A Language Modeling Approach to Entity Recognition and Disambiguation for Search Queries

Bhavana Dalvi
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bbd@cs.cmu.edu

Chenyan Xiong
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
cx@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
callan@cs.cmu.edu

ABSTRACT

The Entity Recognition and Disambiguation (ERD) problem refers to the task of recognizing mentions of entities in a given query string, disambiguating them, and mapping them to entities in a given Knowledge Base(KB). If there are multiple ways to interpret the query, then an ERD system is supposed to group candidate entity annotations into consistent interpretations.

In this paper, we propose a four step solution to this problem. First, we generate candidate entity strings by segmenting queries in different ways. Second, we retrieve candidate entities by searching for these candidate entity strings in Freebase. Third, we rank the candidate entities using language model based query likelihood scores. Finally, we group the entity annotations into interpretations. We also present both quantitative and qualitative evaluation of our methods based on 91 training, 500 validation and 1000 test queries. Our system achieved an F1 score of 0.42 on the set of validation queries, whereas the *NULL* baseline which returns no annotations for any query achieved an F1 score of 0.3. Similarly, on the test queries, our method achieved an F1 score of 0.36 and outperformed the *NULL* baseline which achieved an F1 score of 0.2.

1. INTRODUCTION

The recent advancements in search engine technology have led to the use of knowledge bases (KB) for adding semantic aspect to the query processing. To achieve this, an important first step is to understand the user queries and indexed documents semantically by annotating entity mentions with entities in the KB. According to the analysis done by Guo et al. [5] about 71% of search queries contain named entities. Semantic understanding of user needs (i.e., queries) can help the search engines to narrow down the entities or the type of entities the user is looking for. Further, these annotations are used by the commercial engines to augment their search results with this semantic information.

The Short Text track of "Entity Recognition and Disambiguation Challenge 2014" [4] focuses on recognizing mentions of entities in a given query string, disambiguating them and mapping them to entities in a given KB (Freebase in this case). E.g., for a query "total recall arnold schwarzenegger," an ideal ERD system would recognize two entities "total recall," and "arnold schwarzenegger" and disambiguate "total recall" to be a 1990 movie with the actor "Arnold Schwarzenegger" in its lead role.

This task is challenging due to multiple reasons:

1. The query strings are short and lack proper punctuation and capitalization.
2. Entities might appear in different surface forms, and ambiguous surface forms may match to multiple entity interpretations.
3. There could be multiple entities in a short query.

Problem Definition

Given a short fragment of text, the task is to produce a set of valid entity linking interpretations. For each query, the ERD system should use all available context to produce a set of valid entity linking interpretations.

For a query "total recall arnold schwarzenegger," an ideal ERD system would output a single interpretation consisting of two entities "total recall (1990)" and "arnold schwarzenegger". However, if the query is "total recall movie", then there are two interpretations containing one entity each from "total recall (1990)" and "total recall (2012)".

Example query	Interpre -tation	Entity Annotation	
		FB ID	Mention Text
total recall arnold schwarzenegger	0	/m/0fd4x	total recall
	0	/m/0tc7	arnold schwarzenegger
total recall movie	0	/m/0fd4x	total recall
	1	/m/0gvrws1	total recall

Table 1: Example query annotations. Here, 'Interpretation' means one of the senses of the query if there are multiple possible senses. 'FB ID' is the Freebase ID for the entity. 'Mention text' is the sub-string of the query that corresponds to the annotated entity.

Table 2 lists few additional queries from TREC corpus along with their ideal entity annotations. This table contains an additional column "FB entity title". We can notice that FB entity title can be significantly different compared

ERD'14, July 06 - 11 2014, Gold Coast , QLD, Australia

http://dx.doi.org/10.1145/2633211.2634347.

to Mention Text. Hence it is required to also match the mention text with alias of the entity names, and considering possible abbreviations. E.g. entity "Barack Obama id:/m/02mjmr" has a name alias "Obama" that has an exact match with the mention text. Similarly, entity "Tennessee id:/m/07h34" can match the mention text "tn" if we have a list of standard abbreviations corresponding to USA states.

Outline:

Rest of the paper is organized as follows: We discuss the related work in Section 2. In Section 3, we present our ERD system and the topic modeling based entity ranking technique. Experimental results and error analysis are presented in Section 4, followed by conclusions in Section 5.

2. RELATED WORK

There has been a lot of research in the area of entity recognition. David and Satoshi [12] presented a survey of named entity recognition and classification techniques. They discussed word, dictionary and corpus level representations of words in a document; and presented various evaluation techniques ranging from intuitive exact match to adjustable cost of errors technique. Ratinov and Roth [13] studied the design challenges and misconceptions in named entity recognition task. They proposed a state of the art approach by using a simple NER model with excessive features. Further their method explored four design decisions: representation of text chunk, inference method, using non-local features and incorporating external knowledge base. Guo et al. [15] worked on the problem of named entity recognition from search queries. They proposed Weakly Supervised LDA method for detecting and classifying the named entities from search queries.

Another related area of research is Entity Linking that annotates entity mentions in the given text with their corresponding entities in a knowledge base. Kulkarni et al. [8] discussed the trade-off between local mention to entity score, and global consistency measures. Their approach was based on the hypothesis that a document usually contains entities from one or few related topics. Ratinov et al. [14] studied this problem further by using local and global disambiguation approaches together, and further showed that local approaches form a baseline that is hard to beat.

Later, Lin et al. [9] worked on web-scale entity linking task using local as well as global features. In terms local features they considered string match, prominence, and context match. They also used global information in the sense, linking entities using a set of sentences as opposed to individual sentences and using the in-link prominence to detect system errors. Use of graph based techniques for collective entity linking within a document was proposed by Han et al. [6]. Here, they hypothesized that entities within a document are semantically related to each other. They created referent graphs from all entity mentions within a document and use label propagation kind of techniques on these graphs to do collective entity linking. Most of the techniques described here are targeted towards entity linking in a long document, whereas we deal with a much more challenging problem. Entity disambiguation in search queries is harder due to the short length and hence unavailability of enough context to disambiguate entities.

There is a orthogonal set of techniques that uses Knowledge Bases for entity linking. Liu et al. [10] used WordNet for sense-tagging. They used the "hood" algorithm that is based on hypothesis that a set of words co-occurring in a document disambiguate each other, even if individually they are ambiguous. The hood in case of WordNet is the set of nearest ancestors of the senses of all words in a document. Krishnamurthy and Mitchell [7] proposed ConceptResolver, a technique that jointly does both word sense induction and and synonym resolution on extracted relations from a given text corpus. Agirre et al. [3] proposed random walk based algorithms like PageRank, personalized PageRank over large lexical knowledge bases. These techniques are interesting to us because our task involves using Freebase knowledge base for entity linking.

Our approach borrows some ideas from traditional information retrieval research. We used language model based query likelihood score computation for entity ranking. The basic idea behind such approaches is that they estimate a language model for each document and then compute a probability of generating the query given each document and use this score to generate a ranked list of documents. Zhai and Lafferty [16] focused on the problem of retrieval performance being sensitive to the smoothing technique used. They compared several smoothing techniques on a bunch of test collections. They found that retrieval performance is more sensitive to smoothing techniques for the verbose queries as compared to short keyword queries. Manning et al. [11] surveyed various query-likelihood language models. In this paper, we use the linear interpolation language model, also known as Jelinek-Mercer smoothing technique.

3. OUR APPROACH FOR ENTITY RECOGNITION AND DISAMBIGUATION

In this section we describe our four stage system to generate semantic annotations for search queries. Figure 1 shows the flow-chart of our system. Next, we will go through each stage in detail.

3.1 Query Segmentation

If we try to match entire query string to a Freebase entity title, then we will miss out annotating entities if the query has extra words. E.g. if the query is "The last lecture book download", then the entity "The Last Lecture" of type '/book' won't be outputted. Hence trying out different sub-strings of the input query referred to as sub-queries is important.

In the Query Segmentation stage, our system generates different possible sub-strings of the original query string as potential entity strings. If the original query string contains n words, then we generate sub-strings of length $n, n-1, \ldots 1$. Since the task is to find the longest sequence of words that matches an entity, if we find a sub-string s of length k that is an entity mention, then we don't consider any sub-strings of s as a candidate entity string.

Consider an example query "total recall arnold schwarzenegger," if the system discovers an entity corresponding to the mention text "total recall", then it won't consider the sub-queries "total" and "recall" as they are already covered by a longer entity mention "total recall".

3.2 Candidate Entity Retrieval

Here, we query Freebase with candidate entity strings using Google Freebase Search API. This API takes as input the

TREC query	Interpre -tation	Entity Annotation		
		FB ID	FB Entity Title	Mention Text
map of brazil	0	/m/015fr	Brazil	brazil
east ridge high school	0	/m/03ck4lv	East Ridge High School	east ridge high school
	1	/m/027311j	East Ridge High School	east ridge high school
	2	/m/0bs8gsb	East Ridge High School	east ridge high school
ritz carlton lake las vegas	0	/m/0288kpv	Ritz-Carlton Hotel Company	ritz carlton
	0	/m/06y9l6	Lake Las Vegas	lake las vegas
tn highway patrol	0	/m/07h34	Tennessee	tn
	0	/m/024ckj	Highway Patrol	highway partol
sonoma county medical services	0	/m/0l35f	Sonoma County	sonoma county
obama family tree	0	/m/02mjmr	Barack Obama	obama

Table 2: Sample TREC query annotations (used as training queries).


```
{"status":"200 OK",
"result":[
        {"mid":"/m/0cwx_",  "id":"/en/carnegie_mellon_university",
        "name":"Carnegie Mellon University",
        "notable":{  "name":"College/University",  "id":"/education/university"},
        "lang":"en", "score":26.211172,
        "output":{ "description":{"/common/topic/description":["Carnegie Mellon University is a private
research university in Pittsburgh, …."]}
}},
        {"mid":"/m/0467_t",  "id":"/en/cinder_block",  "name":"Concrete masonry unit",
        "notable":{ "name":"Lighthouse construction material",
                "id":"/architecture/lighthouse_construction_material"},
        "lang":"en",  "score":17.739067,
        "output":{ "description":{"/common/topic/description":["A concrete masonry unit \u2013 also
called concrete block, …. "]}
}},
 ….
}
```


Figure 2: Example Google Freebase Search API results for the query "CMU".

query and parameters like number of search results, and returns JSON results corresponding to Freebase entities that are relevant to the input query. An example output from Google Freebase search API for a query "CMU" is shown in Figure 2.

In our experiments, we used Google Freebase Search API and retrieved 50-200 search results per query. The number of search results is a parameter for our method, and we experimented with a range of values 50-200. Next we generate a pool of candidate entities based on the string match between candidate entity string and entity title or alias.

We also keep a list of standard abbreviations like state names and government organizations, which we use while matching entity names. This will allow us to annotate entity "Tennessee id:/m/07h34" for the mention text "tn" in a query "tn highway patrol" in Table 2. Further, if an exact match is not available, we also check whether unordered match is possible. E.g. sub-query "french lick casino and resort" can match with Freebase entity "French Lick Resort Casino" using unordered match and removing stop-words (e.g. 'and') from both the strings.

3.3 Entity Ranking

The Google Freebase Search API returns a score per result document. However, the score returned by the API for an entity is based on the sub-query given as input, and we would ideally like to score an entity based on its overall relevance to the entire query. Hence both entities "The Last Lecture" of type book and TV-episode might be assigned high score for the query "the last lecture book download", and sub-query "The Last Lecture". But we would like to give higher score to "The Last Lecture" of type book. For this purpose, we score the entity results with query-likelihood scores using language modeling approach.

For computing this score we make use of the Google API search result as a result document, and Freebase dump for the corpus statistics. We use a Freebase dump from October 2013 that has a size of 14GB compressed. From this corpus we build corpus statistics of the kind: vocabulary of words, their term frequency (tf) and document frequency (df). We have also built a corpus of DBPedia long abstracts for the target Freebase entities, using the Freebase ID to Wikipedia title mapping given as input. We use this DBPedia long abstract along with description of the Freebase entity returned by Google API, while computing the query-likelihood.

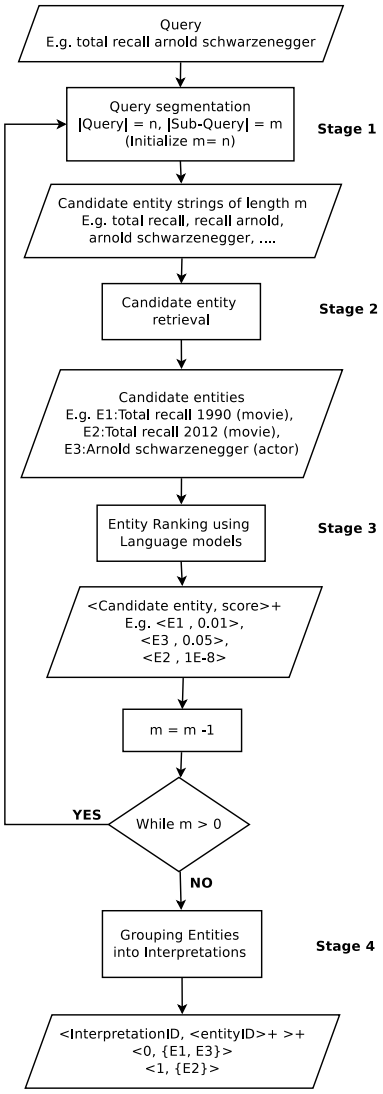

Figure 1: Flow chart of our four-stage ERD System.

Language Model with Jelinek-Mercer Smoothing

Say Q is the input query, d is the result document to be ranked and retrieved using sub-query $subQ$, C is the entire Freebase corpus, and t is a term in Q. Then the likelihood of Q given d can be written as follows:

$$P(Q|d) = \prod_{t \in Q} P(t|d) \tag{1}$$

We compute query-likelihood given a result document by applying linear interpolation or Jelinek-Mercer smoothing [16]:

$$P(t|d) = (1-\lambda)P(t|d) + \lambda * P(t|C)$$

$$P(Q|d) = \prod_{t \in Q} ((1-\lambda)P(t|d) + \lambda * P(t|C))$$

$$P(Q|d) = \prod_{t \in Q} \left((1-\lambda) * \frac{tf(t)}{length(d)} + \lambda * \frac{ctf(t)}{length(C)} \right)$$

$$logP(Q|d) = \sum_{t \in Q} log \left((1-\lambda) * \frac{tf(t)}{length(d)} + \lambda * \frac{ctf(t)}{length(C)} \right) \tag{2}$$

Here λ is a relative weight for corpus statistics vs. document statistics. Note that λ is a parameter for our method, and we experimented with various values in the range $[0, 1]$. We select the best value of λ based on the performance on training queries.

Ignoring the Sub-query during Score Computation

Further since document d is retrieved using sub-query $subQ \subseteq Q$, we rank it by computing likelihood of $Q \setminus subQ$ given d, to make sure that the document is relevant to query terms other than the sub-query.

$$P(Q, subQ|d) = \prod_{t \in Q \setminus subQ} P(t|d)$$

$$= \sum_{t \in Q \setminus subQ} log \left((1-\lambda) * \frac{tf(t)}{length(d)} + \lambda * \frac{ctf(t)}{length(C)} \right) \tag{3}$$

The ranking defined in Equation 3 is referred to as Language Model (LM) method in this paper.

E.g. consider a query "blue throated hummingbird". We retrieve multiple entities for the sub-query "hummingbird": "id:/m/0j91q (/biology/animal)", "id:/m/0f3xgb9 (/music/ album)", "id:/m/051vlvb (/music/music_group)". Since all these entities have the same title "Hummingbird", they all will result in high query-likelihood scores due to repeated use of this title throughout the result text (including description of an entity, alias etc.). Hence instead of computing likelihood of "blue throated hummingbird" given a result entity "id:/m/0f3xgb9 (/music/album)", we compute the likelihood of "blue throated" given "id:/m/0f3xgb9 (/music/album)". We expect that "blue throated" would match well with the "/biology/animal" type of entity as apposed to other "/music" entities. With this approach, we are able to rank the result entities in correct order and also able to generate.

Incorporating Target Freebase Types

We enhance our likelihood functions further by making use of the Freebase type associated with the result entities e.g. "/biology" and "/music" in the above example. To achieve this, we compute corpus statistics per Freebase type of interest. For each target Freebase type, we compute a separate vocabulary of words, and their topic specific term frequency (tf) and document frequency (df). Target Freebase types for this task are listed in Table 3. Also if for some entity, target type's corpus statistics are not available then the algorithm falls back to overall freebase based corpus statistics (Equation 3).

Target Freebase Types
/architecture, /automotive, /aviation, /award, /book, /broadcast, /business, /comic_books, /comic_strips, /commerce, /computer, /cvg, /education, /fictional_universe, /film, /internet, /location, /music, /organization, /people, /sports, /time, /tv

Table 3: List of target Freebase types.

$$P(Q, subQ|d, type(d)) = \prod_{t \in Q \setminus subQ} ((1-\lambda)P(t|d) + \lambda * P(t|C, type(d))) \quad (4)$$

The corpus based term $P(t|C, type(d))$ can be computed as follows:

$$P(t|C, type(d)) = \frac{ctf_{type(d)}(t)}{length(C_{type(d)})} \quad (5)$$

Here, $ctf_{type(d)}(t)$ is the total number of occurrences of term t in the documents of type $type(d)$ in the corpus C, and $length(C_{type(d)})$ is the total length of documents of type $type(d)$ in the corpus C. Hence, the query-likelihood formula becomes:

$$logP(Q, subQ|d) = \sum_{t \in Q \setminus subQ} log(\ (1-\lambda) * \frac{tf(t)}{length(d)} + \lambda * \frac{ctf_{type(d)}(t)}{length(C_{type(d)})}\ ) \quad (6)$$

The ranking defined by Equation 6 is referred to as Type-specific Language Model (TLM) method in this paper.

Score Thresholding

Along with ranking the result entities, we also do some post-processing to discard uninteresting entities. We discard the single word entities that are common words, and entities for which the query-likelihood without sub-query words is less than a pre-defined threshold. This allows us to discard entity annotations for common words like "is (IS id:/m/0cc6nby)", "i (I id:/m/03tjm)" and so on.

Further, we found that if a result for a sub-query is present in the search results for the entire query, then we can relax the score threshold for such results. E.g. for the query "the beatles rock band", in the search results for sub-query "the beatles" there is a result "the beatles id:/m/07c0j", which got a score of 1.37E-14, however this result was present in the search results for the entire query "the beatles rock band". Hence we define two distinct score thresholds: First threshold called $Thresh_1$ is a lower threshold for strict sub-strings of the given query and when the result occurs in the result set of the query. Otherwise, we apply a second threshold called $Thresh_2$ which is higher than $Thresh_1$ and is used to filter all other results.

3.4 Grouping Entities into Interpretations

If there are multiple entities corresponding to various sub-queries in the given query, then we need to group them into interpretations. Grouping entity annotations into interpretations is an important last step of the ERD process.

Example query	Ranked list of entity annotations		
	FB ID	Mention text	Score
total recall arnold schwarzenegger	/m/0fd4x	total recall	0.5
	/m/0tc7	arnold schwarzenegger	0.2
	/m/0gvrws1	total recall	1E-5

Table 4: Entity ranking output for the query "total recall arnold schwarzenegger".

Consider an example query "total recall arnold schwarzenegger". Table 4 shows the result of entity ranking stage for this query, with 3 possible entity annotations. Note that, we are ranking the candidate entities using the query likelihood score of entire query except the sub-query (mention text). Hence the entity "total recall (1990) id:/m/0fd4x" has higher query-likelihood score given the query except sub-query i.e. "arnold schwarzenegger".

To summarize, we rank the candidate entities using the query likelihood score of entire query except the sub-query used to retrieve the entity. We then group the entity annotations into interpretations by going through this ranked list once. We start with $Interpretation_0$, and add entities into this bucket till we reach an entity whose mention string is already covered by the current interpretation. In this case this entity starts a new bucket: $Interpretation_1$, and so on.

For our example query, we start with the first entity from the ranked list in Table 4, i.e. $Interpretation_0 = \{$/m/0fd4x$\}$ and covered words = {total, recall}. We then visit entity '/m/0tc7' and check that the words 'arnold' and 'schwarzenegger' are uncovered. Hence we add this entity to the same interpretation. So now, $Interpretation_0 = \{$/m/0fd4x, /m/0tc7$\}$ and covered words = {total, recall, arnold, schwarzenegger}. When we visit entity '/m/0gvrws1', the words 'total' and 'recall' are already covered, hence we create a new interpretation: $Interpretation_1 = \{$/m/0gvrws1$\}$ and initialize covered words = {total, recall}. The resultant interpretations (entity groups) are shown in Table 5

Interpretation ID	FB ID	Mention text	Score
0	/m/0fd4x	total recall	0.5
	/m/0tc7	arnold schwarzenegger	0.2
1	/m/0gvrws1	total recall	1E-5

Table 5: Interpretations(entity groupings) for the query "total recall arnold schwarzenegger".

In the end, if there are multiple entities in any interpretation, then we output a single multi-entity interpretation. However, if there are multiple possible single entity annotations then we output all of the interpretations that satisfy a predefined score threshold. The hypothesis here is that if there are multiple entities in the query then they will disambiguate each other and result in a single multi-entity interpretation. However, in the presence of a single entity we might not always be able to disambiguate it using the context text in the query, hence all interpretations that cross the score threshold will be outputted.

From Table 5 we can see that there are two possible interpretations, and one of them contains 2 entities. Hence according to our hypothesis, we will output a single multi-entity interpretation containing two entities "total recall (1990) id:/m/0fd4x" and "arnold schwarzenegger id:/m/0tc7".

3.5 Overall Flow of the System

To summarize, our approach consists of four stages: query segmentation, candidate entity retrieval, entity ranking, and entity grouping. Henceforth, we refer to this method as "Type-specific Language Model" (*TLM*) method. The flowchart of our approach is shown in Figure 1. Note that the stages 1,2, and 3 are iterated till the sub-query size goes to zeros or all the sub-query strings are covered with entity annotations, followed by stage 4 that constructs and outputs consistent interpretations.

Since the ERD challenge has a timeout of 12 seconds for each query, we keep track of the time-budget while executing this iterative procedure. We measure relative time spent since the query was issued, and if the time budget is about to expire before entering the stage 2 of any iteration, then our system quits the iteration and runs stage 4 on whatever entity annotations are available till that point in time.

4. EXPERIMENTAL RESULTS

In this section we will first look at all the data sources we used in our approach for entity recognition and disambiguation. We then go through the quantitative as well as qualitative evaluation of our method on the ERD task.

4.1 Data Sources

In our approach we make use of following data sources for the entity recognition and disambiguation task:

- *ERD'14 data:* The ERD'14 challenge [4] has released an entity snapshot of 2.3M target Freebase entities, and mapping to their Wikipedia pages. They have also given a set of Freebase target types e.g. "/architecture/building", "/book/magazine" etc. However, in our approach we consider coarse-grained types e.g. "/architecture", "/book".
- *Freebase corpus:* This is a dump of Freebase triples from October 2013 downloaded from [2] (size 14GB compressed).
- *Freebase corpus statistics:* We created a vocabulary, term frequency and document frequency from the entire Freebase corpus with vocabulary size of 989K. Vocabulary size refers to the number of distinct terms in the corpus.
- *Freebase type specific corpus statistics:* For each target Freebase type from Table 3, we created a type-specific vocabulary and corpus statistics. Table 6 shows the vocabulary size per type.
- *Freebase entity ID to type mapping:* We go through entire Freebase triples corpus and create a map of entity ID to its Freebase type.
- *DBPedia long abstracts:* For each target Freebase entity, we have been given the Wikipedia title mapping. The DBPedia abstracts are downloaded from [1]. We matched the DBPedia long abstracts to Freebase ids using this mapping. We were able to match DBPedia long abstracts for 1.95M out of 2.3M entities. Some example mappings are shown in Table 7.
- *Standard abbreviations for named entities:* We created a list of standard abbreviations available on the Web. Table 8 shows a sample of these abbreviations. While matching the entity titles to sub-queries, we check whether there is an exact match possible by replacing a named entity with its abbreviation or vice versa.

Freebase type	Vocabulary size
/architecture	227.3K
/automotive	33.7K
/aviation	26.1K
/award	25.7K
/book	257.8K
/broadcast	77.0K
/business	125.9K
/computer	40.9K
/cvg	61.8K
/education	69.7K
/fictional_universe	231.1K
/film	266.1K
/internet	62.9K
/location	1.19M
/music	160.4K
/organization	585.7K
/people	1.42M
/sports	469.1K
/time	6.1K
/tv	155.3K

Table 6: Type-specific corpus statistics from Freebase.

4.2 Method Parameters

Our proposed method referred to as Type-specific Language Model (*TLM* method) has following four parameters:

1. λ parameter in the language modeling formula (Equation 6),
2. Top-K parameter that defines #search results per Google Freebase Search API call,
3. Threshold on the score of a result given a sub-query
   - $Thresh_1$: a threshold given (sub-query != query) and the result occurred in the result set of the query
   - otherwise, $Thresh_2$: a threshold on any other result score.

We compare our method with a simple baseline called *NULL* method; the baseline does not return any annotations corresponding to any input query.

4.3 Evaluation Results

Table 9 describes the performance of the above mentioned methods on a set of 500 validation queries. Setting values of score thresholds $Thresh_1$ and $Thresh_2$ is the most important parameter tuning for our system. If these values are set too high (1E-12, 1E-8) then it results in very few annotations and hence the F1 score drops to 0.39. If these values are set too low (1E16, 1E-12) the it outputs too many noisy annotations and score again drops to 0.36. Thresholds set to (1E-14, 1E-9) results in the best performance on the validation query-set with F1 score of 0.42.

The next most important parameter is λ. We can see that if λ is set to either of the extreme values like 0 or 1, which

Freebase ID	Entity Title	DBPedia long abstract
/m/06tq74f	Autism	Autism is a disorder of neural development characterized by impaired social interaction and communication, and by restricted and repetitive ...
/m/0492krs	Achilles	In Greek mythology, Achilles was a Greek hero of the Trojan War and the central character and greatest warrior of Homer's Iliad. ...
/m/03cmrp6	Aristotle	Aristotle (384 BC - 322 BC) was a Greek philosopher and polymath, a student of Plato and teacher of Alexander the Great. His writings cover many subjects, including physics, ...

Table 7: Example mappings from Freebase ID to DBPedia long abstract.

Named Entity	Abbreviation
Colorado	CO
Connecticut	CT
Delaware	DE
Florida	FL
United States Air Force	USAF
United States Coast Guard	USCG
United States Marine Corps	USMC
Institute	Inst
Irish Republican Army	IRA
Internal Revenue Service	IRS

Table 8: Sample standard abbreviations.

selects only one of the result term or corpus term to rank the entity, it results in a poor performance. Whereas, setting $lambda = 0.1$, i.e. giving most of the weight(0.9) to the term frequency and a small weight(0.1) to the corpus frequency results in the best performance. The system is not very sensitive to the Top-K parameter. The more the value of this parameter, more the number of candidate entities considered by our entity ranking stage. We observed that Top-K set to 100 resulted in good performance for our training and validation set queries.

Based on the performance observed on the validation set, we submitted our *TLM* system with following parameter settings for the final test run: 1) $\lambda = 0.1$, 2) Top-K = 100, 3) $Thresh_1$ = 1E-14, and 4) $Thresh_2$ = 1E-9. Some sample queries from ERD'14 challenge's test run are shown in Table 10. The performance of our final submission on the set of 1000 test queries is shown in Table 11.

Further, we found that for 85 out of 1000 test queries, our system decided to preempt the algorithm, to return the available entity annotations and avoided query timeouts. Table 11 also shows the performance of the SMAPH Team which performed the best on this challenge with 0.68 F1 score. Carmel et al. [4] present a detailed overview of this challenge along with the description of methods used by other participating teams.

Test ERD query
where did the maple leaf flag idea originated
cosmos cell phone
2 chainz out of town
elmos world muooet wiki
you tube miss suriname in bikini
transit system in milwaukee wi
linda evans on dallas
shay mitchell parents
register for selective service
chocolate chip cookie recipe

Table 10: Sample ERD'14 test queries.

Method	F1 on test-queries
NULL	0.20
TLM (our approach)	0.36
SMAPH Team	0.68

Table 11: Performance of our final submission on ERD'14 test queries. SMAPH Team performed the best on the test-queries in this challenge.

4.4 Discussion

We did a detailed error analysis of the performance of our system on training TREC queries. This query-set consisted of 91 TREC queries with total 61 entity annotations. Example queries and annotations are shown in Table 2.

4.4.1 Manual Error Analysis of Training Queries

For manual evaluation, we used 20 randomly chosen training queries with at least one entity annotation, and 15 randomly chosen queries without any entity annotations. For 14 out of 15 queries without any gold-standard annotations our system correctly returned no annotations. For one query "adobe indian houses", we annotate entity "Adobe id:/m/0vlf". We could not filter it out due to its high query-likelihood score(1.83E-8).

We did a detailed analysis of how well each stage of our system performed for these 35 (20+15) TREC queries. In particular, we gave binary scores for each stage per query, indicating whether that stage generated a desired output. If applicable, we also wrote notes about what went wrong by analyzing the detailed log outputs. Many of our design decisions about what features and scoring model should be used are inspired by such analysis. As our system evolved, we did such analysis multiple times for the same set of queries.

An example of a decision inspired by our per stage error analysis is the use of standard abbreviations while matching the query to an entity title. Another example is keeping two thresholds ($Thresh_1$ ad $Thresh_2$) for filtering entity annotations: $Thresh_1$ is applied when a result for a strict sub-query appears in the result set for the entire query, and $Thresh_2$ is applied for all other cases.

4.4.2 Failure Cases and Future Directions

For a query "condos in florida", we could not detect and link the entity mention "condos" to Freebase entity "Condominium id:/m/020ys5" because it does not satisfy our criterion to string match with the entity title. For another

Method	Parameters				Performance		
	$Thresh_1$	$Thresh_2$	Top-K	λ	Expected F1	Latency (sec.)	Timeout count
NULL	-	-	-	-	0.30	0.12	0
TLM	**1E-14**	**1E-9**	**100**	0.0	0.36	4.17	0
				0.1	**0.42**	6.72	3
				0.2	0.41	4.09	0
				0.6	0.41	4.00	0
				1.0	0.38	4.17	0
			50	0.1	0.42	5.25	1
	1E-16	1E-12	100	0.1	0.36	4.05	0
			50	0.1	0.39	3.64	0
	1E-12	1E-8	100	0.1	0.39	6.19	3
			50	0.1	0.39	3.80	0

Table 9: Comparison of our method with different parameter settings in terms of expected F1 scores on a validation query set of size 500.

query "ritz carlton lake las vegas" we could not annotate "Ritz-Carlton Hotel Company id:/m/0288kpv" because of no string match with the sub-query "ritz carlton". We could not resolve this problem because enabling a partial string match results in many noisy annotations for other queries in the training set.

For a query "bowflex power pro", our system could not produce a score above threshold for entity "bowflex id:/m/04cnvy", because the words "power" and "pro" are mentioned way down in the wikipedia page; and neither the DBPedia long abstract nor the freebase result text mentions these context words. Our system might get better at the given task using additional sources of information like Wikipedia pages for entities and syntactic parses of queries. Access to the entire Wikipedia page for an entity can help generate better likelihood scores for query given an entity text.

Another possibility of improvement lies in making the technique more efficient. E.g. for a query "corruption and its consequences in a developing country like bangladesh", our system takes more than 12 seconds to respond due to large number of possible sub-queries that can be derived from this query. However, since we are looking for mentions of named entities, a sub-query like "consequences in a" does not make any sense. Syntactic parsing of the query (E.g. POS tagging) might help us discard syntactically incorrect candidate sub-query strings from the given query, and hence reduce the query-response time. Right now, we avoid the timeouts by keeping a time budget, and as soon as the time budget is about to expire, we output the annotations based on information processed till that point in time.

5. CONCLUSIONS

We described a four step algorithm called Topic-specific Language Model (TLM method) for doing Entity Recognition and Disambiguation from search queries. First stage generates candidate entity strings by segmenting the query in different ways. Second stage retrieves candidate entities by searching these candidate entity strings in Freebase. Third stage ranks the candidate entities using language model based query likelihood scores. Finally, the forth stage groups the entity annotations into interpretations.

In the manual evaluation and error analysis, we observed that each stage of this system contributes positively to the overall system performance. The design decision to do title based string match using abbreviations, alias and unordered match enabled us to increase the coverage of candidate entities. Further, using typed corpus statistics, DBPedia long abstracts, and carefully selecting the score thresholds helped us achieve the F1 score of 0.42 on a set of validation queries. Finally, in the final test evaluation, our proposed *TLM* method achieved an F1 score of 0.36, outperforming the *NULL* baseline with 0.2 F1 score.

6. REFERENCES

[1] DBPedia data dumps. 2013. `http://wiki.dbpedia.org/Downloads39`.

[2] FreeBase data dumps. 2013. `https://developers.google.com/freebase/data`.

[3] E. Agirre, O. L. de Lacalle, and A. Soroa. Random walks for knowledge-based word sense disambiguation. In *Computational Linguistics*, 2013.

[4] D. Carmel, M.-W. Chang, E. Gabrilovich, B.-J. P. Hsu, and K. Wang. ERD 2014: Entity recognition and disambiguation challenge. *SIGIR Forum*, 2014 (forthcoming).

[5] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, 2009.

[6] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011.

[7] J. Krishnamurthy and T. M. Mitchell. Which noun phrases denote which concepts? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011.

[8] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[9] T. Lin, O. Etzioni, et al. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, 2012.

[10] Y. Liu, P. Scheuermann, X. Li, and X. Zhu. Using wordnet to disambiguate word senses for text classification. In *Computational Science ICCS*. 2007.

[11] C. D. Manning, P. Raghavan, and H. Schtze. Introduction to information retrieval. In *Cambridge University Press*, 2008.

[12] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 2007.

[13] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, 2009.

[14] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011.

[15] S. Sekine and E. Ranchhod. *Named entities: recognition, classification and use*. John Benjamins Publishing, 2009.

[16] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 2004.