TWENTY-FIRST INTERNATIONAL
JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE
PASADENA, CALIFORNIA USA

Workshop W35

User-Contributed Knowledge and Artificial Intelligence: An Evolving Synergy

Organizing Committee
Razvan Bunescu (Ohio University, USA)
Evgeniy Gabrilovich (Yahoo! Research, USA)
Rada Mihalcea (University of North Texas, USA)
Vivi Nastase (EML Research gGmbH, Germany)

Sponsored by the International Joint Conferences on Artificial Intelligence (IJCAI) and the Association for the Advancement of Artificial Intelligence

Organizing Committee

Razvan Bunescu (Ohio University, USA)
Evgeniy Gabrilovich (Yahoo! Research, USA)
Rada Mihalcea (University of North Texas, USA)
Vivi Nastase (EML Research gGmbH, Germany)

Invited Speakers

Eugene Agichtein (Emory University, USA)
Timothy Chklovski (Factual Inc., USA)

Program Committee

Michele Banko (University of Washington, USA)
Misha Bilenko (Microsoft Research, USA)
Yunbo Cao (Microsoft Research, USA)
Timothy Chklovski (Factual Inc., USA)
Andras Csomai (Google, USA)
Silviu Cucerzan (Microsoft Research, USA)
James Fan (IBM, USA)
Iryna Gurevych (TU Darmstadt, Germany)
Eduard Hovy (USC / Information Sciences Institute, USA)
Rohit Kate (University of Texas at Austin, USA)
Ravi Kumar (Yahoo! Research, USA)
Oren Kurland (Technion, Israel)
Lillian Lee (Cornell University, USA)
Daniel Marcu (USC / Information Sciences Institute, USA)
Shaul Markovitch (Technion, Israel)
Hwee Tou Ng (National University of Singapore, Singapore)
Vincent Ng (University of Texas at Dallas, USA)
Bo Pang (Yahoo! Research, USA)
Patrick Pantel (Yahoo! Labs, USA)
Marius Pasca (Google, USA)
Simone Paolo Ponzetto (University of Heidelberg, Germany)
John Prager (IBM, USA)
Michael Strube (EML Research gGmbH, Germany)
Mihai Surdeanu (Stanford University, USA)
Peter Turney (National Research Council, Canada)
Dan Weld (University of Washington, USA)
Michael Witbrock (CYC, USA)
Qiang Yang (HKUST, Hong Kong)
Hugo Zaragoza (Yahoo! Research, Spain)

Website

http://lit.csci.unt.edu/~wikiai09

Preface

The performance of an Artificial Intelligence system often depends on the amount of world knowledge available to it. During the last decade, the AI community has witnessed the emergence of a number of highly structured knowledge repositories whose collaborative nature has led to a dramatic increase in the amount of world knowledge that can now be exploited in AI applications. Arguably, the best-known repository of user-contributed knowledge is Wikipedia. Since its inception less than eight years ago, it has become one of the largest and fastest growing on-line sources of encyclopedic knowledge. One of the reasons why Wikipedia is appealing to contributors and users alike is the richness of its embedded structural information: articles are hyperlinked to each other and connected to categories from an ever expanding taxonomy; pervasive language phenomena such as synonymy and polysemy are addressed through redirection and disambiguation pages; entities of the same type are described in a consistent format using infoboxes; related articles are grouped together in series templates.

Many more repositories of user-contributed knowledge exist besides Wikipedia. Collaborative tagging in Delicious and community-driven question answering in Yahoo! Answers and Wiki Answers are only a few examples of knowledge sources that, like Wikipedia, can become a valuable asset for AI researchers. Furthermore, AI methods have the potential to improve these resources, as demonstrated recently by research on personalized tag recommendations, or on matching user questions with previously answered questions.

The goal of this workshop was to foster the research and dissemination of ideas on the mutually beneficial interaction between AI and repositories of user-contributed knowledge. The workshop took place on July 13, 2009, in Pasadena CA, immediately preceding the International Joint Conference on Artificial Intelligence – IJCAI 2009.

This volume contains papers accepted for presentation at the workshop. We issued calls for regular papers, short late–breaking papers, and demos. After careful review by the program committee of the 20 submissions received – 13 regular papers, 6 short papers and 1 demo – 5 regular papers and 3 short papers were accepted for presentation. Consistent with the original aim of the workshop, the accepted papers address a diverse set of problems and resources, although Wikipedia-based systems are still dominant. The accepted papers explore leveraging knowledge induced and patterns learned from Wikipedia and apply them to the web or untagged text collections, using such knowledge for tasks such as information extraction, entity disambiguation, terminology extraction and analysing the structure of social networks. We also learn of useful methods that integrate Wikipedia with structured resources, in particular relational databases.

The members of the program committee provided high quality reviews in a timely fashion, and all submissions have benefited from this expert feedback.

For a successful event, having high quality invited speakers is crucial. We were lucky to have two excellent speakers for this year's event. We thank Eugene Agichtein and Timothy Chklovski for their enthusiastic acceptance and presentations.

Razvan Bunescu, Evgeniy Gabrilovich, Rada Mihalcea, Vivi Nastase
July 2009

Program

8:45 - 9:00	Opening remarks
9:00 - 10:00	**Invited talk:** Towards Clean, Structured Data from the Web and Volunteers *Timothy Chklovski*
10:00 - 10:30	Coffee break
10:30 - 11:00	LRTwiki: Enriching the Likelihood Ratio Test with Encyclopedic Information for the Extraction of Relevant Terms *Niklas Jakob, Mark-Christoph Müller, Iryna Gurevych*
11:00 - 11:30	Filtering Information Extraction via User-Contributed Knowledge *Thomas Lin, Oren Etzioni, James Fogarty*
11:30 - 12:00	Using an SVM Classifier to Improve the Extraction of Bilingual Terminology from Wikipedia *Maike Erdmann, Kotaro Nakayama, Takahiro Hara, Shojiro Nishio*
12:00 - 13:30	Lunch
13:30 - 14:30	**Invited talk:** Modeling Information Seeking Behavior in Social Media *Eugene Agichtein*
14:30 - 15:00	Scaling Wikipedia-based Named Entity Disambiguation to Arbitrary Web Text *Anthony Fader, Stephen Soderland, Oren Etzioni*
15:00 - 15:30	Coffee break
15:30 - 16:00	Learning to Integrate Relational Databases with Wikipedia *Doug Downey, Arun Ahuja, Mike Anderson*
16:00 - 16:20	Leveraging Social Networking Sites to Acquire Rich Task Structure *Yolanda Gil, Paul Groth, Varun Ratnakar*
16:20 - 16:40	Compact Hierarchical Explicit Semantic Representation *Sonya Liberman, Shaul Markovitch*
16:40 - 17:00	WikiSLE: Mapping Wikipedia Infobox information onto the article text *Sedat Gokalp, Syed Toufeeq Ahmed, Suvitha Vijayarajan, Hasan Davulcu*
17:00 - 17:30	Free discussion and closing remarks

Contents

Invited Talk

Modeling Information Seeking Behavior in Social Media

Eugene Agichtein
Emory University
http://www.mathcs.emory.edu/~eugene

Abstract

Social media is transforming how we seek and find information online: it is now a prominent part of the web information ecosystem, a powerful platform for information seeking, and a planet-scale experimental testbed for studying information seeking behavior and interactions. In particular, the Collaborative Question Answering (CQA) model has emerged as a potential alternative to automatic web search: participants post questions and answers, and rate and evaluate each other's contributions. The resulting archives of both the content and the context of the interactions complement the more limited web search interaction data, and provide a testbed for development of novel natural language processing, text mining, and information retrieval techniques. I will describe our progress on mining various traces of information seeking behavior in social media for tasks such as estimating content quality, search intent, and searcher satisfaction with the obtained results.

Bio

Dr. Eugene Agichtein is an Assistant Professor in the Mathematics and Computer Science Department at Emory University, where he founded the Intelligent Information Access Laboratory. He is also affiliated with the Computational & Life Sciences Initiative and the Linguistics program at Emory, and with the Web Science Initiative at the Georgia Institute of Technology. After obtaining a Ph.D. from Columbia University in 2005, Eugene was a Postdoctoral Researcher at Microsoft, and recently a visiting researcher at Yahoo!. Eugene’s research interests are in information retrieval, user modeling, text and data mining, and their applications to social computing and medical informatics.

Invited Talk

Towards Clean, Structured Data from the Web and Volunteers

Timothy Chklovski
Factual, Inc.

Abstract

Having access to large amounts of clean, structured data is an important step towards disruptive, useful AI applications. The large amount of data on the Web and the advent of crowdsourcing have created unprecedented opportunities to aggregate large amounts of knowledge in simple structured format. This talk will review some highlights in collecting knowledge from volunteers and extraction from the Web. Collecting from volunteers will cover exciting developments in the field, as well as touch on the author's prior work on using acquired knowledge to ask better and better knowledge acquisition questions, work on collecting linguistic information (much of it joint with Rada Mihalcea), and work on collecting task knowledge. On the extraction front, we will touch on some work on extracting verb relation semantics, and quantification of adjectives. The talk will include a demo of the recent work at Factual, Inc. that seeks to empower broader audiences to both extract factual information from the Web and improve quality of the data through direct contributions. We will conclude with brief remarks on what the near future might bring.

Bio

Timothy Chklovski works in artificial intelligence, focusing on creating large collections of knowledge and algorithms for reasoning over them. His interests include broad-scale knowledge acquisition from volunteers, text mining, and ways to apply various knowledge sources in natural language understanding. Timothy completed his PhD at MIT in 2003, where he focused on using similarity-based reasoning to collect commonsense knowledge from volunteers, and later worked as a Senior Research Scientist at USC ISI. Since 2007, he has been working at Factual as a Founding Engineer, where he has been leading a small team developing practical applications for collecting and organizing open datasets.

LRT$_{wiki}$: Enriching the Likelihood Ratio Test with Encyclopedic Information for the Extraction of Relevant Terms

Niklas Jakob and **Mark-Christoph Müller** and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab
Technische Universität Darmstadt
Hochschulstr. 10, 64289 Darmstadt, Germany
{njakob, chmark, gurevych}@tk.informatik.tu-darmstadt.de

Abstract

This paper introduces LRT$_{wiki}$, an improved variant of the Likelihood Ratio Test (LRT). The central idea of LRT$_{wiki}$ is to employ a comprehensive domain specific knowledge source as additional "on-topic" data sets, and to modify the calculation of the LRT algorithm to take advantage of this new information. The knowledge source is created on the basis of Wikipedia articles. We evaluate on the two related tasks product feature extraction and keyphrase extraction, and find LRT$_{wiki}$ to yield a significant improvement over the original LRT in both tasks.

1 Introduction

The identification of the most relevant terms[1] in a document collection is one of the pervasive tasks in natural language processing. A fundamental application of this task is keyphrase extraction [Turney, 2000], which aims at determining the most important terms in a document. The resulting keyphrases are assumed to reflect the document topic, and they are typically used for summarization, clustering, or search. Another application, which is gaining importance due to the popularity of Web 2.0, is product feature extraction [Yi *et al.*, 2003], which is often performed on customer reviews. Here, identified product features are used to create feature-oriented summaries of customer review collections, or as the basis for extracting opinions about features. Keyphrase extraction and product feature extraction mainly differ in their definition of "relevance". In keyphrase extraction, the goal is to identify those terms in a given document which best describe its topic by distinguishing it from documents with different topics. Individual *mentions* of the same term are not considered. In product feature extraction, on the other hand, the goal is to extract *all mentions* of features of a given product. At the same time, it is important to only extract features of the product *under review*, and not of any other products mentioned e.g. in comparisons. Approaches which rely on statistical information have been successfully employed for both applications in previous research [Tomokiyo and Hurst, 2003; Yi *et al.*, 2003]. Both of the above approaches utilize the Likelihood Ratio Test [Dunning, 1993] (LRT), which is well suited for the identification of relevant terms from document collections, since it does not assume a normal distribution of the variables (= frequencies of the terms) in the data. LRT compares the frequencies of candidate terms in the document collection to be analyzed with their frequencies in a general-language corpus and calculates the likelihood that a term is relevant for the given document collection.

For product feature extraction in customer reviews, other statistical methods have also been used: Hu and Liu [2004] try to summarize customer reviews and present a system which uses association mining to extract product features in opinionated sentences. They only present an evaluation of the combined product feature extraction and opinion mining steps, and report an average F-measure of 0.79 with the best configuration. One advantage of their system is that it does not rely on any pre-built knowledge base, but only uses statistical information. Popescu and Etzioni [2005] employ Point-wise Mutual Information to compute the probability that a candidate term is a feature of a given product, and they report an average F-measure of 0.758. Since the calculation of Point-wise Mutual Information requires a very large corpus, they use the web and a web search engine. Wong et al. [2008] model product feature extraction as a Dirichlet process prior which they then use in an Expectation Maximization algorithm. They reach an average F-measure between 0.58 and 0.95 on their four datasets. The *Sentiment Analyzer* system by Yi et al. [2003] includes a product feature extraction component utilizing base noun phrase patterns and LRT. For their product feature extraction step (on two datasets), Yi et al. [2003] report precision values of 0.97 and 1.0, but no recall. In [Ferreira *et al.*, 2008] we present a comparative evaluation of the approaches by Hu and Liu [2004] and Yi et al. [2003], this time evaluating product feature extraction independently of opinion detection. They find two limitations of LRT: 1) it often fails to identify **rare** product features, and 2) it also often fails to identify terms that are **both product features and general vocabulary items** (e.g. *weight*, *speed*, *option*).

This paper introduces LRT$_{wiki}$, our extension of the Likelihood Ratio Test algorithm, which addresses the above limitations by enriching LRT with encyclopedic information drawn from Wikipedia. In LRT$_{wiki}$, Wikipedia is employed as a

[1] We use *term* here to cover both single terms and multi-term expressions.

general-purpose source of domain knowledge. We analyze the performance of LRT$_{wiki}$ in two different tasks: In the first scenario we employ the algorithm for product feature extraction as in [Ferreira *et al.*, 2008], in the second scenario we employ it for keyphrase extraction as in [Tomokiyo and Hurst, 2003]. The remainder of this paper is structured as follows: Section 2 gives an overview of the data we use in our experiments. Section 3 describes LRT$_{wiki}$, our proposed extension of LRT, and Section 4 contains the experimental results and discussion. Conclusions and future work can be found in Section 5.

2 Employed Corpora

2.1 Data for Product Feature Extraction

We use the dataset as in [Ferreira *et al.*, 2008], for which a corpus originally annotated by [Hu and Liu, 2004] was reannotated. In contrast to the original annotations, in [Ferreira *et al.*, 2008] we annotated *all* mentions of product features, irrespective of there being an opinion expressed about them. Table 1 outlines some statistics on the dataset. We did not report any inter-annotator agreement statistics before, but since we are interested in the agreement (e.g. as an upper bound for the evaluation of the product feature extraction task), we reannotated a subset of the corpus in a controlled manner. First, we randomly selected 60 sentences from each of the five product review sets, and then we had two human subjects annotate them following the guidelines presented in [Ferreira *et al.*, 2008]. Due to the skewed class distribution (the vast majority of terms in product reviews are *not* product features), we simply calculated Precision, Recall, and F-measure (instead of e.g. Kappa) on the overlap between the two annotators. The overlap was calculated rather strictly by considering only exact matches in the product feature annotation. The results of the annotation overlap measurements are shown in Table 2.

Table 1: Product review datasets

Dataset	Documents	Sentences	Feature Mentions
Digital camera 1 (DC1)	45	597	594
Digital camera 2 (DC2)	34	346	340
Cell phone (CP)	41	546	471
MP3 player (MP3)	95	1716	1031
DVD player (DVD)	99	739	519

Table 2: Annotation overlap for product feature mentions

Dataset	Sentences	Words	Features	F-measure
DC1	60	980	67	0.736
DC2	60	1029	69	0.747
CP	60	1001	63	0.825
MP3	60	830	46	0.745
DVD	60	883	52	0.477

An analysis of the annotation overlap shows that product feature extraction is not a trivial task. F-measure on the DVD player dataset is particularly low. We observe that this is due to excessive usage of abbreviations regarding the product in this document collection (e.g. referring to the product with just its model number) and some disagreement regarding their annotation.

2.2 Data for Keyphrase Extraction

The data we employ in our keyphrase extraction experiments is originally from the DUC2001 dataset [Over, 2001]. The corpus consists of 309 news articles with keyphrases annotated by Wan and Xiao [2008]. The articles cover 30 different news topics and have an average length of 740 words. The annotation involved two annotators, who were allowed to select a maximum of 10 distinct keyphrases per document. Wan and Xiao report an inter-annotator agreement of 0.70 κ. After the annotation, the annotators created the final gold standard by resolving conflicting annotations in a discussion. The average number of keyphrases per document is 8.08, and the average number of words per keyphrase is 2.09 [Wan and Xiao, 2008]. Since LRT requires a collection of "on-topic" documents for extracting the most relevant terms, we selected the two largest subsets of the DUC2001 datasets. Each of the two subsets (DUC IDs: d06a & d34f) contains 16 documents.

3 LRT and LRT$_{wiki}$

3.1 LRT

LRT was introduced by Dunning [1993] and has been employed for many different NLP-related tasks, since the algorithm does not assume that the population it operates on is distributed normally or approximately normally, which is true for the frequencies of terms in a text. In short, LRT identifies relevant terms from a document collection by comparing the frequencies of the candidate terms in the "on-topic" documents with their frequencies in a general language "off-topic" document collection. It uses a contingency table for the current candidate term T, for which the frequency related values C_{11} to C_{22} are extracted from the "on-topic" document collection D_+ and the "off-topic" document collection D_-. Table 3 outlines the different elements of the contingency table, the LRT definition is shown in Equation (1).

Table 3: Contingency table for candidate term T

	D_+	D_-
T	C_{11}	C_{12}
$\overline{T}$	C_{21}	C_{22}

$$r_1 = \frac{C_{11}}{C_{11}+C_{12}}, r_2 = \frac{C_{21}}{C_{21}+C_{22}}, r = \frac{C_{11}+C_{21}}{C_{11}+C_{12}+C_{21}+C_{22}}$$

$$\begin{aligned} lr =& (C_{11}+C_{21})\log(r) + (C_{12}+C_{22})\log(1-r) - C_{11}\log(r_1) \\ & - C_{12}\log(1-r_1) - C_{21}\log(r_2) - C_{22}\log(1-r_2) \end{aligned}$$

$$-2\log\lambda = \begin{cases} -2 * lr & \text{if } r_2 < r_1 \\ 0 & \text{if } r_2 \geq r_1 \end{cases} \quad (1)$$

In their application of LRT to product feature extraction, Yi et al. [2003] and Ferreira et al. [2008] report high precision but low recall. As already described in Section 1, in [Ferreira *et al.*, 2008] we observe that LRT typically misses product features that have a low frequency in the "on-topic" document collection - e.g. because only very few customers comment on them - even if they do not occur in the "off-topic" document collection at all. In addition, LRT also misses terms which are both product features and general vocabulary items, such as *speed, option, flexibility*. Section 3.2 describes LRT$_{wiki}$, which is our proposed enhancement of LRT specifically targetting these two shortcomings.

3.2 LRT$_{wiki}$

LRT$_{wiki}$ aims at improving the ranking of candidate terms of two problematic candidate classes:

1. Candidate terms which occur in the "on-topic" document collection with low frequency and not at all or with a low frequency in the "off-topic" document collection
2. Candidate terms which occur both in the "on-" and "off-topic" document collection with medium or high frequency

The central idea of LRT$_{wiki}$ is to employ a comprehensive domain specific knowledge source containing the terminology typically used in the current domain as the source of additional "on-topic" data sets, and to modify the calculation of the LRT algorithm to take advantage of this new knowledge source. The knowledge source is created on the basis of Wikipedia.

Wikipedia Data

We chose the free online encyclopedia Wikipedia for two reasons: 1) Due to its broad coverage, we can expect it to contain articles about many topics. 2) Due to the encyclopedic style of Wikipedia articles, they tend to focus on a single topic, and normally do not contain irrelevant information.

Since our goal is to extract an additional corpus about the pre-defined topic(s) dealt with in the document collection to be analysed, we assume the topic to be known in advance. We then query Wikipedia in order to retrieve one article for each topic as the seed for retrieving the new "on-topic" data sets. For the product classes from the Hu and Liu dataset, we used the names provided in their paper (*digital camera*, *dvd player*, *mp3 player*, *cell phone*). For each of these topics there is either a Wikipedia article with the same title or an automatic redirect page (*mp3 player* → *Digital audio player*). For the DUC data, we read the documents and inferred the topics "police brutality" (d06a), for which Wikipedia contains an article, and "atlantic hurricanes" (d34f), which is redirected to "North Atlantic tropical cyclone". The Wikipedia-based document collection for each topic is built by extracting the categories to which the seed article belongs and then extracting all articles found in these categories. We performed a simple ad-hoc filtering only: We ignored all subcategories of "Wikipedia administration", since they do not carry any semantic content, and all categories with more than 200 articles, since we regard them as too broad. The article pages retrieved in this manner were then automatically cleaned of all Wikipedia markup, metainformation, references, and hyperlinks. The data we retrieved is extracted from a Wikipedia dump from Feb. 2007. Some statistics about the resulting data is given in Table 4.

Table 4: Content retrieved from Wikipedia

Wikipedia Seed Article	Retrieved Articles	Word Count
digital camera	263	161459
cell phone	250	204410
digital audio player	64	79099
dvd player	100	99898
north atlantic tropical cyclone	403	459046
police brutality	166	127216

Modifying the LRT Algorithm

The new Wikipedia content provides an additional document collection D_W on the basis of which we can calculate C_{13} for a given term T, which are defined in analogy to Table 3. With these new values we modify the calculation of the original LRT lr as follows:

$$\begin{aligned} lr_{mod} =& (C_{11mod} + C_{21}) \log(r) + (C_{12mod} + C_{22}) \log(1-r) \\ & - C_{11mod} \log(r_1) - C_{12mod} \log(1-r_1) - C_{21} \log(r_2) \\ & - C_{22} \log(1-r_2) \end{aligned}$$

$$C_{11mod} = \begin{cases} C_{11} + C_{13} & \text{if } C_{11} < t_1 \text{ and } C_{12} < t_1 \\ C_{11} + C_{13} & \text{if } C_{11} > t_2 \text{ and } C_{12} > t_2 \end{cases}$$

$$C_{12mod} = \begin{cases} 0 & \text{if } C_{11} < t_1 \text{ and } C_{12} < t_1 \\ \max(0, C_{12} - C_{13}) & \text{if } C_{11} > t_2 \text{ and } C_{12} > t_2 \end{cases} \quad (2)$$

The two thresholds t_1 and t_2 are used to set the boundaries for terms with low frequency (t_1) and terms with medium or high frequency (t_2).

4 Experiments

We model term extraction as a two-step process. In the first step, candidate terms are extracted, and in the second step, these candidates are ranked on the basis of their LRT / LRT$_{wiki}$ values. An overview of our architecture is shown in Figure 1. As in [Ferreira *et al.*, 2008], we employed the 600 randomly selected documents from the UKWaC British English web corpus [Ferraresi *et al.*, 2008] as an "off-topic" corpus (D_-).

4.1 Product Feature Extraction

For the product feature extraction evaluation, we follow the approach of Yi et al. [2003]. They specify their candidate patterns as the following Base Noun Phrases (BNP): NN, NN NN, JJ NN, NN NN NN, JJ NN NN, JJ JJ NN. These patterns are applied inclusively, i.e. multi-term expressions (e.g. *digital camera*) are allowed to be matched more than once. This way, occurrences of terms as parts of multi-term expressions (e.g. *camera*) are counted several times, thus boosting the extraction of the embedded term. After extracting the candidate terms, both LRT versions calculate the likelihood score for

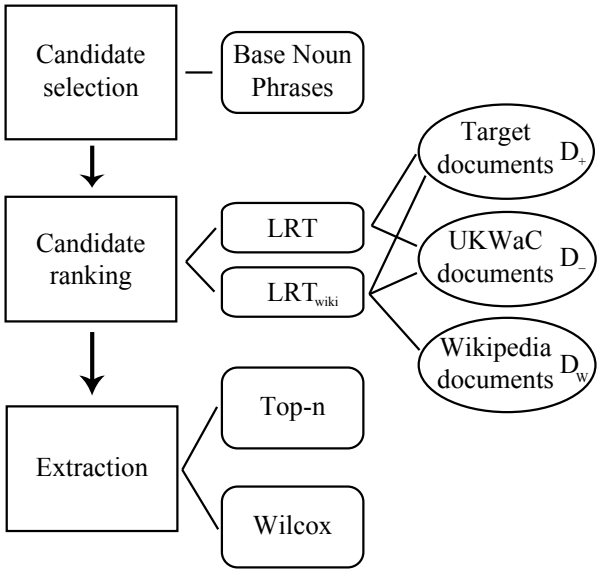

Figure 1: Term extraction architecture

each. The resulting ranked list has then to be transformed into a set of relevant and irrelevant features. Yi et al. [2003] address this issue by selecting the top n features as relevant, where n is the number of candidate features selected by the most restrictive BNP pattern set of "beginning definite Base Noun Phrases" [Yi *et al.*, 2003]. While this approach can lead to a high precision extraction, it unfortunately suffers from an extremely low recall. Therefore, we propose a different method for selecting the threshold for distinguishing relevant from irrelevant terms, which is based on the algorithm for outlier detection presented in [Wilcox, 2001, page 38]. According to this method, the threshold t_{LRT} for feature extraction is set to:

$$t_{LRT} = m_{lr} + sd_{lr} \quad (3)$$

where m_{lr} is the mean likelihood value and sd_{lr} is the standard deviation.

Table 5 shows the results obtained in [Ferreira *et al.*, 2008] and our experiments. The column "LRT Wilcox Threshold" shows the effect of the dynamic threshold calculation while using our reimplementation of the original LRT algorithm. The column "LRT$_{wiki}$ Wilcox Threshold" shows the results obtained by employing the dynamic threshold calculation and LRT$_{wiki}$. Following [Ferreira *et al.*, 2008], we perform an evaluation on each mention of a product feature, comparing the lowercased and lemmatized forms of the automatically extracted features with those in the gold standard. For LRT$_{wiki}$, we set t_1 to 5, which was empirically defined in order to reflect a threshold under which we consider a term to be rare. Likewise, the threshold t_2 was set to 10, meaning we consider terms which are found more often to be frequently occurring. These thresholds are optimal to the corpora we experimented with while smaller or larger values might make sense for different input corpora D_+.

4.2 Keyphrase Extraction

As we are interested in an state-of-the-art approach for keyphrase extraction which is also unsupervised, we employ the TextRank system [Mihalcea and Tarau, 2004]. We follow Mihalcea & Tarau by selecting only adjectives and nouns as candidate terms. The matching is done in a greedy fashion on the terms' POS tags with the following regular expression: $(JJ|JJR|JJS)^*(NN|NNS|NP|NPS)^+$. Greedy matching makes sure that only the longest matching phrases in a sentence are selected as candidates. This matching strategy is based on the observation that *complete* noun phrases are typically annotated as keyphrases in the DUC data set. For example "accidental shooting death" is annotated as a keyphrase and not just "shooting death" or "death". Contrary to product features, there is no clear-cut definition of what is and what is not regarded as a keyphrase for a document. Therefore, during our evaluation, we did not employ a threshold like in 4.1. Alternatively, we evaluate Precision, Recall and F-measure of the *top-n* extracted keyphrases. As a baseline system we employ TextRank in its default configuration. The keyphrases extracted by the TextRank system, the two versions of LRT, and the keyphrases in the gold standard are lemmatized and lowercased before comparison. When employing LRT$_{wiki}$, we use the same thresholds t_1 and t_2 as described in Section 4.1. We evaluate the top-n keyphrases (n $\leq$ 10) on the two datasets each containing 16 documents as described in Section 2.2. The results of the keyphrase extraction evaluation are shown in Table 6.

4.3 Error Analysis

As evident from Tables 5 and 6, LRT$_{wiki}$ consistently and significantly[2] improves F-measure in both applications. In the following, we perform an error analysis in the two applications separately.

Product Feature Extraction Error Analysis

When comparing the results of "LRT Wilcox" with "LRT in [Ferreira *et al.*, 2008]", one can already observe a constant improvement in precision and recall. This shows that the extraction strategy is also an important aspect of the LRT which might deserve further research. In the task of product feature extraction, the recall slightly decreases when comparing LRT and LRT$_{wiki}$ on two of the datasets (DC2, MP3). However, the concurrent gains in precision outweigh them, leading to an overall higher F-measure. The decrease of recall on some datasets can be explained as follows: A substantial amount of terms belonging to the specific vocabulary of the domain have C_{11} and C_{12} values smaller than t_1 and therefore receive a *boosting* from the new Wikipedia content. The boosting often pushes their lr_{mod} values into regions of other terms which have a $C_{11} > t_1$ and which would therefore typically be extracted as relevant. However, the boosting effect raises the overall average likelihood ratio, which we use to separate the relevant from the irrelevant terms. At the same time, there are typically quite a few terms which occur

[2]Significance of improvement in F-measure is tested using a paired one-tailed t-test and p $\leq$ 0.05 (*), p $\leq$ 0.01 (**), and p $\leq$ 0.005 (***)

Table 5: Product Feature Extraction

	LRT in [Ferreira *et al.*, 2008]			LRT Wilcox Threshold			LRT_{wiki} Wilcox Threshold			
Dataset	P	R	F	P	R	F	P	R	F	Δ F
DC1	0.671	0.495	0.570	0.750	0.513	0.609	0.760	0.574	0.654	**+0.045***
DC2	0.634	0.347	0.449	0.800	0.485	0.604	0.875	0.474	0.615	**+0.011***
CP	0.659	0.459	0.541	0.579	0.535	0.556	0.813	0.544	0.651	**+0.095***
MP3	0.339	0.408	0.370	0.513	0.665	0.579	0.560	0.661	0.606	**+0.027***
DVD	0.506	0.243	0.328	0.633	0.416	0.502	0.667	0.458	0.543	**+0.040***

Table 6: Keyphrase Extraction

		TextRank			LRT			LRT_{wiki}			
Dataset	**n**	P	R	F	P	R	F	P	R	F	Δ F
d06a	1	0.188	0.029	0.051	0.000	0.000	0.000	0.062	0.010	0.017	**+0.017*****
(16 docs)	2	0.125	0.039	0.060	0.094	0.030	0.045	0.375	0.119	0.180	**+0.135*****
	3	0.083	0.039	0.053	0.250	0.119	0.161	0.333	0.158	0.215	**+0.054*****
	4	0.094	0.059	0.072	0.281	0.178	0.218	0.328	0.208	0.255	**+0.037*****
	5	0.088	0.069	0.077	0.250	0.198	0.221	0.325	0.257	0.287	**+0.066*****
	6	0.073	0.069	0.071	0.250	0.238	0.244	0.312	0.297	0.305	**+0.061*****
	7	0.071	0.078	0.075	0.232	0.257	0.244	0.295	0.327	0.310	**+0.066*****
	8	0.070	0.088	0.078	0.234	0.297	0.262	0.273	0.347	0.306	**+0.044*****
	9	0.069	0.098	0.081	0.222	0.317	0.261	0.250	0.356	0.294	**+0.033*****
	10	0.075	0.118	0.092	0.219	0.347	0.268	0.238	0.376	0.291	**+0.023*****
d34f	1	0.467	0.053	0.096	0.062	0.008	0.014	0.062	0.008	0.014	**+0.000**
(16 docs)	2	0.500	0.115	0.186	0.062	0.015	0.024	0.062	0.015	0.024	**+0.000**
	3	0.500	0.168	0.251	0.042	0.015	0.022	0.062	0.023	0.033	**+0.011*****
	4	0.448	0.198	0.275	0.062	0.030	0.041	0.078	0.038	0.051	**+0.010*****
	5	0.431	0.237	0.305	0.100	0.061	0.075	0.150	0.091	0.113	**+0.038*****
	6	0.393	0.252	0.307	0.115	0.083	0.096	0.167	0.121	0.140	**+0.044*****
	7	0.396	0.290	0.335	0.125	0.106	0.115	0.170	0.144	0.156	**+0.041*****
	8	0.374	0.305	0.336	0.148	0.144	0.146	0.172	0.167	0.169	**+0.023*****
	9	0.350	0.313	0.331	0.139	0.152	0.145	0.167	0.182	0.174	**+0.029*****
	10	0.325	0.313	0.319	0.138	0.167	0.151	0.169	0.205	0.185	**+0.034*****

in almost every sentence (e.g. the product under review) and which therefore influence the standard deviation. In general, the boosting modification leads to a substantial increase in the average likelihood ratio, while hardly affecting the standard deviation. This leads to a slight increase in the threshold for the extraction of terms, with some relevant terms no longer reaching it. On the DC1 dataset, e.g. LRT_{wiki} extracts the correct product features "control", "film", and "sensor", while the original LRT misses them. At the same time, using LRT_{wiki}, the correct features "external flash" and "lcd screen" do not reach the threshold any more while the original LRT extracts them.

This effect on the average likelihood ratio (which is even more pronounced for the standard deviation) is caused by the modification which aims to improve the extraction of relevant terms also occurring in the general language corpus. Such candidates typically have a rather high C_{11} value, and due to the Wikipedia content their C_{12} value is reduced, leading to a very high likelihood ratio, which in turn leads to a higher standard deviation.

The inclusion of the Wikipedia documents also exacerbates one of the issues mentioned in [Ferreira *et al.*, 2008]: If several reviews mention products of another manufacturer or a different model (e.g. in comparisons), LRT (and also LRT_{wiki}) will extract them. Since the documents from Wikipedia are typically not limited to a single product, but rather to a product class, they tend to contain names of several different models and manufacturers. If such a name is mentioned in the "on-topic" documents, its likelihood ratio will be boosted due to our algorithm modification. Overall, however, LRT_{wiki} still leads to a significant improvement over LRT regarding F-measure.

Keyphrase Extraction Error Analysis

When comparing the results of the keyphrase extraction based on both versions of LRT with TextRank as a baseline, we observe that on the d06a dataset both LRT versions perform considerably better and on the d34f dataset considerably worse than the TextRank system. However, the performance of TextRank on the d34f dataset is much better than its average on the entire DUC2001 dataset: TextRank yields an overall F-measure of 0.132 at 10 extracted keyphrases on the entire DUC2001 dataset[3], while on the d34f dataset it reaches an F-Measure of 0.319 at 10 extracted keyphrases. This is due to the fact that, with three to five words, the keyphrases on the d34f subset are rather long compared to those in the other document sets (overall average keyphrase length in words:

[3]We obtained this result in a separate experiment.

2.0). When employed in a keyphrase extraction task, both versions of LRT have the limitation that the relevance of a term is calculated on the overall document collection. However, keyphrases are annotated with respect to their importance in individual documents. Therefore, LRT often fails to extract keyphrases which are only relevant for one document. This definition of relevance is different from the product feature extraction task, as terms are regarded as relevant over the entire document collection.

5 Conclusions and Future Work

In this paper we presented LRT_{wiki}, an enhancement of the Likelihood Ratio Test, which makes use of encyclopedic documents retrieved from Wikipedia. The enhanced algorithm leads to an improvement regarding the relevance ranking of terms. Since Wikipedia is available in many languages and its content is very broad, is seems to be a well suited resource for extending a statistical method for term mining. We also propose a method to calculate the threshold for the separation of relevant and irrelevant terms. In our empirical evaluation, LRT_{wiki} yielded a significant improvement regarding F-measure in two different tasks: keyphrase extraction and product feature extraction. A limitation which remains for product feature extraction regards terms which belong to the current domain, but which are not features of the product under review. Although somewhat less consistent, our results show that LRT_{wiki} might also be a promising candidate for keyphrase extraction. While our enhancements consistently improve the results regarding F-measure over the original LRT, both LRT and LRT_{wiki} are inferior to the non-LRT baseline on one of two datasets. In future work, we plan to further investigate this issue.

Other lines of future work include the following: We plan to improve the threshold calculation for the separation of relevant and irrelevant terms, since the current approach is optimized for the original LRT, and not yet for LRT_{wiki}. We will also address remaining issues regarding the product feature extraction task. For this, we plan to extract additional information regarding individual products (e.g. product and manufacturer names) from Wikipedia, and to use this information to filter out feature candidates not pertaining to the product under review. We plan to extract this additional information from *semi-structured* Wikipedia content (e.g. links, lists, and tables), thus going beyond treating Wikipedia articles as flat documents.

Acknowledgements

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The authors take the responsibility for the contents.

References

[Dunning, 1993] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[Ferraresi *et al.*, 2008] Adriano Ferraresi, Eros Zanchetta, Silvia Bernardini, and Marco Baroni. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google?*, pages 47–54, Marrakech, Morocco, June 2008.

[Ferreira *et al.*, 2008] Liliana Ferreira, Niklas Jakob, and Iryna Gurevych. A comparative study of feature extraction algorithms in customer reviews. In *Proceedings of the 2nd IEEE International Conference on Semantic Computing*, pages 144–151, Santa Clara, California, USA, August 2008.

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, Seattle, Washington, USA, August 2004.

[Mihalcea and Tarau, 2004] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing 2004*, pages 404–411, Barcelona, Spain, July 2004.

[Over, 2001] Paul Over. Introduction to duc-2001: an intrinsic evaluation of generic news text summarization systems. In *DUC 2001 Workshop on Text Summarization*, New Orleans, Louisiana, USA, September 2001.

[Popescu and Etzioni, 2005] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, Canada, October 2005.

[Tomokiyo and Hurst, 2003] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 33–40, Sapporo, Japan, July 2003.

[Turney, 2000] Peter D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303–336, 2000.

[Wan and Xiao, 2008] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 855–860, Chicago, Illinois, USA, July 2008.

[Wilcox, 2001] Rand R. Wilcox. *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy*. Springer, 2001.

[Wong *et al.*, 2008] Tak-Lam Wong, Wai Lam, and Tik-Shun Wong. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–42, Singapore, July 2008.

[Yi *et al.*, 2003] Jeonghee Yi, Tetsuya Nasukawa, Razvan C. Bunescu, and Wayne Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 427–434, Melbourne, Florida, USA, December 2003.

Filtering Information Extraction via User-Contributed Knowledge

Thomas Lin, Oren Etzioni, James Fogarty
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{tlin, etzioni, jfogarty}@cs.washington.edu

Abstract

Large repositories of knowledge can enable more powerful AI systems. Information Extraction (IE) is one approach to building knowledge repositories by extracting knowledge from text. Open IE systems like TextRunner [Banko et al., 2007] are able to extract hundreds of millions of assertions from Web text. However, because of imperfections in extraction technology and the noisy nature of Web text, IE systems return a mix of both useful, informative facts (e.g., "*the FDA banned ephedra*") and less informative statements (e.g., "*the FDA banned products*").

This paper investigates using user-contributed knowledge from Wikipedia and from TextRunner website visitors to train classifiers that automatically filter extracted assertions. In a study of human ratings of the interestingness of TextRunner assertions, we show that our approach substantially enhances the quality of results. Our relevance feedback filter raises the fraction of *interesting* results in the top thirty from 41.6% to 64.1%.

1 Introduction

Information extraction (IE) is a subfield of natural language processing that seeks to obtain structured information from unstructured text. IE can be used to automate the tedious and error prone process of collecting facts from the Web. Open IE is a relation-independent form of IE that scales well to large corpuses. Figure 1 presents the output of the TextRunner Open IE system [Banko et al., 2007] in response to the question "*What has the FDA banned?*". TextRunner homes in on such answers as "*ephedra*" and "*most silicone implants*" and frees people from sifting through many Web pages to find the desired answers.

Unfortunately, extraction engines, like search engines, intermix relevant information with irrelevant information. This problem is exacerbated in IE systems because they use heuristic methods to extract phrases that are meant to denote

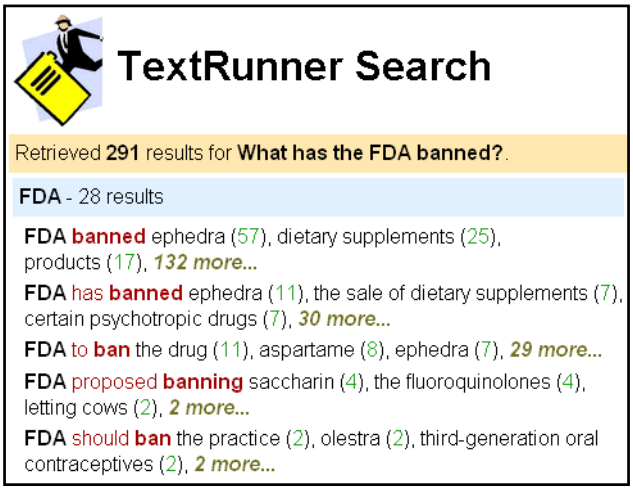

Figure 1. TextRunner results for the question "*What has the FDA banned?*". This paper examines the filtering of such results to focus on *interesting* assertions.

entities and relationships. Thus, in response to the above question, an extraction engine like TextRunner also returns such uninformative answers as "*products*" and "*the drug*". Experiments presented in this paper show that people find 58.4% of the thirty top-ranked answers returned by TextRunner to be uninformative.

Extraction engines therefore could be improved by filtering based on models of which extracted assertions are of interest and which are not. See Figure 2 for an overview of this idea. Of course, the notion of *interestingness* is subjective, personal, and context specific. Nevertheless, any system that returns ranked results, from Google to TextRunner, either implicitly or explicitly utilizes a model of *what is interesting* in its ranking function.

A challenge here is that while people can identify what is interesting to them, it is less clear how computers can do this algorithmically. We could implement several theories of interestingness from psychology such as *complexity*, *novelty*, *uncertainty*, and *conflict* [Silvia, 2006], but it is unclear which, if any, is best. The most accurate method would be if we could have people go through and specify which of the extracted assertions are of interest.

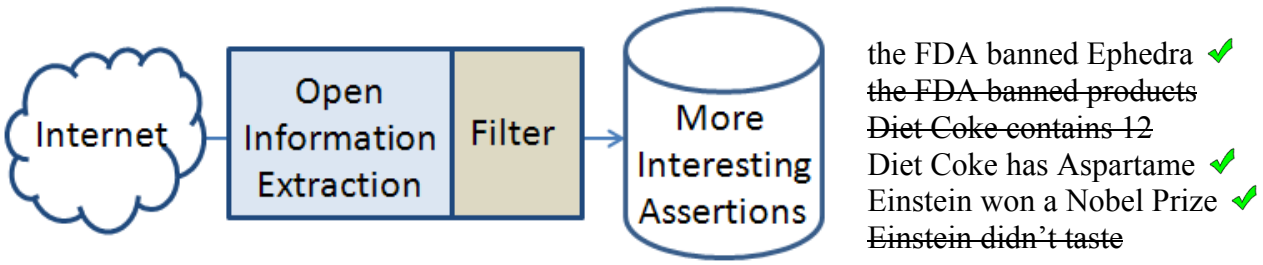

Figure 2. Filtering the output of Open IE enables it to better focus on extracted assertions that are more *interesting*.

However, as TextRunner has over 800 million extracted assertions, it would be prohibitively expensive to hand-label every assertion. Instead, we collect labels on a small but general subset of the data, and then generalize by using this as training data for a classifier filter that combines basic features and psychology theories. We also leverage large available repositories of user-contributed knowledge. Using Wikipedia Infobox matching to automatically classify assertions, we are able to inexpensively generate additional sizable sets of assertions that are likely to be interesting and not interesting to train a classifier. Figure 3 shows our process for using user-contributed knowledge.

This paper proposes several models of *what is interesting*, presents our implementation of these models as filters, and reports on measurements of their efficacy on a sample of queries. The main contributions of this paper are to:

- Introduce several practical models of interestingness that, when implemented as filters, offer substantial improvements over the existing technique of sorting assertions by frequency. These models are informed by previous work, theories, and user-contributed knowledge. Our models could be easily adapted to aid other Web-based extraction systems, such as PowerSet.
- Utilize a machine-learning method that leverages all the models, together with relevance feedback, to filter out uninteresting assertions resulting from extraction.
- Report on the first study of interestingness in extraction. For this study we compare the efficacy of our models to each other and the TextRunner baseline. Among other findings, we show that our filtering significantly improves the fraction of interesting results contained within TextRunner's top thirty results from 41.6% interesting to 64.1% interesting.

The remainder of this paper is organized as follows. Section 2 introduces TextRunner and related work. Section 3 describes our models of interestingness and how we operationalize them as filters. Section 4 presents a study evaluating those models. We then conclude with a discussion.

2 Background

The TextRunner system crawls the Web and extracts information as triples that take the form **(entity, relation, entity).** The *relation* string is meant to denote the relationship between the two *entities*. For example, if the sentence *"Franz Kafka was born in Prague, at the time part of Austria"* were found on a webpage, then one extraction would be (*"Franz Kafka", "was born in", "Prague"*).

This extraction process is based in an automatically trained extractor [Banko and Etzioni, 2008]. The extractor is domain independent and relation independent (Open IE), and has been run on 500 million high-quality webpages yielding over 800 million extractions. These are indexed in Lucene and can be queried by entity or relationship.

As shown in Figure 1, TextRunner results are returned ranked by frequency. TextRunner ranks results by frequency because, all other things being equal, extractions that appear frequently on high-quality Web pages are more likely to be correct [Downey et al., 2005]. However, experience has shown that this technique also yields many vague or otherwise uninteresting assertions.

2.1 Related Work

Traditional Information Extraction

A key aspect of this study is that in order to scale better to the full Web, we are studying models that can improve the interestingness of Web extractions in a domain independent and relation independent way. This is important because lexical rules (e.g. "all assertions about what companies Microsoft has bought are interesting") might work well for particular domains or relations but not apply more generally.

In traditional IE systems, system developers pre-specify relations of interest and then provide training examples. For example, the NAGA system [Kasneci et al., 2008] has considered methods for evaluating quality of web extractions, but their work is grounded in a graph representation based on the specific set of relationships that they chose to extract. This limited set of relationships meant that they could only evaluate 12 of 50 queries for one of their benchmarks.

Systems that Consider Interestingness

The general concept of using interestingness as a metric has value and applicability to a wide range of domains. For instance, Flickr recently launched a new feature[1] for identifying "Interestingness" in photos on its site. The Flickr

[1] http://www.flickr.com/explore/interesting/

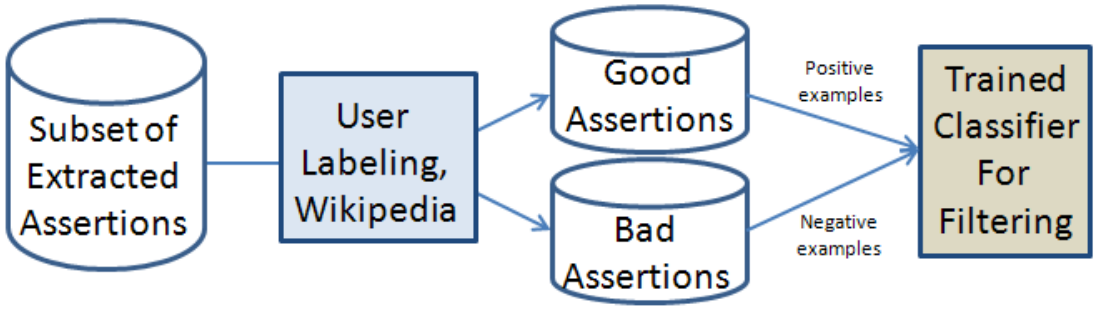

Figure 3. We train our filter using examples of good and bad assertions from TextRunner. We have filters based on both user labels and a Wikipedia Infobox approach that allowed us to easily leverage a large amount of user-contributed knowledge.

notion is based on social feedback such as click data and comments, supporting the idea that people care about what's interesting and leave indirect clues to where interesting content can be found. We use a similar concept later in learning from how people populate Wikipedia infoboxes.

Similarly, automated mathematical discovery programs require a notion of interestingness in order to identify which potential conjectures and concepts will be of interest to people. Colton and Bundy's survey [1999] identified several key concepts that these programs tended to use in deciding what would be interesting, including plausibility, novelty, surprisingness, comprehensibility and complexity. Liu et al. [2000] found that unexpected database association rules are more interesting to users.

What Makes Text Interesting?

Beyond the psychological work mentioned in the introduction, there has also been research into what makes text more interesting. It is important that text be the right level of complexity. Sentences with concrete words were found to be more interesting than abstract sentences [Sadoski et al., 1993]. Texts that are more coherent and easier to comprehend are more interesting [Schraw, 1997]. Prior knowledge in the subject generally increases interest. These ideas inform some of our classifier features later.

3 What's Interesting?

This section describes the problem of *interestingness*, then introduces three practical models for identifying interesting assertions. For each model, we first present an intuition behind characteristics that can make assertions interesting. We then operationalize those characteristics so that we can express them algorithmically.

We want to capture the interesting assertions, but what exactly does this mean? At the most general level, we define interesting assertions to be those that a person may find useful or engaging. For any particular query (e.g., "*Einstein*"), the extent to which possible assertions are interesting may vary greatly. A good set of results might, for example, include a mix of biographical facts like "*Einstein was born in Germany*" and other interesting facts like "*Einstein's favorite color was blue*". On the other hand, "*Einstein turned 15*" or "*Einstein wrote the paper*", would be less interesting because they express little useful information.

In a discussion of what is interesting, personalization is one approach to consider. Different people will find different topics to be interesting. We consider personalized notions of interesting to be a future direction, but currently focus on what characteristics make an assertion broadly interesting to a variety of people.

3.1 Specific Assertions

One quality of interesting assertions is that they tend to provide more *specific* information. For example, "*Albert Einstein taught at Princeton University*" is more interesting than "*Albert Einstein taught at a university*" because identifying Princeton as the university is informative. We hypothesize this is one characteristic that can make assertions interesting more broadly in TextRunner.

To operationalize this quality, we define a *specific* assertion as an assertion that either relates multiple proper nouns or an assertion that contains a year. If an assertion relates multiple proper nouns, it is specific because it expresses information about one specific entity relative to another. Similarly, an assertion that contains a year is specific because it contains specific temporal information.

3.2 Distinguishing Assertions

Another quality of interesting assertions might be providing *distinguishing* information about an object. This is related to novelty and surprisingness. Einstein may be a physicist who was born in Germany, but what really sets him apart and makes him interesting are his contributions to relativity theory and that he won the Nobel Prize. Conversely, assertions that do not set an object apart from other objects are often uninteresting.

We operationalize this notion of *distinguishing* using a technique similar to TF-IDF (term frequency – inverse document frequency) weighting [Salton and Buckley, 1988]. In IR, term frequency refers to the number of times a term occurs in a document. For our term frequency component, we define *AssertionFrequency* as the number of times an assertion occurs in the TextRunner set of assertions (e.g.,

the number of times TextRunner found that "*Einstein won the Nobel Prize*"). For our document frequency component, we define *ObjectFrequency* as the number of times the object (e.g., "*the Nobel Prize*") appears in a sample of ten million random TextRunner assertions. We define an *AFOFRatio(Extraction)* as follows[2]:

$$AFOFRatio(E) = \frac{AssertionFrequency(E)}{ObjectFrequency\big(object(E)\big) + 1}$$

For assertions, the *AFOFRatio* compares how often the assertion appears with how often we would expect the assertion to appear given its object. If the object has extremely high *ObjectFrequency* (e.g., a common word like "*food*"), the *AFOFRatio* will be low. If the object has extremely low *ObjectFrequency* (e.g., a misspelling or obscure term), then the *AFOFRatio* will be high. In the case of average *ObjectFrequency*, the *AFOFRatio* will reflect whether the assertion appears more often than one would normally expect.

Informal experimentation confirmed that extremely low *AFOFRatio* values often indicate an assertion is too vague to be interesting, while the very highest *AFOFRatio* values generally indicated assertions that were not well formed or well expressed. We chose a middle range ($1 < AFOFRatio \leq 10$) that seemed to generally yield interesting assertions from the *distinguishing* perspective.

3.3 Basic Assertions using Wikipedia

The final quality of interesting assertions that we focus on here are *basic* facts, definitional assertions that, for example, might be interesting to a person learning about an object. A person learning about Einstein, for example, might look up such facts as "*Einstein was a physicist*" or "*Einstein was born in 1879*". Although it would be difficult to define all-encompassing rules for what makes an assertion *basic*, we can take advantage of the fact that user-contributed knowledge bases like Wikipedia provide high quality examples of *basic* knowledge. Many Wikipedia articles contain infoboxes, tabular summaries of *basic* information about objects. Our operationalization of *basic* assertions, is therefore based in learning a classifier to identify assertions like those that human editors have decided to include in infoboxes.

Training a classifier with Wikipedia infobox data allows us to automatically leverage enough existing high quality human-generated knowledge to bootstrap learning [Wu and Weld, 2007]. We first obtain training data by automatically finding TextRunner assertions that reflect the information found in infoboxes. Notable people generally have populated Wikipedia infoboxes, so we obtain our training data using queries for famous people. We used the DBPedia Wikipedia Infobox database [Auer et al., 2007] and applied a series of filters to isolate a set of notable people with high quality infoboxes. We then matched the infobox data against TextRunner to obtain 1,584 assertions that reflected information found in infoboxes, to use as positive training examples, and an even larger set of assertions that did not match infoboxes, to use as negative training examples.

We train our *basic* classifier using around ten domain-independent features, such as the number of words in the assertion, whether the assertion relates proper nouns, and the estimated frequency of the assertion's object argument in TextRunner. Lexical features (those specific to the query terms, such as learning that any assertion with the relation "was born in" is interesting), are intentionally omitted because we are interested in a *generally applicable* classifier that is effective regardless of whether it was trained on assertions similar to those that it will classify (e.g., "was born in" is useless on assertions about fruits). An experiment showed that the lexical features enable greater precision at the cost of reduced recall and generality.

If we already have Wikipedia and we hypothesize that the infobox attributes are what's interesting, then why not just use all the Wikipedia data instead of using Web extraction? The key point here is that compared to the full Web, Wikipedia is incomplete. Many entities do not have Wikipedia articles, either because they have not been written yet or because they do not belong in a general encyclopedia. Even when an entity does have a Wikipedia article, often the infobox is incomplete or even missing. Also, while infobox attributes are good starting point for *basic* knowledge, there exist many additional similar attributes that also express *basic* knowledge. Web extraction has the potential for much greater coverage, both in terms of entities covered and attributes per entity.

4 Evaluating Human Ratings of Interesting

In order to evaluate our *specific*, *basic*, and *distinguishing* models, we used them each as the basis for filters that discard TextRunner results that fail to satisfy each model. To assess the quality of each filter, we conducted a study to collect human ratings of the interestingness of assertions.

4.1 Method and Procedure

We first selected a set of ten study query terms including famous people (*Albert Einstein, Bill Gates, Thomas Edison*), other proper nouns (*Beijing, Brazil, Microsoft, Diet Coke*), improper nouns (*sea lions*), and also relationship queries (*invented, destroyed*). This query set is meant to provide a varied sample of the sorts of queries for which TextRunner can provide interesting results. Our analyses are based on the top thirty assertions resulting from each of these queries, because the top results have the greatest impact on utility and about thirty results can be seen at a glance on a TextRunner results page.

As a baseline for comparison, we first obtain the number of times each assertion was found by TextRunner, and so

[2] We add 1 in the denominator to prevent possible division by 0.

our *AssertionFrequency* condition examines the thirty most frequently occurring assertions. We next obtain assertions for our *specific*, *distinguishing*, and *basic* conditions by applying each of our filters in order of assertion frequency, discarding results that fail the filter, until we obtain thirty results that satisfy the filter. The study therefore focuses on 1200 assertions (10 queries * 4 conditions * 30 assertions).

We recruited 12 study participants (7 female), who had a variety of backgrounds including math, marketing, finance, music, and nursing. Participants were each asked to rate 200 assertions on a scale from 1 (labeled "Least Interesting") to 5 (labeled "Most Interesting"). Assertions were presented one at a time, drawn randomly without replacement between participants. We gathered two or three ratings for every assertion, helping to account for individual differences in what people consider interesting.

4.2 Results

We analyze participant ratings of interestingness using a mixed-model analysis of variance. We model our variable of interest, *condition* (values *AssertionFrequency*, *simple*, *distinguishing*, and *basic*), as a fixed effect. To account for learning or fatigue effects, we model *trial number* as a fixed effect. Similarly, we account for the possibility that how long a person viewed an assertion might impact their rating by modeling *time to rate* as a fixed effect. Finally, we account for variations in the interestingness of queries and variations in the ratings given by different people by modeling both *query* and *participant* as random effects.

We found no significant effect of either *trial number* or *time to rate*, and so remove both of them from the remainder of our analysis. The omnibus test reveals a significant main effect of *condition* ($F(4, 3542) = 15.6$, $p < .0001$), leading us to investigate pairwise differences. We use Tukey's Honestly Significant Difference (HSD) procedure to account for increased Type I error in unplanned comparisons. This shows *basic* yielded the most interesting assertions, significantly more interesting than *AssertionFrequency* ($F(1,3545) = 55.0$, $p < .0001$), *specific* ($F(4,3539) = 7.7$, $p \approx .005$), and *distinguishing* ($F(1,3547) = 10.3$, $p \approx .001$). Our other filters also significantly improved interestingness, as both *specific* ($F(1,3544) = 21.2$, $p < .0001$) and *distinguishing* ($F(1,3539) = 18.7$, $p < .0001$) were significantly more interesting than *AssertionFrequency*.

4.3 Relevance Feedback

Although our results showed that *basic* assertions are the most interesting and that all of our filters yield results that are significantly more interesting than *AssertionFrequency*, inspection of our data suggested that our filters identify *different* interesting assertions. We found that only 21% of the *interesting* assertions would be identified by all three filters. We therefore consider whether a learning-based method, using a classifier to combine information from all three filters, might perform better than any single filter.

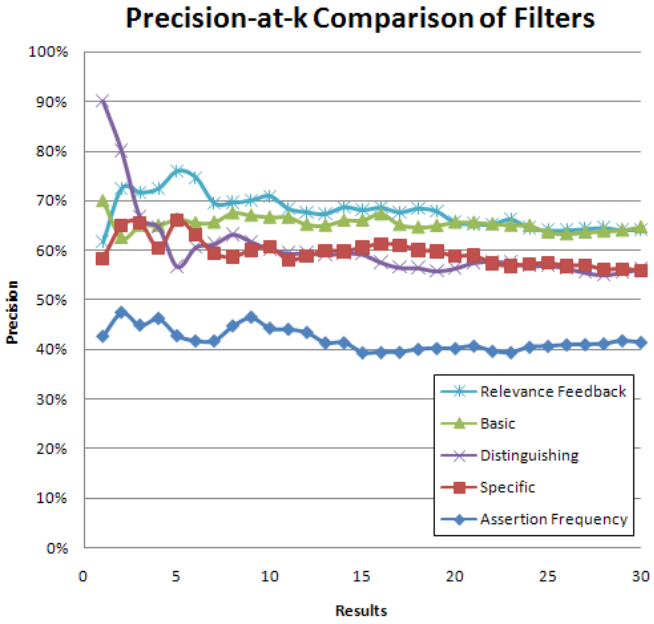

Figure 4. Our trained filters led to significantly higher mean average precisions for whether top assertions were interesting. Relevance Feedback (67.9%) was the best ($p \approx .005$). *Basic* (65.4%) was second best ($p < .0001$). *Specific* (59.5%) and *distinguishing* (60.3%) were also better ($p < .0001$) than Assertion Frequency (the results without any filtering), which had the lowest mean average precision at 41.9%.

In this case our training data is from study participants, but the same ideas would apply to collecting interactive feedback from users via a Web interface. We have developed such an interactive Web interface for TextRunner where people browsing query results can click on assertions to highlight them and specify whether they are of interest.

In order to simplify this and our remaining analyses, we first reduce our five-point scale to a binary classification. We define ratings of *4* or *5* to be *interesting*, define ratings of *1* or *2* to be *not interesting*, and ignore ratings of *3*. This discretization creates a nearly even split of our collected human labels, which we then use as positive and negative training examples for a relevance feedback classifier.

We make the output of our *specific*, *distinguishing*, and *basic* filters available as features to this classifier. We also provide the same features that are used by the *basic* classifier. Because we are interested in a *generally applicable* classifier of *interesting* assertions, we evaluate trained relevance feedback classifiers using ten-fold cross-validation such that we only test on the assertions from query terms not used to train the filter. This allows us to estimate performance on queries for which the system has not been trained, as we would expect improved performance on any queries for which the system has been trained.

Several classifiers from the WEKA toolkit [Witten and Frank, 2005] all had comparable *precision at k* values, averaging over ten-fold cross-validation, from k=1 to k=30. Decision Tree [Quinlan, 1993] was slightly better than the rest with an average precision at 67.9%. The *precision at k*

measure illustrates the percentage of the first *k* results that are *interesting*, and is an appropriate and important measure because it corresponds to the quality of the top results.

4.4 Analysis

Figure 4 plots the *precision at k* for the *relevance feedback* decision tree classifier versus our *specific*, *distinguishing*, and *basic* filters as well as against *AssertionFrequency*. To test for difference between these curves, we conduct an analysis of variance for the precision at each plotted point, treating *condition* and *k* as fixed effects. The omnibus test reveals a significant main effect of *condition* ($F(4, 4) = 285$, $p < .0001$), leading us to investigate pairwise differences. We use Tukey's HSD procedure to account for increased Type I error in unplanned comparisons. This shows that *relevance feedback* yields significantly more interesting assertions than *specific* ($F(1,144) = 95.4$, $p < .0001$), *distinguishing* ($F(1,144) = 78.6$, $p < .0001$), *basic* ($F(1,144) = 8$, $p \approx .005$) and *AssertionFrequency* ($F(1,144)=926$, $p<.0001$).

The largest differences in Figure 4 are between our filter-based approaches and TextRunner's original use of *AssertionFrequency*, indicating the advantage of filtering. The classifier filters trained with user-contributed knowledge (*relevance feedback* and *basic*) performed significantly better than all other approaches, indicating the utility of user-contributed knowledge for this task. Our *relevance feedback* classifier achieves a precision at 30 of 64.1% and a mean average precision of 67.9%. This is comparable to human level performance, as we measured inter-annotator agreement in our label set to be approximately 70%.

5 Conclusions

Extraction engines such as TextRunner are a promising avenue towards improving Web search and generating large knowledge bases. However, such systems are currently hamstrung by the fact that they often return uninformative results that are vague or uninteresting. Web extraction systems are particularly prone to this problem because of the general methods they use to extract entities and relationships [Banko and Etzioni, 2008]. This paper has developed filters based on user contributions that allow TextRunner to better focus on assertions that are *interesting*. These filters raised the average percentage of interesting results on a sample of queries from 41.6% to 64.1%.

Leveraging user-contributed knowledge to improve the quality of Open IE presents an interesting synergy. Open IE draws knowledge from the Web, and at the same time it contributes back to the Web in terms of smarter searching and question answering abilities and the ability to empower better AI applications via a large knowledge base. Our filtering uses knowledge from Wikipedia to enable higher quality Open IE, which in turn could lead to good contributions back to Wikipedia via projects such as those that use IE on Wikipedia article text to populate Wikipedia Infoboxes [Hoffmann et al., 2009].

One avenue of future work is to incorporate research on entity ranking [Zaragoza et al., 2007], which might provide valuable additional input in areas such as entity generality.

Acknowledgements

This research was supported in part by NSF grants IIS-0803481 and IIS-0812590, ONR grant N00014-08-1-0431, and Google and was carried out at the University of Washington's Turing Center.

References

[Auer et al., 2007] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives. DBPedia: A Nucleus for a Web of Open Data. In *Proc ISWC 2007*.

[Banko and Etzioni, 2008] M. Banko and O. Etzioni. The Tradeoffs Between Open and Traditional Relation Extraction. In *Proc ACL 2008.*

[Banko et al., 2007] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni. Open Information Extraction from the Web. In *Proc IJCAI 2007.*

[Colton and Bundy, 1999] S. Colton and A. Bundy. On the Notion of Interestingness in Automated Mathematical Discovery. In *Proc AISB 1999*.

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

[Hoffmann et al., 2009] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, D. Weld, Amplifying Community Content Creation Using Mixed-Initiative Information Extraction. In *Proc CHI 2009.*

[Kasneci et al., 2008] G. Kasneci, F. Suchanek, G. Ifrim, M. Ramanath, G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc ICDE 2008*.

[Liu et al., 2000] B. Liu, W. Hsu, S. Chen, Y. Ma, Analyzing the Subjective Interestingness of Association Rules. *IEEE Intelligent Systems 15*, 47-55, 2000.

[Quinlan, 1993] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[Sadoski et al., 1993] M. Sadoski, E.T. Goetz, J.B. Fritz. A causal model of sentence recall: Effects of familiarity, concreteness, comprehensibility and interestingness. *Journal of Reading Behavior, 25*, 5-16, 1993.

[Salton and Buckley, 1988] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval, *Information Processing & Management, 24*, 1988.

[Schraw, 1997] G. Schraw. Situational interest in literary text. *Contemporary Educational Psychology, 22*, 436-456, 1997.

[Silvia, 2006] P.J. Silvia, *Exploring the Psychology of Interest*. Oxford University Press, New York, NY, 2006.

[Witten and Frank, 2005] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, 2005.

[Wu and Weld, 2007] F. Wu, and D. Weld. Autonomously Semantifying Wikipedia. In *Proc CIKM 2007.*

[Zaragoza et al, 2007] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, G. Attardi. Ranking Very Many Typed Entities on Wikipedia. In *Proc CIKM 2007*.

Using an SVM Classifier to Improve the Extraction of Bilingual Terminology from Wikipedia

Maike Erdmann[1], **Kotaro Nakayama**[2], **Takahiro Hara**[1], **Shojiro Nishio**[1]
[1]Graduate School of Information Science and Technology, Osaka University
[2]Center for Knowledge Structuring, The University of Tokyo

Abstract

Bilingual dictionaries are usually constructed from large parallel corpora, but since these corpora are available only for selected text domains and language pairs, the potential of other resources is being explored as well.

In this paper, we want to further pursue the idea of using Wikipedia, a large-scale multilingual encyclopedia, as a corpus for bilingual terminology extraction. We propose a method that extracts term-translations pairs from different types of Wikipedia link information and lets an SVM classifier trained on the features of manually labeled training data determine the correctness of unseen term-translation pairs. In an experiment, we showed that our proposed method is effective for constructing bilingual dictionaries.

1 Introduction

Bilingual dictionaries hold great potential for emerging research areas such as machine translation and human-aided translation as well as cross-language information retrieval. Unfortunately, the manual construction of bilingual dictionaries is inefficient, since linguistic knowledge is expensive and new or domain-specific terminology is difficult to cover. Therefore, a lot of research has been conducted on the automatic extraction of bilingual dictionaries. In particular, the extraction from large parallel corpora [van der Eijk, 1993] has achieved impressive results. However, parallel corpora are available for only selected text domains and language pairs, since their construction is time-consuming and labor-intensive.

To solve that problem, we propose the extraction of bilingual terminology from Wikipedia or other large multilingual encyclopedias, in order to complement bilingual dictionaries with accurate term-translation pairs for languages and text domains where no parallel corpora exist. Wikipedia is a very promising resource as the continuously growing encyclopedia already contains more than 10 million articles in over 250 languages, has a dense link structures and covers a wide variety of topics.

In Wikipedia, there are many links between articles in different languages. If we regard the titles of Wikipedia articles as terminology, it is easy to extract translation relations by analyzing the interlanguage links, assuming that if two articles are connected by an interlanguage link, their titles are translations of each other. In addition, we can analyze redirect page and anchor text information to extend the number of term-translation pairs in the dictionary while maintaining a relatively high accuracy.

Since not all redirect page and anchor text information is suitable to extend our dictionary, we manually labeled a small number of extracted term-translation pairs and trained an SVM (Support Vector Machine) classifier on the characteristics (features) of that data. After that, the classifier can predict the correctness of unseen term-translation pairs with high accuracy. In an experiment, we proved the advantages of our new approach compared to previous methods on extracting bilingual terminology from Wikipedia.

2 Related Work

The traditional way of constructing bilingual dictionaries is by human effort. For the language pair German and English, one of the most popular freely accessible dictionaries is the online dictionary BEOLINGUS (http://dict.tu-chemnitz.org) from Chemnitz University of Technology, which contains more than 900,000 entries. The main disadvantage of manually created dictionaries is that new terms as well as domain-specific terms are difficult to cover.

In order to reduce the burden of manual dictionary construction, a lot of research has been conducted on the construction of bilingual dictionaries from parallel corpora, which are bilingual text collections consisting of the same text in two or more different languages. For German-English dictionary extraction, the largest available parallel corpus is the EUROPARL corpus [Koehn, 2005] consisting of documents from the European Parliament. Besides, other German-English corpora such as the OPUS collection of parallel texts [Tiedemann and Nygaard, 2004] (documents of the European Medicines Agency, open source software documentations, etc.) are also available. One of the main issues of bilingual dictionary extraction from parallel corpora is that sufficiently large parallel corpora are not available for all text domains, even though attempts have been made to construct parallel corpora automatically [Resnik and Smith, 2003].

Recently, several attempts have been made to use other types of corpora, such as Wikipedia, for automatic bilingual dictionary construction. For instance, [Adafre and de Rijke, 2006] have created a bilingual dictionary from Wikipedia in order to construct a parallel corpus from Wikipedia articles. [Bouma *et al.*, 2006] have extracted bilingual terminology in a multilingual question answering system and [Schönhofen *et al.*, 2007] for cross-language information retrieval. [Ferrández *et al.*, 2007] have used Wikipedia to translate and disambiguate named entities. In addition, [Erdmann *et al.*, 2008] used not only interlanguage links for extracting bilingual terminology, but also analyzed redirect page and anchor text information to increase the number of extracted translations.

3 Proposed Method

Our proposed method extracts term-translation pairs from Wikipedia. In the following, we will describe how we extract and filter this bilingual terminology.

3.1 Wikipedia Link Structure

In order to create a high accuracy and high coverage dictionary, we have analyzed several types of link information. Prior to describing our method, we explain the used link information.

Interlanguage Links
An interlanguage link in Wikipedia is a link between two articles in different languages. In most cases, the titles of two articles connected by an interlanguage link are translations of each other. The accuracy of interlanguage links is very high, although sometimes mistakes occur. For instance, if an article in one of the languages does not exist, a similar article is sometimes linked instead.

Redirect Pages
Redirect pages in Wikipedia are pages that link to a different article (target page) in order to facilitate the access to Wikipedia content. Redirect page titles are usually strongly related to the title of the target page. They often indicate synonym expressions, but can also be e.g. more general or more specific terms for which no separate Wikipedia articles exist.

Anchor Texts
An anchor text, also called link text, is the text part of a link that is presented to the user in the browser. Anchor texts are usually strongly related to the title of the linked page. They can be synonym expressions of the linked article, but are sometimes changed to fit in the sentence structure of the linking article.

Forward/Backward Links
For all the above mentioned types of links, we can distinguish the link direction. As shown in Figure 1, a forward link is an outgoing link and a backward link is an incoming link of a page. Both forward and backward links are useful information for extracting translation candidates. Furthermore, the number of backward links is a valuable factor for estimating the reliability of a redirect page or anchor text, as we will describe later.

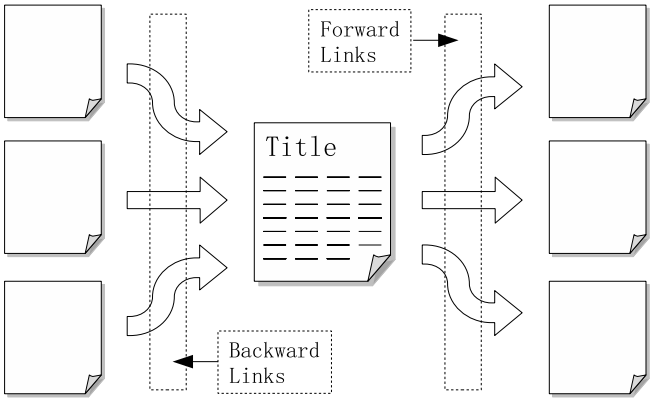

Figure 1: Forward and Backward Links

3.2 Extraction of Translation Candidates

We use the method for extracting bilingual terminology from interlanguage link, redirect page and anchor text information described by [Erdmann *et al.*, 2008]. At first, we create a baseline dictionary from Wikipedia interlanguage links. Then, assuming that redirect page titles are often synonym expressions of the target page title, we add new term-translation pairs to our dictionary in which we replace the target page title with a redirect page title. In the same way, we assume that anchor texts of Wikipedia pages can be synonym expressions of the page title, thus create term-translation pairs by replacing the page title by an anchor text.

3.3 Filtering Incorrect Term-Translation Pairs

Unfortunately, not all redirect page titles and anchor texts are synonym expressions of a target page title, thus not all term-translation pairs are correct. [Erdmann *et al.*, 2008] thus calculated a score for each term-translation pair based on the number of backward links of a redirect page or anchor text. After that, they manually labeled some of the extracted term-translation pairs in order to determine a threshold for filtering out unsuitable term-translation pairs based on this score.

However, in this paper we propose the usage of an SVM classifier for estimating the correctness of term-translation pairs automatically. The advantage of using a classifier is that we can easily add other features to identify and remove incorrect term-translation pairs more accurately. In total, our proposed method uses 12 different features from source and target language information which we will list up in the following, assuming a term-translation pair $\langle s, t \rangle$ with s being a term in the source language and t being a term in the target language.

Feature 1
(Whether s and t are connected by an interlanguage link)
Term-translation pairs that are directly derived from interlanguage links are more reliable than pairs extracted using redirect page or anchor text information.

Feature 2
(Whether t is redirect page title AND anchor text)
Terms that exist both as a redirect page title and an anchor

text are more reliable than terms that are only either redirect page title or anchor text.

Feature 3

(Number of backward links of t as a redirect page)
A large number of backward links of a redirect page indicates a high reliability of the redirect page title being a synonym of the target page title. We adjusted the number of backward links by taking the logarithm.

Feature 4

(Relative no. of backward links of t as a redirect page)
The relative number of backward links of a redirect page also indicates the reliability of the redirect page title being a synonym of the target page title. It is calculated by dividing the number of backward links of the redirect page by the number of backward links of the target page. This feature is calculated in the same way as the redirect page score described in [Erdmann *et al.*, 2008].

Feature 5

(Number of backward links of t as an anchor text)
A large number of backward links of a page with a certain anchor text indicates a high reliability of that anchor being a synonym of the page title. We adjusted the number of backward links by taking the logarithm.

Feature 6

(Relative no. of backward links of t as an anchor text)
The relative number of backward links of a page with a certain anchor text also indicates the reliability of that anchor text being a synonym of the page title. It is calculated by dividing the number of backward links having the specified anchor text by the number of backward links with any anchor text. This feature is calculated in the same way as the anchor text score described in [Erdmann *et al.*, 2008].

Feature 7

(Whether s OR t is derived from redirect page / anchor text)
Redirect pages and anchor texts in the source language can be more/less reliable than those in the target language.

Feature 8

(Whether s is redirect page AND anchor text)
Refer to the description of Feature 2.

Feature 9

(Number of backward links of s as a redirect page)
Refer to the description of Feature 3.

Feature 10

(Relative no. of backward links of s as a redirect page)
Refer to the description of Feature 4.

Feature 11

(Number of backward links of s as an anchor text)
Refer to the description of Feature 5.

Feature 12

(Relative no. of backward links of s as an anchor text)
Refer to the description of Feature 6.

We used the software LIBSVM [Chang and Lin, 2001] to train the classifier, since it comes with scripts that automate e.g. normalization of the feature values and optimization of the C and γ parameters. After training the classifier on a small amount of training data including examples of both correct and incorrect term-translation pairs, the classifier can estimate the correctness of unseen term-translation pairs with high accuracy.

4 Evaluation

[Erdmann *et al.*, 2008] have already proved that a bilingual dictionary constructed from Wikipedia can outperform dictionaries constructed from a large parallel corpus and as well as manually created dictionaries. Therefore, in this experiment, we focus on showing the advantage of our proposed method, which is using an SVM classifier to determine the correctness of the term-translation pairs, compared to manually setting a threshold as proposed by Erdmann et al. In the following, we will describe the experiment and discuss its results.

4.1 Training/Test Set Construction

We downloaded the German and English Wikipedia dump data from January 2008 containing 753,197 German and 2,393,591 English articles. From that data, we extracted all interlanguage links, anchor texts and redirect pages as well as the number of backward links for each page.

We randomly extracted term-translation pairs from German-English interlanguage links. On average, more than 95% of all interlanguage links in Wikipedia connect articles on proper nouns such as names of persons, places and products. However, since we are interested mainly in the translation of common nouns, we automatically filtered out proper nouns by analyzing the capitalization of English anchor texts (proper nouns always start with a capital letter whereas common nouns usually start with a small letter). In total, we extracted 300 term-translation pairs, of which a few examples are shown in Table 1.

In the next step, for each of the term-translation pairs extracted from interlanguage links, we extracted additional term-translation pairs from redirect page and anchor text information in both the German and the English Wikipedia. Some examples are shown in Table 2. The total number of term-translation pairs thus increased to 6266 pairs (on average 20.89 pairs per interlanguage link).

All term-translation pairs were manually categorized into "correct" and "incorrect". Of course, this categorization is not always clear-cut. The categorization has to be adopted to the type of application for which the dictionary is intended to be used. Apart from that, due to staff shortage the evaluation was done by only one judge. However, we tried to categorize the term-translation pairs as consistently as possible.

The labeling of all 6266 term-translation pairs took only a few hours, since most pairs could be labeled easily. Only a few pairs required the judge to consult additional information such as the respective Wikipedia article or a monolingual

Table 1: Term-Translation Pairs From ILL

German	English
Zweikeimblättrige	dicotyledon
Schneeschuh	snowshoe
Chylothorax	chylothorax
Ultimatum	ultimatum
Thema-Rhema-Gliederung	topic-comment
Individualismus	individualism
Diapsida	diapsid
Ausgangssperre	curfew
Ladeluftkühler	intercooler
Epische Vorausdeutung	foreshadowing
...	...

Table 2: Additional Term-Translation Pairs

German	English
Zweikeimblättrige	dicotyledon
Zweikeimblättrige	dicotyledonous
Zweikeimblättrige	broadleaf
Zweikeimblättrige	liliopsida
Zweikeimblättrige	magnoliopsida
...	...
Zweikeimblättrige Pflanzen	dicotyledon
Dikotyledonen	dicotyledon
Dicotyledoneae	dicotyledon
Dicotyledonen	dicotyledon
Zweikeimblättriger	dicotyledon
...	...

dictionary prior to making a decision. From the manually labeled data, we formed training and test sets to conduct cross-evaluation experiments.

4.2 Term-Translation Pair Distribution

During manual evaluation, we collected information on the distribution of term-translation pairs, as shown in Table 3. From the term-translation pairs extracted directly from interlanguage links, 92.3% were correct. By adding term-translation pairs from redirect pages and anchor texts, the percentage of correct pairs decreased drastically to only 23.1%. In the following, we will analyze the different types of incorrect term-translation pairs.

In 5.7% of term-translation pairs derived from interlanguage links and 61.9% of all pairs in total, one of the terms had a more specific or more general meaning than the other. For instance, the German term "Silikonarmband" (silicone wristband) was incorrectly translated as "wristband". These kinds of incorrect translations were extracted from interlanguage links, redirect pages as well as from anchor texts. One of the reasons is that in cases where a required article does not exist, sometimes an article with more general or more specific content is linked.

In 0.7% of term-translation pairs derived from interlanguage links and 7.9% of all pairs in total, one of the terms was a noun whereas the other term was a verb or adjective. For instance, the German term "Verdunstung" (evaporation) was incorrectly translated as "vaporized". These kinds of translation mistakes were often derived from anchor texts, since anchor texts have to be adapted to fit into the sentence structure of the linking article.

Table 3: Term-Translation Pair Distribution

Category	ILL only	All pairs
Correct	277 (92.3%)	1450 (23.1%)
Noun / Other than noun	2 (0.7%)	494 (7.9%)
Action / Actor	2 (0.7%)	177 (2.8%)
General / Specific	17 (5.7%)	3876 (61.9%)
Other	2 (0.7%)	269 (4.3%)
All categories	300 (100%)	6266 (100%)

In 0.7% of term-translation pairs derived from interlanguage links and 2.8% of all pairs in total, one of the terms described an action whereas the other term described an actor. For instance, the German term "Ehebruch" (adultery) was incorrectly translated as "adulterer".

Other incorrect term-translation candidates were caused by e.g. rarely used terms or misspellings. They were often extracted from redirect page titles, since redirect pages are intended to facilitate access to Wikipedia content by forwarding all article requests to the actual article. For instance, the German term "Aspic" is usually spelled "Aspik", and the English term "tounge twister" is a common misspelling of "tongue twister".

Overall, incorrect translations were extracted from redirect page and anchor text information much more often than from interlanguage links.

4.3 Methods

We compared the following four methods in our experiment:

- **BEOLINGUS**: Baseline method that assumes a term-translation pair is correct if it can be found in the BEOLINGUS dictionary.
- **ILL**: Baseline method that assumes that all term-translation pairs directly extracted from interlanguage links are correct.
- **SVM 2 Features**: Reproduces the results of the method proposed in [Erdmann *et al.*, 2008] by creating a classifier based on only the 2 features used in that work. We assume that the performance of both methods is comparable, thus we can avoid the time-consuming process of manual threshold determination.
- **SVM 12 Features**: Our proposed method which creates a classifier based on 12 different features from information in source and target language.

4.4 Comparison Criteria

We calculated the two standard criteria precision and recall to compare the accuracy and coverage of the different methods.

Precision measures the accuracy by calculating how many of the extracted term-translation pairs are correct. Recall measures the coverage by calculating how many correct term-translation pairs were extracted by a method compared to the total number of existing correct term-translation pairs.

Table 4: Experimental Results

Method	Precision	Recall	F_{10}-Measure	F_1-Measure	$F_{0.1}$-Measure
BEOLINGUS	1.000	0.126	0.137	0.224	0.613
ILL	0.923	0.191	0.206	0.317	0.685
SVM 2	0.841	0.303	0.322	0.446	0.725
SVM 12	0.772	0.350	0.369	0.482	0.696

It is not trivial to estimate the total number of correct term-translation pairs, since it cannot be calculated automatically. In our experiment, we decided to use a relative recall, assuming that the number of correct term-translation pairs found in Wikipedia is the total number of existing correct term-translation pairs.

The relative recall is sufficient to compare the different methods with each other. Relative recalls are often used where the absolute recall is difficult to estimate, e.g. for search engine evaluation [Goncalves *et al.*, 1998].

We further evaluated the balance of precision and recall by using the F_α-measure which is defined as:

$$F_\alpha = \frac{(1+\alpha) \cdot (\text{precision} \cdot \text{recall})}{\alpha \cdot \text{precision} + \text{recall}} .$$

In order to reflect the different requirements of applications using the bilingual dictionary, we calculated the F_{10}-measure (weighs recall ten times as much as precision), the F_1-measure (weighs precision and recall equally), and the $F_{0.1}$-measure (weighs precision ten times as much as recall).

4.5 Experimental Results

In the following, we will present and discuss the results of our experiment shown in Table 4.

Precision

The BEOLINGUS dictionary not surprisingly achieved the highest possible precision, since it has been constructed manually. The baseline dictionary that is using only inter-language links (ILL) also achieved a high precision. Our proposed method (SVM 12 Features) as well as the method that imitates the approach described in [Erdmann *et al.*, 2008] (SVM 2 Features) achieved a lower precision, since term-translation pairs extracted from redirect page and anchor text information are less accurate.

Recall

Our proposed method (SVM 12 Features) achieved a much higher recall than the reproduction of the method of Erdmann et al. (SVM 2 Features). The ILL method achieves an even lower recall, since the number of term-translation pairs that can be extracted by the ILL method is very limited. The BEOLINGUS dictionary achieved the lowest recall, because it covers almost none of the correct term-translation pairs that can be extracted from Wikipedia.

F-Measures

Our proposed method achieved a significantly higher f_{10}-measure and f_1-measure than the baseline methods and the SVM 2 Features method. Thus, our proposed method is very suitable for creating dictionaries where the recall is more important than the precision or where precision and recall are equally important, such as in dictionaries used in cross-language information retrieval.

The highest $f_{0.1}$-measure was achieved by the SVM 2 Features method. From that result, we can understand that for creating dictionaries where the precision is more important than the recall, e.g. in dictionaries used for machine translation, feature quality is more important than feature quantity, thus only the most reliable features should be used to train the classifier.

4.6 Statistical Significance

We applied the McNemar's test in order to estimate the significance of the improvement by the SVM 12 Features method compared to the results of the SVM 2 Features method. Both two-tail-based and one-tail-based P values fell below 0.000001, thus the difference between both methods can be considered statistically relevant by conventional criteria.

4.7 Training/Test Time

Training time is affected by the amount of training data and the number of features. For the largest amount of training data and the largest number of features, the training time took 7.5 hours on a standard PC. However, since the training has to be conducted just once, only the testing time (time to determine the correctness of unknown term-translation pairs) is important. In our experiment, the average test time took less than one second for even large amounts of test data.

5 Conclusion and Future Work

In this paper, we presented our approach of bilingual dictionary construction from Wikipedia, in order to help satisfy the demand of bilingual dictionaries in research areas such as machine translation or cross-language information retrieval. As opposed to previous research, we used an SVM classifier to determine the correctness of term-translation pairs automatically rather than manually determining a threshold. This enabled us to take into account not only the number of backward links but various other features of a term-translation pair.

In order to prove the advantages of our proposed method, we conducted an experiment on a randomly extracted and manually labeled set of mostly domain-specific term-translation pairs from Wikipedia. The experiment proved that our SVM classifier trained on 12 different features achieved a significantly higher recall than a method where a threshold is set based on only 2 different features, and is thus very suitable for creating dictionaries where the recall is at least as important as the precision. The experiment also showed that many of the extracted term-translation pairs are not covered in even

comprehensive manually created dictionaries. Furthermore, since Wikipedia is growing continuously, both accuracy and coverage of our dictionary will become even better in near future.

We believe that it is promising to combine our dictionary with manually constructed dictionaries such as the BEOLINGUS dictionary in order to enhance the coverage for common terms, especially for word groups other than nouns. In the future, we want to further improve our method, e.g. by using additional features or by experimenting with different feature combinations. Another goal is to apply dictionaries constructed from Wikipedia to applications such as machine learning or cross-language information retrieval.

Acknowledgments

This research was supported in part by Grant-in-Aid on Priority Areas (21300032, 20500093), and by the Microsoft Research IJARC CORE project.

References

[Adafre and de Rijke, 2006] Sisay Fissaha Adafre and Maarten de Rijke. Finding similar sentences across multiple languages in wikipedia. In *Proceedings of the EACL Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, 2006.

[Bouma *et al.*, 2006] Gosse Bouma, Ismail Fahmi, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. The university of groningen at qa@clef 2006 using syntactic knowledge for qa. In *Working Notes for the Cross Language Evaluation Forum Workshop*, 2006.

[Chang and Lin, 2001] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[Erdmann *et al.*, 2008] Maike Erdmann, Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. Extraction of bilingual terminology from a multilingual web-based encyclopedia. *Journal of Information Processing (JIP)*, 16, 2008.

[Ferrández *et al.*, 2007] Sergio Ferrández, Antonio Toral, Óscar Ferrández, Antonio Ferrández, and Rafael Mu noz. Applying wikipedias multilingual knowledge to crosslingual question answering. In *Natural Language Processing and Information Systems*, 2007.

[Goncalves *et al.*, 1998] P. Goncalves, J. Robin, T. Santos, O. Miranda, and S. Meira. Measuring the effect of centroid size on web search precision and recall. In *Proceedings of the Annual Conference of the Internet Society (INET)*, 1998.

[Koehn, 2005] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*, 2005.

[Resnik and Smith, 2003] Philip Resnik and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3), 2003.

[Schönhofen *et al.*, 2007] Péter Schönhofen, András Benczúr, István Bíró, and Károly Csalogány. Performing cross-language retrieval with wikipedia. In *Workshop of the Cross-Language Evaluation Forum (CLEF)*, 2007.

[Tiedemann and Nygaard, 2004] Jörg Tiedemann and Lars Nygaard. The opus corpus - parallel & free. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2004.

[van der Eijk, 1993] Pim van der Eijk. Automating the acquisition of bilingual terminology. In *Proceedings of the Conference on European chapter of the Association for Computational Linguistics (EACL)*, 1993.

Scaling Wikipedia-based Named Entity Disambiguation to Arbitrary Web Text

Anthony Fader, Stephen Soderland, and Oren Etzioni
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{afader, soderlan, etzioni}@cs.washington.edu

Abstract

This paper investigates the "named-entity disambiguation" task on the Web—identifying the referent of a string, found on an arbitrary Web page. The GROUNDER system, introduced in this paper, addresses two challenges not considered by previous work: how to utilize *a priori* information (*e.g.*, Bill Clinton is more prominent on the Web than Clinton County) to improve disambiguation, and how to compose this prior information with contextual evidence.

GROUNDER addresses both challenges by leveraging the user-contributed knowledge in Wikipedia and providing a novel formulation of the task. On a sample of strings drawn from the Web, GROUNDER achieves precision of 1.0 at recall 0.34, and precision 0.90 at recall 0.60.

1 Introduction and Motivation

The problem of determining the referent of a word or phrase has its roots in the philosophy of language where Gottlob Frege analyzed the distinction between the meaning and referent of the phrase "the morning star", and Hilary Putnam considered whether "water" refers to the same substance in a hypothetical "twin earth" where water has the same functional role but a different chemical composition [Putnam, 1975]. In the AI and database literatures, more pragmatic versions of the problem have been explored under the headings entity deduplication, reference reconciliation, and more [Yates and Etzioni, 2007; Singla and Domingos, 2005]. We are interested in the problem as it manifests on arbitrary Web text, which means that we cannot restrict the entities to particular types as in [Singla and Domingos, 2005] for example. In contrast with [Yates and Etzioni, 2007], we leverage the user-contributed knowledge in Wikipedia.

Thus, this paper investigates the "named-entity disambiguation" task on the Web—identifying the referent of a string, found on an arbitrary Web page, leveraging the set of entities described by Wikipedia articles. Seminal work by Bunescu and Pasca [2006] and Cucerzan [2007] introduced this task. However, their work suffers from a key limitation: it does not factor *a priori* information into the disambiguation decision.

Not all entities are "created equal"—some are *a priori* more likely to serve as the referents of textual strings. Consider, for example, the string "Clinton"—it could potentially refer to Bill, Hillary, or Roger Clinton or even to Clinton County. However, *a priori* the string is much more likely to refer to Bill or Hillary Clinton than to Clinton County given Bill and Hillary's prominence on the Web. Of course, we can't ignore the possibility that, in some contexts, the string *does* refer to Clinton County. Contextual evidence might cause us to map the string "Clinton" to Clinton County, but it would have to be fairly strong evidence. Thus, we need a means of quantifying *a priori* information, and a method for combining it with contextual evidence to yield disambiguation decisions.

This paper reports on the GROUNDER system, the first named-entity disambiguation system that composes *a priori* prominence information with contextual evidence to yield superior disambiguation decisions. Our contributions are as follows:

- We introduce a novel formulation of the task, and highlight the value of a prior over entities for this task.
- We present an overview of the GROUNDER system, which operationalizes this formulation into a working system that draws its prior automatically from Wikipedia.
- We report on a set of experiments that demonstrate the value of using prior information in concert with contextual evidence. GROUNDER achieves a precision of 90% at a recall of 60%.

The remainder of the paper is organized as follows. Section 2 describes closely related work. Section 3 provides an overview of the GROUNDER system. Section 4 describes our experimental results, and Section 5 concludes with a discussion of future work.

2 Previous Work

The problem of named entity disambiguation has a long history in the database community, where the task is presented as a classification problem: given a vector of similarities between two records in a database, output whether they refer to the same entity or not [Fellegi and Sunter, 1969]. Work that is closer to ours leverages a graph of known relationships between entities to find the entity that best matches a

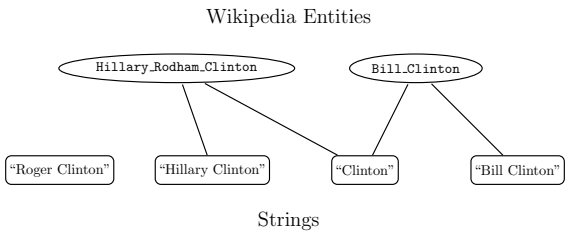

Figure 1: A possible relationship between Wikipedia entities (top) and a set of strings (bottom). Mapping ambiguous strings such as "Clinton" to the correct entity pose a harder problem than unambiguous strings such as "Hillary Clinton".

given reference [Bhattacharya and Getoor, 2006a]. Our work is similar in that it relies on the Wikipedia graph of entities to obtain a measure of entity prominence.

There has also been work on providing a probabilistic framework for named entity disambiguation in text [Li *et al.*, 2004; Bhattacharya and Getoor, 2006b]. However, these frameworks make assumptions about the types of named entities and cannot be applied directly to the problem of named entity disambiguation on the Web.

The work presented in this paper builds on the ideas described in papers by Bunescu and Pasca [2006] and by Cucerzan [2007], both of which attempt to disambiguate named entities by mapping them to Wikipedia articles. Each of these papers make use of the fact that the context surrounding an ambiguous string gives useful evidence for disambiguating it. Given an ambiguous string s and its context, their systems find all Wikipedia articles that can be referred to as s, and "ground" s to the article whose content best overlaps with the context of s.

Bunescu and Pasca [2006] measure this overlap using tf-idf cosine similarity. Bunescu and Pasca found that the text of Wikipedia articles is often not enough to disambiguate an ambiguous string, despite its sense being clear from the context. To address this issue, they use a supervised learning technique to enrich a given Wikipedia article's term vector with words from articles in the same category. They evaluate their system on Wikipedia articles and obtain accuracies ranging from 77.2% to 84.8%.

Cucerzan [2007] takes a similar approach, but uses context vectors consisting of key words and short phrases extracted from Wikipedia. Cucerzan's system also attempts to disambiguate all named entities in a single context simultaneously, adding the constraint that the target Wikipedia articles should be from the same categories. He evaluated his system on Wikipedia articles and a set of 100 news articles, obtaining accuracies of 88.3% and 91.4%, respectively.

Our work builds on these two approaches and follows their lead on using Wikipedia as a database of candidate entities. However, these previous systems focus only on contextual evidence for disambiguation and fail to include an explicit notion of prior information about the relationship between a string and its referent entities. We show that for the named entity disambiguation problem, prior information about entity prominence turns out to be very useful. This prior information can be used in conjunction with contextual information and we show that doing so leads to better performance than either component in isolation.

Another important difference between this work and the previous two is that we evaluate our system on domain-independent Web text, as opposed to only news and Wikipedia articles. We obtained performance on arbitrary Web text that is consistent with the performance on the previous systems.

3 The GROUNDER System

This section introduces the GROUNDER system, which uses a novel formulation of the named entity disambiguation problem. The key insight used in GROUNDER is that *a priori* information about the ambiguity of a string is valuable for named entity disambiguation. We call this type of information a *prominence prior* over strings and entities. We can think of a prominence prior as representing the GROUNDER system's real-world knowledge about the ambiguity of a string s: what are the entities s could possibly refer to and with what probability?

In order to leverage this type of prior information, we are faced with two challenges. The first challenge is finding a source of information to use as our prominence prior. Our solution to this problem relies on the fact that Wikipedia's network structure naturally encodes a measure of entity prominence: highly linked articles tend to be about more prominent entities than articles with fewer incoming links. In addition to this, Wikipedia's article titles and redirect pages provide a large amount of information about the entities a string can refer to. We combine these two sources of information to compute a prominence prior, which lets us answer questions about the likely referents of a string independent of its context.

Given a source of information for our prominence prior, the second challenge we face is the question of *how* to combine it with contextual evidence. We solve this problem by combining contextual and prior information via Bayes' theorem.

In the following sections, we describe in detail GROUNDER's novel model of named entity ambiguity, our source of contextual evidence for disambiguation, and the Wikipedia-based prominence prior outlined above.

3.1 A Novel Model for Named Entity Disambiguation

Suppose that we observe a named entity string s in a document D. Given a database E of candidate entities, we define the *named entity disambiguation problem* as the task of maximizing $P(s \rightarrow e^*|D)$, *i.e.* finding the entity $e \in E$ that s most likely refers to in the context given by D. We can express this as the problem of maximizing the probability $P(s \rightarrow e|D)$ over all entities $e \in E$, where the notation $s \rightarrow e$ represents the event that s refers to e. Rewriting this using Bayes' theorem, we can restate named entity disambiguation as finding the entity e^* in the following optimization problem:

$$e^* = \arg\max_{e \in E} P(D|s \rightarrow e)P(s \rightarrow e). \quad (1)$$

We make the simplifying assumption that the normalizing constant $P(D)$ in Bayes' theorem is uniform in order to

PROBDISAMBIG(s, D, τ):

1. $e^* = \arg\max_{e \in E} P(D|s \rightarrow e)\ P(s \rightarrow e)$
2. **if** $P(s \rightarrow e^*|D) > \tau$, **return** e^*
3. **else return** $NoEntity$

Figure 2: Pseudocode for a named entity disambiguation algorithm based on the model defined in Section 3.1. The inputs are: s, a named entity string; D, the document containing s; and τ, the minimum value that the posterior probability $P(s \rightarrow e|D)$ must obtain for e^* to be returned.

threshold the value of $P(s \rightarrow e|D)$ to control precision and recall.

The two factors on the right hand side of Equation 1 correspond to the two sources of information that we include in GROUNDER. The first factor $P(D|s \rightarrow e)$ represents the likelihood of seeing the document D given that we know it contains a reference to e. We can interpret the role of this in the optimization as the source of *contextual evidence* that the document D gives us. The second factor $P(s \rightarrow e)$ is the prior probability of the string s referring to e. This is the *prominence prior* that was introduced in the previous section and can be thought of as a measure of the ambiguity of s.

This view of named entity disambiguation has three clear benefits. First, it corresponds to our intuition about how human readers disambiguate a string s. If we believe that s refers to e with high probability, then it would take a lot of contextual evidence to convince us otherwise. On the other hand, if s is ambiguous, then the role of context becomes more important for our decision.

The second benefit of GROUNDER's model is that it gives us a measure of the uncertainty of the most likely entity e^* in Equation 1. This is useful for controlling the precision and recall of a system that uses the model. For example, we could choose a threshold parameter $\tau \in [0, 1]$ and instruct the computer to ground only the strings s such that $P(s \rightarrow e^*|D) > \tau$.

Lastly, a third benefit of our model is that it provides a very general framework for incorporating different types of information into the disambiguation process. By changing the details of the contextual and prior model components, we can account for new types of evidence. In this sense, our model describes a family of algorithms for named entity disambiguation. The pseudocode for an algorithm based on this model is shown in Figure 2.

3.2 GROUNDER Implementation

In the following sections, we describe our implementation of the GROUNDER system, which consists of three parts: the entity database E, the context model component $P(D|s \rightarrow e)$, and the prominence prior model component $P(s \rightarrow e)$.

Wikipedia as an Entity Database

Following the work of Bunescu and Pasca [2006] and Cucerzan [2007], we can treat Wikipedia as an entity database, where each article corresponds to an entity. Wikipedia is well-suited to act as an entity database for named entity disambiguation on the Web: it is domain-independent and contains millions of articles, so many of the named entities mentioned on the Web have a Wikipedia article. Let E represent the set of all entities in Wikipedia. Define $Article(e)$ to be the text of the article on entity $e \in E$. We use a basic filter to remove articles that do not describe entities (e.g., list and category pages).

Cosine Similarity Context Model

Our implementation of the context model component $P(D|s \rightarrow e)$ is based on the assumption that if D contains a reference to e, then the words used in D will tend to overlap with the words used in the Wikipedia article $Article(e)$. To measure this overlap, we treat D and $Article(e)$ as tf-idf vectors and compute the cosine similarity of D and $Article(e)$:

$$P(D|s \rightarrow e) = \cos(D, Article(e)). \quad (2)$$

We computed the idf scores for each term using a Lucene index of Wikipedia as a corpus (see the following section for more information), which defines the idf score of a term t as

$$idf(t) = 1 + \log\left(\frac{N+1}{n_t}\right),$$

where n_t is the number of Wikipedia articles containing the term t and N is the total number of Wikipedia articles. We note that our model gives a simple measurement of similarity between the context and the Wikipedia articles and is not a probability measure. Future work would include extending this to a full generative model of the context.

In order to compare the behavior of the GROUNDER system to one that ignores the prominence prior information, we define the COSINE-SIM algorithm, which simply returns the entity s that maximizes the cosine similarity in equation (2):

$$\text{COSINE-SIM}(s, D) = \arg\max_{e \in E} \cos(D, Article(e)).$$

The COSINE-SIM algorithm is similar to the baseline method used in [Bunescu and Pasca, 2006], which also uses tf-idf cosine similarity to compare contexts to Wikipedia articles.

Search Engine Prominence Prior

Given a string s and an entity $e \in E$, how do we come up with a good prior probability $P(s \rightarrow e)$? In order to answer this, consider the following types of evidence that would suggest that s refers to e:

1. e is known to be referred to as s or a string similar to s
2. e is referred to with high frequency relative to other entities in E that can be referred to as s

The first type of information is a necessary condition for the event $s \rightarrow e$ to have a non-zero probability: there must be some evidence that e can be referred to as s or else $P(s \rightarrow e)$ should be 0. The second type of information is based on the intuition that if e is more prominent than another entity e', then $P(s \rightarrow e)$ should be greater than $P(s \rightarrow e')$.

In the GROUNDER system, we use an existing Lucene-based Wikipedia search engine[1] to calculate these two types

[1]Available at `http://www.mediawiki.org/wiki/Extension:Lucene-search`.

of information. This software is currently being used as the public search engine for the English-language Wikipedia. Given a query string s, the search engine returns a list of scored entities. The search score of an entity e with respect to the query string is given by

$$\begin{aligned}\text{search-score}(e,s) &= \text{query-match}(e,s) \\ &\times \left(1+\log\left(1+\frac{\text{in-deg}(e)}{\alpha}\right)\right).\end{aligned}$$

The value $\text{search-score}(e,s)$ can be interpreted as how likely it is that s refers to e in the absence of context. The first factor $\text{query-match}(e,s)$ is a non-negative score representing how well the query s matches the article name of e, the titles of any redirect pages to e, and the words of the article of e. The second factor boosts the score of e proportional to the number of incoming links on Wikipedia, written as $\text{in-deg}(e)$. This can be interpreted as a measure of prominence of e in the Wikipedia hyperlink network. We used the search engine's default value of $\alpha = 15$ in our experiments. We define $\text{search-score}(e,s)$ to be 0 for any e that is not in the set of articles returned when searching for s. We also normalize the search scores such that $\sum_{e' \in E} \text{search-score}(e',s) = 1$.

In our experiments, we were interested in testing the effects of this measure of prominence on performance, so we introduce a smoothed version of the search score given by

$$\text{search-score}(e,s|\lambda) = \frac{\lambda}{N_{s,e}} + (1-\lambda)\cdot \text{search-score}(e,s) \quad (3)$$

where $N_{s,e}$ is the number of entities e such that $\text{search-score}(e,s) \neq 0$. We can think of this as a mixture of a uniform score over all candidate entities with the search score. A value of $\lambda = 0$ corresponds to the full search score and a value of $\lambda = 1$ corresponds to a uniform score. We can now define the prominence prior as

$$P(s \rightarrow e|\lambda) = \text{search-score}(e,s|\lambda).$$

To compare the GROUNDER system to one that ignores contextual evidence, we introduce the PRIOR algorithm, which simply returns the entity that maximizes the prior probability in (3), given a smoothing parameter λ:

$$\text{PRIOR}(s,\lambda) = \arg\max_{e \in E} P(s \rightarrow e|\lambda).$$

The GROUNDER Algorithm

Now that we have defined the necessary components of the model described in Section 3.1, we can combine them and formally define GROUNDER. The GROUNDER assigns each entity $e \in E$ a score $score[e]$, which is the product of its local context probability and the prior prominence probability. Figure 3 shows the pseudocode of GROUNDER, which includes a threshold parameter $\tau \in [0,1]$ to control precision and recall.

4 Experimental Results

This section evaluates GROUNDER's performance in grounding strings drawn from a "random" set of Web pages. Section 4.1 describes our method of creating a dataset and Section 4.2 uses this dataset to characterize the problem space for grounding arbitrary proper nouns to Wikipedia concepts. Sections 4.3 and 4.4 present our experimental results.

GROUNDER(s, D, λ, τ):

1. **for** $e \in E$:

   $score[e] = \cos(D, Article(e)) \cdot \text{search-score}(e, s|\lambda)$

3. $e^* = \arg\max_{e \in E} score[e]$
4. **if** $score[e] > \tau$ **return** e^*
5. **else return** $NoEnt$

Figure 3: Pseudocode for GROUNDER. The inputs are: s, a named entity string; D, the document containing s; λ, the prior smoothing parameter; and τ, the minimum value that $P(s \rightarrow e|D,\lambda)$ must obtain for e^* to be returned.

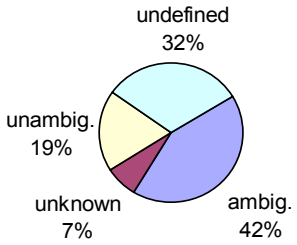

Figure 4: Nearly one third of proper nouns in our dataset have no Wikipedia page (undefined). Only 19% are the easy cases that unambiguously refer to a single Wikipedia page.

4.1 Creating a Web-based Dataset

We began with a corpus of approximately 500 million Web pages of arbitrary topics and genres, including blogs, news articles, and online stores. Associated with this Web corpus is a set of tuples extracted by the TEXTRUNNER Open IE system [Banko *et al.*, 2007; Banko and Etzioni, 2008]. These tuples are extractions in the form $(arg_1, pred, arg_2)$ where arg_1, $pred$, and arg_2 are text strings and $pred$ expresses a relation between arg_1 and arg_2. (e.g., ("Clinton", "born in", "Hope, AR")).

We collected a set of 500 tuples and their Web page of origin, uniformly sampling from a collection that have a proper noun as arg_1 and were manually verified to be correct extractions. We took the arg_1 values as our set of strings s and the associated Web page as the document D associated with s.

To create a gold standard from this dataset, we manually identified the entity $e \in E$ to which that s refers (and set $e = NoEntity$ if Wikipedia did not contain an article for that entity). This gives us a set of 500 triples (s, D, e), where s is proper noun string occurring on document D, and e is the entity to which that s actually refers.

4.2 Characterizing the Problem Space

This section addresses the following questions: What proportion of the strings s in our dataset are relatively easy to map to the correct e? What proportion present a relatively difficult disambiguation problem? And what proportion are impossible to map to Wikipedia because a Wikipedia page for s does

not exist?

To help answer these questions, we distinguish between four types of s: *undefined*, *unambiguous*, *ambiguous*, and *unknown*. Undefined strings are cases where s refers to an entity that does not have a Wikipedia page. The remaining three cases are defined in terms of the set of Wikipedia entities $Ents(s)$ that can be referred to by s. A string s is unambiguous if $Ents(s)$ contains exactly one Wikipedia page; it is ambiguous if $|Ents(s)| > 1$; and it is unknown if there is a Wikipedia page e that corresponds to s, but s is not among the known ways to refer to e, i.e. $|Ents(s)| = 0$.

We compute $Ents(s)$ from information that Wikipedia provides about the different ways to which an entity $e \in E$ can be referred, following the methodology of Bunescu and Pasca [2006] and Cucerzan [2007]. This information comes from four different sources: article titles, redirect pages, disambiguation pages, and hyperlink anchor text. For each $e \in E$, we can define a set $Names(e)$ containing the strings that are known to refer to e in Wikipedia. We can also define the set $Ents(s) = \{e \in E : s \in Names(e)\}$, which is the collection of entities that s can refer to in Wikipedia.

We can represent the relationship between strings and entities as a bipartite graph, where there is an edge between a string s and an entity e when $s \in Names(e)$. Figure 1 shows a simple example relating a set of entities {`Bill_Clinton`, `Hillary_Rodham_Clinton`} to a set of strings {"Bill Clinton", "Hillary Clinton", "Roger Clinton", "Clinton"}. In this toy example, the strings "Bill Clinton" and "Hillary Clinton" are unambiguous, the string "Clinton" is ambiguous, and the string "Roger Clinton" is undefined.

Let $\mathcal{D}$ be the set of 500 triples (s, D, e), where s is the first argument of the extraction, D is the text of the document containing the extraction, and e is the entity that s actually refers to. Figure 4 shows how the string-entity pairs (s, e) in $\mathcal{D}$ are distributed relative to the information in Wikipedia. 32% of the s in our dataset are undefined – no method using the $Ents$ and $Names$ relations can possibly map them to a Wikipedia page. Only 19% constitute the easy case of strings that unambiguously map to a single Wikipedia page, such as "Hillary Clinton".

The remainder are the hard cases. To make matters worse, we found that 31 of the 208 ambiguous s did not include the correct e in the set of candidate entities $Ents(s)$. Hence, an entity grounding algorithm that relies on the set of known references to s on Wikipedia, will fail on these cases as well as on the *unknown* cases. We need an entity grounding algorithm that is not limited by the incompleteness of $Names(e)$. The GROUNDER system avoids this problem by using a search engine over Wikipedia articles, allowing inexact matches on article titles and redirect pages.

4.3 Evaluation of GROUNDER

This section compares the full GROUNDER algorithm with its two main components—PRIOR and COSINE-SIM. Our experiments utilize recall and precision metrics. To do so, we created a ranked list of results for each method on the dataset $\mathcal{D}_W$, the subset of $\mathcal{D}$ that have a corresponding Wikipedia article (*i.e.*such that $e \in E$). ordered by that method's confidence score. As we vary a threshold τ, we can define recall as the percentage of correctly grounded strings s with confidence greater than τ divided by all possible correct groundings in $\mathcal{D}_W$. Precision is the percentage of correctly grounded s divided by the number of results with confidence greater than τ.

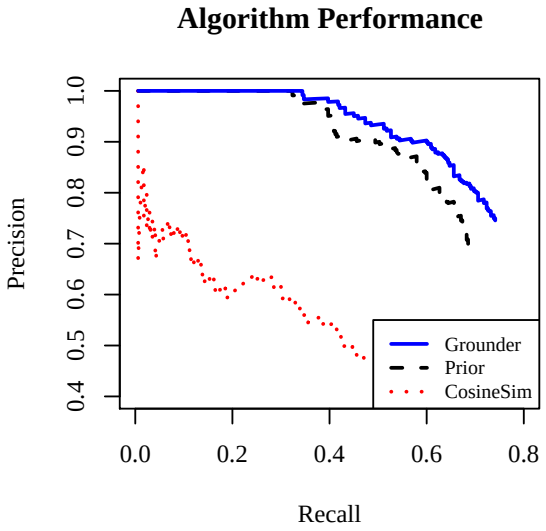

Figure 5: Combining both the search engine score (Prior) and the Cosine Similarity gives better performance on $\mathcal{D}_W$ than either knowledge source alone.

Figure 5 shows a recall-precision curve for COSINE-SIM, which uses Cosine Similarity exclusively; for PRIOR, which uses the prominence prior computed from the search engine scores; and the full GROUNDER system. For this experiment we set $\lambda = 0.0$, which uses the unsmoothed prior.

The COSINE-SIM method suffers from lower recall and precision than either of the other methods. Combining the two knowledge sources gives the best result, with a recall-precision curve that is consistently higher than PRIOR alone. While cosine similarity is informative, it is unable to make many fine-grained distinctions. The Prior given by the Lucene search engine score completely ignores local context from the page D, but succeeds when s is either unambiguous (e.g. "Hillary Clinton") or when an ambiguous s refers to the most prominent Wikipedia page. PRIOR can achieve precision 1.0 for 31% of our dataset $\mathcal{D}_W$. We also evaluated PRIOR on Cucerzan's news dataset [2007], which achieved 0.86 precision at 0.68 recall. This suggests that, as in our dataset, the distribution $P(s \rightarrow e)$ for a given s tends to be skewed towards a single entity.

The remaining cases are the hardest ones to disambiguate, where local context is necessary to find the correct entity. We examined the "hard cases" in which s does not refer to the dominant sense of s. In these cases, where PRIOR is always wrong, we found that COSINE-SIM is able to distinguish the correct entity about 40% of the time. This performance is lower than the previously reported results, however this is expected because the "hard cases" dataset does not include any unambiguous strings.

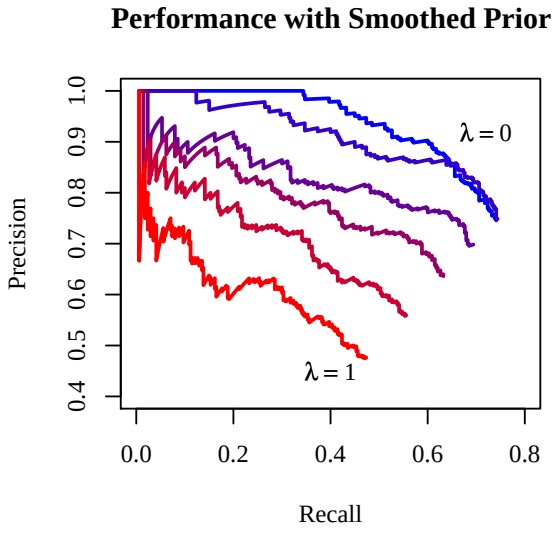

Figure 6: Performance improves as we vary λ from 1.0 to 0.0, at which point equal weight is given to the search engine score (Prior) and the Cosine Similarity.

4.4 Parameter Settings to Combine Knowledge Sources

The key to GROUNDER's success is the appropriate combination of the information in its two components: PRIOR and COSINE-SIM, which is controlled by the smoothing parameter λ. We carried out a sensitivity analysis to demonstrate that we chose a value for λ that is close to optimal.

In the analysis, we varied the parameter λ by increments of 0.2. Figure 6 shows the results for λ from 0.0 to 1.0. The lowest curve in Figure 6 is for $\lambda = 1.0$, which is identical to the COSINE-SIM algorithm. The results improve monotonically until $\lambda = 0.0$ which gives equal weight to both the prior and contextual evidence scores. Thus, we believe we have a value of λ that is close to optimal.

5 Conclusions and Future Work

This paper presented our GROUNDER system, which disambiguates named entities mentioned in text by mapping them to Wikipedia pages. A novel feature of the GROUNDER system is that it combines local contextual evidence with the prior probability given by search engine scores. We show that this method scales well to extractions from arbitrary Web text. GROUNDER achieves precision of 1.0 at recall 0.34 and precision 0.90 at recall 0.60.

We did not include any type-specific or genre-specific knowledge in GROUNDER to ensure that it scales to arbitrary entities. A future direction of research is to incorporate more types of evidence into the contextual component of the probabilistic model. For example, including type-specific coreference resolution to create soft constraints on entity types might be useful.

Another interesting direction would be to explicitly model the joint distribution $P(s_1 \rightarrow e_1, \ldots, s_n \rightarrow e_n | D)$ of a set of ambiguous strings $s_1, \ldots, s_n$ on D. One could imagine that knowledge about the referent entity of s_i would provide information about the referent entities of the other strings in D.

Finally, we plan to embed GROUNDER as a module in a variety of textual inference systems including cross-document reference resolution systems, textual entailment systems.

6 Acknowledgements

We thank Stefan Schoenmackers, Mausam, and Dan Weld for comments on earlier drafts and Silviu Cucerzan for helpful discussion.

This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431 as well as gifts from Google, and carried out at the University of Washington's Turing Center.

References

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

[Banko *et al.*, 2007] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Procs. of IJCAI*, 2007.

[Bhattacharya and Getoor, 2006a] Indrajit Bhattacharya and Lise Getoor. Entity resolutions in graphs. In D. Cook and L. Holder, editors, *Mining Graph Data*. Wiley, 2006.

[Bhattacharya and Getoor, 2006b] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM Conference on Data Mining (SDM)*, April 2006. Winner of the Best Paper Award.

[Bunescu and Pasca, 2006] Razvan C. Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*. The Association for Computer Linguistics, 2006.

[Cucerzan, 2007] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, 2007.

[Fellegi and Sunter, 1969] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[Li *et al.*, 2004] Xin Li, Paul Morie, and Dan Roth. Robust reading: Identification and tracing of ambiguous names. In *HLT-NAACL*, pages 17–24, 2004.

[Putnam, 1975] Hilary Putnam. The meaning of 'meaning'. In K. Gunderson, editor, *Language, Mind, and Knowledge*. University of Minnesota Press, 1975.

[Singla and Domingos, 2005] Parag Singla and Pedro Domingos. Collective object identification. In *IJCAI-2005*, pages 1636–1637, 2005.

[Yates and Etzioni, 2007] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*, 2007.

Learning to Integrate Relational Databases with Wikipedia

Doug Downey, Arun Ahuja
EECS Dept., Northwestern University
Evanston, IL 60208
{ddowney, arun.ahuja}@eecs.northwestern.edu

Mike Anderson
Rexonomy
Heidelberg, Germany
mrander@gmail.com

Abstract

Wikipedia is a general encyclopedia of unprecedented breadth and popularity. However, much of the Web's factual information still lies within relational databases, each focused on a specific topic. While many database entities are described by corresponding Wikipedia pages, in general this correspondence is unknown unless it has been manually specified. As a result, Web databases cannot leverage the relevant rich descriptions and interrelationships captured in Wikipedia, and Wikipedia readers miss the extensive coverage that a database typically provides on its specific topic.

In this paper, we present ETOW, a system that automatically integrates relational databases with Wikipedia. ETOW uses machine learning techniques to identify the correspondences between database entities and Wikipedia pages. In experiments with two distinct Web databases, we demonstrate that ETOW outperforms baseline techniques, reducing error overall by an average of 19%, and reducing false positive rate by 50%. In one experiment, ETOW is able to identify approximately 13,000 correct matches at a precision of 0.97. We also present evidence suggesting that ETOW can substantially improve the coverage and utility of both the relational databases and Wikipedia.

1 Introduction

Wikipedia is arguably the most comprehensive and frequently used knowledge base in existence. The Web-based encyclopedia contains user-contributed entries on a multitude of topics, providing detailed descriptions of millions of distinct entities and their interrelationships.

Nonetheless, it remains the case that much information on the Web resides in relational databases focused on a particular domain. For almost any conceivable topic, the Web contains a corresponding online database; examples include the USDA Nutrient Database for nutrition, the Internet Movie Database for films, and numerous similar databases focused on mountains, music, diseases, castles, digital cameras, and so on. For the most part, each database has coverage that—in its specific domain—greatly exceeds that of Wikipedia. However, because the databases are domain-specific, they lack useful connections to the more general knowledge found in Wikipedia.

In this paper, we present ETOW, a system that automatically *integrates* relational databases with Wikipedia by resolving precisely which Wikipedia page, if any, corresponds to each entity in a given relational database. This integration offers several benefits. For example, ETOW can enhance the relational database with helpful links into the general Wikipedia knowledge base. Likewise, as we illustrate, information from the database can be utilized to augment infoboxes on Wikipedia pages, or to create appropriate new pages. Further, in combination with recent automated techniques for categorizing Web pages in terms of Wikipedia concepts [Gabrilovich and Markovitch, 2007] and identifying mentions of Wikipedia concepts in text [Milne and Witten, 2008; Cucerzan, 2007], ETOW can link entities in relational databases to relevant content in the Web at large.

Resolving correspondences between database entities and Wikipedia pages is challenging primarily because multiple distinct entities may share the same name. For example, consider a "musicals" database containing a record for the Broadway hit "Chicago"; of the more than twenty Wikipedia pages corresponding to different meanings of the word "Chicago" (including a city, a typeface, a poem, a magazine, and so on), only one is a correct match for the musical. While Wikipedia does include a category system for articles, it is known to be both incomplete and unreliable [Wu and Weld, 2008]; further, even an improved category structure is unlikely to exactly match the relational structure employed in a particular database. Thus, identifying the correct page requires utilizing other clues as well, such as whether the Wikipedia page includes text indicative of the entity's type, or whether the page text mentions the attributes and relations of the entity in the database. ETOW employs machine learning techniques to effectively identify correspondences based on these features.

In this paper, we introduce the task of learning to automatically integrate relational databases with Wikipedia. Our contributions are as follows:

1. We present a general method, ETOW, which employs machine learning techniques to automatically resolve relational database entities to Wikipedia pages, using a small number of labeled examples per entity type.

2. In experiments with two distinct databases, we demonstrate that ETOW can effectively resolve thousands of entities to Wikipedia. ETOW is shown to achieve high precision (0.9) on average, at an acceptable level of recall (0.74). Compared with baseline algorithms, ETOW reduces error in terms of F1 score by 19% on average, and reduces false positive rate by 50%.

3. We present evidence suggesting that the integration performed by ETOW can offer substantial improvements to the coverage of both Wikipedia and the database.

The remainder of the paper is organized as follows. We define our task formally in Section 2, and present our system in Section 3. The experimental results are presented in Section 4, and we provide evidence suggesting ETOW's utility in applications in Section 5. Section 6 discusses related work, and the paper concludes with a discussion of future work.

2 Problem Definition

We consider a relational database consisting of a set of *entities* E, *relations* R, and *types* T. Each $r \in R$ is a binary relation over the set of entities.[1] Each entity is of exactly one *type* (analogous to a table in a database implementation). Each type defines a set of *attributes* which have numeric or symbolic values for each entity of the type.

For example, a nutrition database may contain a relation *is_rich_in* which holds between the entities *Dark Chocolate* and *Anti-oxidants*. Further, both *Dark Chocolate* and *Broccoli* may be members of the *Food* type in the database, characterized by attributes such as *Food.calories_per_serving*. Figure 1 shows an example from the company database used in our experiments.

We represent Wikipedia as a set P of pages. Each page $p \in P$ is described by attributes including its title, text, category information, and so on.

Our task is to resolve which Wikipedia page, if any, corresponds to each relational database entity. More formally, we say an entity *matches* a Wikipedia page if the concept described on the page is the same as that represented by the database entity. We then define our task as follows:

Definition 1 *The* **database-to-Wikipedia resolution problem** *is the task of finding a mapping* $\phi : E \rightarrow (P \cup \{\text{null}\})$ *from entities* E *in a given relational database to pages* P *in Wikipedia, such that* $\phi(e)$ *is a Wikipedia page matching the entity* e *if such a page exists, and* $\phi(e)$ *is* null *otherwise.*

Our task definition considers Wikipedia *pages* as the targets of entity resolution. This definition may be too narrow for cases in which database entities refer to concepts described on only a portion of a Wikipedia page (for example, "Dark Chocolate" is described on a portion of the "Chocolate" page). However, as we demonstrate in our experiments, the assumption that entities correspond to individual pages often holds in practice.

[1] Our discussion and experiments focus on binary relations; the extension to relations of higher arity is straightforward.

3 The ETOW system

ETOW, so-called because it maps entities to Wikipedia, solves the database-to-Wikipedia resolution using machine learning. Starting with a small set of seed correspondences, ETOW trains a classifier for each type to estimate whether a given pair $(e, p) \in E \times P$ is a correct match. Below, we describe a set of simplifying assumptions ETOW makes for tractability, and then describe the ETOW algorithm, classifier, and feature set.

3.1 Assumptions

For a reasonably large relational database containing millions of entities, there are *trillions* of potential correspondences between the database entities and Wikipedia's millions of pages. Clearly, narrowing the space of possible matches is required. We employ the following simplifying assumptions:

- We assume that each entity e has a *name* attribute, such that if e matches a page p, then the name of e is the title of p, with the potential addition of disambiguating text in trailing parentheses (as is standard in Wikipedia to denote specific senses of a term, e.g. "Chicago (2002 film)").
- We assume each database entity matches at most one Wikipedia page.

These assumptions dramatically simplify the resolution task, and hold the vast majority of the time in practice. In our experiments, the first assumption reduced the number of matches ETOW considered by more than four orders of magnitude, and was in fact true for more than 95% of the entities. The percentage is so high partly because for entities referred to by multiple distinct names, Wikipedia typically includes redirect pages linking the multiple names to a single, unified page. For example, the page titled "William Henry Gates" redirects to a page titled with the more common name of the founder of Microsoft, "Bill Gates." This practice helps ensure that if an entity e is described on Wikipedia page p, either p or some page redirecting to p will be titled with the name for e employed in the database.

The second assumption held in all cases we examined. By eliminating many potential matches, this assumption improved precision in our experiments considerably.

3.2 Algorithm

The algorithm ETOW follows is shown in Figure 2. ETOW begins by applying the first assumption detailed above to obtain a set C of *candidate matches*: all pairs (e, p) such that the entity e and the page p refer to the same name. ETOW invokes a classifier (detailed below) that assigns a probability to each candidate match. For entities e for which some page p is a greater than τ likelihood match, ETOW applies the second assumption, choosing as the match for entity e the most probable page p according to the classifier. The use of a probabilistic classifier and threshold τ allows ETOW to trade-off precision and recall according to application requirements—we illustrate this capability of ETOW in our experiments.

3.3 Classifier and Feature Set

ETOW employs inductive learning to train the probabilistic classifiers it utilizes to identify matches. In our experiments,

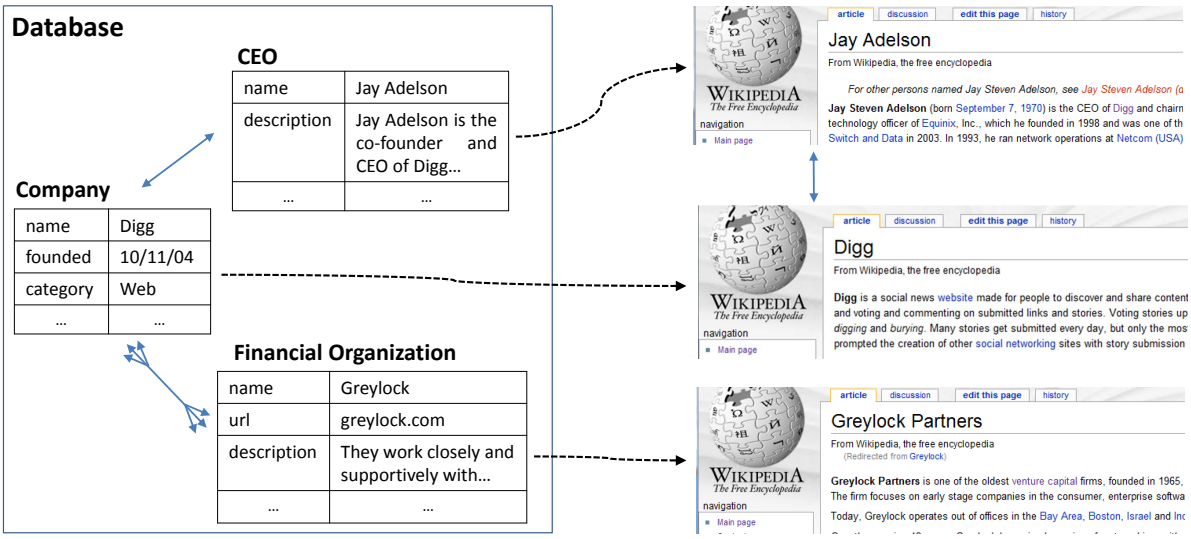

Figure 1: The database-to-Wikipedia entity resolution task. The goal is to obtain links between database entities of various types to their corresponding Wikipedia pages (indicated with dashed lines). Connections between two database entities indicate relationships between the entity types (multiple arrows signify a mapping to potentially many entities); connections between Wikipedia pages indicate hyperlinks.

ETOW(*Pages* P, *Entities* E, *Classifier* Φ_E)
  C = (e, p) such that $p \in P$ is titled with the name of e (modulo trailing parenthetical text)
  for $e \in E$:
    $\phi(e) := \arg\max_p \Phi_E((e, p) \in C)$
    if $\phi(e) < \tau$
      $\phi(e) :=$ null
  output ϕ

Figure 2: Pseudocode for ETOW at run-time. The classifier Φ_E assigns probabilities to candidate matches for entities in E, and the threshold τ is a parameter of the system.

we train a Support Vector Machine classifier for each entity type, using a small number of hand-labeled examples per type. We utilize the libSVM package, configured to produce probabilistic output [Chang and Lin, 2001].

ETOW's classifier estimates the probability that a given pair (e, p) is in fact a correct match. The features for this classification task were chosen based on two primary criteria. First, because ETOW is intended to be widely applicable, the features should be general-purpose and not tied to a specific database or domain. Second, as we wish train the classifier using only a small number of labeled examples, the feature space cannot be too large.

The features we employ for a given candidate match (e, p) are detailed below. Many of the features are computed based on "known" matches of similar types. In the standard ETOW algorithm, the known matches are simply those in the training set; however, as we describe in Section 4.3, the values can also be updated dynamically in an iterative, self-training configuration.

Entity Name/Page Title Features

Ambiguous entity names are disambiguated in Wikipedia page titles through the addition of text in trailing parentheses, as in "Chicago (2002 film)." Although the added text does not follow any consistent standard, it can be informative for identifying matches. For each pair (e, p), we created two features: a binary feature indicating whether p's title has text in trailing parentheses, and a continuous feature measuring the similarity between any parenthetical text of p to that of known matches of entities of e's type. We compute this similarity as the cosine measure between bag-of-words representations of the texts.

We would expect that candidate matches for more obscure or less ambiguous entity names are more likely to be correct. Thus, we include a feature giving the frequency of the entity name on the Web, as estimated from the Google n-grams data set,[2] as well as a feature giving the number of distinct Wikipedia pages titled with the entity name.

Textual Features

For correct matches (e, p), we expect the text of p to include some of e's attribute values or related entity names. Let a *related entity name* of an entity e be all names of entities e' where $r(e, e')$ or $r(e', e)$ occurs for some $r \in R$. We include a feature giving the cosine similarity between e's related entity names and a bag of words representing its attributes, a feature equal to the fraction of e's related entity names that appear in p, and a feature giving the Web frequency of the least-frequent related entity name of e found on p.

[2]For entity names longer than the five word limit of the data set, we estimate the frequency using a five-gram language model.

Relational Features

As shown in Figure 1, it may be the case that the relational structure of the database is reflected in the link structure of Wikipedia. Thus, we include features expressing how well the relational structure corresponds to Wikipedia links. Specifically, for each entity type related to e, we add a feature giving the fraction of matched entities e' related to e for which e''s match has a hyperlink to or from p.

Category Features

Wikipedia's pages are organized into a hierarchical category structure, where each page may be included in an arbitrary number of categories. Although the structure is inconsistent and incomplete, entities of a given type tend to be mapped to similar branches of the structure, generally speaking. We compute a "bag of categories" for each page p consisting of its categories and up to three parent categories. Our category feature is then the cosine similarity between the bag of categories for p and the bag of categories for all known matches of entities of e's type. For category features, we employ TF/IDF normalization in the cosine similarity computation.

Popularity Features

We expect measures of the *popularity* of the Wikipedia page p and the entity e to be informative for the classification of a candidate match. More popular database entities are more likely to have a corresponding Wikipedia page. Also, the popularity of an entity and its corresponding page should exhibit some correlation. As direct popularity information is not externally available, we use surrogate measures. For the media products database, we treat the *sales rank* attribute as a measure of an entity's popularity; for the companies database we use the string length of the database's content describing the entity. We approximate the popularity of a Wikipedia page p by the number of Wikipedia pages linking to p.

4 Experiments

In this section, we describe experiments measuring ETOW's effectiveness in the database-to-Wikipedia resolution task for two distinct databases. We begin by describing our data sets, and then present our results.

4.1 Data Sets and Experimental Setup

We experimented with two distinct relational databases. The first, *media products*, is derived from the Amazon.com product database. Our version of the database included three types: a *products* type consisting primarily of recordings and films; an *artist* type of contributors to the products (bands, composers, actors, directors, etc.); and a *track* type representing each recording's individual tracks. The second database, *companies*, is a subset of CrunchBase, an online database of information on companies and their funding sources. As illustrated in Figure 1, the three entity types in this database were *companies*, *CEOs*, and *financial organizations*.

The media products database consisted of about 961,000 products, 390,000 artists, and 7.8 million tracks. From this large database, we sampled a set of about 2,400 products which had at least one candidate Wikipedia page match. These products and their associated artists and tracks comprised a working set for the products database, which we employ in our experiments. All feature values for the media products experiments were computed relative to this working set of entities. The companies database consisted of about 15,000 company entities, 1,700 financial organizations, and 4,000 CEOs; we used this database in its entirety.

To obtain training and test data, we selected a set of 40 entities of each type from the media products database, and 80 entities of each type from the companies database, with the requirement that each entity have at least one candidate match in Wikipedia. We hand-labeled the resulting candidate matches, producing a data set of 720 labeled match candidates for the media products database, of which 67 were correct, and 346 match candidates for the companies database, of which 157 were correct. Thus, our experiments test ETOW on two data sets with very different characteristics, as seen in the fraction of correct candidate matches (0.09 for the media products database, vs. 0.45 for companies) and the degree of ambiguity (6 possible matches per entity on average for media products, vs. 1.44 for companies).[3]

In our experiments, we measure performance via five-fold cross-validation over the labeled data (thus, training sets are relatively small—32 or 64 hand-examined entities per type). We partition the candidate matches by database entity, meaning that the same database entity never appears in the training set and the test set at the same time.

We compare the performance of ETOW with two intuitive baselines. The first, *All Exact*, marks a candidate match as positive *iff* its title exactly matches the name of the entity (i.e., the Wikipedia page title has no trailing parenthetical text). The second, *All Unambiguous*, marks a candidate as positive *iff* it is an exact match *and* the entity name is unambiguous in Wikipedia (note that this baseline can still generate false positives, because frequently the database refers to an entity other than the one appearing in Wikipedia).

We used a Gaussian kernel for the SVM employed in ETOW, with parameters chosen via grid search and 2-fold cross-validation on the training set. The parameter τ is set to 0.5 (placing equal emphasis on false positives and false negatives) unless indicated otherwise.

4.2 Results

We first investigate how well ETOW performs in the database-to-Wikipedia resolution task relative to baseline techniques. The results of this experiment are shown in Table 1. We measure performance in terms of accuracy (the fraction of candidate matches correctly classified as positive or negative), precision (the fraction of positively classified candidates that are in fact correct), recall (the fraction of correct matches that are classified as positive) and F1 (the harmonic mean of precision and recall).

The results indicate that ETOW is substantially more effective than the baseline methods in both domains. In terms of F1, ETOW reduces error (deviation from 1.0) by 23% over the best performing baseline on the media products data, and

[3] On average, Wikipedia contains 1.05 distinct pages per concept name, so both databases exhibit above-average ambiguity.

	Media Products				Companies			
	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
All Exact	0.543	**0.657**	0.595	0.918	0.702	**0.962**	0.812	0.798
All Unambiguous	0.722	0.582	0.645	0.941	0.816	0.904	0.858	0.864
ETOW	**0.930**	0.597	0.727	**0.958**	**0.864**	0.892	**0.878**	**0.887**
ETOW + self-training	0.911	0.612	**0.732**	**0.958**	0.842	0.917	**0.878**	0.884

Table 1: Performance of ETOW on the database-to-Wikipedia resolution task. ETOW outperforms the baselines by a substantial margin on both data sets, reducing error in terms of F1 by an average of 19%, and false positive rate by an average of 50%, when compared with the best performing baseline. Self-training has only a minor impact on performance.

by 14% for the companies data, for an average error reduction of 19%. On both data sets, neither baseline has a level of precision that is likely to be high enough for many data integration applications; ETOW improves on this substantially, reducing the false positive rate of the most precise baseline by an average of 50% across the two data sets.

As noted in Section 5, different applications may have very different requirements for precision and recall. The second question we investigate is whether ETOW can be used to cater output toward high-recall or high-precision performance, by manipulating the probabilistic threshold τ. As shown in Table 2, the precision of ETOW does in fact increase markedly if we increase τ to 0.95, at the cost of some recall. When we lower τ to 0.05, we see that recall increases greatly at the cost of some precision.

	Media Products		Companies	
	Precision	Recall	Precision	Recall
τ=0.95	0.969	0.463	0.895	0.108
τ=0.05	0.594	0.851	0.726	0.994

Table 2: Performance of ETOW when varying the classification threshold τ. The precision or recall of ETOW can be increased substantially as τ varies.

4.3 Enhancement to ETOW: self-training

Several of ETOW's features for a given candidate match become more informative when other matches are known. For example, the relational features for a candidate match (e, p) are only helpful when matches for entities related to e are known. Similar are the categorical and title-based features that compare aspects of p to other pages known to match to entities of e's type. This suggests a strategy of first identifying the easy-to-classify matches, and using these to inform the classification of the more difficult candidates. As an example, the entity *Catherine Zeta-Jones* is unambiguous and easy to match. We would like to leverage this easy match to help us find the correct Wikipedia page for more difficult-to-match entities related to Zeta-Jones, like the ambiguously-named 2002 film "Chicago." This strategy seems promising, because the correct matching page ("Chicago (2002 film)") is indeed linked with the Zeta-Jones page, whereas the page for the city of Chicago, for example, is not.

We incorporate this intuition in ETOW using a semi-supervised self-training approach. After training ETOW on the training set, we apply the system to the unlabeled candidate matches. Those candidate matches with probability greater than δ are added as positive training examples, and those with probability less than $1 - \delta$ are added as negative examples, where δ is a parameter of the system. We then re-compute the feature values using the new matches, and re-train the classifier on the new features and augmented training set. After repeating this process for k iterations, we measure performance on the test set.

The results of this enhancement are shown in the bottom row of Table 1, using values of $\delta = 0.95$ and $k = 10$.[4] Self-training does *not* provide substantial improvement, on average. Overall performance is essentially unchanged, with recall increasing somewhat and precision falling.

We believe the reasons self-training does not improve performance are two-fold. First, any benefits from exploiting similarities between the Wikipedia hyperlink structure and the relational database structure are to a large degree obviated by the textual features we employ: when a page links to a related entity page, the anchor text is typically the related entity's name. Second, the feature space is small enough that the original labeled training data is relatively representative, so the additional training examples produced by self-training are less beneficial. Exploring self-training with larger feature spaces is an item of future work.

5 Applications

In this section, we investigate ETOW's value in applications. We illustrate how the integration performed by ETOW can provide new capabilities for online databases, and improve the coverage of both the databases and Wikipedia.

ETOW offers a number of possibilities for enhancing online relational databases. By augmenting the relational database with links to Wikipedia pages, or by directly harvesting Wikipedia links or content, we could dramatically improve the generality of a database's content and search capabilities. Extrapolating from our experiments with the high-precision version of ETOW, we estimate that the system is able to correctly identify matches for 13,000 artist entities in the media products database, at precision of approximately 0.97. Further, we find that the matching Wikipedia pages for the entities contain many outgoing links, for an average of 56 per page. The linked pages cover a multitude of topics not found in the database, such as the artist's educational background and related artistic movements. The ability

[4] In previous experiments, adjusting the threshold or the number of iterations did not result in substantial changes in performance.

to search a database based on these general relationships—retrieving all recordings by artists linked to one's hometown, for example—would offer an enriched user experience.

"Infoboxes" on Wikipedia pages list relevant attribute/value pairs in an organized format. Recent efforts have attempted to increase the coverage of infoboxes automatically using text extraction [Wu *et al.*, 2008]. Could ETOW be employed for the same task? In measurements with the companies database, we find this approach holds remarkable promise. For CEOs, the majority of the Wikipedia pages (64%) do not contain infoboxes at all, and in each of these cases an infobox could be created containing at least one attribute from the database. For companies and financial organizations, infoboxes are more common, and contain between 6-7 attributes on average. Many infoboxes are missing particular attribute values, and we find that in the correspondences identified by ETOW, the database information can augment the infoboxes with 2.6 values for financial organizations on average, and 2.5 values for companies, for an average of a 40% improvement in coverage.

ETOW also detects database entities that are *not* currently found in Wikipedia (i.e., those with $\phi(e) = \text{null}$). In these cases, the database information can be used to generate high-quality "stub" pages. Based on a random sample of existing Wikipedia pages, we expect that the stub pages generated from database information would be at least as comprehensive as that of 60% of existing CEO pages, and 53% of company pages. To avoid creating duplicate pages, for this task we employ the high-recall version of ETOW (with $\tau = 0.05$); for the companies database, this approach can create thousands of stub pages with a duplicate rate of less than 5%.

Lastly, the infoboxes in the Wikipedia pages ETOW identifies as matches can improve the coverage of the database. For the companies database, the Wikipedia infoboxes contain multiple fields—e.g., net worth for CEOs, or revenue for companies—not found in the database. For the matches ETOW identified, we found an average of 1.9 values per infobox that could be added to the database.

6 Related Work

To our knowledge, this work is the first attempt to automatically integrate relational databases with Wikipedia. Recent work aimed at integrating Wikipedia with the Cyc ontology provides strategies for a disambiguation problem similar to ours [Medelyan and Legg, 2008]. However, that work targets the Cyc common-sense ontology, in contrast to our goal of a general architecture for integrating relational databases with Wikipedia.

Section 5 establishes the potential value of ETOW for automatically augmenting Wikipedia infoboxes. Another strategy for this a task is to extract information from text on the Web [Wu *et al.*, 2008]. Our work is complementary to this approach. ETOW augments Wikipedia using relational databases, which (even when online) are often not amenable to extraction methods that detect assertions in running text, like those employed in [Wu *et al.*, 2008]. Databases also offer higher precision than that of current extraction techniques.

Recent efforts to automatically construct a database from the information in Wikipedia infoboxes [Auer and Lehmann, 2007] suggests an alternative strategy for integrating databases with Wikipedia: first construct a database from Wikipedia infoboxes, and then apply well-studied methods of database integration (see e.g., [Doan and Halevy, 2005]). In contrast to this strategy, ETOW can be applied in the many cases in which no Wikipedia infobox is present. Lastly, in contrast to recent efforts toward linking mentions of concepts in text to their corresponding Wikipedia page [Milne and Witten, 2008; Cucerzan, 2007], our focus is on integrating relational databases, rather than textual content.

7 Conclusions and Future Work

In this paper, we presented ETOW, a general-purpose mechanism for integrating relational databases with Wikipedia. ETOW uses machine learning techniques to identify correspondences between database entities and Wikipedia pages. In experiments with two distinct databases, ETOW was shown to outperform baseline techniques.

ETOW and related research efforts present exciting possibilities for utilizing Wikipedia to perform large-scale semantic integration of online databases and Web content in general. In future work, we plan to experimentally investigate applications of ETOW and evaluate on additional data sets. We also plan to investigate active learning techniques, which offer the promise of improved accuracy while maintaining ETOW's limited need for human-annotated input.

References

[Auer and Lehmann, 2007] Sören Auer and Jens Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *Proc. of ESWC*, 2007.

[Chang and Lin, 2001] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[Cucerzan, 2007] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proc. of EMNLP*, 2007.

[Doan and Halevy, 2005] AnHai Doan and Alon Y. Halevy. Semantic-integration research in the database community. *AI Mag.*, 26(1):83–94, 2005.

[Gabrilovich and Markovitch, 2007] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of IJCAI*, 2007.

[Medelyan and Legg, 2008] O. Medelyan and C. Legg. Integrating cyc and wikipedia: Folksonomy meets rigorously defined common-sense. In *Proc. of WIKIAI*, 2008.

[Milne and Witten, 2008] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proc. of CIKM*, 2008.

[Wu and Weld, 2008] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proc. of WWW*, 2008.

[Wu *et al.*, 2008] Fei Wu, Raphael Hoffmann, and Daniel S. Weld. Information extraction from wikipedia: moving down the long tail. In *Proc. of KDD*, 2008.

Leveraging Social Networking Sites to Acquire Rich Task Structure

Yolanda Gil, Paul Groth and Varun Ratnakar
Information Sciences Institute
University of Southern California
gil,pgroth,varunr@isi.edu

Abstract

Task management is a core part of knowledge work. However, intelligent assistance for task management is hampered be the lack of large amounts of structured knowledge about user tasks. In this paper, we present a novel approach, Social Task Networks, for obtaining rich user contributed task information by integrating task management with social networking sites.

1 Introduction

A central part of knowledge work is the collection, assignment, sharing, tracking and scheduling of tasks. Such *task management* activities are performed using a variety of tools, most commonly to-do lists, calendars and email. Both commercial companies[1] and the AI community have developed systems to extend these tools to provide intelligent assistance for task management [Myers, K. et al. 2007] These systems take advantage of both the content of task artifacts but also their structure [Kushmerik, N., et al. 2005]. For example, a system may analyze the text of an email but will also derive information from how the text is organized into sender, receiver, subject, and body fields. Indeed, this structured information can make the development intelligent assistants significantly easier.

However, the structure available to intelligent assistants is often limited because of the nature of the task management tool being used. For example, the explicit structure in to-do lists, the most popular task management tool [Jones, S.R. et al. 1997], is limited to the order in which the entries occur, which we note is often not meaningful. In addition to this lack of structure, task artifacts are often not collated either through tool limitations or isolation of the data in personal repositories (notepads, sticky notes or individual email accounts). This limits the information that systems can leverage.

Thus, our goal is to investigate mechanisms to enable task management using richly structured user contributed task knowledge. To achieve this, we have designed a new approach to task management, termed a Social Task Network, that combines ideas from task representations in hierarchal planning, scripting in distributed environments and sharing in social networking sites. This approach allows the acquisition of knowledge about how tasks are situated in a social network, the hierarchical organization of tasks in real world settings and the ability for particular tasks to be automated. In this paper, we detail this new approach to task management and how it will enable the acquisition of richly structured task knowledge. The approach is grounded in a study of to-do lists and an initial prototype system.

2 The Social Structure of Tasks

Tasks are inherently collaborative. Whether scheduling a meeting, writing a portion of a document for a colleague, or asking a family-member to pick up milk at the grocery store, tasks often require the interaction of multiple people in order to be accomplished. Indeed, a central part of task management is the tracking of how tasks have been delegated and shared.

To confirm this intuition, we performed an analysis of a corpus of to-do list items gathered during CALO, a large project to develop intelligent assistants for office-related tasks. The corpus of 1200 to-dos was gathered over a period of several months from a dozen users. [Gil and Ratnakar, 2008], present a detailed analysis of the corpus. Here we revisit the corpus focusing on collaboration.

We manually checked the entire corpus and found that 17.5% of the to-dos were collaborative in nature. Either the to-do referred to a task to be accomplished for another person (e.g. read John's document), scheduling a meeting (e.g. discuss program with Mark), or assigning a task to another person (e.g. email Mary about John). While 17.5% is a significant percentage of the tasks, we believe that this understates the number of collaborative tasks. From our observation, it seems that many tasks, while not specifically identifying collaborators, are sub-tasks of larger collaborative tasks. For example, background reading necessary to participate in a meeting.

Thus, given the collaborative nature of tasks, social relationships provide a key structure for tasks. However, this social structure is not explicit in common task management tools such as to-do lists and calendars. Social relationships are more evident in email through addresses and have been taken advantage to create task management interfaces [Bellotti V., et al. 2003]. However, email addresses do not capture the nature of the social relationships or information about the participants themselves. Groupware systems do a

[1] http://www.reqall.com/

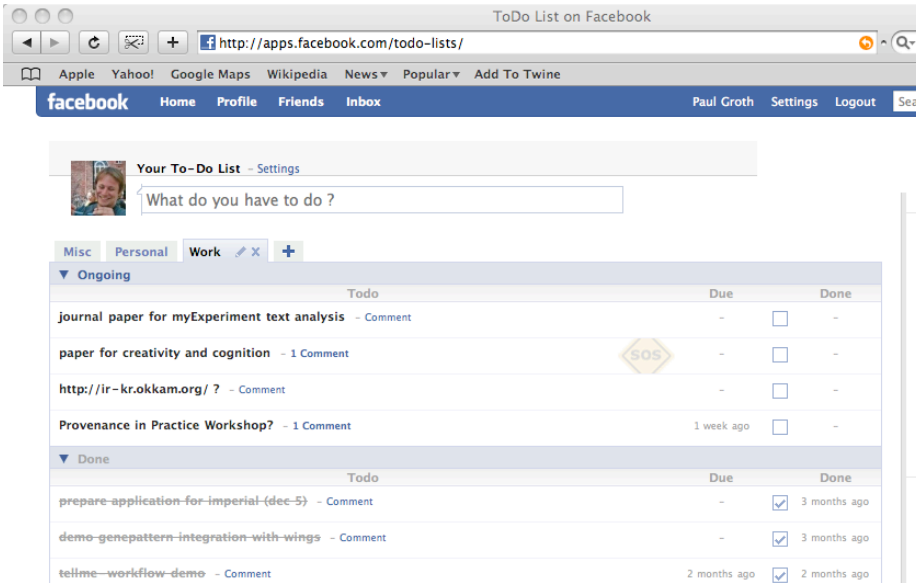

Figure 1: To Do List Facebook Application

better job of explicitly capturing social relationships but are limited to the particular organizations that adopt them.

To address this lack of social relationship information, we decided to integrate task management with a social networking site. We present our initial prototype in the next section.

3 Task Management in a Social Network

As previously mentioned, to-do lists are probably the most widely used task management tool. Therefore, we developed a To-Do List application for Facebook, a very popular social networking site. The application is instrumented to collect data from actual use. We obtained preliminary feedback from a small user group. This application was released early December, and advertised to gather a substantial user base. Currently, there are 97 monthly active users. The application is accessible at http://apps.facebook.com/todo-lists/. A screenshot of the interface is shown in Figure 1.

The application provides standard to-do features including setting dates, priorities, categories and comments. Beyond these features, we have also added Facebook specific features for sharing to-do list entries. Users can share their to-do lists with everyone on Facebook, people just in a particular network (i.e. larger organizations of people like a university), group or with only their friends. Additionally, users can post what we term a "SOS", which is a broadcast to all the users friends that they need help with a particular task. Anecdotally, these sharing features have been of real use to the users. In particular, we have seen users organize everything from evenings out to computer LAN parties. Additionally, we have received requests for more sharing features such as the following:

> "Is there any way to possibly choose more selectively who can see each item in the to do list? I am thinking about surprise parties and secret meetings, and right now its either everyone or no one. [sic]"

Thus, the sharing features enabled by Facebook are a core feature set for attracting users. Moreover, they allow us to unobtrusively gather detailed information about how users share tasks and connect that to profiles of the user. For each to-do entry, we maintain the Facebook id of the user and can thus access pertinent user profile information, everything from the age of the user, to their list of friends, and interests. We also maintain which users comment on other users shared to-do entries to track which users are actively sharing tasks.

Thus, this prototype was a good first start towards collecting rich task knowledge structured around social relationships. It also revealed a great deal about the kinds of task people jot in their to-do lists, and about the potential for automatic assistance.

- Many tasks had very coarse granularity. Examples are "Get Christmas presents", "Study nursing", and "Get back in shape". These are high-level tasks that involve many substeps and activities. There are no concrete first steps enumerated, which would be useful for a user to get started on the overall goal expressed in the to-do.
- Many entries were in other languages.
- Many tasks that were concrete had only some aspects that could be automated. For example, "Write Christmas cards" or "Read Dune" could involve some on-line purchasing that could be automated but most of the activity was meant to be done by the person themselves.
- Some tasks could be fully or mostly automated. Examples include "Renew my driver's license", "Buy iPhone", "Rent Aliens movie".

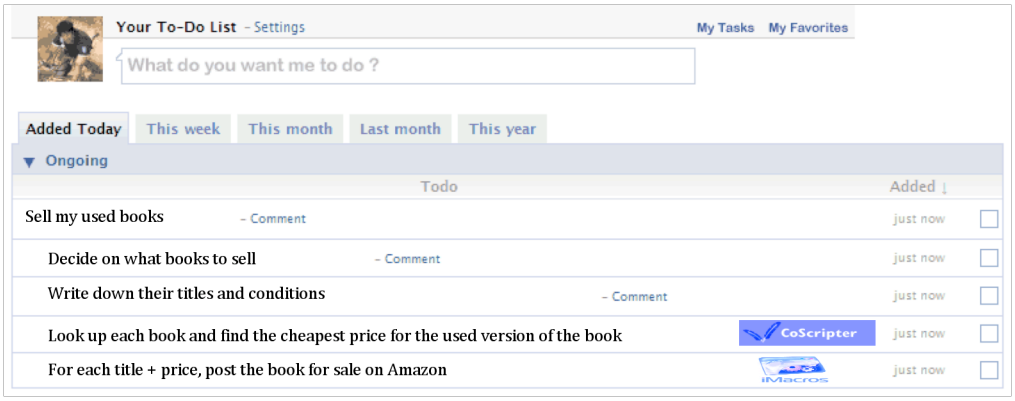

Figure 1: Redesigned To-Do List Facebook Application

- Many tasks were not amenable to automation, for example, "Go to the mall".
- Some tasks could be accomplished by friends of the users. For example, "Get Volunteers for a Meal".

With these lessons in mind, we devised a new approach to personal task management: "social task networks".

4 Social Task Networks

In social task networks to-do lists and other task artifacts are organized around an explicit social network. Tasks are decomposed into subtasks with enough detail to allow tracking and sharing in the network. More specifically:

- To-dos (i.e. tasks) are organized hierarchically and are described in terms of their constituent subtasks. The set of to-dos that are active at any given time are often part of enveloping tasks. This will allow users to express tasks at coarser and finer granularity, giving the task a high level coarser description when it is first jotted down and later drilling down into details
- Tasks include both automatable steps and non-automatable steps. The latter provides context for the user and serves as a reminder that the task is pending their attention and is up to them and not the system.
- To-dos should be assignable so that assistance in terms of automation can be provided by other individuals in the social network. For example, a project assistant may provide the maximum allowable amount to spend on a new laptop purchase, which may be just one step in the overall task of purchasing a new computer. Other users can decline assigned tasks, but if they accept the user should have visibility about its status.
- To-dos should be shareable so that assistance may be provided by other individuals in the social network. For example, if a to-do entry is to find a hotel in Washington DC, someone else may have a list of favorites that they are willing to suggest.
- To-do decompositions should be shareable, so that know-how can be shared. For example, if someone is looking for job announcements someone else may have just looked and have a task description to share: a set of steps that they followed searching diverse web sites and mailing different individuals.

Thus, social task networks not only provide the context of a social network but also explicitly represent the hierarchal nature of tasks as well as the possible mixture of automated and non-automated tasks. Figure 2 shows a redesigned version of our To-Do List application that follows the aforementioned desiderata. In particular, the design supports hierarchal tasks where both automated and non-automatable steps are mixed together. In particular, we show that some steps could be automated through web scripting applications such as IBM's Coscripter or Mozilla's iMacros.

Just as Wikipedia has become a valuable knowledge source for AI researchers, social task networks will provide a powerful new knowledge source for developing intelligent assistance for task management.

Acknowledgements

We gratefully acknowledge funding for this work by DARPA under contract no. NBCHD030010.

References

[Bellotti V., et al. 2003] Bellotti V., Ducheneaut N., Howard M., and Smith I.. Taking email to task: the design and evaluation of a task management centered email tool. In CHI-03, 2003.

[Gil and Ratnakar, 2008] Yolanda Gil and Varun Ratnakar. Proceedings of the *2008 International Conference on Intelligent User Interfaces (IUI-2008),* January 2008.

[Jones, S.R. et al. 1997] Jones, S. R. and P. J. Thomas. "Empirical assessment of individuals personal information management systems." *Behaviour & Information Technology* 16(3), 1997.

[Kushmerik, N., et al. 2005] Kushmerick, N. and Lau, T. 2005. Automated email activity management: an unsupervised learning approach. In *Proceedings of the 10th international Conference on Intelligent User Interfaces* January 2005.

[Myers, K. et al. 2007] Myers, K., P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine*, 28(2), 2007.

Compact Hierarchical Explicit Semantic Representation

Sonya Liberman and Shaul Markovitch
Department of Computer Science
Technion - Israel Institute of Technology, 32000 Haifa, Israel
{sonyal,shaulm}@cs.technion.ac.il

1 Introduction

It has long been recognized that in order to process natural language, computers require access to vast amounts of common-sense and domain-specific world knowledge [8]. The method of Explicit Sematic Analysis (ESA) [3; 4] incorporated natural concepts derived from Wikipedia to represent the meaning of words and texts, thus explicitly using knowledge defined and manipulated by humans. Results achieved on semantic relatedness tasks using ESA were highly correlated with human judgements emphasizing how beneficial external world knowledge is.

However, in addition to using background knowledge in the form of independent natural concepts, humans use another innate ability, extremely beneficial for judging semantic relatedness, the ability to *generalize*. For instance, correctly identifying the relation between the words *cat* and *dog* requires a process of generalization followed by a recognition of an intersecting higher level concept, *Pets* or *Animals*.

ESA represents semantic meaning as a flat *interpretation vector* in the multi-dimensional concepts space. Consider the top twenty concepts generated by ESA for the word *cat*: 1. CAT (UNIX) 2. CHESHIRE CAT 3. COOL CAT 4. PLASAN SAND CAT 5. CLAUDE CAT 6. BIG CAT 7. STRAY CATS 8. FELIDAE 9. CAT'S EYE (FILM) 10. CAT SCRATCH FEVER 11. SABER-TOOTHED CAT 12. NEW BRITAIN ROCK CATS 13. CATS (MUSICAL) 14. CATS & DOGS 15. CLAN NOVA CAT 16. CAT ON A HOT TIN ROOF 17. SACRAMENTO RIVER CATS 18. WILDCAT 19. JUNGLE CAT 20. LEOPARD CAT

This example demonstrates several drawbacks in constructing a flat semantic representation. First, observe that 13 out of the top 20 concepts are not related to the core meaning of the word *cat*, as they describe plays, films, rock bands etc.

Moreover, the representation contains overly specific concepts such as NEW BRITAIN ROCK CATS, a minor league baseball team, that add noise. Finally, ESA assumes concept independency while obviously, the concept BIG CAT is related to WILDCAT and LEOPARD CAT.

These observations emphasize the importance of generalization and motivate the construction of a semantic interpreter which incorporates the inner structure of Wikipedia-based concepts. This is particulary important within the framework of semantic relatedness tasks where a high-level conceptual recognition is often required.

In this work, we present a novel approach called CHESA[1], for representing meaning within a hierarchical compact structure, capturing semantics at different levels of specificity. This is done by incorporating additional information from Wikipedia categories graph. Instead of an *interpretation vector* constructed with ESA, our method represents meaning through an *interpretation rooted directed acyclic graph* where every node is assigned with an association score.

2 Compact Hierarchical Representation

Wikipedia articles are part of a large taxonomy-like structure imposed by a network of Wikipedia categories, where leaf nodes correspond to articles and inner nodes correspond to categories.

We convert the Wikpedia category graph into a simpler form of multiple-inheritance hierarchy. We do so by traversing it starting from the MAIN TOPIC CLASSIFICATIONS category in a BFS order. When a category is reached we add it to the hierarchy unless it was already included into the hierarchy through a shorter path from the root. All reached articles are automatically added to the representation.

Let $\mathcal{C}$ be the complete set of nodes in a given hierarchical taxonomy. Let $c : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ be a function from a node to its children. Let $l : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ be a function from a node to the set of all leaves reached from it, defined directly by c. Let $r \in \mathcal{C}$ be the root node of the hierarchy. We denote by $H(\mathcal{C}, c, r)$ a multiple-inheritance hierarchy with a set of nodes $\mathcal{C}$, children mapping function c and a root r.

Now, let $\mathcal{T}$ be the set of all terms in the collection of articles corresponding to leaves in the hierarchy, and denote by $count(t, n)$ the number of times the term t appears in a node n. Since only leaf nodes have $count(t, n) > 0$ for some t, we define the term count for inner nodes as $count(t, n) = \sum_{\ell \in l(n)} count(t, \ell)$. We denote by $tf(t, n)$ the term frequency of the term t in node n, defined by $tf(t, n) = count(t, n) / \sum_{t' \in \mathcal{T}} count(t', n)$.

In the following, we present two strategies for constructing compact hierarchical representations as well as an algorithm for constructing semantic interpretations of any given granularity level. All three algorithms receive a term t and a hierarchy $H(\mathcal{C}, c, r)$ as input.

[1] Compact Hierarchical Explicit Semantic Analysis

The TD-CHESA algorithm, performs a DFS top-down traversal over the nodes of the hierarchy. The algorithm starts with a representation containing the root r only. When a node n is reached, for every child $d \in c(n)$ the algorithm applies Pearson's Chi-square test to determine whether d is significantly more associated with t than n. If so, d is included in the representation and DFS traversal proceeds within the subtree of d. Intuitively, TD-CHESA performs specification by demand, namely, it starts by representing the word on a very high conceptual level, and iteratively reveals more specific meanings of the word, until it reaches the point where a more specific description does not contribute to the semantic interpretation of the word.

The BU-CHESA algorithm, starts by including all nodes in $\mathcal{C}$ in the representation. Then, it iteratively prunes nodes starting from the leaves, bottom-up, according to the aforementioned statistical test. If a node is not pruned, all its ancestors automatically remain in the representation.

It is evident that in many cases the two algorithms produce different representations for the same term. TD-CHESA produces more compact representations and usually focuses on the primary meanings of the word, efficiently filtering out over-specific concepts even if their association score is high. For instance the concepts PLASAN SAND CAT and STRAY CATS for the word *cat* are filtered out by that strategy. BU-CHESA trims the representation hierarchy at places where generalization is suitable. For example, the concepts BIG CAT, WILDCAT and LEOPARD CAT are trimmed by the bottom up strategy while their mutual category FELINES remains.

Often, we need to represent semantics in various levels of granularity due to time and space efficiency and comprehensibility needs. We present an iterative algorithm, K-CHESA that, given a size k, constructs a compact hierarchical semantic interpretation with at most k nodes.

Let Q be a priority queue of hierarchy nodes. The algorithm begins with a representation containing the node r only. Then for every child $d \in c(r)$, it computes the significance score of d with respect to r, and inserts it into a Q. Now, let p be the node which is currently placed at the front of Q. Then p is included in the representation and is removed from Q, while $c(p)$ are inserted into Q. The algorithm stops when the representation reaches the size of k, or once Q is empty. Figure 1 shows the representation of the word *cat* with $k = 20$.

For all three methodologies, once a representation is constructed for a given term, we assign association scores to all nodes in the representation. We define the association score of a term t with a node n within its hierarchical representation as $s_{t,n} = \log \frac{tf(t,n)}{tf(t,r)}$ when r is the root of the hierarchy.

Our first usage of CHESA is for computing semantic relatedness between words. Intuitively, two words are related when their hierarchical representations are similar with respect to structure, having high association scores for intersecting concepts. We define the following relatedness score, inspired by the cosine similarity metric for vector representations:

$$rel(t_1, t_2) = \frac{\sum_{n \in \mathcal{C}} s_{t_1,n} s_{t_2,n}}{\sqrt{\sum_{n \in \mathcal{C}} s_{t_1,n}^2} \sqrt{\sum_{n \in \mathcal{C}} s_{t_2,n}^2}}$$

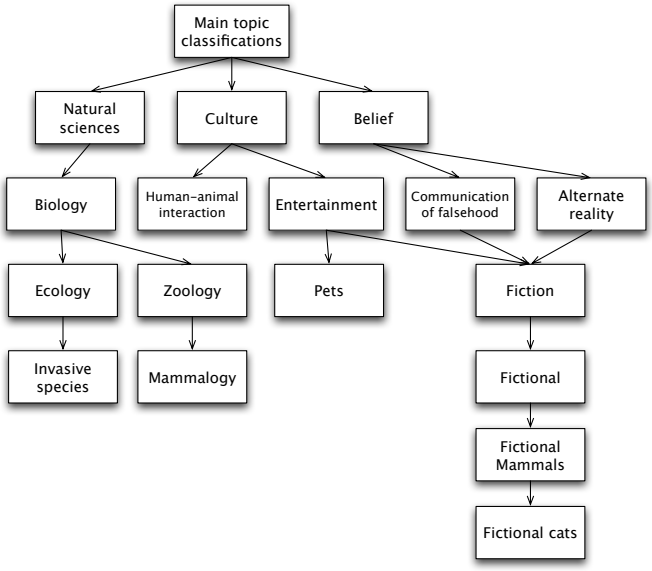

Figure 1: CHESA Representation of *cat* with $k = 20$

3 Preliminary Evaluation

We implemented our CHESA approach using a Wikipedia snapshot as of October 18, 2007. After parsing the Wikipedia XML dump, we removed small and overly specific articles as described in [3] as well as several types of categories such as lists, stubs, and categories which names contained a specific year. At the end of that process 497,153 articles and 125,542 categories were remained. The text of the articles was processed as described in [3] and term counts were computed for every term and article.

We then constructed a hierarchy, based on the Wikipedia category graph, rooted at the MAIN TOPIC CLASSIFICATIONS category and removed all articles and categories not reached from that category. The final hierarchy contained 557,113 nodes.

We evaluated our approach on the the WordSimilarity-353 collection. Spearman rank-order correlation coefficient was used to compare computed relatedness scores with human judgements. We compare our results to ESA results on the same Wikipedia dump as well as to other methods for semantic relatedness computation.

Table 1 shows the results of applying TD-CHESA and BU-CHESA methodologies to estimate relatedness of individual term. The results show that BU-CHESA yields substantially better results than TD-CHESA. Both strategies outperform all presented methods but ESA.

We also compared the performance of K-CHESA with ESA trimmed to top k concepts (with respect to their association scores). Figure 2 shows the correlation coefficients for k between 10 and 500 for the two methods.

The results show that for $k < 300$, K-CHESA is superior to ESA, and for $k > 100$ K-CHESA outperforms other methods presented in Table 1. Interestingly, CHESA is able to capture semantic relatedness (correlation of 0.33) with extremely small representations of size 10.

Algorithm	Correlation
WordNet [6]	0.35
Rogets Thesaurus [5]	0.55
LSA [12]	0.56
WikiRelate! [13]	0.50
MarkovLink [10]	0.55
ESA [3]	0.74
TD-CHESA	0.63
BU-CHESA	0.71

Table 1: Computing word relatedness

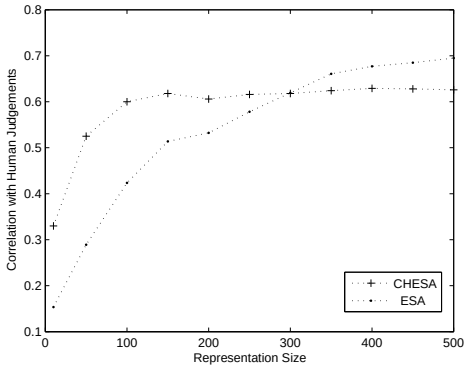

Figure 2: The impact of representation size

4 Related Work

Several previous methods [2; 6; 1; 11; 9; 7] used hierarchical taxonomies to represent the semantic meaning of words. Methods using lexical databases map text words into word senses and use the latter as concepts. They then use various metrics to compute relatedness using the properties of the underlying graph structure.

These methods substantially differ from CHESA. They represent a term by a node in the hierarchy while CHESA uses a hierarchical representation. Consequently, they use different metrics to evaluate relatedness. While CHESA compares the hierarchies representing the terms, the other methods compare nodes using various graph-distance metrics. Also, CHESA, like ESA, has a much richer vocabulary, as it generates its semantics from a large collection of text articles.

WikiRelate! [13] also uses Wikipedia categories to compute semantic relatedness. However, it differs from CHESA as it represents a term by short list of articles containing the term in their title, and computes semantic relatedness based on path distance and information content measures.

5 Conclusions and Future Work

We proposed a novel approach to represent semantic meaning in the form of compact hierarchical structure derived from natural concepts and the taxonomy imposed on them by the Wikipedia category system. Our approach allows description of semantics in varying specification levels, capturing relations between concepts and having the ability to generalize. This is in contrast to the Explicit Semantic Analysis approach which constructs a flat vector representation. We observed that the compact representations constructed by CHESA are much more intuitive and comprehensible than those constructed by ESA. Empirical evaluation confirms that using CHESA leads to substantial improvements in computing word relatedness when representation size is limited. It shows CHESA's ability to capture semantic meaning on high conceptual levels. ESA still outperforms CHESA when using its full interpretation vector. We believe that the main reason are deficiencies in the category graph of Wikipedia. We intend to apply more sophisticated preprocessing to amend some of these problems.

References

[1] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI*, pages 805–810, 2003.

[2] A.Budanitsky and G. Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *NAACL*, 2001.

[3] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, January 2007.

[4] E. Gabrilovich and S Markovitch. Wikipedia-based semantic interpretation for natural language processing. *JAIR*, 2009.

[5] M. Jarmasz and S. Szpakowicz. Roget's thesaurus and semantic similarity. *RANLP*, 2003.

[6] M. Jarmasz. Roget's thesaurus as a lexical resource for natural language processing. Master's thesis, University of Ottawa, 2003.

[7] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *ROCLING*, pages 57–63, 1997.

[8] D. Lenat and R. Guha. *Building Large Knowledge Based Systems*. Addison Wesley, 1990.

[9] D. Lin. Automatic retrieval and clustering of similar words. In *COLING*, pages 768–774, 1998.

[10] H. Thad and R. Daniel. Lexical Semantic Relatedness with Random Graph Walks. *EMNLP-CoNLL*, pages 581–589, 2007.

[11] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130, 1999.

[12] L. Finkelstein and E. Gabrilovich and Y. Matias and E. Rivlin and Z. Solan and G. Wolfman and E. Ruppin. Placing Search in Context: The Concept Revisited. *WWW*, pages 406–414, 2001.

[13] M. Strube and S. P. Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. *IJCAI*, July 2006.

WikiSLE: Mapping Wikipedia infobox information onto the article text

Sedat Gokalp **Syed Toufeeq Ahmed** **Suvitha Vijayarajan** **Hasan Davulcu**
Department of Computer Science and Engineering
Arizona State University
{sgokalp, toufeeq, svijaya4, hdavulcu}@asu.edu

Abstract

Great amount of user-contributed information published on Wikipedia[1] makes it a good source for knowledge and fact corpus. The unstructured text (articles text) and marked labeled data (infoboxes) can be used together as a corpus for variety of Artificial Intelligence and Machine Learning tasks, when the mapping between infobox (attribute-values) and the article text is done. This paper presents WikiSLE (Wikipedia Semantic Label Extractor) along with the results and extracted dataset (made publicly available) that maps infobox information onto the article text.

1 Introduction

With an extensive set of articles numbering almost 2.7 million today, Wikipedia is one of the major user-contributed knowledge sources. The articles contain detailed information about the entities in natural language, and the infoboxes contain semi-structured information. We observed that the information in the infoboxes also occur in the plain text of the articles in an unstructured nature. Therefore, it is possible to mark up the infobox information in the article plain texts. For example, the first sentence of the article for Chris Buttars is *"Chris Buttars (born April 1, 1942) is a Republican state senator for Utah representing senate district 10."* and the Wikipedia infobox contains the information *"Utah State Senator"* (Figure 1). It can be seen that it is possible to tag the exact occurrences of the words *"state senator"* and *"Utah"* in the article plain text as the label of the entity. This can be done by basically finding the matching words on the plain text and the infobox label. We further extend this idea by finding the name phrase chunks in the plain text and then try to find the possibly non-exact matches. In our example, this process would give us *"a Republican state senator for Utah"* as the *semantic label* for *Chris Buttars.*

Large-scale information extraction for web is an active research area. Web information extraction systems such as KnowItAll [Etzioni et al., 2004] extract facts and relationships from text using set of extraction rules for each class and relation from a set of generic, domain independent templates. Pasca et al. [2006] utilized generalized contextual extraction patterns to extract large amount of facts (of type person-born-in-year) from web. Turney [2001] used large number of mentions of entities on web as a measure to recognize synonyms. With exponential growth of Wikipedia articles, it is one of the largest sources of general knowledge on the web. And now, it is becoming the center of attention for the researchers to leverage this enormous corpus of facts to train machine learning algorithms, to extract information from, and to utilize for word sense disambiguation (WSD). For most of these tasks, the Wikipedia data needs to be preprocessed before it can be used by learners or classifiers. Suchanek et al. [2007] extracted facts (*Is-A* and *hasWonPrize* type) from Wikipedia to enrich YAGO. Wu et al. [2008] and [Wu and Weld, 2007] focused on automatic extraction of Wikipedia infoboxes and attribute values. In [Wu et al., 2008], the focus was extraction of infoboxes for non-infobox articles and automatic link generation for Wikipedia articles. In [Wu and Weld, 2007], training data was generated by finding exact matches of the attributes in the infobox, whereas in our approach we find the matches where words might be added, removed or altered.

Ponzetto et al. [2007] built a large scale taxonomy (*Is-A* relations) by using category system of Wikipedia as a conceptual network. Cucerzan [2007] used web and Wikipedia for disambiguation of named entities, where as Bunescu and Pasca [2006], used Wikipedia as training and testing data-

Infobox Data

```
{{Infobox_Politician
| name = D. Chris Buttars
| birth_date = April 1
| residence = [[West Jordan, Utah|West Jordan]]
| office = Utah State Senator
| term_start = 2001
| term_end = Current
| predecessor = L. Alma Mansell
…
```

Article Text

D. Chris Buttars (born April 1, 1942) is a ***Republican member of the Utah State Senate*** *representing senate district 10…*

Figure 1: Infobox and Article text example

[1] Wikipedia: http://www.wikipedia.org/

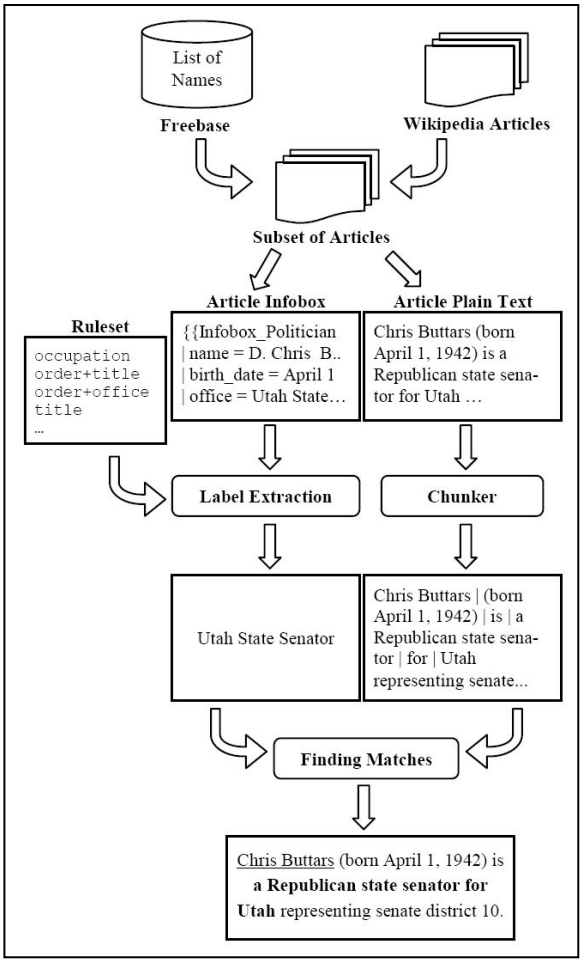

Figure 2: Labeling Entity names with Semantic Labels

sets for SVM kernel classifier to disambiguate named entities.

With the help of large scale automated algorithms making use of the largest corpora created by humans (like Wikipedia), we are getting close to achieving the dream of *Semantic Web* (Berners-Lee et al. [2001]). A step towards this effort is to extract structured information from Wikipedia and make it available to the research community, as shown by the DBpedia[2] group [Auer et al., 2007]. In this work, we follow similar footsteps by providing processed, machine accessible (XML) dataset of entity names (person) and their *Semantic Label*s (occupation or role), and believe that this dataset (along with other similar datasets) will help us realize our goal of connected, semantic web.

In Section 2 of this paper, the methodology is explained in three steps. Section 3 is about experimental evaluation and Section 4 concludes the paper.

2 Extraction from Infoboxes

As can be seen on Figure 2, we first collect a subset of articles from Wikipedia to work on. This subset contains articles about people. For each article, we extract the *semantic label* of the person by using a rule set on the infobox. Then the first paragraph of the article is chunked into noun phrases and the best combination of the chunks that match the *semantic label* is tagged as `sem_label`. The details are explained in the following subsections.

2.1 Populating Input Data

We worked on the 2007 dump of Wikipedia archive which contains more than 400,000 articles about people. We collected the list of person names from Freebase[3], which is an open database of general information organized in the form of categories like movies, people, books, etc. Using Freebase's Metaweb Query Language[4], we extracted a list of person names (840,265 names). Then, we used the JWPL[5] to check if an article exists for a given name and to fetch the corresponding article. The articles with an infobox are then stored in a database for further processing.

2.2 Extracting Labels from Infoboxes

We made a survey on the infobox attributes for the articles about people to make a rule set which would give the *semantic label* for the person. The rule set is shown below:

1- Value of `occupation`
2- Value of `order+title`
3- Value of `order+office`
4- Value of `title`
5- Value of `office`
6- Value of `order`
7- Infobox template name

For the rules 1-6, we search for an attribute on the infobox data. If a value for a rule is found, the system assigns that value as the *semantic label* and ignores the rest of the rules. If none of the attributes given in the rules 1-6 are found, then the template name of the infobox is assigned as the *semantic label* for the article.

2.3 Mapping Labels in the Articles

Once the *semantic label* is extracted from infoboxes, the next step is to find the occurrence of the *semantic label* in the plain text, and tag the occurrence with "`<sem_label>`" and "`</sem_label>`". The baseline approach is to find the exact or similar occurrences of the words by using a distance measure. We used the levenshtein distance[6] algorithm to find the minimum edit distance and used the below similarity formula to find the similar words with a threshold of 80%.

$$similarity(s_1, s_2) = 1 - \frac{levenshtein(s_1, s_2)}{\max(|s_1|, |s_2|)}$$

[2] http://wiki.dbpedia.org/Datasets

[3] Freebase: http://www.freebase.com/

[4] MQL: http://www.freebase.com/view/en/documentation

[5] JWPL: http://www.ukp.tudarmstadt.de/software/jwpl/

[6] http://en.wikipedia.org/wiki/Levenshtein_distance

The *semantic label* may consist of multiple words and the matches in the plain text might be distant from each other. For example, the *semantic label* extracted from infobox for *Chris Buttars* is "*Utah State Senator*" and the article contains the sentence "*Chris Buttars (born April 1, 1942) is a Republican state senator for Utah representing senate district 10*". The matches of the words in the *semantic label* which are underlined in the example are not only distinct from each other, but also incomplete to be tagged as label.

To solve this, we first split the plain text into Noun Phrase chunks using the MontyLingua[7] library and then find the best combination of the chunks to mark as *semantic label*. The example illustrates the process:

Original sentence:
Chris Buttars (born April 1, 1942) is a Republican state senator for Utah representing senate district 10.
Chunked sentence with candidates marked:
Chris Buttars | *(born April 1, 1942)* | *is* | ***a Republican state senator*** | *for* | ***Utah*** | *representing* | ***senate district 10***.
Candidate chunk groups:
Chris Buttars | *(born April 1, 1942)* | *is* | ***a Republican state senator for Utah*** | *representing* | ***senate district 10***.
Labeled Sentence:
Chris Buttars (born April 1, 1942) is ***a Republican state senator for Utah*** *representing senate district 10.*

Chunking helps with finding the range for the `sem_label` tag. Moreover, it yields us to extract more information than the *semantic label* extracted from infobox.

3 Dataset and Evaluation

The extracted dataset is described in Table 1, which can be downloaded at: *http://cips.eas.asu.edu/WikiSLE/*. The output files are in XML format, and are organized as 26 files with approximately 1,000 entries each of: *1) Entity name, 2) Infobox Role 3) Tagged text with semantic labels.* We randomly selected 500 entries from this dataset to check for extraction accuracy, and found that 16 of them had incorrect labels, approximately giving the dataset **97**% accuracy.

Number of articles in Wikipedia	1,660,067
Number of person names in Freebase	840,265
Number of people that has a Wikipedia page	410,148
Wikipedia articles with an infobox	53,713
Articles of which infobox labels were extracted	49,260
Articles of which labels are tagged in plain text	27,168

Table 1: Dataset Description, (extracted from 2007 Wikipedia)

4 Conclusions

In this paper, we presented WikiSLE, which makes Wikipedia a source and a corpus for training and testing AI and Machine learning algorithms in information extraction and WSD domain. Specifically, we presented our approach to map infobox attribute values with entities and *semantic label*s in the article text. We also have made the extracted dataset available to galvanize the research utilizing sources like Wikipedia.

[7] MontyLingua: http://web.media.mit.edu/~hugo/montylingua/

References

[Etzioni et al., 2004] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. 2004. Web-scale information extraction in knowitall: (preliminary results). In Proceedings of the 13th international Conference on World Wide Web, 2004. ACM, New York, NY, 100-110.

[Pasca et al., 2006] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In Proc. of AAAI-2006, 2006.

[Turney, 2001] Turney, P. D. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning* (September 05 - 07, 2001).

[Suchanek et al., 2007] Suchanek, F. M., Kasneci, G., and Weikum, G. 2007. Yago: a core of semantic knowledge. In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 697-706.

[Wu et al., 2008] Wu, F., Hoffmann, R., and Weld, D. S. 2008. Information extraction from Wikipedia: moving down the long tail. In: 14th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Las Vegas, Nevada, USA, August 24 - 27, 2008). KDD '08. ACM, New York, NY, 731-739.

[Wu and Weld, 2007] Wu, F. and Weld, D. S. 2007. Autonomously semantifying wikipedia. In: Sixteenth ACM Conference on Conference on information and Knowledge Management (Lisbon, Portugal, November 06 - 10, 2007). CIKM '07. ACM, New York, NY, 41-50.

[Ponzetto et al., 2007] Ponzetto, S.P. and Strube, M., Deriving a large-scale taxonomy from wikipedia. In: AAAI, pp. 1440-1445.

[Berners-Lee et al., 2001] Berners-Lee, T., J. Hendler & O. Lassila (2001). The semantic web. Scientific American, 284(5):34-43.

[Cucerzan, 2007] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In EMNLP-CoNLL '07, pages 708--716, 2007.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z., DBpedia: a nucleus for a web of open data. In: The 6th International Semantic Web Conference (ISWC 2007).

[Bunescu and Pasca, 2006] R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In: 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), pp. 9--16, Trento, Italy, April 2006.