Beginning to Understand Unstructured, Ungrammatical Text: An Information Integration Approach

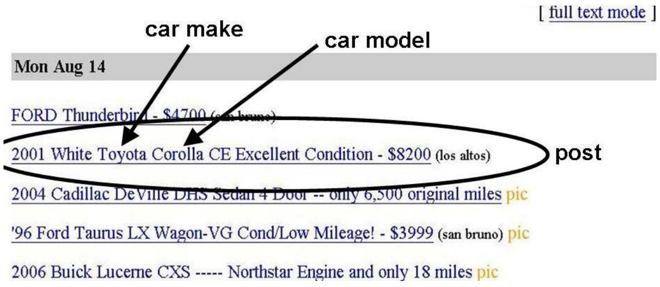
Matthew Michelson and **Craig A. Knoblock** *
University of Southern California
Information Sciences Institute,
4676 Admiralty Way
Marina del Rey, CA 90292 USA
{michelso,knoblock}@isi.edu

Abstract

As information agents become pervasive, they will need to read and understand the vast amount of information on the World Wide Web. One such valuable source of information is unstructured and ungrammatical text that appears in data sources such as online auctions or internet classifieds. One way to begin to understand this text is to figure out the entities that the text references. This can be thought of as the semantic annotation problem, where the goal is to extract the attributes embedded within the text and then annotate the text with these extracted attributes. If enough attributes can be extracted, then the entity referenced in the text can be determined. For example, if we have a used car for sale in a classified ad, and we can identify the make, model and year within the post, we can identify the car for sale. However, information extraction is difficult because the text does not contain reliable structural or grammatical clues. In this paper we present an unsupervised approach to semantically annotating such unstructured and ungrammatical text with the intention that this will help in the problem of machine understanding on the Web. Furthermore, we define an architecture that allows for better understanding over time. We present experiments to show our annotation approach is competitive with the state-of-the-art which uses supervised machine learning, even though our technique is fully unsupervised.

Introduction

Understanding unstructured and ungrammatical text such as auction listings from EBay or classified listings from Craig's List[1] requires deciding what entity is being described. For example, consider the used car classifieds shown in Figure 1. We decide what type of car is for sale by determining attributes that define it such as the car make, model and year. In this sense, the understanding can be thought of as semantic annotation, since we are annotating each piece of text, called a "post," with the attributes that define the entity. For instance, Figure 2 shows a semantically annotated post. Once we semantically annotate a set of posts, we can query them structurally and expand an agent's knowledge through the understanding (via annotation) of the posts.



[full text mode]

Mon Aug 14

FORD Thunderbird - $4700 (san bruno)

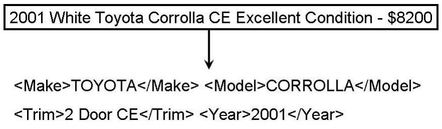
2001 White Toyota Corolla CE Excellent Condition - $8200 (los altos)

2004 Cadillac DeVille DHS Sedan 4 Door -- only 6,500 original miles pic

'96 Ford Taurus LX Wagon-VG Cond/Low Mileage! - $3999 (san bruno) pic

2006 Buick Lucerne CXS ----- Northstar Engine and only 18 miles pic

Figure 1: Posts from Craig's List.



2001 White Toyota Corrolla CE Excellent Condition - $8200

<Make>TOYOTA</Make> <Model>CORROLLA</Model>
<Trim>2 Door CE</Trim> <Year>2001</Year>

Figure 2: A semantically annotated post

However, for very large scale understanding of posts we must come up with an algorithm that scales in both the ability to semantically annotate the data and the coverage of what can be annotated. To scale the annotation we have developed an unsupervised method that builds upon previous work of exploiting "reference sets" for semantic annotation (Michelson & Knoblock 2005). Furthermore, we argue for an architecture that automatically expands the coverage of what can be annotated by collecting new reference sets from the web and plugging them into our approach for semantic annotation.

First we must define a reference set and its uses. A reference set is a collection of known entities and their attributes. For example, a reference set of cars might come from a semi-structured data source such as the Edmunds car buy-

*This research is based upon work supported in part by the National Science Foundation under award number IIS-0324955, and in part by the Air Force Office of Scientific Research under grant number FA9550-04-1-0105. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] www.craigslist.org

ing guide,[2] from which we can extract the make, model, trim and year for all cars from 1990 to the current year. However, there is no restriction on the forms a reference set could take. It can be a structured or semi-structured source. The key is that it can be thought of as a relational set of data with a defined schema and consistent attribute values.

Although our previous work (Michelson & Knoblock 2005) uses reference sets, our new approach differs substantially. Previously we used supervised machine learning, while this paper employs an information retrieval approach instead, making the annotation unsupervised. Second, in the previous work the reference sets were provided to the system *a priori*. In this work the reference sets are selected automatically from a dynamic repository of reference sets. This way, as new reference sets are added to the repository they can extend the coverage of the annotation without user involvement. So, overall the process of semantically annotating posts breaks into three main pieces. First we select the reference sets to use. Next we exploit these reference sets during annotation. Lastly, we automatically discover, model and include new reference sets, expanding the system to understand posts it could not previously.

Automatically Choosing the Reference Sets

The first step is to choose the relevant reference set. Since our repository of reference sets grows over time, we want the system to choose the reference sets automatically, improving our annotation over time. To do this, we choose the reference sets based on the similarity between the set of posts and the reference sets in the repository.

Intuitively, the most appropriate reference set is that which has the most useful tokens in common with the posts. We assume that if there is enough overlap between the posts and the reference set, then they probably refer to the same types of entities. For example, if we have a set of posts about cars, we expect a reference set with car makes, such as Honda or Toyota, to be more similar to the posts than a reference set of hotels.

To choose the reference sets, we treat all of the records in a reference set as a single document and we treat the full set of posts as a single document. Then we compute the similarity between these two documents. Next we select the reference sets that have the highest relative scores as the relevant reference sets.

When selecting the highest relative score we first we sort the similarity scores in descending order. Then we traverse this list, computing the percent difference between the current similarity score and the next. If this percent difference is above some threshold, and the score of the current reference set is greater than the average similarity score for all reference sets, the algorithm terminates. Upon termination, the algorithm returns the current reference set and all reference sets that preceded it as matching reference sets. If the algorithm traverses all of the reference sets without terminating, then no reference sets are relevant to the posts. Table 1 shows the algorithm.

We assume that the correct reference sets are relatively that much better than the irrelevant ones. This motivates the percent difference as the splitting criterion between the relevant and irrelevant reference sets. Just comparing the actual similarity values might not capture how much better one reference set is compared to another.

```
Given posts $P$, threshold $T$, and reference set repository $R$
$p \leftarrow$ SingleDocument($P$)
For all reference sets ref $\in R$
    $r_i \leftarrow$ SingleDocument(ref)
    $SIM(r_i, p) \leftarrow$ Similarity($r_i, p$)
For all $SIM(r_i, p)$ in descending order
    If PercentDiff$(SIM(r_i, p), SIM(r_{i+1}, p)) > T$ AND
    $SIM(r_i, p) > AVG(SIM(r_n, p)), \forall n \in |R|$
        Return all $SIM(r_x, p), 1 > x > i$
Return nothing (No matching reference sets)
```

Table 1: Automatically choosing a reference set

Although the algorithm does not impose any particular similarity measure, we use the Jensen-Shannon distance (JSD) (Lin 1991) between the reference set and the set of posts. The Jensen-Shannon metric is an information theoretic measure that quantifies the difference in probability distributions. Since JSD requires probability distributions, we define our distributions as the likelihood of tokens occurring in each document.

There are a few other points that require clarification about this algorithm. For one, this algorithm requires the setting of a threshold to be completely automatic. In this light, we do not employ any machine learning to learn this threshold, although that could be done. Instead, we simply set it to a “reasonable” value, by which we mean one that is neither very high and very specific, nor very low passing almost anything through. Our experiments show that a value of 0.6 works well across domains.

Also, we employ the heuristic for terminating that both the percent difference is above the threshold *and* the score is above the average. We do this since toward the end of the sorted list of similarities the numbers start to get small so that the percent differences might suddenly increase. However, this does not mean that these are good, matching reference sets. It’s just that the next reference set is that much worse than the current, bad one.

Matching Posts to the Reference Set

Once we choose the relevant reference sets, the algorithm then matches each post to the best matching members of the reference set, using these reference set member’s schema and attributes as values for semantic annotation. In the case of selecting multiple reference sets, the matching posts algorithm runs iteratively, matching the set of posts once to each chosen reference set. However, if two chosen reference sets have the same schema, we only select the one ranked higher in terms of relevance to prevent redundant matching.

To match the reference set records to the posts, we employ

[2] www.edmunds.com

a vector space model. A vector space model, rather than machine learning, makes the algorithm unsupervised and scalable. If we envision semantic annotation of unstructured and ungrammatical data on the scale of the World Wide Web, then these two characteristics are necessary.

To match posts to the records of the reference set, we treat each post as a query and each record of the reference set as its own document. We define the similarity between the post and a reference set record using the Dice similarity. Let us call our post p and a record of the reference set r, where both p and r are sets of tokens. Dice similarity is defined as:

$$Dice(p, r) = \frac{2 * (p \cap r)}{|p| + |r|}$$

We slightly change the classic Dice similarity in that we treat two tokens as belonging to the intersection between p and r if the Jaro-Winkler similarity between them is greater than or equal to 0.95. This ensures that we capture tokens that might be misspelled or abbreviated as matches, since misspellings are common in posts. Using this definition of Dice similarity, we compare each post, p_i to each member of the reference set, and we return the reference set matches that have the maximal similarity, called r_{max_i}. For each post, we also store the matches found in R_{max} which is a collection of matches for the whole set of posts. This way, we can compute the average Dice similarity scores for the matches, which we will use later in the algorithm.

Once all of the posts are scored, we must separate the true matches from the false positives. One major difference between machine learning for matching and information retrieval is that machine learning can explicitly denote a non-match, while information retrieval always returns the maximally similar document, if it exists. When matching posts, this means many false positives might be matched. For example, if we have a post “Some car is for sale and very cheap” then machine learning could determine it would match nothing, but a vector space model might assign some member of the reference set as a match, albeit with a very low similarity score. To prune these false positives, we calculate the average similarity score for all $r_{max_i} \in R_{max}$. Then we remove the r_{max_i} that are less than this average, removing those matches for those posts p_i.

One final step remains for the semantic annotation. It is possible for more than one reference set record to have a maximum similarity score with post (r_{max_i} is a set). This means we have a problem with ambiguity with the annotation provided by the reference set records. For example, consider a post “Civic 2001 for sale, look!” Now, assume our reference set has 3 records, each with a make, model, trim, and year. Record 1 might be $\{$HONDA, CIVIC, 4 Dr LX, 2001$\}$, record 2 might be $\{$HONDA, CIVIC, 2 Dr LX, 2001$\}$ and record 3 might be $\{$HONDA, CIVIC, EX, 2001$\}$. If all 3 of these records have the maximum similarity to our post, then we have a problem with some ambiguous attributes.

We can confidently assign HONDA as the make, CIVIC as the model and 2001 as the year, since all of the matching records agree on these attributes. We call these attributes *in agreement*. However, there is disagreement on the trim because we can not determine which value is best for this attribute, based on the matches from the reference set, since all values are equally valid from the vector space perspective. So, we leave this attribute out of the annotation. This is a reasonable approach since in many real world posts, not all of the detailed attributes are specific. For example, the first post of Figure 1 shows a Ford Thunderbird, but nothing else, so we can not make a claim about its trim or even its year. So the final step is to remove all attributes from our annotation that do not agree across all matching reference set records. Our vector space approach to unsupervised annotation is shown in Table 2.

```
Given posts $P$ and reference set $R$
$R_{max} \leftarrow \{\}$
For all $p_i \in P$
    $r_{max_i} \leftarrow$ MAX( DICE ($p_i$,R))
    $R_{max} \leftarrow R_{max} \cup r_{max_i}$
For all $p_i \in P$
    Prune $r_{max_i}$ if DICE($r_{max_i}$) $<$ AVG(DICE($R_{max}$)
    Remove attributes not in agreement from $r_{max_i}$
```

Table 2: Our Vector Space approach to automatic semantic annotation

One aspect of this approach that requires some discussion is the use of Dice similarity. While it has been used in the past for information retrieval, we choose the Dice similarity based on a few additional reasons. Once choice would be to use TF-IDF weighting with cosine similarity. However, we found that in reference sets, matching at such a fine level of individual records to a post, the weighting schemes will overemphasize unimportant tokens, while discounting the important ones. For example, in a reference set of cars, the token Honda will occur much more frequently than Sedan. In this case, a reference set record might incorrectly match a post simply because it matches on Sedan, rather than the more important token Honda. Dice, on the other hand, does not exploit frequency based weights.

The other similarity measure we could use would be the Jaccard similarity, which uses the intersection of the tokens divided by their union. However, Jaccard penalizes having only a small number of common tokens, which could be a problem when matching to posts since often times posts contain just a few important tokens for matching, such as “Civic 2001.” Since Jaccard contains a union in the denominator, the denominator is affected by the size of the intersection, since union is the sum of the sizes of the sets minus the intersection. So, if many tokens are in common, the denominator is shrunk, resulting in a higher score, but if there are only a few in common, the denominator is barely affected. Meanwhile, using Dice similarity the number of tokens in common does not affect the denominator. So, if only a few tokens are in common, Dice boosts this number in the numerator by 2 while leaving the denominator unnafected. Meanwhile, Jaccard does not boost the numerator and barely affects the denominator. So, with only a few tokens, Dice gets a higher score than Jaccard.

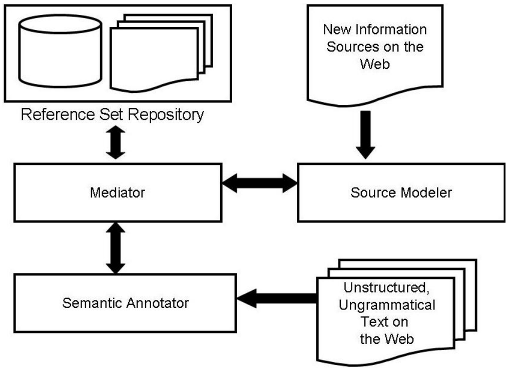

New Information Sources on the Web
Reference Set Repository
Mediator
Source Modeler
Semantic Annotator
Unstructured, Ungrammatical Text on the Web

Figure 3: An architecture for expanding the coverage of annotation

Extending the Coverage

For our approach to scale to the size of the Web, our repository of reference sets must cover as many different entities as possible. For this to happen, we need a mechanism to discover, model and integrate new sources of data into the repository so we can improve our ability to understand posts by improving our coverage in the repository.

We propose hooking our semantic annotation algorithm to systems that can automatically discover and understand web sources by semantically modeling what the services do and what their attributes are(Carman 2006; Lerman, Plangrasopchok, & Knoblock 2006). However, just discovering and modeling new sources is not enough. We must also connect our annotation approach to a system that can manage, integrate and connect them to the repository, such as a mediator system, e.g. (Thakkar, Ambite, & Knoblock 2005). Interesting pieces of future work are using the mediator to combine different sources into more useful reference sets and deciding which reference sets could be ignored since sources such as weather providers would not be useful for semantic annotation.

In this approach, shown in Figure 3, new sources are constantly discovered, modeled, and integrated into the repository of reference sets, allowing for expanded coverage of the semantic annotation. This way, the understanding of posts improves over time as the repository of reference sets improves over time.

Experimental Results

This section presents results for our unsupervised approach to selecting a reference set and semantically annotating each post. However, before we examine the results of our experiments, we describe the reference sets and sets of posts we will use in testing. All data sources, whether posts or reference sets, were collected from data sources that exist on the World Wide Web.

Reference Sets

For our experiments we use four different reference sets, crossing multiple domains. Most of these reference sets have been used in the past in either the information extraction literature or record linkage literature. The first reference set is the *Hotels* reference set which consists of 132 hotels in the Sacramento, San Diego and Pittsburgh areas culled from the Bidding For Travel website's Hotel List. These records contain a star rating, a hotel name and a local area and were used previously as a reference set in (Michelson & Knoblock 2005).

Another reference set comes from the restaurant domain and previous work on record linkage (Bilenko & Mooney 2003). We call this reference set *Fodors*, which consists of 534 restaurants, each having a name, address, city and cuisine as the attributes.

Next, we have two reference sets containing information about cars. The first, called *Cars*, contains all of the cars from the Edmunds Car Buying Guide from 1990-2005. From this data set we extracted the make, model, year and trim for all cars from 1990 to 2005, resulting in 20,076 records. We supplement this reference set with cars from before 1990, taken from the auto-accessories company, Super Lamb Auto. This supplemental list contains 6,930 cars from before 1990, each having a make, model year and trim. We consider this combined reference set of 27,006 records as the *Cars* reference set.

Our last reference set is about cars having make, model, year and trim as its attributes as well. However, it is a subset of the cars covered by the *Cars* reference set. This data set comes from the Kelly Blue Book car pricing service containing 2,777 records for Japanese and Korean makes from 1990-2003. We call this set *KBBCars*. This data set has also been used in the record linkage community (Minton *et al.* 2005).

Post Sets

The set of posts for our experiments show the different cases that exist for finding the appropriate reference sets. One set of our posts matches only a single reference set in our collection. It contains 1,125 posts from the internet forum Bidding For Travel. These posts, called *BFT* match the *Hotels* reference set only. This data set was used previously in (Michelson & Knoblock 2005).

Our approach can also select multiple relevant reference sets. So we use a set of posts that matches both car reference sets. This set contains 2,568 posts about cars from the internet classifieds Craig's List. We call this set of posts *Craigs List*. Note, however, that while there may be multiple reference sets that are appropriate, they also might have an internal ranking. As an example of this, we expect that the *Craigs List* posts selects both the *Cars* and *KBBCars* reference sets, but *Cars* should be ranked first.

Lastly, we need to check whether the algorithm can suggest that there is no relevant reference set in our repository. To test this idea, we collected 1,099 posts about boats from Craig's List, called *Boats*. Intuitively boats are similar enough to cars to make this a non-trivial test, since boats and cars are both made by Honda, for example, so that keyword appears in both sets of posts. However, boats are also different enough from all the reference sets that there should not be an appropriate reference set selected.

Results for Choosing Relevant Reference Sets

In this section we provide results to show the algorithm successfully performs on multiple cases and across multiple domains. For all experiments we keep the threshold at 0.6. We expect that the last appropriate reference set should be roughly 60% better than the first inappropriate one.

Table 3 shows our results using the Jensen-Shannon distance (JSD) which validate our approach to automatically choosing relevant reference sets. The reference set names in bold reflect those that are chosen as appropriate. (This means boats should have no bold names). The scores in bold are the similarity scores for the chosen reference sets, and the percent difference in bold is the point at which the algorithm breaks out and returns the appropriate reference sets. In particular, JSD successfully identifies the multiple cases where we might have a single appropriate reference set, multiple reference sets, or no reference set. Also, the results show that across domains using the intuitive and simple threshold of 0.6 works well.

The results also justify the need of including a double stopping criteria for the algorithm. It is not enough to just consider the percent difference as an indicator of relative superiority amongst the reference sets. The scores must also be compared to an average to make sure that the algorithm is not errantly supplying a bad reference set as appropriate just because it is relatively better than an even worse one. For an example of this, consider the last two rows of the *Boats* posts in Table 3.

BFT Posts		
Ref. Set	Score	% Diff.
Hotels	**0.622**	**2.172**
Fodors	0.196	0.050
Cars	0.187	0.248
KBBCars	0.150	
Average	0.234	

Craig's List		
Ref. Set	Score	% Diff
Cars	**0.520**	**0.161**
KBBCars	**0.447**	**1.193**
Fodors	0.204	0.561
Hotels	0.131	
Average	0.326	

Boat Posts		
Ref. Set	Score	% Diff.
Cars	0.251	0.513
Fodors	0.166	0.144
KBBCars	0.145	0.723
Hotels	0.084	
Average	0.162	

Table 3: Results using Jensen-Shannon distance

Results for Semantic Annotation

Once the relevant reference sets are chosen, we use our vector space model of semantic annotation for each post, and in this section we present results showing that our approach to semantic annotation is valid. To do this, we take the true matches between the posts and the reference sets, and for each set of true matches for each post, we use the attributes in agreement, as stated above. Then we compare these to the attributes in agreement for our matches chosen using our vector space model. We present only results for the BFT and Craig's List posts, since Boats have no relevant reference set.

To evaluate the semantic annotation, we use the traditional information extraction measures of precision, recall and f-measure, the harmonic mean between precision and recall. We define a correct match for an attribute when the attributes in agreement predicted by the vector space model matches that from the true matches. In some sense these are field level extraction results. In many extraction experiments, just finding a token in common between the truly extracted attribute and the predicted is counted as a match, but in our case, we are considering matches only where the whole attribute matches (all of the tokens), which is a stricter rubric, and more truly indicates the accuracy for searching based on the extracted attributes. The results are shown in Table 4.

We compare our results to those from (Michelson & Knoblock 2005), for the BFT Posts to show our automatic approach is competitive with a machine learning approach. For this we use the F-measure record linkage results from our previous paper, since in that work we used the matching reference set record's attributes as normalized semantic annotation. This allows us to compare our new semantic annotation using the attributes in agreement to our old annotation stemming from the record matching. Table 4 reports our old record linkage F-measure as *Prev. F-Mes.*

BFT Posts				
Attribute	Recall	Prec.	F-Measure	Prev. F-Mes.
Hotel Name	88.23	89.36	88.79	92.68
Star Rating	92.02	89.25	90.61	92.68
Local Area	93.77	90.52	92.17	92.68
Craig's List Posts				
Make	93.96	86.35	89.99	N/A
Model	82.62	81.35	81.98	N/A
Trim	71.62	51.95	60.22	N/A
Year	78.86	91.01	84.50	N/A

Table 4: Semantic Annotation Results

However, a direct comparison between the two results is slightly skewed because our system is unsupervised, where as the old system is not. So previously we only reported on 70% of the data used for testing, while this paper uses 100% of the posts. Nonetheless, we believe these are encouraging results given that the new approach is fully automatic in both the selection of reference sets and the matching, while our previous approach requires a user to provide both the correct reference set and labeled training data.

One interesting difference between the two approaches is in disambiguating false positives, which leads to some problems with the vector space model. When we used machine learning we could learn directly that some candidate matches are true matches while others are false positives, especially given that certain matching attributes are more indicative of a match than others. For example, matching on a car model is probably better at indicating a match than matching on a car make. This allowed us to be sure that the reference set attributes for a match were correct, since we were confident of the match, so there was not a problem with attributes not being in agreement. However, with the vector space model, we have a disambiguation problem because certain reference set records that score the same for a given post are all equally as likely to be a match, especially since we have no notion of certain attributes being more indicative than others. Clearly this is a limitation with our approach and it requires us to either select the intersection or the union of the attributes for all returned reference set attributes. We select the intersection because we want to limit the false positives for given attributes, but this leads to problems, especially with attributes that are short and ambiguous. For example, the trim attribute in Craig's List Cars is a short pieces of text, usually just a few letters. So, in some cases, different sets of a few letters in a post could match multiple reference set trims, even though only one is correct, so these attributes are not in agreement and get removed, hindering their accuracy. Overcoming this issue is something we plan to investigate in the future.

Related Work

Performing semantic annotation automatically is a well studied field of research, especially as researchers develop the Semantic Web. According to a recent survey (Reeve & Han 2005), systems that perform automatic semantic annotation break into 3 categories: rule based, pattern based, and wrapper induction based methods. However, the rule based and pattern based methods rely on regularity within the text, which is not the case with posts. Also, the wrapper induction methods use supervised machine learning instead of unsupervised methods.

The most similar system to ours is the SemTag (Dill *et al.* 2003) system, which first identifies tokens of interest in the text, and then labels them using the TAP taxonomy, which is similar to our exploitation of reference sets.

However, there are a few key differences between this approach and ours. First, the noisy nature of the posts does not allow for an exact lookup of the tokens in the reference set. So our approach emphasizes this aspect as a contribution, while SemTag's focus is more on disambiguating the possible labels. Second, their disambiguation comes about because of synonymy that our approach avoids entirely. In their paper they mention the token Jaguar might mention a car or an animal, since they disambiguate after labeling. In our case, we perform disambiguation before the labeling procedure, during the selection of the relevant reference sets. If we had a reference set of animals and one of cars, and we chose cars as the relevant reference set, then we would not have this type of synonymy since animal labels would not be an option.

Lastly, our approaches differ in their outside knowledge. SemTag uses a well defined, carefully crafted taxonomy. This gives their reference set good accuracy and well defined labels with lots of meaning. Our approach is just to incorporate any reference sets that we can collect automatically from the web. Including new reference sets in our repository has no effect on the data already in it (since the reference sets are independent). So our approach to data collection is not as careful as using a full taxonomy, but we can much more easily and quickly gather lots and lots of reference data, greatly increasing our coverage of items we can annotate.

Another unsupervised system that we might compare our research to is the KnowItNow system (Cafarella *et al.* 2005), which extracts entities from online sources automatically. However, a direct comparison is not appropriate because our focus is on unstructured posts, while they are extracting entities seeded by patterns and using natural language, so the domain of text is different. However, our approach to expanding our coverage by discovering and incorporating new reference sets is somewhat similar to KnowItNow in that the goal is to discover and incorporate new information automatically. In fact, the two approaches complement each other very well for automatically constructing reference sets and extracting data from different types of sources. We could use the source modeling component for discovering structured reference sets from sources such as Web Services, while we could use KnowItNow to build reference sets from semi-structured web pages.

Conclusion

In this paper we present a system for understanding unstructured and ungrammatical text via unsupervised semantic annotation. Our annotation method exploits reference sets, which are automatically chosen, and can be continuously harvested, showing that the coverage, and thus annotation, can improve over time.

References

Bilenko, M., and Mooney, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings KDD*.

Cafarella, M. J.; Downey, D.; Soderland, S.; and Etzioni, O. 2005. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of HLT-EMNLP*.

Carman, M. J. 2006. *Learning Semantic Definitions of Information Sources on the Internet*. Ph.D. Dissertation, University of Trento.

Dill, S.; Gibson, N.; Gruhl, D.; Guha, R.; Jhingran, A.; Kanungo, T.; Rajagopalan, S.; Tomkins, A.; Tomlin, J. A.; and Zien, J. Y. 2003. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of WWW*.

Lerman, K.; Plangrasopchok, A.; and Knoblock, C. A. 2006. Automatically labeling the inputs and outputs of web services. In *Proceedings of AAAI*.

Lin, J. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory* 37(1):145–151.

Michelson, M., and Knoblock, C. A. 2005. Semantic annotation of unstructured and ungrammatical text. In *Proceedings of IJCAI*.

Minton, S. N.; Nanjo, C.; Knoblock, C. A.; Michalowski, M.; and Michelson, M. 2005. A heterogeneous field matching method for record linkage. In *Proceedings of ICDM*.

Reeve, L., and Han, H. 2005. Survey of semantic annotation platforms. In *Proceedings of ACM Symposium on Applied Computing*.

Thakkar, S.; Ambite, J. L.; and Knoblock, C. A. 2005. Composing, optimizing, and executing plans for bioinformatics web services. *The VLDB Journal, Special Issue on Data Management, Analysis, and Mining for the Life Sciences* 14(3):330–353.