Learning Weighted Entity Lists from Web Click Logs for Spoken Language Understanding

Dustin Hillard[1], *Asli Celikyilmaz*[1], *Dilek Hakkani-Tür*[1,2], *Gokhan Tur*[1,2]

[1]Microsoft Speech Labs, [2]Microsoft Research
Mountain View, CA, 94041
{hillard,asli,dilek,gokhan.tur}@ieee.org

Abstract

Named entity lists provide important features for language understanding, but typical lists can contain many ambiguous or incorrect phrases. We present an approach for automatically learning weighted entity lists by mining user clicks from web search logs. The approach significantly outperforms multiple baseline approaches and the weighted lists improve spoken language understanding tasks such as domain detection and slot filling. Our methods are general and can be easily applied to large quantities of entities, across any number of lists.

Index Terms: spoken language understanding, domain detection, slot filling, named entity lists, click logs

1. Introduction

Identifying spoken entities is a critical component of Spoken Language Understanding (SLU) systems. Correctly extracted entities can provide valuable information to many of the statistical classifiers that enable understanding in automated conversational systems. In this paper we will focus on two components of SLU systems that enable basic machine understanding: domain detection and slot filling. Domain detection is often a top level component of SLU systems that predicts an utterance's domain and provides a guide for subsequent understanding components, such as by limiting the types of information the system can extract from that utterance. A dialog system can invoke different information extraction depending on an automated domain classification of the utterance. For instance, given that an utterance is classified into the restaurant domain, the slot filling task is then to extract important phrases and detect their type (such as the restaurant name or the style of food desired). These statistical classifiers for domain detection and slot extraction can further be significantly improved when provided with accurate dictionaries that identify terms of interest. For instance, a domain detection classifier that decides between the restaurant and movie domain could clearly benefit from knowing if a movie name is present in the current utterance. Similarly, identifying if a phrase is a known movie name would provide an important feature for a slot filling classifier that extracts movie names.

Given a clean and unambiguous list of names, both domain and slot detection performance can improve significantly. The primary difficulty is in obtaining a high quality list without investing significant human effort to collect, clean, and curate it. Even if significant effort is invested in creating a useful list of movie names, ongoing effort will be required to keep the list up to date with new movies. This process does not scale well when a large set of lists is needed; for instance it would also be useful to have high quality lists for actors, directors, restaurant names, city names, brand names, products, and more. In order to provide broad coverage for the many types of lists that could be useful, one approach is to leverage lists that have been crowd sourced and are available online at websites such as Freebase.com, or by mining the web for tables and lists within websites. The challenge with incorporating such lists into statistical classifiers is that they are often extremely noisy due to ambiguous entries, mistakes during human generation, or errors resulting from automatic extraction. In the case of movies, Freebase provides a list of about 150k movie names, but these include rare movie titles that would typically not be a strong indicator of a movie title (such as "chrome" or "holiday inn"). Similar issues exist across most crowd sourced or automatically generated lists, which motivates an automated approach to predict the ambiguity of a phrase or entity within a list.

In this paper we present a general approach to automatically refine entity lists based on information available from web search click logs. Given a noisy seed list, we score each member in the list based on the click behavior of web search users and then show that statistical models for SLU improve when using the scored lists as features. In the remaining sections, we first describe a general approach for learning weighted lists from user click logs in Section 2. Section 3 then describes our baseline domain and intent detection models. We describe experiments and results (focusing on restaurant and movie tasks) in Section 4. We review previous related work in Section 5 and then summarize our conclusions in Section 6.

2. Refining noisy entities with search logs

There are plentiful resources on the web that provide typed entity lists, such as for movies or restaurants. These lists can be powerful features for many natural language processing tasks, but only when the content is sufficiently free of ambiguity and noise. A typical approach is to spend significant human effort in cleaning entity lists before including them in statistical systems. Using just the raw list can even hurt system performance. While expert human cleaning is possible when creating a small number of entity dictionaries, the approach does not scale well for potentially hundreds of dictionaries with thousands of entities. Maintenance is also a significant issue, given the constantly changing nature of many lists, such as with new movie releases or emerging celebrities.

This paper presents an automated process for refining entity lists for use in statistical systems. Click logs from large web search engines implicitly encode information that can be automatically extracted and processed to refine entity lists from noisy sources. Users of web search engines provide information about entities in the course of typical search sessions by click-

movies	restaurants	random
wikipedia.org (.120)	yelp.com (.027)	wikipedia.org (.027)
imdb.com (.093)	wikipedia.org (.017)	yahoo.com (.011)
amazon.com (.020)	citysearch.com (.012)	youtube.com (.009)
netflix (.004)	urbanspoon.com (.005)	facebook.com (.005)

Table 1: Frequently clicked websites in the context of various lists (with probability of click).

ing on relevant websites, and this is recorded in search engine logs. We will present results for improving lists of movie and restaurant names as examples, but the approach is general and could apply to a large set of potential typed lists. Our approach mines the click logs of a large commercial search engine, but similar approaches could be applied by instead mining the sites returned by search engines in response to relevant queries.

2.1. Scoring entity lists with click logs

Given a noisy typed list, our goal is to generate a score for each entity in the list that reflects the likelihood that the entity is a good member of the list. The resulting score should be high for entities that have unambiguous membership in the list, and low for entities that have ambiguous or incorrect membership. In the case of the movie title list from Freebase, "The Dark Knight" is a phrase that uniquely references a movie, but the list also contains the title "Hotel" (a small movie from 2003) that has meaning in many other contexts. The difference between these two phrases can be easily identified by examining the search click logs. Three quarters of user clicks for the query "The Dark Knight" belong to just three sites: Wikipedia, IMDB, and Warner Brothers. For the query "Hotel" user clicks are more evenly distributed: one quarter of the clicks go to hotels.com, followed by a broad distribution of clicks on sites such as travelocity.com, orbitz.com, etc. (with negligible clicks on any movie related sites).

Whether or not a phrase belongs in an entity list can be evaluated by considering the websites users click when searching for that phrase. In order to have a scalable process for any type of list, a method for automatically determining the relationship between a website and a list is required. Equation 1 proposes such a measure, which computes the probability of a click on a particular website (url_i) given a proposed list of queries. This measure aggregates the clicks received by a particular website for queries on the candidate list ($querySeedSet$), divided by clicks received for all websites in the context of the candidate list:

$$p(url_i|querySeedSet)=\frac{clicks(url_i|querySeedSet)}{\sum_j clicks(url_j|querySeedSet)} \quad (1)$$

where $clicks$ is the sum of all clicks that a particular website received over all queries in the seed query list:

$$clicks(url_i|querySeedSet)= \sum_{query_m \in querySeedSet} clickCount(url_i|query_m) \quad (2)$$

Given a candidate list, the most related sites can be automatically inferred from the user click logs in this way. In this paper we simplify websites to just the basic domain name from the url, although future work could explore ways to determine the optimal granularity for url simplification. Simplification is an important aspect of the approach, because it provides a method to generalize specific web pages to broader websites that can be associated with entity lists. In the case of movies, imdb.com and netflix.com are reasonable generalizations, but for larger sites such as google.com it would be preferable to have a more granular form (such as google.com/movies). Table 1 presents the top sites associated with three lists: movie titles from Freebase, restaurant names generated from automatic web mining, and a set of 100k random web queries. We collect over 1 billion clicks for these lists from about one year of click logs and keep the top 10k domains (the remaining domains are treated as unknown).

movies/random	restaurants/random
netflix.com (93|4.5)	yelp.com (14.7|2.7)
imdb.com (32|3.5)	opentable.com (12.8|2.6)
wikipedia.com (4.3|1.5)	wikipedia.com (.6|-.4)
yellowpages.com (.02|-4)	answers.com (.2|-1.7)

Table 2: Example sites in the context of seed lists (likelihood ratio | log likelihood ratio).

These click distributions can then be compared to the click distribution for a particular phrase to evaluate if the phrase is a good match to an overall list. A phrase that is a high quality match would be expected to have a click distribution similar to the list as a whole, while a low quality match would be more similar to a background model of random queries. This behavior can be represented with likelihood ratios comparing the probability of a click on a site in the context of the seed entity list versus the probability of a click on a site in the context of any random web search. For example, the likelihood ratio for a site (url_i) in the context of a seed set of movie names, against a background model of random queries, would be as in Equation 3:

$$movieRatio(url_i)=\frac{p(url_i|movieSeedSet)}{p(url_i|randomSeedSet)} \quad (3)$$

The log likelihood ratio is an alternative form, where $movieLogRatio(url_i) = log(movieRatio(url_i))$. Example likelihood ratios are given in Table 2. Comparing clicks on a site in the context of a list versus clicks on the site for a set of random queries emphasizes the relationship of a particular site to a list of entities more so than just the raw click probabilities given in Table 1. For example, while netflix.com receives a smaller percentage of the overall clicks for the movie domain than wikipedia.com, it is clear that a click on netflix.com is a stronger indicator that a search is related to movies when compared with a background distribution of random web queries.

Finally, for each individual candidate phrase in a list, a score can be generated based on the sites clicked on for that phrase. One potential score can be viewed as a weighted click vote over the sites clicked for that phrase, where $p(url_i|phrase)$ is the portion of clicks on a particular site in the context of a single phrase and $movieRatio(url_i)$ weights the site with the likelihood ratio that the site is a movies site (from Equation 3):

$$diffRatio(phrase)=\sum_i p(url_i|phrase)*movieRatio(url_i) \quad (4)$$

When weighted by the $logMovieRatio$, the score can be interpreted as the difference of cross entropies, comparing the cross entropy of the phrase click distribution and the movie click distribution with the cross entropy of the phrase click distribution and the random query click distribution:

$$logDiffRatio(phrase)= \sum_i p(url_i|phrase)*log(p(url_i|movieSeedSet))- p(url_i|phrase)*log(p(url_i|randSeedSet)) \quad (5)$$

This factors to Equation 6 below, which is equivalent to the (log) likelihood ratio weighted voting proposed in Equation 4.

$$logDiffRatio(phrase)= \sum_i p(url_i|phrase)*log(\frac{p(url_i|movieSeedSet)}{p(url_i|randSeedSet)}) \quad (6)$$

Typical $diffRatios$ are: 20 for "grumpier old men", 9.3 for "the dark knight", while "hotel" is 0.25 and "chrome" is 0.11.

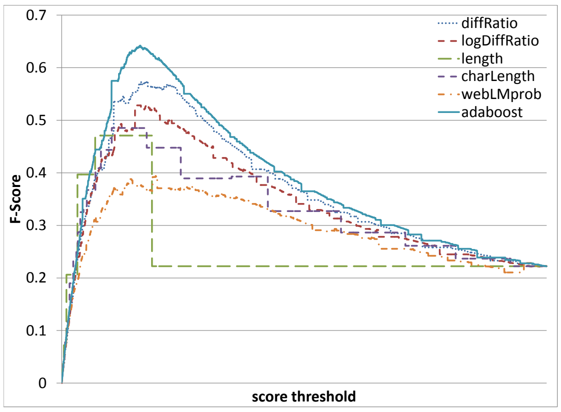

Figure 1: Movie list weights comparison

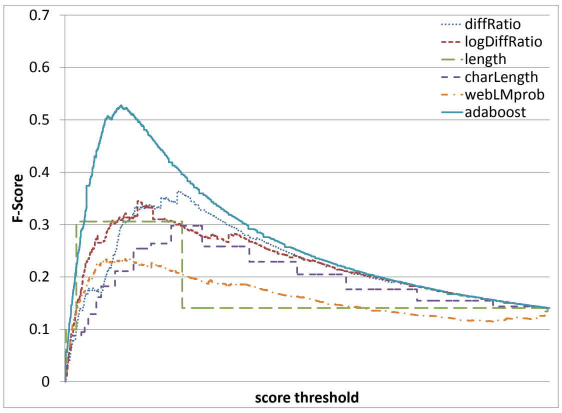

Figure 2: Restaurant list weights comparison

2.2. Evaluating weighted entity lists

The above scoring alternatives can be directly evaluated by comparing the scored lists on a simple entity extraction task (Section 4 evaluates the approach in the context of downstream SLU components). After computing the $diffRatio$ and $logDiffRatio$ scores for the lists, we normalize the phrases with standard text to speech processing and then evaluate using labeled speech data on correct transcripts from a conversational dialog system. To evaluate each scoring function, potential entities are labeled in the training data if a phrase's score is above some threshold. F-Score can then be computed while sweeping across all possible score thresholds to generate a learning curve. In addition to the two scores derived from the click logs, we also compare to three additional baselines: word length, character length, and the phrase's probability from a language model trained on web search queries. Word and character length provide reasonable baselines because longer entities tend to be less ambiguous. The probability from the web query language model provides a score related to the likelihood of the phrase, where rarer phrases can be considered less ambiguous (and therefore more likely to be a good member for the entity list).

Finally, we also train a classifier with ten fold cross-validation on the data set, which combines the five different scoring functions as features in a model that predicts list membership. The classifier training set is the set of candidate phrases extracted from our dialog training set with the complete list of entities, where the target is true or false (whether that instance of the phrase was labeled as an entity by a human annotator). The data for both restaurants and movies contains about 6k instances that occur in the candidate dictionaries, with about 1k instances labeled as true entity occurrences (recall is limited because an additional few hundred true entities do not occur in our dictionaries). The classifier used in these experiments learns boosted decision tree stumps (adaboost [1]), as implemented by the $icsiboost$ tool. The plots present F-score across varying score thresholds for the five individual scores, as well as the combination adaboost classifier. We preprocess the raw lists to include only those phrases that received ten or more clicks in our logs, which significantly cleans irrelevant noise from the lists with little loss in recall. Figure 1 compares results for movies, while restaurant results are in Figure 2.

Both figures show that using just word or character thresholds provides reasonable performance with a large improvement over using the original entire list (this corresponds to no threshold, at the far right of each graph). The blocky steps in performance for the length scores are a result of the relatively limited granularity available in the simple scores. The probability generated by the web language model performs worse than any other score because it only indicates something about the general probability of the phrase occurring; however, tests on a separate held-out set showed that it proved to be a useful feature in the overall combination classifier. The $diffRatio$ and $logDiffRatio$ achieve the best F-score among the individual score types across most operating points. The $diffRatio$ score slightly outperforms, perhaps as a result of the sharper scores used in weighting that were otherwise dampened with a log function in the $logDiffRaio$ score. In both cases the combination classifier significantly outperforms any individual score, making it an appealing approach if some small amount of labeled entity data is available. The combination can be trained on a relatively small amount of data because only the scores are used as input features, but the resulting classifier can be used for cleaning an entire entity list because the component scores can be automatically generated for any unseen candidate entity phrase. The learning curve plots show that performance on name extraction with a simple dictionary look up can be significantly improved by making use of the proposed scoring approaches. In the next sections we describe our statistical spoken language understanding models and then describe how the scored entity lists can be incorporated to improve the models.

3. Domain and Slot Filling Models

A comprehensive survey of previous approaches for domain and slot detection can be found in [2]. We follow these state-of-the-art approaches for our baseline domain and slot detection.

The domain detection classifier uses boosted decision tree stumps (adaboost [1]). Baseline features incorporate all unigrams, bigrams, and trigrams from each utterance and the classifier learns among five potential domains (although we focus on two: movies and restaurants). Including basic entity lists can improve performance, so we add binary features from already available human curated dictionaries that indicate presence for: cities, hotels, actors, directors, movie genre, movie awards, and movie characters. We also include unweighted versions of our restaurant and movie name lists, as an additional baseline. Finally, we incorporate the weights for restaurant and movie entities by providing a real valued feature that is populated with the max score for each entity type (movie and restaurant) that is found in the corresponding scored entity list. Domain models take scores from all of the proposed weighting types and the

Model	overall F-Score	movie F-Score	restaurant F-Score
baseline	.926	.907	.891
+ unweighted lists	.928	.919	.889
+ weighted lists	.932	.933	.900

Table 3: Domain detection error and F-score results

classifier learns how the scores should be combined.

We compare two modeling approaches for the sequence tagging task of slot extraction. A standard linear chain CRF model [3] is trained using the CRF++ tool to learn a tag for each word in an utterance: in name, outside name, or begin name (IOB representation). Features are generated from the current word, previous word, previous previous word, next word, and next next word, as well as the previous predicted tag. Simple entity list features are incorporated by adding the same contextual features as in the case of words, but based on IOB tags that encode whether the word sequence is present in an entity list. Weighted entity lists are then incorporated in much the same way, but first the scored list is clustered into ten clusters with K-means clustering and then ten lists of increasing precision are produced where the first includes the entire list and the last includes only those entities that occur in the highest scoring cluster. Then ten features are added to the CRF, based on the tags produced by the dictionaries of increasing precision. We compare with a semi-CRF model [4] that incorporates the same word based features (the implementation from [5]). Unweighted entity lists provide binary features for phrases that occur in the list, while weighted lists provide a real valued feature equal to the score of an occurring phrase. The weighted lists use only the score produced from the combined adaboost classifier, which performed slightly better than each individual score (and better than providing all scores as features to the CRFs).

4. SLU Experiments

We evaluate the impact of weighted entity lists in the context of domain and slot models for a spoken dialog system. Our domain training data consists of 16k utterances, each annotated with one of five domains. We report on a test set of 2k utterances, but we will primarily focus on F-score for the movie and restaurant domains (about 300 utterances each). Table 3 presents results for our baseline model, adding unweighted lists, and then with all unweighted and weighted lists. Unweighted lists improve performance for movie utterances, although not for restaurant utterances. Improvements in movie F-score are nearly double the gain achieved by adding a simple unweighted list. Overall five class domain F-Score is also improved by incorporating the weighted lists.

Slot filling results are presented in Table 4. Results are derived from five fold cross-validation on two separate sets: 2,400 movie utterances which contain 2,000 movie names and 3,000 restaurant utterances which contain 1,350 restaurant names. The unweighted movie list provides a large performance improvement for both CRF and semi-CRF models; then, incorporating the weighted list provides an additional 1.5% absolute improvement for the CRF model and 2.5% improvement for the semi-CRF model. The unweighted restaurant list provides a large performance improvement for both CRF and semi-CRF models, and incorporating the weights provides further large gains for the semi-CRF model (although not the CRF).

	Model	CRF F-Score	semi-CRF F-Score
movies	baseline	.658	.667
	+unweighted list	.736	.717
	+weighted list	.751	.742
rests.	baseline	.715	.737
	+unweighted list	.745	.752
	+weighted list	.745	.776

Table 4: Name slot tagging

5. Related Work

Much previous work has shown that dictionaries and lists can improve language segmentation tasks. Lists extracted from structured databases provide gains in information extraction tasks [6, 7] and named entities help in dialog systems [8]. More recent work has shown that unsupervised extraction from websites and query logs provides valuable features for information extraction [9] and weighting these features is better than simple lists [10]. Recent work uses search click logs [11, 12], although their approach focuses on $(query, document)$ pairs, rather than our $(list, siteDomain)$ pairs that better generalize for scoring lists. The most similar approach to ours is [13], which associates user browsing history with product attributes (although they constrain their approach to very small lists). Other previous approaches have shown the benefit of processing the click graph for extracting related queries [14, 15], but to our knowledge none have enriched the approach for scoring lists.

6. Conclusions

We present a general approach to score entity lists by learning from user clicks in web search logs. The method can be applied to any typed list that garners a reasonable amount of web search activity. Our contribution builds on previous click mining approaches and proposes a novel algorithm that compares click distributions on a phrase to aggregate click distributions of typed lists. We then show that weights improve example spoken language understanding tasks of domain detection and slot filling for movies and restaurants.

7. References

[1] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[2] G. Tur and R. De Mori, Eds., *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech.* Wiley and Sons, 2011.

[3] J. Lafferty, A. Mccallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

[4] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction," in *NIPS*, 2004.

[5] X. Li, "Understanding the semantic structure of noun phrase queries," in *ACL*, 2010.

[6] E. Agichtein and V. Ganti, "Mining reference tables for automatic text segmentation," in *KDD*, 2004, pp. 20–29.

[7] S. Canisius and C. Sporleder, "Bootstrapping information extraction from field books," in *EMNLP*, 2007, pp. 827–836.

[8] F. Bechet *et al.*, "Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken dialogue context: How May I Help You?" *Speech Communication*, vol. 42, no. 2, 2003.

[9] M. Pasca and B. V. Durme, "Weakly-supervised acquisition of open-domain classes and attributes from web documents and query logs," in *ACL*, 2008.

[10] Y. Wang, R. Hoffmann, X. Li, and J. Szymanski, "Semi-supervised learning of semantic classes for query understanding," in *CIKM*, 2009.

[11] X. Li, Y. Wang, and A. Acero, "Extracting structured information from user queries with semi-supervised conditional random fields," in *SIGIR*, 2009.

[12] J. Liu, X. Li, A. Acero, and Y. Wang, "Lexicon modeling for query understanding," in *ICASSP*, 2011.

[13] D. Panigrahi and S. Gollapudi, "Result enrichment in commerce search using browse trails," in *WSDM*, 2011.

[14] N. Craswell and M. Szummer, "Random walks on the click graph," in *SIGIR*, 2007.

[15] T. Anastasakos *et al.*, "A collaborative filtering approach to ad recommendation using the query ad click graph," in *CIKM*, 2009.