Kylin.

Background: Extraction in Kylin

Following (Wu & Weld 2007) we are interested in the problem of infobox completion. An infobox is a relational summary of an article: a set of attribute / value pairs describing the article's subject (see (Wu & Weld 2007) for an example). Not every article has an infobox and some infoboxes are only partially instantiated with values. We seek to create or complete infoboxes whenever possible. Before explaining how K2 extracts data from the general Web to supplement that found in Wikipedia, we review the basic Kylin architecture (Figure 1), upon which we build.

Preprocessor: The preprocessor selects and refines infobox schemata, choosing relevant attributes; 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.

Next, the preprocessor constructs two types of training datasets — those for sentence classifiers, and CRF attribute extractors. For each article with an infobox mentioning one or more target attributes, Kylin tries to find a unique sentence in the article that mentions that attribute's value. The resulting labelled sentences form positive training examples for each attribute; 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 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. For this, Kylin uses a maximum entropy model (Nigam, Lafferty, & McCallum 1999) 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-labelling problem. Kylin uses conditional random fields (CRFs) (Lafferty, McCallum, & Pereira 2001) 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 master extractor to clip all attributes, Kylin trains a different CRF extractor for each attribute, ensuring simplicity and fast retraining.

Shrinkage: Although Kylin performs well when it can find enough training data, it flounders on sparsely populated infobox classes — the majority of cases. We partially mitigated this problem by refining Wikipedia's infobox ontology (Wu & Weld 2008) and improving Kylin's performance using shrinkage, a general statistical technique for improving estimators in the case of limited training data (McCallum *et al.* 1998). K2 uses shrinkage when training an extractor of a instance-sparse infobox class by aggregating data from its parent and children classes.

Shrinkage improves Kylin's precision, but more importantly, it increases recall, when extracting from the long tail of sparse infobox classes. Because this technique leads to extractors with improved robustness, we use shrinkage in K2, when extracting information from the general Web.

Retraining

We now consider how to improve extractor robustness by harvesting additional training data from the *outside* Web. Leveraging information outside Wikipedia to help training extractors, could improve Kylin's recall. To see why, we note that the wording of texts from the greater Web are more diverse than the relatively strict expressions used in many places in Wikipedia.[2] Training on a wider variety of sentences would improve the robustness of Kylin's extractors, which would potentially improve the recall.

The trick here is determining how to automatically identify relevant sentences given the sea of Web data. For this purpose, K2 utilizes TextRunner, an open information extraction system (Banko *et al.* 2007), which extracts semantic relations $\{r|r = \langle obj_1, predicate, obj_2\rangle\}$ from a crawl of about 10 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. In the next subsection, we explain the details of our retraining process; then we follow with an experimental evaluation.

Using TextRunner for Retraining: Recall that each Wikipedia infobox implicitly defines a set of semantic triples $\{t|t = \langle subject, attribute, value\rangle\}$ where the subject corresponds to the entity which is the article's title. These triples have the same underlying schema as the semantic relations extracted by TextRunner and this allows us to generate new training data.

The retrainer iterates through each infobox class C and again through each attribute, $C.a$, of that class collecting a set of triples from existing Wikipedia infoboxes: $T =$

[2] It is possible that Wikipedia's inbred style stems from a pattern where one article is copied and modified to form another. A general desire for stylistic consistency is another explanation.

$\{t|t.attribute = C.a\}$.[3] The retrainer next iterates through T, issuing TextRunner queries to get a set of potential matches $R(C.a) = \{r|\exists t : r.obj_1 = t.subject, r.obj_2 = t.value\}$, together with the corresponding sentences which were used by TextRunner for extraction. The K2 retrainer uses this mapped set $R_{C.a}$ to augment and clean the training data set for C's extractors in two ways: by providing additional positive examples for the learner, and by eliminating false negative examples which were mistakenly generated by Kylin from the Wikipedia data.

Adding Positive Examples: Unfortunately, TextRunner's raw mappings, $R(C.a)$, are too noisy to be used as positive training examples. There are two causes for the noise. The most obvious cause is the imperfect precision of TextRunner's extractor. But false positive examples can also be generated when there are multiple interpretations for a query. Consider the TextRunner query $\langle r.obj_1 = A, r.predicate =?, r.obj_2 = B\rangle$, where A is a person and B is his birthplace. Since many people die in the same place that they were born, TextRunner may well return the sentence "Bob died in Seattle." which would be a poor training example for birthplace.

Since false positives could greatly impair training, the K2 retrainer morphologically clusters the predicates which are returned by TextRunner (e.g., "is married to" and "was married to" are grouped). We discard any predicate that is returned in response to a query about more than one infobox attribute. Only the k most common remaining predicates are then used for positive training examples; in our experiments we set $k = 1$ to ensure high precision.

Filtering Negative Examples: As explained in (Wu & Weld 2007), Kylin considers a sentence to be a negative example unless it is known to be positive or the *sentence classifier* labels it as potentially positive. This approach eliminates many false negatives, but some remain. A natural idea is to remove a sentence from the set of negative examples if it contains the word denoting the relation itself. Unfortunately, this technique is ineffective if based soley on Wikipedia content. To see why, consider the "Person.spouse" attribute which denotes the "marriage" relation —because the word "spouse" seldom appears in natural sentences, few false negatives are excluded. But by using TextRunner, we can better identify the phrases (predicates) which are harbingers of the relation in question. The most common are used to eliminate negative examples. By adding new positive examples and excluding sentences which might be false negatives, retraining generates an improved training set, whose benefit we now quantify.

Retraining Experiments: We pose two questions: 1) Does K2's retraining improve over Kylin's extractors? 2) Do the benefits from retraining combine synergistically with those from shrinkage? Before addressing those questions we experimented with different retraining alternatives (e.g., just adding positive examples and just filtering negatives). While both approaches improved extractor performance, the combination worked best, so the combined method was used in the subsequent study.

We evaluate retraining in two different cases. In the first case, we use nothing but the target class' infobox data to prime TextRunner for training data. In the second case, we first used uniform-weight shrinkage to create a training set which was then used to query TextRunner. To compute precision and recall, we manually verified the attribute values contained in the articles. This made it necessary to limit the evaluation set and so we tested on four randomly picked classes of different sizes. We got improved results on each; Figure 2 shows the results on the sparsest and on the most popular class.

We note that in most cases retraining improves the performance, in both precision and recall. When compared with shrinkage, retraining provides less benefit for sparse classes but helps more on the popular class "writer." This makes sense because without many tuples to use for querying TextRunner, retraining has little effect. We suspect that full cotraining would be more effective on sparse classes when shrinkage was unavailable. Finally, we observe that the combination of shrinkage and retraining is synergistic, always leading to the biggest improvement. Particularly, on the two sparsest classes "Irish Newspaper" and "Performer", it substantially improved recall by 590% and 73.3% respectively, with remarkable improvement in precision as well; and the areas under the precision and recall curve improve 1753% and 66% respectively. For the more popular classes "Baseball Stadium" and "Writer" recall improved by 41% and 11% respectively.

Extracting from the Web

While shrinkage and retraining improve the quality of Kylin's extractors, the lack of redundancy of Wikipedia's content makes it increasingly difficult to extract additional information. Facts that are stated using uncommon or ambiguous sentence structures often hide from the extractors.

In order to retrieve facts which can't be extracted from Wikipedia, we extract from the general Web: We train extractors on Wikipedia articles and then apply them to relevant Web pages. The challenge — as one might expect — is maintaining high precision. Since the extractors have been trained on a very selective corpus, they are unlikely to discriminate irrelevant information. For example, an extractor for a person's birthdate will have been trained on a set of pages all of which have that person's life 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.

To ensure extraction quality, it is crucial to carefully select which content is to be processed by K2's extractors. We viewed this task as an information retrieval problem, and solved it in the following steps: K2 generates a set of queries and utilizes a general Web search engine, namely Google, to identify a set of pages which are likely to contain the desired information. The top-k pages are then downloaded, and the

[3] We note that another way of generating the set, T, would be to collect baseline Kylin extractions for $C.a$ instead of using existing infoboxes. This would lead to a *cotraining* approach rather than simple retraining. One could iterate the process of getting more training date from TextRunner with improvements to the Kylin extractor (Blum & Mitchell 1998).

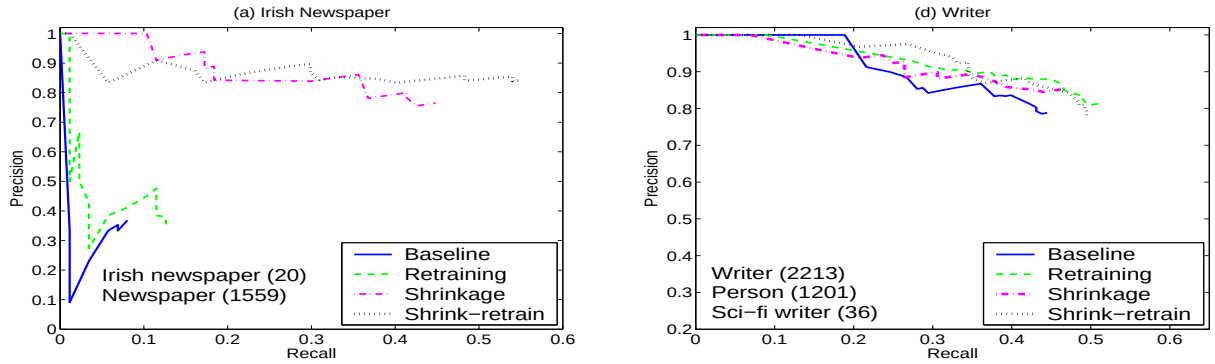

Figure 2: Used in isolation, retraining enables a modest but marked improvement in recall. And combining retraining with shrinkage yields substantially improved extractors with dramatic improvements to precision in the sparse Irish Newspaper domain (only 20 infoboxes) and improved recall in both domains. Note that Irish Newspaper used shrinkage from the paper Newspaper class (1559 infoboxes), while Writer used shrinkage from both a parent and a child class.

text on each page is split into sentences, which K2 processes in turn. Finally, each extraction is weighted using a combination of factors which we will explain shortly.

Choosing Search Engine Queries: The first step is ensuring that the search engine returns highly relevant pages. A simple approach is to use the article title as a query. Suppose we are interested in finding the birth date of Andrew Murray, a writer, whose Wikipedia page is titled "Andrew Murray (minister)". Wikipedia uses information in parentheses to resolve ambiguities, but K2 removes it to increase recall. To improve result relevance, quotes are placed around the remaining string, here ``andrew murray''.

Although such a query might retrieve many pages about Murray, it is possible that none of the top k contains the attribute value in question. K2 therefore runs several more restrictive queries which contain additional keywords to better target the search.

One such query is the quoted article title followed by the attribute name, as in ``andrew murray'' birth date. While this increases the chance that a returned page contains the desired information, it also greatly reduces recall, because the terms 'birth date' might not actually appear on a relevant page. For example, consider the sentence "Andrew Murray was born in 1828.".

But note that K2 has already computed a list of predicates which are indicative of each attribute (e.g. 'was born in' for the birth date), as explained in our section on retraining. Thus, K2 generates appropriate queries for each predicate, which combines the quoted title with these predicates. The combined results of all queries (title only, title and attribute name, as well as title and any attribute predicate) are retrieved for further processing.

Weighing Extractions: Pages which do not contain the preprocessed article title, here 'Andrew Murray', are discarded. Then, formatting commands and scripts are removed, and sentences in the remaining text are identified.

Since most sentences are still irrelevant, running Kylin's extractors on these directly would result in many false positives. Recall that unlike Wikipedia's articles, web pages often compare multiple related concepts, and so we would

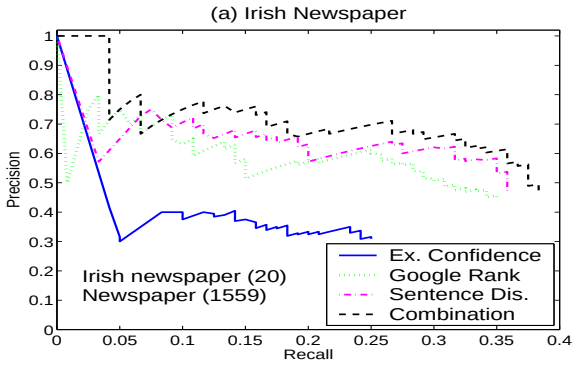

Figure 3: When applying K2 to Web pages, the CRF extractor's confidence is a poor choice for scoring competing extractions of the same attribute. By factoring in IR features, performance improves substantially.

like to capture the likeliness that a sentence or extraction is relevant to a concept. A variety of features may be indicative of content relevance, but K2 uses two in particular:

- The number of sentences δ_s between the current sentence and the closest sentence containing the (preprocessed) title of the article.
- The rank of the page δ_r on Google's results lists returned in response to our queries.

Each retrieved sentence is processed by Kylin's extractors, and for each extraction a combined score is computed. This score takes into account both factors δ_s and δ_r as well as the confidence δ_c reported by Kylin's extractors. The combined score is obtained in the following way: First, each of the three parameters $\delta_s, \delta_r, \delta_c$ is normalized by applying a linear mapping into the intervals $[\alpha_s, 1]$, $[\alpha_r, 1]$, and $[\alpha_c, 1]$ respectively, where 1 corresponds to the optimal value and α_s, α_r, and α_c are user-defined parameters. With δ_s^*, δ_r^*, and δ_c^* denoting the normalized weights, the combined score is then obtained as $score_{web} := \delta_s^* * \delta_r^* * \delta_c^*$.

Combining Wikipedia and Web Extractions: Finally, K2 combines extraction results from Wikipedia and Web pages. Extractions from Wikipedia are ranked by extractor confidence $score_{wiki} := \delta_c$ and extractions from the Web by

$score_{web}$ as defined above. But what is the overall best extraction? We expect that extractions from Wikipedia tend to be more precise (a given Wikipedia article is known to be relevant, of high quality, and of consistent structure for which Kylin's extractors have been trained), but fewer.

K2 applies a simple normalization and always returns the extraction with highest score. To be able to balance the weight of one extractor versus the other, K2 adjusts the score of extractions from the Web to $1 - (1 - score_{web})^{\lambda}$, where λ is a new parameter. If $\lambda = 0$, extractions from the Web are not considered, and if $\lambda = 1$, $score_{wiki}$ and $score_{web}$ are compared directly.

Web Experiments: In our experiments we investigated 1) how to best weigh extractions from the Web, and 2) if our techniques for combining extractions from Wikipedia and Web pages improve recall while maintaining precision.

We assume that there exists some correct value for each attribute contained in the infobox template for an article and define recall to be the proportion of correct attribute values relative to all attributes. Note that most infoboxes in Wikipedia do not provide a value for each attribute contained in the corresponding infobox template. For example, the attribute "spouse" does not make sense for people who are not married, and "death date" for people who are still alive. Therefore, our recall estimates are conservative, but enable a relative comparison of our proposed techniques.

For all experiments, we queried Google for the top-100 pages containing the article title, and the top-10 pages containing the article title and the attribute name or any associated predicate. Each new extraction — for which no ground truth existed in Wikipedia — was manually verified for correctness by visiting the source page.

In our first series of experiments (Figure 3), we used Shrink-Retrain — the best extractors trained on Wikipedia — and applied different weighting functions to select the best extraction for an attribute. The CRF extractor's reported confidence performed poorly in isolation. Giving priority to extractions from pages at a higher position in Google's returned result lists and resolving ties by confidence, yielded a substantial improvement. Similarly, we tried giving priority to extractions which were fewer sentences apart from the occurrence of the Wikipedia article title on a page, again resolving ties by extractor confidence. The large improvements in precision and recall (as highlighted in Figure 3) show that much of the returned text is irrelevant, but can be re-weighted using simple heuristics. Finally, we were interested if a weighted combination of these factors would lead to synergies. We set $\alpha_s = .1$, $\alpha_r = .7$, $\alpha_c = .9$, so that each factor was roughly weighted by our observed improvement (results were not sensitive to minor variations). On all datasets, performance was comparable or better than the best factor taken in isolation.

In our second series of experiments (Figure 4), we combined extractions from Wikipedia and the Web. In both cases, we applied the Shrink-Retrain extractor, but scored extractions from the Web using the weighted factor combination with $\lambda = .4$. The results, shown in Figure 4, show large improvements in recall at higher precision for the popular "Writer" (42%) dataset, and at moderately reduced precision for the sparse "Irish Newspaper" dataset. The area under the curve was substantially expanded in both cases, by 58% and 15% respectively. Compared to the original baseline system, the area has expanded 93% and 771% respectively. On the "Baseball Stadium" and "Performer" classes, the area has expanded 91% and 102% respectively.

In future work, we would like to automatically optimize the parameters α_s, α_r, α_c, λ based on comparing the extractions with values in the infobox.

Related Work

In the preceding sections we have discussed how our work relates to co-training. In this section, we discuss the broader context of previous work on unsupervised information extraction and other Wikipedia-based systems.

Unsupervised Information Extraction: Since the Web is large and highly heterogeneous, unsupervised and self-super-vised learning is necessary for scaling. Several systems of this form have been proposed. SNOWBALL (Agichtein & Gravano 2000) iteratively generates extraction patterns based on occurrences of known tuples in documents to extract new tuples from plain texts. MULDER (Kwok, Etzioni, & Weld 2001) and AskMSR (Brill, Dumais, & Banko 2002) 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. Instead of utilizing redundancy, K2 exploits Wikipedia's unique structure and the presence of user-tagged data to train machine learners. Patwardhan et al. proposed a decoupled information extraction system by first creating a self-trained relevant sentence classifier to identify relevant regions, and using a semantic affinity measure to automatically learn domain-relevant extraction patterns (Patwardhan & Riloff 2007). K2 uses the similar idea of decoupling when applying extractors to the general Web. However, K2 uses IR-based techniques to select relevant sentences and trains CRF extractors.

Other Wikipedia-Based Systems: Bunescu and Pasca utilized Wikipedia to detect and disambiguate named entities in open domain documents (Bunescu & Pasca 2006). Ponzetto et al. derived a large scale taxonomy based on the Wikipedia category system by identifying the IS-A relationships among category tags (Ponzetto & Strube 2007). Auer and Lehmann developed the DBpedia (Auer & Lehmann 2007) system which extracts information from existing infoboxes within articles and encapsulate them in a semantic form for query. In contrast, K2 populates infoboxes with *new* attribute values. Suchanek et al. implement the YAGO system (Suchanek, Kasneci, & Weikum 2007) which extends WordNet using facts extracted from Wikipedia's category tags. But in contrast to K2, which can learn to extract values for *any* attribute, YAGO only extracts values for a limited number of predefined relations.

Conclusion

Wu and Weld's Kylin system demonstrated the ability to perform self-supervised information extraction from Wikipedia (Wu & Weld 2007). While Kylin achieved high precision and reasonable recall on popular infobox classes, most

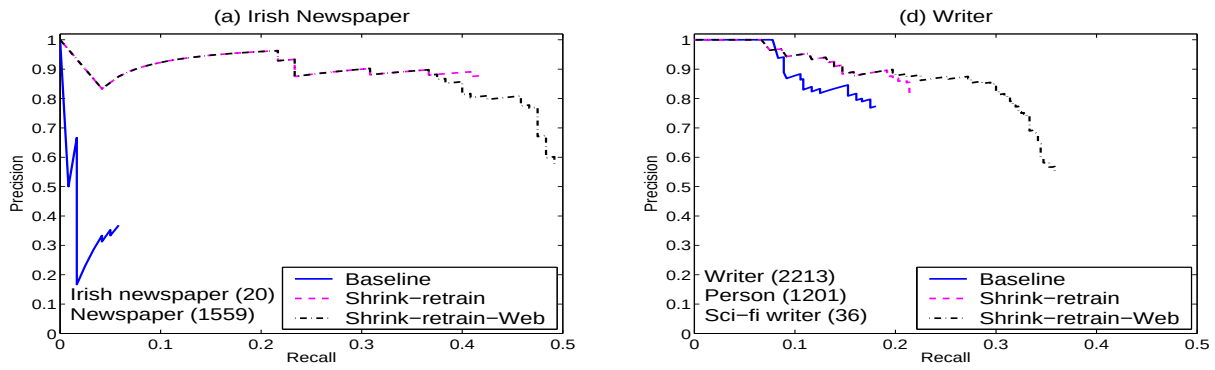

Figure 4: Combining Kylin's extractions from Wikipedia and the Web yields a substantial improvement in recall without compromising precision. Already, shrink-retrain improved recall over the original Kylin system, here the baseline, but the combination of extractions from Wikipedia and the Web, shrink-retrain-Web, performs even better. Note that recall is substantially improved, even for the Writer class, which has many infoboxes (2213) for training.

classes (i.e., 82%) provide fewer than 100 training examples; on these classes, Kylin's performance is unacceptable.

This paper describes the K2 system, which extends Kylin by supplementing Wikipedia extractions with those from the Web. There are two keys to effective (self-supervised) Web extraction: 1) careful filtering to ensure that only the best sentences are considered for extraction and 2) a novel retraining technique which generates more robust extractors. While these techniques are useful individually, their combination is synergistic (Figure 4):

- Precision is modestly improved in most classes, with larger gains if sparsity is extreme ("Irish Newspaper").
- Recall sees extraordinary improvement with gains from 0.06% to 0.49% (a factor of 8.4) in extremely sparse classes such as "Irish Newspaper." Even though the "Writer" class is populated with over 2000 infoboxes, its recall improves from 18% to 32% (a factor of 1.77) at equivalent levels of precision.
- Calculating the area under the precision / recall curve also demonstrates substantial improvement, with an improvement factor of 16.71, 2.02, 1.91, and 1.93 for "Irish Newspaper," "Performer," "Baseball Stadium," and "Writer," respectively.

Much remains to be done. For example, we wish to extend our retraining technique to full cotraining. There are ways to better integrate extraction of Web content with that of Wikipedia, ranging from improved querying policies to DIRT-style analysis of extraction patterns (Lin & Pantel 2001).

References

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

Auer, S., and Lehmann, J. 2007. What have Innsbruck and Leipzig in common? Extracting semantics from wiki content. In *ESWC07*.

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the Web. In *Proceedings of IJCAI07*.

Blum, A., and Mitchell, T. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, 92–100.

Brill, E.; Dumais, S.; and Banko, M. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of EMNLP02*.

Bunescu, R., and Pasca, M. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL06*.

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

Giles, J. 2005. Internet encyclopaedias go head to head. *Nature* 438:900–901.

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

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of WWW01*.

Lin, D., and Pantel, P. 2001. DIRT @SBT@discovery of inference rules from text. In *Knowledge Discovery and Data Mining*, 323–328.

McCallum, A. K.; Rosenfeld, R.; Mitchell, T. M.; and Ng, A. Y. 1998. Improving text classification by shrinkage in a hierarchy of classes. In Shavlik, J. W., ed., *Proceedings of ICML-98, 15th International Conference on Machine Learning*, 359–367. Madison, US: Morgan Kaufmann Publishers, San Francisco, US.

Nigam, K.; Lafferty, J.; and McCallum, A. 1999. Using maximum entropy for text classification. In *Proceedings of Workshop on Machine Learning for Information Filtering, IJCAI99*.

Patwardhan, S., and Riloff, E. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-07*.

Ponzetto, S. P., and Strube, M. 2007. Deriving a large scale taxonomy from wikipedia. In *Proceedings of AAAI07*.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In *Proceedings of WWW07*.

Wu, F., and Weld, D. 2007. Autonomouslly semantifying wikipedia. In *Proceedings of CIKM07*.

Wu, F., and Weld, D. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of WWW08*.