**International Journal of Document Analysis and Recognition (IJDAR) manuscript No.**
(will be inserted by the editor)

**Matthew Michelson · Craig A. Knoblock**

# Unsupervised Information Extraction from Unstructured, Ungrammatical Data Sources on the World Wide Web

Received: March 2007 / Accepted: August 2007

**Abstract** Information extraction from unstructured, ungrammatical data such as classified listings is difficult because traditional structural and grammatical extraction methods do not apply. Previous work has exploited reference sets to aid such extraction, but it did so using supervised machine learning. In this paper, we present an unsupervised approach that both selects the relevant reference set(s) automatically and then uses it for unsupervised extraction. We validate our approach with experimental results that show our unsupervised extraction is competitive with supervised machine learning approaches, including the previous supervised approach that exploits reference sets.

**Keywords** Information Extraction · Unsupervised · Semantic Annotation · Information Integration · Unstructured Data Sources

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Air Force Office of Scientific Research under grant numbers FA9550-04-1-0105 and FA9550-07-1-0416, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. 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.

Matthew Michelson
University of Southern California
Information Sciences Institute
Tel.: +1-310-448-8719
Fax: +1-310-822-0751
E-mail: michelso@isi.edu

Craig A. Knoblock
University of Southern California
Information Sciences Institute
Tel.: +1-310-448-8786
Fax: +1-310-822-0751
E-mail: knoblock@isi.edu

## 1 Introduction

The huge amounts of unstructured and ungrammatical data on the World Wide Web could be useful if the information contained within them could be extracted. Examples of such data sources are internet classified listings, such as those on Craigslist,[1] internet auction listings such as those found on eBay, or internet forum postings like those posted to the Bidding For Travel forum.[2] We consider all of the listings within these sources as "posts." Figure 1 shows example posts from Craigslist. In the circled post we want to extract 02 as the year, M3 as the car model, and Convertible as part of the trim.

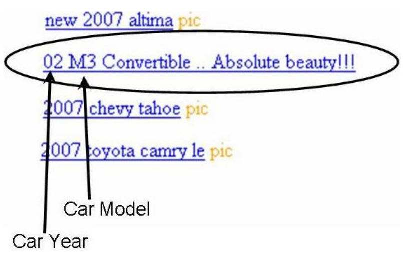

**Fig. 1** Posts from Craigslist

However, information extraction from posts poses difficulties. The posts are not structured enough for wrapper methods, such as HLRT wrappers [11]. Neither are the posts grammatical enough to support natural language based techniques, such as Amilcare [5]. In previous work, the Phoebus system [17] exploited reference sets to overcome these challenges. A reference set is a relational data set that contains entities and their attributes. An example of a reference set is a set of cars, each with a make, model, and year. The Phoebus algorithm exploits the reference set by first matching each

[1] www.craigslist.org
[2] www.BiddingForTravel.com

post to the members of the reference set, and then using the attributes from these matching members to aid the extraction. The main benefit of Phoebus is its ability to extract attributes without any assumptions about the structure of the text, because it does not consider grammar or structure for extraction.

However, the Phoebus system is limited because it requires much user input. First, the user provides the reference set. Then, the user labels matches between the posts and the members of the reference set. Finally, the user labels examples of the extracted attributes. Only after training, can the Phoebus system extract data from unstructured, ungrammatical data sources.

This paper presents an alternative, unsupervised approach to exploiting reference sets for extraction. Using reference sets allows for unsupervised extraction without any structural assumptions on the text. Our contributions replace each supervised step of the Phoebus algorithm with an unsupervised alternative, relieving the human dependence from all aspects of Phoebus' extraction algorithm. This operation improves the scalability, cost, and robustness of extraction. In our approach, first the reference sets are selected by the system from a growing repository of many reference sets. After choosing the reference sets, the system selects the matches using a vector-space model and performs the extraction in an unsupervised manner.

This work extends our previous work on unsupervised semantic annotation of unstructured sources [18]. In that work we used the average values of string similarities as one of the splitting criteria for finding matches versus non-matches between the reference set and the set of posts. In this work, we remove that restriction. Furthermore, we present extensive experiments in this paper investigating and justifying different heuristic choices such as thresholds and similarity metrics. In some cases, we are able to make generalizations about the types of similarity metrics that should be used at different steps in our approach. Lastly, we extend our previous work by describing how to collect reference sets automatically and how to use the technique presented in this paper to automatically include the unstructured, ungrammatical data sources in information integration systems.

Unsupervised Information Extraction (UIE) has recently witnessed significant progress [3,10,21]. However, current work on UIE relies on the redundancy of the extractions to learn patterns in order to make further extractions. Such patterns rely on the assumption that similar structures will occur again to make the learned extraction patterns useful. Initially, the approaches can be seeded with manual rules [3], example extractions [21], or sometimes nothing at all, relying solely on redundancy [10]. Regardless of how the extraction process starts, extractions are validated via redundancy, and they are then used to generalize patterns for extractions.

This approach to UIE differs from ours in important ways. First, the use of patterns makes assumptions about the structure of the data. We cannot make any structural assumptions because our target data for extraction is defined by its lack of structure and grammar. Second, since these systems are not clued into what can be extracted, they rely on redundancy for their confidence in their extractions. In our case, our confidence in the extracted values comes from their similarity to the reference-set attributes exploited during extraction. Lastly, the goals differ. Previous UIE systems seek to build knowledge bases through extraction. For instance, they aim to find all types of cars on the Web. Our extraction creates relational data, which allows us to classify and query posts. For this reason, we believe the previous work complements ours well because we could use their techniques to automatically build our reference sets.

## 2 Unsupervised Extraction of Unstructured, Ungrammatical Data

Our algorithm for unsupervised information extraction of unstructured, ungrammatical data has three distinct steps. Given a set of posts, the first step is to have the system choose the applicable reference sets from a repository of reference sets. Once the reference sets are chosen, the system must then match each post to members of the reference set, which allows it to use these members' attributes as clues to aid in the extraction. Finally, the system exploits these reference-set attributes to perform the unsupervised extraction.

### 2.1 Automatically Choosing the Reference Sets

Because our repository of reference sets grows over time, the system should choose the reference sets to exploit for a given set of posts. The algorithm chooses 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 the one with the most useful tokens in common with the posts. 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 each reference set in the repository as a single document and the set of posts as a single document. We calculate a similarity score between each reference set and the set of posts. Then, we sort the similarity scores in descending order, and traverse this list, computing the percent difference between the current similarity score and the next. If this percent difference is above a 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 as matches the current reference set and all reference sets that preceded it. 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.

**Table 1** Automatically choosing a reference set

```
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
    $SIM(r_i, p) > AVG(SIM(R, p))$
        Return $SIM(r_x, p), 1 > x > i$
Return nothing (No matching reference sets)
```

We use the percent difference as the splitting criterion between the relevant and irrelevant reference sets because it is a relative measure. Comparing only the actual similarity values might not capture how much better one reference set is as compared to another. Further, we require that the score at the splitting point be higher than the average score. This requirement is needed in cases where the scores are so small at the end of the list that their percent differences can suddenly increase, even with a small difference in score. This difference does not mean that we have found relevant reference sets, rather it just means that the next reference set is that much worse than the current, bad one.

By treating each reference set as a single document, the algorithm of Table 1 scales linearly with the size of the repository. That is, each reference set in the repository is scored against the posts only once. Furthermore, as the number of reference sets increases, the percent difference still determines which reference sets are relevant. If an irrelevant reference set is added to the repository, it will score low, so it will still be relatively that much worse than the relevant one. If a new relevant set is added, the percent difference between the new one and the one already chosen will be small, but both of them will still be much better than the irrelevant sets in the repository. Thus, the percent difference remains a good splitting point.

We do not require a specific similarity measure for this algorithm. Instead, our experiments of Section 3 find that certain classes of similarity metrics can perform well. So, rather than picking just one as the best we try many different metrics and draw conclusions about what types of similarity scores should be used and why.

## 2.2 Matching Posts to the Reference Set

After choosing the relevant reference sets, the algorithm matches each post to the best matching members of the reference set. When selecting multiple reference sets, the matching algorithm executes 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 higher ranked one to prevent redundant matching.

To match the reference set records to the posts, we employ a vector-space model. Using a vector-space model, rather than machine learning, makes the algorithm unsupervised. Furthermore, a vector-space model allows us to use information-retrieval techniques such as inverted indexes, which are fast and scalable.

In our model, we treat each post as a query and each record of the reference set as a document, and we use a token-based similarity to define their likeness. Again, we do not tie this part of the algorithm to a specific similarity metric because we find a class of token-based similarity metrics works well, which we justify in our experiments.

However, for our algorithm we modify the similarity metrics we use. Our modification considers two tokens as matching if their Jaro-Winkler [25] similarity is greater than a threshold. For example, consider the classic Dice similarity, defined over a post $p$ and a record of the reference set $r$, as:

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

If the threshold is 0.95, two tokens are put into the intersection of the modified Dice similarity if those two tokens have a Jaro-Winkler similarity above 0.95. This Jaro-Winkler modification captures tokens that might be misspelled or abbreviated, which is common in posts. The underlying assumption of the Jaro-Winkler metric is that certain attributes are more likely similar if they share a certain prefix. This works particularly well for proper nouns, which many reference set attributes are, such as “Honda Accord” cars. Using our modified similarity metric, we compare each post, $p_i$, to each member of the reference set and return the reference set records with the maximum score, called $r_{max_i}$. In our experiments, we vary this threshold to test its effect on the performance of matching the posts. We also justify using the Jaro-Winkler metric to modify the Dice similarity, rather than another edit distance metric.

However, because more than one reference set record can have a maximum similarity score with a post ($r_{max_i}$ is a set), an ambiguity problem exists with the attributes provided by the reference set records. For example, consider a post “Civic 2001 for sale, look!” If we have the following 3 matching reference records: {HONDA, CIVIC, 4 Dr LX, 2001}, {HONDA, CIVIC, 2 Dr LX, 2001} and {HONDA, CIVIC, EX, 2001}, then we have an ambiguity problem with the trim. We can confidently assign HONDA as the make, CIVIC as the model, and 2001 as the year, because all of the matching records agree on these attributes. We say that these attributes are “in agreement.” Yet, there is disagreement on the trim because we cannot determine which value is best for this attribute. All the reference records are equally acceptable from the vector-space perspective, but they differ in value for this attribute. Therefore, we remove from our annotation all attributes that do not agree across all matching reference set records (e.g., the trim in our example). Once this process executes, we have all of the attributes from the reference set that we can use for extraction. The full algorithm is shown in Table 2.

**Table 2** Vector-space approach to finding attributes in agreement

```
Given posts $P$ and reference set $R$
For all $p_i \in P$
    $r_{max_i} \leftarrow$ MAX( SIM ($p_i$,R))
    Remove attributes not in agreement from $r_{max_i}$
```

Selecting all the reference records with the maximum score, without pruning possible false positives, introduces noisy matches. These false positives occur because posts with small similarity scores still 'match' certain reference set records. For instance, the post "We pick up your used car" matches the Renault Le Car, Lincoln Town Car, and Isuzu Rodeo Pick-up, albeit with small similarity scores. However, since none of these attributes are in agreement, this post gets no annotation. Therefore, by using only the attributes in agreement, we essentially eliminate these false positive matches because no annotation will be returned for this post. That is, because no annotation is returned for such posts, it is as if there are no matching records for it. In this manner, by using only the attributes "in agreement," we separate the true matches from the false positives.

Once we have matched the posts to a reference set, we can query the posts structurally, as we would a database. This is a tremendous advantage over the traditional keyword search approach to searching unstructured, ungrammatical text. For example, keyword search cannot return records for which an attribute is missing, whereas our approach can. If a post were "2001 Accord for sale," and the keyword search was Honda, this record would not be returned. However, after matching posts to the reference set, if we select all records where the matched-reference-set attribute is "Honda" this record would be returned. Another benefit of matching the posts is that aggregate queries are possible. This mechanism is not supported by keyword search.

Perhaps one of the most useful aspects of matching posts to the reference set is that we can include the posts in an information-integration system (e.g., [14,24]). In particular, our approach is well suited for Local-as-View (LAV) information integration systems [13], which allow for easy inclusion of new sources by defining their "source description" as a view over known domain relations. For example, let us say we have a reference set of cars from Edmunds car buying guide for the years 1990-2005, in our repository. If we include this source in an LAV integration system, the source description is:

Edmunds(make, model, year, trim) :-
Cars(make, model, year, trim) $\wedge$
year $\geq$ 1995 $\wedge$ year $\leq$ 2005

Now, let us assume a set of posts comes in and we match them to the records of this Edmunds source. We had no idea previously how to include this source in our integration system, but after matching we can use the same source description of the matching reference set, along with a variable to output the "post" attribute, to define a source description of our unstructured, ungrammatical source. We can do this because we are appending records from the unstructured source with the attributes in agreement from the reference set matches. So, the new source description becomes:

UnstructuredSource(post, make, model, year, trim) :-
Cars(make, model, year, trim) $\wedge$
year $\geq$ 1995 $\wedge$ year $\leq$ 2005

In this manner, we can collect and include new sources of unstructured, ungrammatical text in LAV information integration systems without human intervention.

## 2.3 Unsupervised Extraction

Sometimes a user wants to see the actual extracted values for a given attribute, rather than the reference set attribute from the matching record. Therefore, once the system retrieves the attributes in agreement from the reference set matches, it exploits these attributes for information extraction. First, for each post, each token is compared to each of the attributes in agreement from the matches. The token is labeled as the attribute for which it has the maximum Jaro-Winkler similarity. We label a token as 'junk' if it receives a score of zero against all reference-set attributes.

However, this initial labeling generates noise because the attributes are labeled in isolation. To remedy this, we use our modified Dice similarity, and generate a baseline score between the extracted field and the reference-set field. Then, we go through the extracted attribute, removing one token at a time, and calculate a new Dice similarity value. If this new score is higher than the baseline, the removed token is a candidate for permanent removal. Once all tokens are processed in this way, the candidate for removal that yielded the highest new score is removed. Then, we update the baseline to the new score, and repeat the process. When none of the tokens yield improved scores when removed, this process terminates. This cleaning is shown in Table 3. For this part of the algorithm, we use the modified Dice since our experiments in the annotation section show this to be one of the best performing similarity metrics. Note, our unsupervised extraction is $O(\|p\|^2)$ per post, where $\|p\|$ is the number of tokens in the post, because, at worst, all tokens are initially labeled, and then each one is removed, one at a time. However, because most posts are relatively short, this running time is acceptable.

The whole multi-pass procedure for unsupervised extraction is shown in Figure 2. By exploiting reference sets, we perform unsupervised information extraction without assumptions regarding repeated structure in the data.

**Table 3** Cleaning an extracted attribute

```
CLEAN-ATTRIBUTE(extracted attribute $E$, reference set attribute $A$)
ExtCands ← {}
baseline ← DICE(E,A)
For all tokens $e_i \in E$
    score ← DICE(E/$e_i$,A)
    IF score > baseline
        ExtCands ← ExtCands ∪ $e_i$
IF ExtCands is empty
    return $E$
ELSE
    select $c_i \in$ ExtCands that yieds max score
    CLEAN-ATTRIBUTE($E/c_i$,A)
```

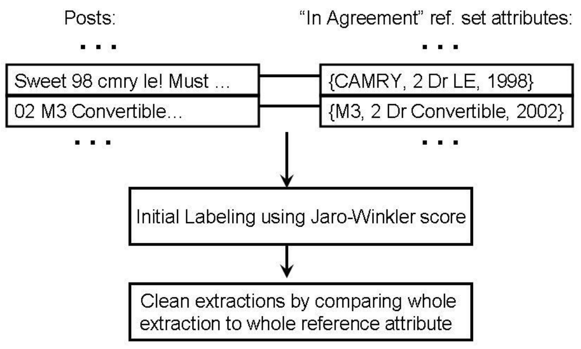
Posts:
"In Agreement" ref. set attributes:
Sweet 98 cmry le! Must ...
02 M3 Convertible...
{CAMRY, 2 Dr LE, 1998}
{M3, 2 Dr Convertible, 2002}
Initial Labeling using Jaro-Winkler score
Clean extractions by comparing whole extraction to whole reference attribute

**Fig. 2** Unsupervised extraction with reference sets

## 3 Experimental Results

This section presents results for our unsupervised approach to selecting a reference set, finding the attributes in agreement, and exploiting these attributes for extraction. Before examining the results, we describe the reference sets and posts used in testing.

### 3.1 Reference Sets

We use six reference sets, most of which have been used in the past. First, we use the *Hotels* reference set from [17], which consists of 132 hotels each with a star rating, a hotel name, and a local area. Another reference set from the same paper is the *Comics* reference set, which has 918 Fantastic Four and Incredible Hulk comics from the Comic Books Price guide, each with a title, issue number, and publisher. We also use two restaurant reference sets, which were both used previously for record linkage [2]. One we call *Fodors*, which contains 534 restaurant records, each with a name, address, city, and cuisine. The other is *Zagat*, with 330 records.

We also have two reference sets of cars. The first, called *Cars*, contains 20,076 cars from the Edmunds Car Buying Guide for 1990-2005. The attributes in this set are the make, model, year, and trim. 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. We call this combined set of 27,006 records the *Cars* reference set. The other reference set of cars also has the attributes make, model, year, and trim. However, it is a subset of the cars covered by *Cars*. This data set comes from the Kelly Blue Book car pricing service containing 2,777 records for Japanese and Korean cars from 1990-2003. We call this set *KBBCars*. This data set has also been used in the record linkage community [20]. A summary of the reference sets is given in Table 4.

### 3.2 Post Sets

We chose sets of posts to test different cases that exist for finding the appropriate reference sets. One set of posts matches only a single reference set in our collection. It contains 1,125 posts from the forum Bidding For Travel. These posts, called *BFT*, match only the *Hotels* reference set. This data set was used previously in [17].

Because our approach can also select multiple relevant reference sets, we use a set of posts that matches both reference sets of cars. This set, which we call *Craig's Cars*, contains 2,568 car posts from Craigslist classifieds. Note that while there may be multiple, appropriate reference sets, they also might have an internal ranking. In this case we expect that both the *Cars* and *KBBCars* reference sets are selected, but *Cars* should be ranked first.

Lastly, we must examine whether the algorithm suggests that no relevant reference sets exist in our repository. To test this feature, we collected 1,099 posts about boats from Craigslist, called *Craig's Boats*. Boats are similar enough to cars to make this task non-trivial, because boats and cars are both made by Honda, for example, so that keyword appears in both sets of posts. However, boats also differ from each of the reference sets so that no reference set should be selected. All three sets of posts are summarized in Table 5.

**Table 4** Reference Set Descriptions

| **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 5** Posts Set Descriptions

| **Name** | **Source** | **Website** | **Reference Set Match** | **Records** |
|---|---|---|---|---|
| BFT | Bidding For Travel | www.biddingfortravel.com | Hotels | 1,125 |
| Craig's Cars | Craigs List Cars | www.craigslist.org | Cars, KBBCars | 2,568 |
| Craig's Boats | Craigs List Boats | www.craigslist.org | | 1,099 |

## 3.3 Results

As stated previously, our algorithm for choosing the reference sets is not tied to a particular similarity function. Rather, we apply the algorithm using many different similarity metrics and draw conclusions about which ones work best and why. This gives us an idea of what types of metrics we can plug in and have the algorithm perform accurately. Note that for the following metrics: Jensen-Shannon distance, Jaro-Winkler similarity, TF/IDF, and Jaro-Winkler TF/IDF, we use the SecondString package's implementation [6].

The first metric we use is the Jensen-Shannon distance (JSD) [15]. This information-theoretic metric quantifies the difference in probability distributions between the tokens in the reference set and those in the set of posts. Because JSD requires probability distributions, we define our distributions as the likelihood of tokens occurring in each document.

Table 6 shows our results for choosing relevant reference sets using JSD. The reference set names in bold reflect those that are chosen as appropriate. (This means Craig's Boats should have no bold names.) The scores in bold are the similarity scores for the chosen reference sets. The percent difference in bold is the point at which the algorithm breaks out and returns the appropriate reference sets. In particular, using JSD we successfully identify the multiple cases where we might have a single appropriate reference set, multiple reference sets, or no reference set. Note that, in all experiments we maintain the percent difference splitting threshold at 0.6.

The next metric we use is cosine similarity using TF/IDF for the weights. These results are shown in Table 7. Similarly to JSD, TF/IDF is able to identify all of the cases correctly. However, in choosing both car reference sets for the *Craig's Cars* posts, TF/IDF incorrectly determines the internal ranking, placing KBBCars ahead of Cars. This is probably due to the IDF weights that are calculated for the reference sets. Although we treat the set of posts and the reference set as a single document for comparison, the IDF weights are based on individual

**Table 6** Using Jensen-Shannon distance 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 | |

| Craig's Cars | | |
|---|---|---|
| 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 | |

| Craig's Boats | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| Cars | 0.251 | 0.513 |
| Fodors | 0.166 | 0.144 |
| KBBCars | 0.145 | 0.088 |
| Comics | 0.133 | 0.025 |
| Zagat | 0.130 | 0.544 |
| Hotels | 0.084 | |
| Average | 0.152 | |

records in the reference set. Therefore, since Cars is a superset of KBBCars, certain tokens are weighted more in KBBCars than in Cars, resulting in higher matching scores. These results also justify the need for a double stopping criterion. It is not sufficient to only consider the percent difference as an indicator of relative superiority amongst the reference sets. The scores must also be compared to an average to ensure that the algorithm does not errantly choose a bad reference set simply because it is relatively better than an even worse one. The last two rows of the *Craig's Boats* posts and the *BFT Posts* in Table 7 show this behavior.

We also tried a simpler bag-of-words metric that does not use any sort of token probabilities or weights. To do this, we use the Jaccard similarity, which is defined as the tokens in common divided by the union of the token sets. That is, given token set $S$ and token set $T$, Jaccard is:

$$Jaccard(S,T) = \frac{S \cap T}{S \cup T} \tag{1}$$

The results using the Jaccard similarity are shown in Table 8. One of the most surprising aspects of these

**Table 7** Using TF/IDF as the similarity measure

| BFT Posts | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| **Hotels** | **0.500** | **1.743** |
| Fodors | 0.182 | 0.318 |
| Comics | 0.134 | 0.029 |
| Zagat | 0.134 | 0.330 |
| Cars | 0.100 | 1.893 |
| KBBCars | 0.035 | |
| Average | 0.182 | |

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

| Craig's Boats | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| Cars | 0.200 | 0.189 |
| Comics | 0.168 | 0.220 |
| Fodors | 0.138 | 0.296 |
| Zagat | 0.107 | 0.015 |
| KBBCars | 0.105 | 0.866 |
| Hotels | 0.056 | |
| Average | 0.129 | |

results is that the Jaccard similarity actually does pretty well. It gets all but one of the cases correct. It is able to link the *BFT Posts* to the Hotels set only and it is able to determine that no reference set is appropriate for the *Craig's Boats* posts. However, with the *Craig's Cars* posts, it is only able to determine one of the reference sets. It ranks the Fodors and Zagat restaurants ahead of KBBCars because the restaurants have city names in common with some of the classified car listings. However, it is unable to determine that city name tokens are not as important as car makes and models. This is a problem if, for instance, the post is small and it contains a few more city name tokens than car tokens.

**Table 8** Using Jaccard as the similarity measure

| BFT Posts | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| **Hotels** | **0.272** | **1.489** |
| Comics | 0.109 | 0.166 |
| Fodors | 0.094 | 0.004 |
| Zagat | 0.093 | 0.640 |
| Cars | 0.057 | 0.520 |
| KBBCars | 0.037 | |
| Average | 0.110 | |

| Craig's Cars | | |
|---|---|---|
| Ref. Set | Score | % Diff |
| **Cars** | **0.207** | **1.339** |
| Fodors | 0.088 | 0.126 |
| Zagat | 0.078 | 0.005 |
| KBBCars | 0.078 | 0.089 |
| Comics | 0.072 | 2.536 |
| Hotels | 0.020 | |
| Average | 0.091 | |

| Craig's Boats | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| Cars | 0.129 | 0.366 |
| Comics | 0.095 | 0.347 |
| Fodors | 0.070 | 0.112 |
| Zagat | 0.063 | 0.261 |
| KBBCars | 0.050 | 1.917 |
| Hotels | 0.017 | |
| Average | 0.071 | |

Lastly, we investigate whether requiring that tokens match strictly, as in the above metrics, is more useful than a soft matching technique that considers tokens matching based on string similarities. To test this idea we use a modified version of TF/IDF where tokens are considered a match when their Jaro-Winkler score is greater than 0.9. These results are shown in Table 9. This metric is able to correctly retrieve the reference set for the *BFT Posts*, and for the *Craig's Boats* posts this metric chooses no appropriate reference set. However, for the Craig's Boats case, this metric is close to returning many incorrect reference sets since with the Zagat reference set the score is very close to the average while the percent difference is huge. The largest failure is with the *Craig's Cars* posts. For this set of posts the ordering of the reference sets is correct because the Cars and KBBCars have the highest scores, but no reference set is chosen because the percent difference is never above the threshold with a similarity score above the averge. The percent differences are low because of the soft nature of the token matching. For example, the Comics contains many matches because the issue number of the comics, such as #99, #199, #299, etc. match an abbreviated car year in a post such as '99. So, the similarity scores between the cars and comics reference sets are close. From these sets of results we see that it is better to have strict token matches, although they may be less frequent. The strict nature of the matches ensures that the tokens particular to a reference set are used for matches, which helps differentiate reference sets.

**Table 9** Using Jaro-Winkler TF/IDF as the similarity measure

| BFT Posts | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| **Hotels** | **0.593** | **1.232** |
| Fodors | 0.266 | 0.173 |
| Zagat | 0.227 | 0.110 |
| Cars | 0.204 | 0.068 |
| Comics | 0.191 | 0.777 |
| KBBCars | 0.108 | |
| Average | 0.265 | |

| Craig's Cars | | |
|---|---|---|
| Ref. Set | Score | % Diff |
| Cars | 0.699 | 0.174 |
| KBBCars | 0.595 | 0.174 |
| Comics | 0.505 | 0.050 |
| Fodors | 0.481 | 0.107 |
| Zagat | 0.435 | 0.948 |
| Hotels | 0.223 | |
| Average | 0.490 | |

| Craig's Boats | | |
|---|---|---|
| Ref. Set | Score | % Diff. |
| Cars | 0.469 | 0.096 |
| Comics | 0.428 | 0.106 |
| Fodors | 0.387 | 0.110 |
| KBBCars | 0.349 | 0.082 |
| Zagat | 0.322 | 1.212 |
| Hotels | 0.146 | |
| Average | 0.350 | |

The performance of each metric is shown in Table 10. We see each metric and whether or not it correctly identified the reference set(s) for that set of posts. In the case of Craig's Boats, we consider it correct if no reference set is chosen. The last column shows whether the method was able to also rank the chosen reference sets correctly for the Craig's Cars posts.

**Table 10** A summary of each method choosing reference sets

| Method | BFT | Craig's Boats | Craig's Cars | Craig's Cars rank |
|---|---|---|---|---|
| JSD | √ | √ | √ | √ |
| TF/IDF | √ | √ | √ | X |
| J-W TF/IDF | √ | √ | X | X |
| Jaccard | √ | √ | X | X |

Based on these sets of results, we draw some conclusions about which metrics should be used and why. Comparing the Jaccard similarity results to the success of both JSD and TF/IDF, we see that it is necessary to include some notion of importance regarding the matching tokens. The results argue that probability distributions of the tokens as defined in JSD are a better metric, since TF/IDF can be overly sensitive, i.e., ignoring tokens that may be important, even though they are frequent. This claim is also justified by the fact that using JSD we do not run into the situation where we need to use the average stopping criterion, although we do with TF/IDF. However, since JSD and TF/IDF both do well, we can say that if a similarity metric can differentiate important tokens from those that are not, then it can be used successfully in our algorithm to choose reference sets. This is why we do not tie this algorithm to any particular metric, since many could work. Another interesting aspect of these results is the poor performance of TF/IDF using the Jaro-Winkler modification. It seems that boosting the number of tokens that match by using a less strict token matching method actually harms the ability to differentiate between reference sets. This suggests that the tokens that define reference sets need to be emphasized by matching them exactly. Lastly, across different domains and even across different similarity metrics, we see our chosen threshold of 0.6 is appropriate for the cases where the algorithm chooses the correct reference set. In all of the cases where the correct reference set(s) is chosen, this threshold is exceeded.

Once the relevant reference sets are chosen, we use them to find the attributes in agreement for the different sets of posts, because we use these attributes during extraction. For the set of posts with reference sets (i.e., BFT and Craig's Cars), we compare the attributes in agreement found by the vector-space model to the attributes in agreement for the true matches between the posts and the reference set. To evaluate the annotation, we use the traditional measures of precision, recall, and F-measure (i.e., the harmonic mean between precision and recall). A correct match occurs when the attributes match between the true matches and the predictions of the vector-space model. As stated previously, we tried multiple modified token-similarity metrics to draw conclusions about what types of metrics work and why.

Table 11 shows our results using the modified Dice similarity for the BFT and Craig's Cars posts, varying the Jaro-Winkler token-match threshold from 0.85 to 0.99. That is, above this threshold two tokens will be considered a match for the modified Dice. In both cases we see an improvement in F-measure as the threshold increases. This is because more relevant tokens are being included when the threshold increases. If the threshold is low then many irrelevant tokens are considered matches, so the algorithm makes incorrect record level matches. For instance, as the threshold becomes low, the results for the year attribute drop steeply because often the difference in years is a single digit, which would often yield errantly matching tokens for a low threshold string similarity. Since errant matches are ignored (because they are not "in agreement") the scores are low. Note, however, that once the threshold becomes too high, at 0.99, the results start to decrease. This is because now the edit-distance is too restrictive so it is not capturing some of the possible matching tokens that might be slightly misspelled.

The only attribute where the F-measure increases at 0.99 versus 0.95 is the year attribute of the cars. In this case, the recall slightly increases at 0.99, but the precisions are almost the same, yielding a higher F-measure for 0.99. This is due to more year values being "in agreement" with the higher threshold since there will be less variation in terms of which reference set values can match for this attribute, so those that do will likely be in agreement. For an example where 0.95 includes years that are not in agreement with high Jaro-Winkler scores, consider a post with the token "19964" which might be a price or a year. If the reference set record's year attribute is "1994," the Jaro-Winkler score between "19964" and "1994" is 0.953. If the reference set record's year is "1996" the Jaro-Winkler score is 0.96. In both cases, a threshold of 0.95 includes both years, so if this post matches two reference set records with the same make, model and trim, but differing years of 1994 and 1996, then the year is discarded because it is not in agreement. We almost see the same behavior with the trim attribute as well. This is because with both of these attributes, a single difference in a character, say "LX" versus "DX" for a trim (or a digit for the year) yields a completely different attribute, which can then become not "in agreement."

Table 12 shows our results using the modified Jaccard similarity. As with the Dice similarity, the modification is such that two tokens are put into the intersection of the Jaccard similarity if their Jaro-Winkler is above the threshold. The most striking result is that the scores match exactly to those using the Dice similarity. The Dice similarity and Jaccard similarity can be used interchangeably. Further investigation revealed that the actual similarity scores between the posts and their reference set matches are different, which should be the case, but the resulting attributes that are "in agreement" are the same using either metric. Therefore, they yield the same annotation from the matches.

Table 13 shows results using the Jaro-Winkler TF/IDF similarity measure. Similarly to the other metrics, for the BFT domain we see an improvement in F-measure as the threshold increases, until the threshold peaks at 0.95 after which it decreases in accuracy. However, with the Cars domain the modified TF/IDF seems to perform the best with a threshold of 0.99.

From these results, across all metrics, a threshold of 0.95 performs the best for the BFT domain. In the Cars domain, the 0.95 threshold works best for the modified Dice and Jaccard, and at this threshold both methods

**Table 11** Annotation results using modified Dice similarity

| **BFT posts** | | | | |
|---|---|---|---|---|
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | Hotel name | 76.46 | 78.13 | 77.29 |
| | Star rating | 80.74 | 77.86 | 79.27 |
| | Local area | 88.04 | 85.46 | 86.73 |
| 0.9 | Hotel name | 88.23 | 88.49 | 88.36 |
| | Star rating | 91.73 | 87.64 | 89.64 |
| | Local area | 93.09 | 89.36 | 91.19 |
| 0.95 | Hotel name | 88.42 | 88.51 | **88.47** |
| | Star rating | 92.32 | 87.79 | **90.00** |
| | Local area | 93.97 | 89.44 | **91.65** |
| 0.99 | Hotel name | 87.84 | 88.36 | 88.10 |
| | Star rating | 92.02 | 87.76 | 89.84 |
| | Local area | 93.39 | 89.39 | 91.34 |
| **Craig's Cars posts** | | | | |
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | make | 81.57 | 77.64 | 79.55 |
| | model | 57.61 | 61.15 | 59.33 |
| | trim | 38.76 | 29.80 | 33.70 |
| | year | 2.38 | 8.14 | 3.68 |
| 0.9 | make | 88.41 | 82.82 | 85.52 |
| | model | 76.28 | 77.16 | 76.72 |
| | trim | 65.57 | 48.12 | 55.51 |
| | year | 69.45 | 82.88 | 75.58 |
| 0.95 | make | 93.96 | 86.35 | **89.99** |
| | model | 82.62 | 81.35 | **81.98** |
| | trim | 71.62 | 51.95 | **60.22** |
| | year | 78.86 | 91.01 | 84.50 |
| 0.99 | make | 93.51 | 86.33 | 89.78 |
| | model | 81.29 | 81.25 | 81.27 |
| | trim | 71.75 | 51.85 | 60.20 |
| | year | 79.14 | 90.94 | **84.63** |

**Table 12** Annotation results using modified Jaccard similarity

| **BFT posts** | | | | |
|---|---|---|---|---|
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | Hotel name | 76.46 | 78.13 | 77.29 |
| | Star rating | 80.74 | 77.86 | 79.27 |
| | Local area | 88.04 | 85.46 | 86.73 |
| 0.9 | Hotel name | 88.23 | 88.49 | 88.36 |
| | Star rating | 91.73 | 87.64 | 89.64 |
| | Local area | 93.09 | 89.36 | 91.19 |
| 0.95 | Hotel name | 88.42 | 88.51 | **88.47** |
| | Star rating | 92.32 | 87.79 | **90.00** |
| | Local area | 93.97 | 89.44 | **91.65** |
| 0.99 | Hotel name | 87.84 | 88.36 | 88.10 |
| | Star rating | 92.02 | 87.76 | 89.84 |
| | Local area | 93.39 | 89.39 | 91.34 |
| **Craig's Cars posts** | | | | |
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | make | 81.57 | 77.64 | 79.55 |
| | model | 57.61 | 61.15 | 59.33 |
| | trim | 38.76 | 29.80 | 33.70 |
| | year | 2.38 | 8.14 | 3.68 |
| 0.9 | make | 88.41 | 82.82 | 85.52 |
| | model | 76.28 | 77.16 | 76.72 |
| | trim | 65.57 | 48.12 | 55.51 |
| | year | 69.45 | 82.88 | 75.58 |
| 0.95 | make | 93.96 | 86.35 | **89.99** |
| | model | 82.62 | 81.35 | **81.98** |
| | trim | 71.62 | 51.95 | **60.22** |
| | year | 78.86 | 91.01 | 84.50 |
| 0.99 | make | 93.51 | 86.33 | 89.78 |
| | model | 81.29 | 81.25 | 81.27 |
| | trim | 71.75 | 51.85 | 60.20 |
| | year | 79.14 | 90.94 | **84.63** |

outperform the TF/IDF metric, even when its threshold is at its best at 0.99. Therefore, the most meaningful comparisons between the different metrics can be made at the threshold 0.95. In the BFT domain, the modified TF/IDF outperforms the Dice and Jaccard metrics, until this threshold of 0.95. At this threshold level, the Jaccard and Dice metrics outperform the TF/IDF metric on the two harder attributes, the hotel name and the hotel area. In the Cars domain, the TF/IDF metric is outperformed at every threshold level, except on the year attribute. Interestingly, at the lowest threshold levels TF/IDF performs terribly because the tokens that match in the computation have very low IDF scores since they match so many other tokens in the corpus, resulting in very low TF/IDF scores. If the scores are low, then many records will be returned and almost no attributes will ever be in agreement, yielding very few correct annotations.

**Table 13** Annotation results using modified TF/IDF similarity

| **BFT posts** | | | | |
|---|---|---|---|---|
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | Hotel name | 85.89 | 78.91 | 82.25 |
| | Star rating | 92.32 | 84.81 | 88.40 |
| | Local area | 94.55 | 86.86 | 90.54 |
| 0.9 | Hotel name | 90.76 | 83.83 | 87.16 |
| | Star rating | 95.33 | 88.05 | 91.55 |
| | Local area | 94.36 | 87.15 | 90.61 |
| 0.95 | Hotel name | 90.47 | 83.63 | **86.92** |
| | Star rating | 97.18 | 89.84 | **93.36** |
| | Local area | 94.55 | 87.41 | **90.84** |
| 0.99 | Hotel name | 89.69 | 82.91 | 86.17 |
| | Star rating | 96.89 | 89.57 | 93.08 |
| | Local area | 93.68 | 86.60 | 90.00 |
| **Craig's Cars posts** | | | | |
| Threshold | Attribute | Recall | Precision | F-Measure |
| 0.85 | make | 51.14 | 44.51 | 47.60 |
| | model | 41.93 | 35.54 | 38.47 |
| | trim | 43.10 | 15.50 | 22.80 |
| | year | 35.58 | 29.18 | 32.06 |
| 0.9 | make | 67.07 | 58.53 | 62.51 |
| | model | 61.79 | 52.48 | 56.76 |
| | trim | 67.81 | 22.90 | 34.24 |
| | year | 58.48 | 48.24 | 52.87 |
| 0.95 | make | 88.55 | 77.30 | 82.54 |
| | model | 84.55 | 71.84 | 77.68 |
| | trim | 81.21 | 26.86 | **40.37** |
| | year | 76.29 | 62.78 | 68.88 |
| 0.99 | make | 88.90 | 77.65 | **82.90** |
| | model | 84.74 | 72.02 | **77.86** |
| | trim | 80.29 | 26.52 | 39.87 |
| | year | 76.67 | 63.12 | **69.24** |

These three sets of results allows us to draw some conclusions about the utility of different metrics for our vector-space matching task. The biggest difference between the TF/IDF metric and the other two is that the TF/IDF metric uses term weights computed from the set of tokens in the reference set. The key insight of IDF weights are their ability to discern meaningful tokens from non-meaningful ones based on the frequency. The assumption is that more meaningful tokens occur less frequently. However, almost all tokens in a reference set are meaningful, and it is sometimes the case that very meaningful tokens in a reference set occur very often. The most glaring instances of this occur with the make and year attributes in the Cars reference set used for the Craig's Cars posts. Makes such as "Honda" occur quite frequently in the data set, and given that for 20,076 car records the years only range from 1990 to

2005, the re-occurrence of the same year tokens are very, very frequent. These attributes will be deemphasized significantly because of their frequency. If the matching token metric ignores the year, this attribute will often not be in agreement since multiple records of the same car for different years will be returned. Thus, TF/IDF has low scores for the year attribute. TF/IDF also creates problems by overemphasizing unimportant tokens that occur rarely. Consider the following post, *"Near New Ford Expedition XLT 4WD with Brand New 22 Wheels!!! (Redwood City - Sale This Weekend !!!) $26850"* which TF/IDF matches to the reference set record {VOLKSWAGEN, JETTA, 4 Dr City Sedan, 1995}. In this case, the very rare token "City" causes an errant match because it is weighted so heavily. In the case of the BFT posts, since the Hotel reference set has few commonly occurring tokens amongst a small set of records, this phenomena is not as observable. Since weights, whether based on TF/IDF or probabilities, rely on frequencies, such an issue will likely occur in most matching methods that rely on the frequencies of tokens to determine their importance.

Therefore, we draw the following conclusions. In the matching step, an edit-distance should be used to make soft matches between the tokens of the post and the reference set records. If the Jaro-Winkler metric is used, the threshold should be set to 0.95, since that yields the highest improvement using the best metrics. Lastly, and most importantly, reference sets do not adhere to the assumptions made by weighting schemes, so only metrics that do not use such schemes, such as the Dice and Jaccard similarities, should be used, rather than TF/IDF.

Since the modified Dice and Jaccard metrics work best using a threshold of 0.95, we now compare those results to our previous work on the Phoebus system [17], showing that our unsupervised approach to semantic annotation is competitive with a machine learning approach. We compare against the F-measure of the record linkage results from the Phoebus paper because that work uses the attributes of the matching record(s) from the reference set as semantic annotation. Since the Phoebus work only reports BFT results, we mirror the experimental procedure for the Craig's Cars posts, running Phoebus 10 times using 10% of the data for training. In the cars case, we did not use 30% of the data for training because that produced a larger number of training pairs than Phoebus was built to handle. Table 14 reports the F-measure of Phoebus's record linkage as *Ph. F-Mes.* Note that *Ph. F-Mes* is the same for all attributes within a domain because it is the F-measure for record linkage.

Although a direct comparison between the two systems is skewed because our system is unsupervised while Phoebus is not, the results nonetheless are intriguing. Even though our system selected the reference sets itself and our vector-space model is unsupervised, our unsupervised system remains competitive in selecting the correct attributes to exploit for extraction.

**Table 14** Results of semantic annotation using modified Dice similarity versus Phoebus

| BFT Posts | | | | |
|---|---|---|---|---|
| Attribute | Recall | Prec. | F-Measure | Ph. F-Mes. |
| Hotel Name | 88.42 | 88.51 | 88.47 | 92.68 |
| Star Rating | 92.31 | 87.79 | 90.00 | 92.68 |
| Local Area | 93.97 | 89.44 | 91.65 | 92.68 |
| Craig's Cars Posts | | | | |
| Make | 93.96 | 86.35 | 89.99 | 77.04 |
| Model | 82.62 | 81.35 | 81.98 | 77.04 |
| Trim | 71.62 | 51.95 | 60.22 | 77.04 |
| Year | 78.86 | 91.01 | 84.50 | 77.04 |

While the above results show our unsupervised approach is competitive with supervised approaches, we still need to justify our use of the Jaro-Winkler. Earlier we stated that the Jaro-Winkler metric emphasizes matching proper nouns, rather than more common words, because it considers the prefix of words to be an important indicator of matching. This is in contrast to traditional edit distances that define a transformation over a whole string, which are better for generic words where the prefix might not indicate matching. Table 15 compares using Dice similarity modified with the Jaro-Winkler metric to Dice modified with the Smith-Waterman distance [23]. (Smith-Waterman is a classic edit distance originally developed to align DNA sequences.) As with the Jaro-Winkler score, if two tokens have a Smith-Waterman distance above 0.95 they are considered a match in the modified Dice similarity. As the table shows, the Jaro-Winkler Dice score outperforms the Smith-Waterman variant. Since many of the reference set attributes are proper nouns, the Jaro-Winkler is better suited for matching, which is especially apparent when using the Cars reference set.

**Table 15** Modified Dice using Jaro-Winkler versus Smith-Waterman

| **BFT posts** | | | | |
|---|---|---|---|---|
| Method | Attribute | Recall | Precision | F-Measure |
| Jaro-Winkler | Hotel name | 88.42 | 88.51 | **88.47** |
| | Star rating | 92.32 | 87.79 | **90.00** |
| | Local area | 93.97 | 89.44 | **91.65** |
| Smith-Waterman | Hotel name | 67.80 | 69.56 | 68.67 |
| | Star rating | 75.39 | 73.88 | 74.63 |
| | Local area | 84.34 | 81.72 | 83.01 |
| **Craig's Cars posts** | | | | |
| Method | Attribute | Recall | Precision | F-Measure |
| Jaro-Winkler | make | 93.96 | 86.35 | **89.99** |
| | model | 82.62 | 81.35 | **81.98** |
| | trim | 71.62 | 51.95 | **60.22** |
| | year | 78.86 | 91.01 | **84.50** |
| Smith-Waterman | make | 24.12 | 27.18 | 25.56 |
| | model | 14.71 | 18.82 | 16.52 |
| | trim | 11.17 | 5.81 | 7.64 |
| | year | 29.17 | 52.30 | 37.45 |

Lastly, Table 16 shows our field-level results for performing information extraction exploiting the attributes in agreement. For field-level results, an extraction is correct if and only if all of the tokens that compose that field in the post are correctly labeled, without extra tokens.

Our results are shown as UIE. We compare our results to those obtained using three supervised systems: the Phoebus system, Conditional Random Fields as implemented in MALLET [16], and Amilcare [5], which relies strongly on NLP. We train MALLET and Amilcare on 30% of the data for BFT and Craig's Cars posts. We present the precision, recall, and F-measure for the extractions in Table 16.

**Table 16** Extraction results

| Craig's Cars Posts | | | | |
|---|---|---|---|---|
| Attribute | | Recall | Prec. | F-Mes. |
| Make | UIE | 95.99 | 100.00 | 97.95 |
| | MALLET | 85.68 | 95.69 | 90.39 |
| | Phoebus | 98.21 | 99.93 | **99.06** |
| | Amilcare | 97.58 | 91.76 | 94.57 |
| Model | UIE | 83.02 | 95.01 | 88.61 |
| | MALLET | 78.76 | 91.21 | 84.52 |
| | Phoebus | 92.61 | 96.97 | **94.59** |
| | Amilcare | 78.44 | 84.31 | 81.24 |
| Trim | UIE | 39.52 | 66.94 | 49.70 |
| | MALLET | 55.94 | 66.49 | 60.57 |
| | Phoebus | 63.12 | 70.15 | **66.43** |
| | Amilcare | 27.21 | 53.99 | 35.94 |
| Year | UIE | 76.28 | 99.80 | 86.47 |
| | MALLET | 91.12 | 76.78 | 83.31 |
| | Phoebus | 88.48 | 98.24 | **93.08** |
| | Amilcare | 86.32 | 91.92 | 88.97 |

| BFT Posts | | | | |
|---|---|---|---|---|
| Attribute | | Recall | Prec. | F-Mes. |
| Star Rating | UIE | 83.94 | 99.44 | 91.03 |
| | MALLET | 97.16 | 96.55 | 96.85 |
| | Phoebus | 97.39 | 97.01 | **97.20** |
| | Amilcare | 95.58 | 97.35 | 96.46 |
| Hotel Name | UIE | 70.09 | 77.16 | 73.46 |
| | MALLET | 74.43 | 84.86 | **79.29** |
| | Phoebus | 77.27 | 75.18 | 76.21 |
| | Amilcare | 58.96 | 67.44 | 62.91 |
| Local Area | UIE | 62.23 | 85.36 | 71.98 |
| | MALLET | 78.62 | 83.58 | 80.52 |
| | Phoebus | 83.73 | 84.76 | **84.22** |
| | Amilcare | 64.78 | 71.59 | 68.01 |

As with the semantic annotation results, the comparison between systems is not direct, yet again our unsupervised approach remains competitive. In fact, over the seven attributes, our UIE results have the lowest F-Measure only *once*, for the star rating of a hotel post. Interestingly, our approach has the highest precision for four of the attributes, but for three of those four, it also has the lowest recall. These results suggest that if we can increase the discovery of the extractions, we can increase the recall, and get even better results. For example, one of the attributes with the highest precision but lowest recall is the hotel area. In this attribute, we often see acronyms such as "AP" for airport or "DT" for downtown. Supervised systems can be trained to identify such cases, but our approach would need some sort of acronym and synonym discovery method to find those. We plan to enhance our system with such functionality in the future.

The largest differences in the F-Measure occur when the attribute to be extracted comes from a field that is often not in agreement. For example, in the Craig's Cars domain, the trim is often not in agreement, leading to poor extraction results. Nonetheless, we feel that the cost of labeling data for the supervised systems outweighs the differences in accuracy from our system. As information extraction begins to scale to huge amounts of data on the World Wide Web, unsupervised information extraction methods will be needed because the cost of labeling the data will be overwhelming. We believe these results validate such an approach.

## 4 Related Work

Semantic annotation is an active field of research, particularly as the popularity of the Semantic Web increases. According to a recent survey [22], systems that perform semantic annotation separate into three categories: rule-based, pattern-based, and wrapper induction methods. However, the rule-based and pattern-based methods rely on regularity within the text, which is not the case with posts. Also, beyond exploiting regular structure, the wrapper induction methods use supervised machine learning instead of unsupervised methods.

The system closest to ours is SemTag [9], which first identifies tokens of interest in the text, and then labels them using the TAP taxonomy, which is similar to our reference sets. This taxonomy is carefully crafted, which gives it good accuracy and meaning. In contrast, our reference sets are flexible as we incorporate any that we can automatically collect. Including new reference sets in our repository has no effect on the data already in it (because the reference sets are independent). Although our data collection is not as careful as using a full taxonomy, we can much more easily and quickly gather many reference sets, greatly increasing our coverage of items we can annotate.

Further, SemTag focuses on disambiguation which our approach avoids. If one looks up the token "Jaguar" it might refer to a car or an animal, because SemTag disambiguates after labeling. In our case, we perform disambiguation before the labeling procedure, during the selection of the relevant reference sets. If we had reference sets of animals and cars, and we chose cars as the relevant one, the synonymy is avoided since animal records are ruled out.

Selecting reference sets is similar to resource selection in distributed information retrieval, sometimes used for the "hidden web." In resource selection, different servers are chosen from a set of servers to return documents for a given query. Craswell, Bailey, and Hawking [8] compare three popular approaches to resource selection. However, these retrieval techniques execute probe queries to estimate the resource's data coverage and its effectiveness at returning relevant documents. Then, these coverage and effectiveness statistics are used to select and merge the appropriate resources for a query. This overhead is unnecessary for our task. Because we have full access to all of our reference sets in our repository, we already know the full data coverage, and do not need to estimate our repository's effectiveness, because it always returns all of our sets.

Information extraction has previously incorporated outside information to aid extraction. For example, the CRAM system [1] is unsupervised, and uses reference sets. However, unlike this paper, CRAM is given the reference set and requires that all tokens receive a label, not allowing for 'junk' in the text. Other work in information extraction incorporates reference sets with machine learning, for example the work of Cohen and Sarawagi [7]. However, this work requires human selected reference sets and this technique relies on supervised machine learning.

## 5 Conclusion

We introduce a technique for unsupervised information extraction from unstructured, ungrammatical text. Previously, unsupervised extraction used extraction patterns that make assumptions about the regularity of the structure in the data. We relax this assumption by exploiting reference sets to aid the extraction. These reference sets are chosen by the algorithm, removing the need for any human intervention.

Furthermore, we investigate different methods for choosing the reference sets and finding the matches between the posts and the reference sets. Experimentally we find that the Jensen-Shannon distance is the superior metric for choosing the correct reference sets. We also discover that when matching the posts to the reference set, modifying the similarity metric to use Jaro-Winkler edit-distance works well. However, this is the case only when the similarity metric does not use weighting schemes such as TF/IDF.

This approach describes a full architecture for collecting and exploiting reference sets for semantic annotation and extraction, allowing Local-as-View information integration to automatically incorporate unstructured, ungrammatical data sources which would be inaccessible for structural queries otherwise.

In future work, we plan to investigate methods to improve the accuracy of our extraction and matching. For instance, using domain-specific, text transformations such as acronyms could greatly aid the accuracy of both. For example, in the BFT domain, it is useful for both matching and extraction to know that "DT" is the same as "Downtown." In some of our other research, we have investigated the problem of automatically discovering such textual transformations, but in that research the datasets are already structured, relational data [19]. Studying how to apply such a method to the case where one set of data is unstructured and not delimited, while the other set of data is structured and relational is a challenge for future investigation.

We also intend to analyze how to apply our method to larger pieces of text, such as the paragraphs associated with classified listings. This will likely involve running an entity-extractor over the text first, to pull out the meaningful tokens, and then concatenating these tokens together to form a new post. In fact, we could use the unsupervised extraction methods mentioned previously in this paper to perform this entity extraction [3,10,21].

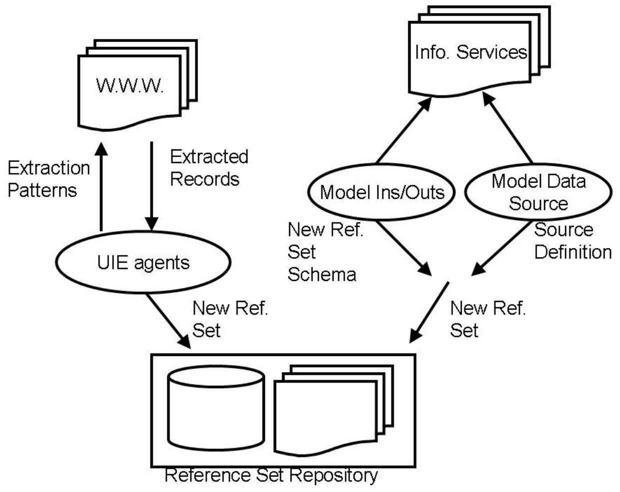

**Fig. 3** Automatically discovering reference sets

Lastly, one aspect to investigate is adding reference sets to our repository automatically, thereby increasing the coverage of sources for which we can perform extraction. The automatic collection of reference sets could be accomplished in two ways. First, as stated previously, we can use previous unsupervised extraction systems to build reference sets. For instance, KnowItNow [3] can extract all cars it can find from the Web, creating a reference set in this manner. Another approach to creating reference sets exploits research on information source discovery and modeling. By combining previous research on labeling the outputs of Web Services [12] with research on modeling the information service provided by Web Services [4] an agent can model the information provided by a source. This combination would allow an agent to define a schema using the Service's inputs and its outputs. Then, modeling the source can classify the source's provided data, therefore creating a new reference set. For example, consider a Web Service takes as input a year and provides all car makes, models, trim and colors for that year. First, the schema is defined based on classifying the inputs and outputs. Then a reference set of records is created by supplying all years and retrieving all car records. Then, a source modeler defines this set of records as cars, and it is included as a new reference set in the repository. The architecture shown in Figure 3 shows how agents can automatically create reference sets by combining the approaches outlined above, using both the unsupervised extraction methods and source modeling methods.

## References

1. Agichtein, E., Ganti, V.: Mining reference tables for automatic text segmentation. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 20-29. ACM Press (2004)
2. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 39-48. ACM Press (2003)
3. Cafarella, M.J., Downey, D., Soderland, S., Etzioni, O.: KnowItNow: Fast, scalable information extraction from the web. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 563-570. Association for Computational Linguistics (2005)
4. Carman, M.J., Knoblock, C.A.: Learning semantic descriptions of web information sources. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2695-2700. (2007)
5. Ciravegna, F.: Adaptive information extraction from text by rule induction and generalisation. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1251-1256 (2001)
6. Cohen, W., Ravikumar, P., Feinberg, S.: A comparison of string metrics for matching names and records. In: Proceedings of the ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation, pp. 13-18. (2003)
7. Cohen, W., Sarawagi, S.: Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 89-98. ACM Press (2004)
8. Craswell, N., Bailey, P., Hawking, D.: Server selection on the world wide web. In: Proceedings of the Conference on Digital Libraries, pp. 37-46. ACM Press (2000)
9. Dill, S., Gibson, N., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In: Proceedings of the International World Wide Web Conference, pp. 178-186. ACM Press (2003)
10. Hassan, H., Hassan, A., Emam, O.: Unsupervised information extraction approach using graph mutual reinforcement. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 501-508. Association for Computational Linguistics (2006)
11. Kushmerick, N., Weld, D.S., Doorenbos, R.: Wrapper induction for information extraction. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 729-737. (1997)
12. Lerman, K., Plangrasopchok, A., Knoblock, C. A.: Automatically labeling the inputs and outputs of web services. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1363-1368. AAAI Press (2006)
13. Levy, A.: Logic-based techniques in data integration. In: J. Minker (ed.) Logic Based Artificial Intelligence, pp. 575-595. Kluwer Academic Publishers (2000)
14. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: Proceedings of the International Conference on Very Large Data Bases, pp. 251-262. Morgan Kaufmann Publishers Inc. (1996)
15. Lin, J.: Divergence measures based on the shannon entropy. IEEE Transactions on Information Theory **37**(1), pp. 145-151. (1991)
16. McCallum, A.: Mallet: A machine learning for language toolkit (2002). http://mallet.cs.umass.edu
17. Michelson, M., Knoblock, C.A.: Semantic annotation of unstructured and ungrammatical text. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1091-1098. (2005)
18. Michelson, M., Knoblock, C.A.: An automatic approach to semantic annotation of unstructured, ungrammatical sources: A first look. In: Proceedings of the IJCAI Workshop on Analytics for Noisy Unstructured Text Data, pp. 123-130. (2007)
19. Michelson, M., Knoblock, C.A.: Mining heterogeneous transformations for record linkage. In: Proceedings of the International Workshop on Information Integration on the Web, pp. 68-73. AAAI Press (2007)
20. Minton, S.N., Nanjo, C., Knoblock, C.A., Michalowski, M., Michelson, M.: A heterogeneous field matching method for record linkage. In: Proceedings of the IEEE International Conference on Data Mining, pp. 314-321. IEEE Computer Society (2005)
21. Paşca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1400-1405. AAAI Press (2006)
22. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: Proceedings of ACM Symposium on Applied Computing, pp. 1634-1638. ACM Press (2005)
23. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology **147**, 195-197 (1981)
24. Thakkar, S., Ambite, J.L., Knoblock, C.A.: Composing, optimizing, and executing plans for bioinformatics web services. The International Journal on Very Large Data Bases, Special Issue on Data Management, Analysis, and Mining for the Life Sciences **14**(3), 330-353 (2005)
25. Winkler, W.E.: The state of record linkage and current research problems. Tech. rep., U.S. Census Bureau (1999)