An Automatic Approach to Semantic Annotation of Unstructured, Ungrammatical Sources: A First Look*

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

Abstract

There exist numerous sources of data on the World Wide Web that contain useful information but are not structured or grammatical enough to support traditional information extraction. Furthermore, even if the information extraction could be done, the extracted values would need to be standardized to ensure the queries over the source are accurate. This paper presents an automatic, scalable approach to semantically annotating such unstructured, ungrammatical sources with standardized values, allowing for accurate, structured queries of the source. Our technique recasts the information extraction problem as an information retrieval problem, treating each entry of unstructured, ungrammatical text as a query and comparing it to a set of known records called a "reference set." Furthermore, given a library of reference sets, the system automatically chooses the correct one, making the technique fully unsupervised. We compare our automatic technique to a previous approach that exploits supervised learning and show that we get comparable results. In the previous approach, beyond providing labeled training data, a user also must supply the "reference set" which is exploited for the extraction and standardization of the values.

1 Introduction

There are numerous data sources on the World Wide Web that are packed with useful information in the form of unstructured and ungrammatical textual data. Examples of such sources are internet classifieds such as Craig's List,[1] internet auctions such as EBay,[2] bulletin boards and forums such as Bidding For Travel,[3] and even the hyperlinks returned by certain queries from some search engines. To generalize this idea, we call each textual entry within such sources a "post." Figure 1 shows some real world posts from Craig's List regarding cars for sale.

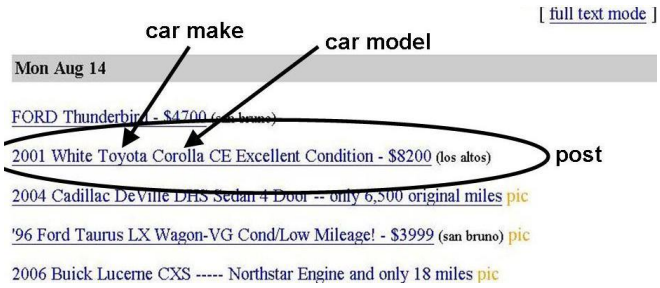

Figure 1: Posts from Craig's List.

The goal is to enable structured queries over these posts, allowing either an agent or a human a deeper ability to gain insight from the information. Before the data can be queried though, attributes and normalized values need to be added to the post to query on. This process is semantic annotation. For example, given the second post, we could annotate the post with a make attribute with the value TOYOTA, a model attribute with the value COROLLA, trim attribute of CE and a year attribute of 2001, as in Figure 2. One way to perform semantic annotation is to do information extraction to pull out and label the attributes. However, the posts present numerous problems to information extraction. First, the structure from post to post is generally not similar enough to support structure based approaches to information extraction, such as the wrapper methods of Stalker [Muslea *et al.*, 2001] or Road-

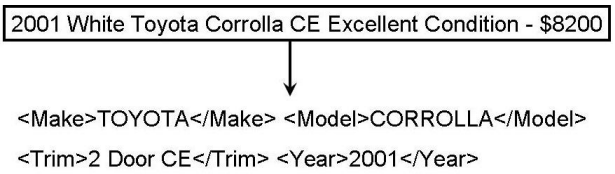

Figure 2: A semantically annotated post

*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.

[1] www.craigslist.org

[2] www.ebay.com

[3] www.biddingfortravel.com

Runner [Crescenzi *et al.*, 2001]. Also, there is generally little grammar within posts to support Natural Language Processing approaches to information extraction such as those used by Whisk [Soderland, 1999] or Rapier [Califf and Mooney, 1999].

However, the difficulty in querying the posts does not stop with just extracting the attributes. The noisy nature of the posts makes them difficult to query, even when information extraction can be performed. For one, many posts are missing key attributes, which are implied. For example, consider the post "2000 Civic for sale, $7,500." The model name Civic implies that the car make is a Honda, even though this is not explicitly stated. So, if we were to query the posts on the make of Honda, this post would not be returned. Beyond the missing attributes, those that are within the post may have many misspellings and the order of the attributes, and even the tokens within the attributes, is not reliable enough to use for extraction. This means that to accurately perform structured queries it is not enough to just extract the attributes, but we need to supply normalized values for these attributes to alleviate the spelling mistakes, myriad token orderings, and missing tokens and attributes.

So, there are two main difficulties in semantically annotating the posts. First, the information extraction is particularly hard since the posts are unstructured and ungrammatical. Furthermore, the extracted attributes need to be normalized. To alleviate both of these problems, we recast the problem to one of information retrieval. If we have a set of normalized values representing the items in the post, then by matching the posts to those items we can get an idea of what attributes are within the post (information extraction) and we have a set of normalized values for those attributes.

The key to our method is exploiting *reference sets* [Michelson and Knoblock, 2005]. A reference set is a known collection of entities, along with the attributes that define these entities. For example, a reference set could be an online collection, such as the Edmunds website,[4] which contains a list of all cars from 1990 to the present. For each car, Edmunds lists the car's make, model, year, and trim. Reference sets can also be online (or offline) data bases, such as the Comics Price Guide,[5] or even collections of documents such as the CIA World Fact Book. The most important factor that defines a reference set is that it can be treated as a relational data set with a well defined schema and normalized attribute values. To exploit reference sets, our method first automatically selects the reference set(s) from a repository that are related to the posts we are examining. We refer to these related reference sets as *relevant*.

Once a reference set(s) is deemed relevant, we then try and match each post to the best matching member(s) of the reference set. To do this, we use a vector space model, treating each post as a query, and each member of the reference set as a document. Then we return the records from the reference set that best match the query post. This vector space model allows for unsupervised matching between the posts and the reference set records. We use the attributes from the returned reference set records as the semantic annotation for the posts. This annotation provides a standard set of attribute values (since they come from the reference sets), and in some cases this annotation provides attributes for a given post that were not explicitly mentioned in the post. The overall architecture is shown in Figure 3.

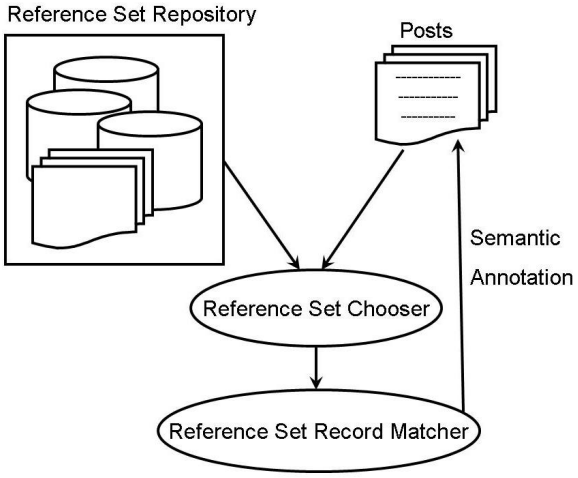

Figure 3: Our architecture for the unsupervised annotation of posts

In the past information extraction and annotation techniques have incorporated outside information to aid in the extraction process. Similar to our approach, some of this extraction work is unsupervised, such as the CRAM system [Agichtein and Ganti, 2004]. However, unlike this paper, CRAM does not select the the appropriate reference set automatically and it assumes the matching between the posts and reference sets has already been accomplished.

Other work in information extraction incorporates reference sets with machine learning techniques. Examples include the Phoebus system [Michelson and Knoblock, 2005] and the work of Cohen and Sarawagi [Cohen and Sarawagi, 2004]. However, aside from requiring human selected reference sets, both of these techniques rely on supervised machine learning for the extraction/annotation. In this paper we present an approach to unsupervised semantic annotation. Our approach is unsupervised in both the selection of the reference sets and their matching.

The rest of the paper is organized as follows. In section 2, we describe our simple, automatic algorithm to choose the relevant reference sets. In section 3 we present our information retrieval approach to semantic annotation. In section 4 we provide experiments evaluating the technique along with a discussion of these results. In sections 5 and 6 we present some more related work and we conclude with some final thoughts.

2 Choosing the Reference Sets

As stated previously, choosing the correct reference sets for the set of posts is a precursor to our information retrieval approach to semantic annotation. To choose reference sets, we have developed a simple, automatic method that exploits the

[4] www.edmunds.com

[5] www.comicspriceguide.com

notions of similarity between the set of posts and the reference sets in our collection.

Intuitively our algorithm decides that the most appropriate reference set is that which has the most useful tokens in common with the posts. The assumption here is that if there is enough overlap between the set of posts and the reference set, then they probably refer to the same things. Furthermore, our assumption is that there are meaningful and distinctive tokens within a reference set, so if those overlap with the posts, they are a good indicator of similarity. For example, if we have a set of posts about cars from Craig's list, 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 in Los Angeles.

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

Selecting the "highest meaningful" scores is done as follows. The algorithm first sorts the similarity scores between all of the reference sets and posts in descending order. Then it traverses 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 terminating in this fashion, the algorithm returns the current reference set and all reference sets that preceded as matching reference sets. If the algorithm traverses all of the reference sets without hitting the termination criteria, then there is not considered a matching reference set for the set of posts. Table 1 shows the algorithm in detail:

Given posts P and threshold T
$p \leftarrow$ *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

There are a few aspects of this algorithm that require clarification. The first is the choice of percent difference as the splitting criterion. This idea captures the intuition that if we have relevant reference sets for the posts, and irrelevant ones, then the relevant ones will be similar while the irrelevant ones will not. However, just comparing the actual similarity values may not capture how much better one reference set is compared to another. Instead, we need a measure of relative similarity, within the set of reference sets, which is why we choose percent difference.

Also, the algorithm requires the setting of a threshold. Since we want this method to be completely automatic, we do not employ any machine learning to learn this threshold. Instead, we simply set it to a "reasonable" value based on the similarity metric used. By reasonable we mean a little better than half, such as 0.6. We consider unreasonable thresholds to be very high and specific numbers, or very low numbers that pass almost anything through. Again, we show that a reasonable value of 0.6 works well in our experiments across domains and different similarity measures.

Furthermore, not only do we require that the percent difference is great, but we also require that the score is above the average. We include this heuristic because in many cases, toward the end of the sorted list of similarities the numbers start to get small so that the percent differences will suddenly jump. 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. We show this property emerging in our experiments, justifying this heuristic.

Lastly, there is no specific similarity metric defined for this algorithm. Our approach is to make the algorithm as general as possible without tying it to a specific measure of similarity. As we show in the experiments section, two different similarity metrics both prove effective. One similarity metric we try is the cosine similarity using TF-IDF to weight the tokens [Salton and McGill, 1983]. We also try the algorithm using the Jensen-Shannon distance (JSD) [Lin, 1991]. Jensen-Shannon distance is an information theoretic measure that quantifies the difference in probability distributions. (Note that since JSD requires probability distributions, we define our distributions as the likelihood of tokens occurring in each document.)

3 Matching Posts to the Reference Set

Once the relevant reference sets have been chosen, the algorithm then attempts to match the posts to best matching members of the reference set, using these reference set member's attributes as normalized values for semantic annotation. In order to match the reference set records to the posts, we employ a vector space model, borrowed from the field of information retrieval. We do this, rather than machine learning, to make the algorithm unsupervised and scalable. We envision semantic annotation of unstructured and ungrammatical data on the scale of the World Wide Web, so 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 r, where both p and r can be considered sets of tokens. Then, Dice similarity is defined as:

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

The one change we make to classic Dice similarity is 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 we have scored all of the posts, and found their matching reference set members in this manner, there are two more important steps. One major difference between machine learning approaches to matching and information retrieval methods is that machine learning can explicitly note when something does not have a match, while information retrieval will always return the maximally similar document, if it exists. This can be a problem when matching posts to the reference sets. For example, if we have a post "Some car is for sale and very cheap" then machine learning would not match it to anything, but a vector space model might assign some member of the reference set as a match, albeit with a very low similarity score. So, we must prune these types of matches where the posts are not actually referring to anything. The pruning is very simple. Using R_{max}, we calculate the average similarity score for all r_{max_i} for all different p_i. Then we prune the r_{max_i} that are less than this average, removing those matches for those posts.

One more step remains for our semantic annotation. As noted above, it is possible that more than one record can have a maximum similarity score as compared to a post. That is, r_{max_i} might be a set of reference set records, rather than just one. 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. For example, 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 say with certainty that we should annotate the make as HONDA, the model as CIVIC and the year as 2001, since all of the matching records agree on these attributes. We call these attributes *in agreement*. However, since there is disagreement on the trim, we leave this attribute out of our annotation, since we can not disambiguate which value is best for this attribute, based on the matches from the reference set. I.e. they are all equally valid from our vector space perspective, so we decide that we do not know that attribute. 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.

One aspect of this approach that requires some discussion

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

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." The reason that Jaccard penalizes having only a few common tokens versus Dice similarity has to do with the treatment of having just a few tokens in common. For Jaccard, the denominator contains a union which can be defined as the sum of the sizes of the sets minus the intersection, so Jaccard's denominator is affected by the size of the intersection. Therefore, 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. With just a few common tokens, Jaccard also has a small numerator, resulting in a lower score than Dice. 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 will not boost the numerator and barely affects the denominator. So, with only a few tokens, Dice gets a higher score than Jaccard.

4 Experiments

To validate our approach we run experiments showing that we can correctly select the relevant reference sets automatically, and then we present results exploiting these relevant reference sets for semantic annotation. 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.

4.1 Reference Sets

For our experiments we use six 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 two reference sets are both from the restaurant domain and come from previous work on record linkage [Bilenko and Mooney, 2003]. The first is a collection of 534 restaurants from Fodors that have a name, address, city and cuisine as their attributes. We call this reference set *Fodors*. The second set of restaurants comes from Zagats, so we refer to it as *Zagat*. This data set also has name, address, city and cuisine as the attributes and contains 330 records.

The next reference set contains 918 records from the Comics Price Guide regarding Fantastic Four and Incredible Hulk comic books. This reference set has a title, issue number (such as vol. 2 or #14), and a publisher. In our results, we call this reference set *Comics*. This reference set was used in [Michelson and Knoblock, 2005]. Also from this paper 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.

Lastly, 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. This reference set is used for the demonstration of the Phoebus system [Michelson and Knoblock, 2006]. 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]. A summary of the reference sets is given in Table 3.

These sets of posts and reference sets demonstrate the different cases that exist for finding the appropriate reference sets, and we describe them next.

4.2 Posts Sets

We have two sets of posts which match only a single reference set in our collection. The first are 1,125 posts from the internet forum Bidding For Travel. These posts, called *BFT* match the *Hotels* reference set only. The other set of single matching posts are auction item posts from EBay regarding Fantastic Four and Incredible Hulk merchandise for sale. There are 776 of these posts. We call this set *EBay* and it should match the *Comics* reference set only. Both the *BFT* and *EBay* posts were used in [Michelson and 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 a set of posts to test 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. Our intuition is that boats are similar enough to cars to make this a non-trivial test, since, for example, boats and cars are both made by Honda, 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. We call this set of posts *Boats*. All of the sets of posts are summarized in Table 4.

4.3 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. We also show the invariance to the similarity metric used, as long as it is not too simple, and we show that an intuitive, reasonable threshold performs well. In fact, 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. Lastly, we show that including the heuristic that a score must be above the average, not just that the percent difference is large, is justified by showing cases where the similarities get small enough to increase the percent differences.

As mentioned previously, two similarity metrics we tried with our approach are cosine similarity using TF-IDF to weight the tokens, and the Jensen-Shannon distance (JSD). Table 5 shows the results when running the algorithm using TF-IDF and Table 6 shows the results using JSD. 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.

TF-IDF successfully performed all of the cases outlined in the descriptions of the posts. The only incorrect reference set selection was ranking *Cars* as the second best reference set for *Craigs List* posts, while incorrectly ranking *KBBCars* as the best. This is due to the fact that we treat the whole reference set as a single document, and cosine similarity prefers shorter strings [Jean *et al.*, 2005], which would be true in comparing *Cars* to *KBBCars*. However, JSD appears to perform better than TF-IDF in that the percent differences are larger, and when comparing the *Craigs List* posts it selects *Cars* ahead of *KBBCars*, which is correct. Also, using JSD we never run into the case where we have a large percent difference between the sets, but the scores are below the average

Name	Source	Website	Attributes	Records
Fodors	Fodors Travel Guide	www.fodors.com	name, address, city, cuisine	534
Zagat	Zagat Restaurant Guide	www.zagat.com	name, address, city, cuisine	330
Comics	Comics Price Guide	www.comicspriceguide.com	title, issue, publisher	918
Hotels	Bidding For Travel	www.biddingfortravel.com	star rating, name, local area	132
Cars	Edmunds and Super Lamb Auto	www.edmunds.com and www.superlambauto.com	make, model, trim, year	27,006
KBBCars	Kelly Blue Book Car Prices	www.kbb.com	make, model, trim, year	2,777

Table 3: Reference Set Descriptions

Name	Source	Website	Reference Set Match	Records
BFT	Bidding For Travel	www.biddingfortravel.com	Hotels	1,125
EBay	EBay Comics	www.ebay.com	Comics	776
Craigs List	Craigs List Cars	www.craigslist.org	Cars, KBBCars	2,568
Boats	Craigs List Boats	www.craigslist.org		1,099

Table 4: Descriptions of the sets of posts

score. This shows that JSD scores allow for a successful relative ranking of their appropriateness.

The results using JSD and TF-IDF validate our approach to automatically choosing relevant reference sets. In particular, they successfully identify the multiple cases where we might have a single appropriate reference set, multiple reference sets, or no reference set. Also, they show that across domains using the intuitive and simple threshold of 0.6 works well, so there is no need for tuning this parameter.

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 5.

BFT Posts		
Ref. Set	Score	% Diff.
Hotels	**0.499**	**1.743**
Fodors	0.182	0.318
Comics	0.138	0.029
Zagat	0.134	0.330
Cars	0.101	1.893
KBBCars	0.035	
Average	0.182	

Ebay Posts		
Ref. Set	Score	% Diff
Comics	**0.402**	**0.973**
Fodors	0.204	0.522
Zagat	0.134	0.057
Cars	0.127	1.567
KBBCars	0.049	0.184
Hotels	0.041	
Average	0.160	

Craig's List		
Ref. Set	Score	% Diff.
KBBCars	**0.122**	0.239
Cars	**0.099**	**1.129**
Zagat	0.046	0.045
Fodor	0.044	0.093
Comics	0.041	0.442
Hotels	0.028	
Average	0.063	

Boat Posts		
Ref. Set	Score	% Diff.
Cars	0.200	0.189
Comics	0.168	0.220
Fodor	0.138	0.296
Zagat	0.107	0.015
KBBCars	0.105	0.866
Hotels	0.056	
Average	0.129	

Table 5: Results using TF-IDF as similarity measure

BFT Posts		
Ref. Set	Score	% Diff.
Hotels	**0.622**	**2.172**
Fodors	0.196	0.050
Cars	0.187	0.248
KBBCars	0.150	0.101
Zagat	0.136	0.161
Comics	0.117	
Average	0.234	

Ebay Posts		
Ref. Set	Score	% Diff
Comics	**0.579**	**2.351**
Fodors	0.173	0.152
Cars	0.150	0.252
Zagat	0.120	0.186
Hotels	0.101	0.170
KBBCars	0.086	
Average	0.201	

Craig's List		
Ref. Set	Score	% Diff.
Cars	**0.520**	0.161
KBBCars	**0.447**	**1.193**
Fodors	0.204	0.144
Zagat	0.178	0.365
Hotels	0.131	0.153
Comics	0.113	
Average	0.266	

Boat Posts		
Ref. Set	Score	% Diff.
Cars	0.251	0.513
Fodors	0.166	0.144
KBBCars	0.145	0.089
Comics	0.133	0.025
Zagat	0.130	0.544
Hotels	0.084	
Average	0.152	

Table 6: Results using Jensen-Shannon distance as similarity measure

4.4 Results for Semantic Annotation

Once the relevant reference sets are chosen, we use these chosen reference sets in 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. Obviously, we present only results for the BFT, Ebay 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. These results are shown in Figure 7.

We compare our results to those from our previous work [Michelson and Knoblock, 2005], for the two experimental domains which we can, the BFT Posts and the EBay posts. To do this, we use the F-measure record linkage results from the previous paper, since we used the matching reference set record's attributes as normalized semantic annotation previously, as in this paper. This allows us to compare our semantic annotation using the attributes in agreement to the annotation stemming from our previous record matching method. In Table 7 we report the previous 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
EBay Posts				
Title	86.08	91.60	88.76	88.64
Issue	70.16	89.40	78.62	88.64
Publisher	86.08	91.60	88.76	88.64
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 7: Semantic Annotation Results

Although, a direct comparison between the two results is slightly skewed because our current system is unsupervised, where as the previous system is not, the results are good. Our results are competitive with those of the supervised system that requires labeling 30% of the data. So, not only are we able to perform semantic annotation well, we can do it in an unsupervised manner.

One interesting difference between the two approaches is in disambiguating false positives, which leads to some problems with this paper's approach. In our previous work, [Michelson and Knoblock, 2005] we used machine learning which learns directly that some candidate matches are true matches while others are false positives, especially given that certain matching attributes might be more indicative of a match than others. This allowed us to be sure that the reference set attributes for a match are correct, so we could return them all confidently. In this sense there was no problem with attributes not being in agreement. In this paper, however, 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 clearly this leads to problems, especially with attributes that are short and ambiguous. For example, the issue attribute in EBay comics and the trim attribute in Craig's List Cars are both very short pieces of text, usually just a number or 1 or 2 letters. So, in some cases, if there is another number in the post or another 1-2 letters that could match another reference set member, then these attributes are not in agreement and they get removed, hindering their accuracy. Overcoming this issue is something we plan to investigate in the future.

5 Related Work

Performing semantic annotation automatically is a well studied field of research, especially as researchers develop the Semantic Web. As stated previously, this work is similar to the Phoebus system in that it exploits reference sets to perform annotation. However, unlike Phoebus, our research is more automatic, given a set of reference sets, making our algorithm much more scalable.

Beyond Phoebus there are other systems that perform semantic annotation in an automatic fashion, and according to a recent survey [Reeve and Han, 2005], these annotation systems break into 3 basic categories: rule based, pattern based, and wrapper induction based methods. Pattern based systems, such as Armadillo [Dingli *et al.*, 2003], can automatically extract entities from text by identifying them based on regular patterns in the text. However, this regularity is not guaranteed in unstructured, ungrammatical text, which makes this approach difficult for posts. This is a similar case for the rule based annotation systems, where the rules rely on regularity. The wrapper induction methods, such as MnM [Vargas-Vera *et al.*, 2002] use supervised machine learning, so they are not fully automatic.

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. To perform its annotation, SemTag tokenizes data, such as a web page, then gives each token(s) a label by looking it up in the taxonomy. It then disambiguates the possible labels (since several may match) using neighbor tokens and corpus statistics, picking the best label for a token.

However, there are 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, using a vector model to select the labels, while SemTag's focus is more on disambiguating the possible labels. Second, their disambiguation comes about because of synonymy that our approach avoids entirely. For instance, 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 this type of disambiguation before the labeling procedure, during the selecting 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.

The automatic selection of reference sets presented in this paper is similar to the problem of resource selection in distributed information retrieval, sometimes used for the "hidden web." For example, [Craswell *et al.*, 2000], provide a comparison of three approaches to this problem that each treat the information sources as documents, like we do. However, one major difference between their resource selection problem and ours is that a major focus of their attention is estimating the data that their sources cover. They must execute probe queries and examine the results to estimate statistics of data coverage and then use these statistics to select the correct resources at query time. In our problem, we have full access to the reference sets, so we can select the resource using the full reference set without any such coverage estimations.

6 Conclusion

In this work we present an unsupervised method to semantically annotate unstructured, noisy text found on the World Wide Web. This scalable approach allows for much richer, structured, accurate querying of these noisy data sources. While the reported results are encouraging, this annotation is not the last step in processing the unstructured sources. In the future, we will investigate how to exploit the matching members of the reference sets to perform unsupervised information extraction. Many of the current unsupervised extraction techniques rely on patterns in the data which would not exist in our unstructured, ungrammatical text. Unsupervised extraction using reference sets would allow for the processing of volumes of new data previously unavailable because of their lack of structure.

References

[Agichtein and Ganti, 2004] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In *the Proceedings of KDD*, Seattle, Washington, August 2004. ACM Press.

[Bilenko and Mooney, 2003] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings KDD*, 2003.

[Califf and Mooney, 1999] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI*, pages 328–334, Orlando, Florida, August 1999.

[Cohen and Sarawagi, 2004] William Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of KDD*, Seattle, Washington, August 2004. ACM Press.

[Craswell *et al.*, 2000] Nick Craswell, Peter Bailey, and David Hawking. Server selection on the world wide web. In *Proceedings of the Conf. on Digital Libraries*, 2000.

[Crescenzi *et al.*, 2001] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of VLDB*, pages 109–118, 2001.

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

[Dingli *et al.*, 2003] Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks. Automatic semantic annotation using unsupervised information extraction and integration. In *Proceedings of the K-CAP Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[Jean *et al.*, 2005] Jiwoon Jean, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. In *Proceedings of CIKM*, 2005.

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

[Michelson and Knoblock, 2005] Matthew Michelson and Craig A. Knoblock. Semantic annotation of unstructured and ungrammatical text. In *Proceedings of IJCAI*, 2005.

[Michelson and Knoblock, 2006] Matthew Michelson and Craig A. Knoblock. Phoebus: A system for extracting and integrating data from unstructured and ungrammatical sources. In *Proceedings of AAAI, Intelligent Systems Demonstrations*, 2006.

[Minton *et al.*, 2005] Steven N. Minton, Claude Nanjo, Craig A. Knoblock, Martin Michalowski, and Matthew Michelson. A heterogeneous field matching method for record linkage. In *Proceedings of ICDM*, 2005.

[Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[Reeve and Han, 2005] Lawrence Reeve and Hyoil Han. Survey of semantic annotation platforms. In *Proceedings of ACM Symposium on Applied Computing*, 2005.

[Salton and McGill, 1983] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[Soderland, 1999] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[Vargas-Vera *et al.*, 2002] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proceedings of EKAW*, 2002.