A Greedy Algorithm for Finding Sets of Entity Linking Interpretations in Queries

Faegheh Hasibi
Norwegian University of Science and Technology
faeghehh@idi.ntnu.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Svein Erik Bratsberg
Norwegian University of Science and Technology
sveinbra@idi.ntnu.no

ABSTRACT

We describe our participation in the short text track of the Entity Recognition and Disambiguation (ERD) challenge, where the task is to find all interpretations of entity-related queries and link them to entities in a knowledge base. We approached this task using a multi-stage framework. First, we recognize entity mentions based on known surface forms. Next, we score candidate entities using a learning-to-rank method. Finally, we use a greedy algorithm to find all valid interpretation sets for the query. We report on evaluation results using the official ERD challenge platform.

Categories and Subject Descriptors

H.4 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval.

Keywords

Entity linking; Query Interpretation; ERD 2014

1. INTRODUCTION

The aim of the Entity Recognition and Disambiguation (ERD) challenge is to recognize entity mentions in a given text, disambiguate them, and link them to entities in a knowledge base [3]. We focused on the short text track of ERD 2014, which targets web search queries. There, the text is typically short and without proper context and linguistic clues.

It is exactly the lack of context that induces that a given query might have multiple valid interpretations. For example, given a knowledge base with several entities named "total recall" and the query "total recall arnold schwarzenegger," the ERD system should return a single interpretation set, which consists of two entities: "Arnold Schwarzenegger," the actor, and "total recall," the movie featuring Arnold Schwarzenegger. On the other hand, the query "total recall movie" does not have a single interpenetration. It can be linked to at least two movies: one from 1990 and one from 2012. Furthermore, the ERD system should handle aliases as well. For example, the query "the governator" should be linked to two entities: the TV program and Arnold Schwarzenegger, who is also known as the "governator."

Linking entity mentions in text to the corresponding nodes in a knowledge base is a well-studied task [4, 5, 8]. However, most of this research is focused on long text, where context can be used to disambiguate entities. The techniques used for long documents may not work well for web queries, due to the short and ambiguous nature of such queries, and because of the lack of linguistic features and lexical context.

We address the problem of named entity recognition and disambiguation in a multi-stage framework. First, we detect candidate entities in the query, based on known surface forms from a knowledge base (DBpedia). Next, we score these candidate entities using a learning-to-rank approach. The features used in our learning algorithm revolve around entity mentions in the query and text-based similarities between the query and a given entity, calculated using language modeling techniques. In the last step of our framework, we use a greedy algorithm to find valid interpretations. The algorithm gets a ranked list of entities as input and extracts the interpretation sets among top ranked entities.

The remainder of the paper is organized as follows. In Section 2 we present a high-level overview of our approach. Next, in Sections 3–5, we describe the three stage of our framework. We present our experiments and results in Section 6, followed by the conclusion our work in Section 7.

2. APPROACH

The objective of our system is to recognize the entity mentions in a given query and provide a set of valid entity linking interpretations with respect to a knowledge base. We used DBpedia (specifically, version 3.9[1]) as the knowledge base and removed entities that are not present in Freebase (i.e., there is no "same-as" link for the given DBpedia entity to Freebase). We note that this is likely to result in lower coverage, but we hope that it is compensated by the higher quality data for the remaining entities (i.e., those in the intersection of DBpedia and Freebase).

In approaching the entity recognition and disambiguation task, our strategy is to employ a pipeline, shown in Figure 1, with the following components:

- *Candidate detection*: Given a query, it detects entity mentions in the query and returns all candidate entities; see Section 3.

[1] http://wiki.dbpedia.org/Downloads39

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ERD'14, July 11, 2014, Gold Coast, Queensland, Australia.
Copyright 2014 ACM 978-1-4503-3023-7/14/07 ...$15.00.
http://dx.doi.org/10.1145/2633211.2634356.

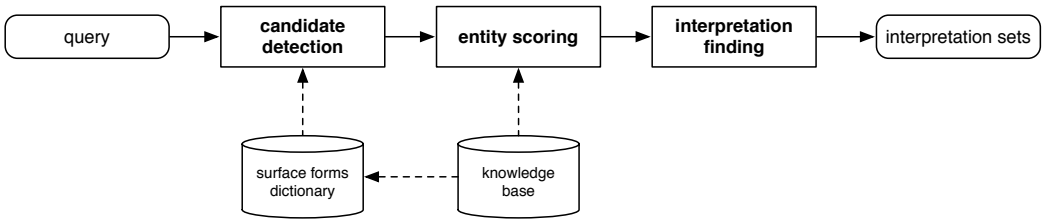

Figure 1: System overview.

Table 1: Lookup table for the surface form "new york."

<*rdfs:label*> <dbpedia:New_York>
<*foaf:name*> <dbpedia:New_York,_Tyne_and_Wear> <dbpedia:Global_Underground_007:_New_York> <dbpedia:New_York,_North_Yorkshire> <dbpedia:New_York_(film)__New_York__1> <dbpedia:New_York,_Lincolnshire> <dbpedia:New_York_(U2_song)> <dbpedia:New_York,_Texas> <dbpedia:New_York_(Ja_Rule_song)> <dbpedia:New_York,_Kentucky> <dbpedia:New_York_(Eskimo_Joe_song)> <dbpedia:Pennsylvania_Station_(New_York_City)> <dbpedia:USS_New_York_(ACR-2)> <dbpedia:Roman_Catholic_Archdiocese_of_New_York> <dbpedia:New_York_(album)> <dbpedia:Province_of_New_York> <dbpedia:New_York_(magazine)> <dbpedia:Episcopal_Diocese_of_New_York> <dbpedia:New_York_wine> <dbpedia:New_York_(film)> <dbpedia:USS_New_York_(1820)> <dbpedia:New_York_(Paloma_Faith_song)> <dbpedia:New_York_(Glee)> <dbpedia:New_York_(Snow_Patrol_song)>"
<*dbo:wikiPageRedirects*> <dbpedia:New_York>

- *Entity scoring*: Given a query and a set of candidate entities as input, it assigns a score to each entity; see Section 4.
- *Interpretation finding*: It finds interpretation sets for the query, given a ranked list of entities; see Section 5.

3. CANDIDATE DETECTION

The objective of this step is to detect all candidate entities that are relevant to the query. Our focus here is on obtaining high recall. To this end, we generate all n-grams of the query and perform lexical matching between n-grams and entity names (surface forms). We collected entity surface forms from DBpedia via three different predicates: titles (<*rdfs:label*>), names from mapping-based properties (<*foaf:name*>), and Wikipedia page redirects (<*dbo:wikiPageRedirects*>). Table 1 displays an example. Each of the entities matching the surface form "new york" are scored.

4. ENTITY SCORING

Once candidate entities are identified, we score them based on their relevance to the query, using a learning-to-rank approach. The input of this step is an input query, q_i, together with a set of candidate entities, $e_j \in E_i$. Our goal is to find a function $h(.)$ that generates a score for each query-entity pair (q_i, e_j); such pairs are considered as instances and are represented by a feature vector $w_{i,j}$.

To find the ranking function $h(.)$, we first perform learning. The input of learning process takes the form $D = \{w_{i,j}, l_{i,j}\}$, where $l_{i,j} \in \{0, 1\}$ is a manually assigned label from a given ground truth. The label $l_{i,j}$ indicates the binary relevance of entity e_j to the query q_i. We used two datasets as our ground truth for assigning labels: the (training) query annotations from the ERD challenge [3] and the Yahoo! webscope dataset [1]. In the remainder of this section we describe our entity representation, the feature vector, and the learning-to-rank method.

4.1 Entity Representation

We construct a fielded document-based representation for each entity in the knowledge base. The following fields are considered: Wikipedia *title* of entity, *short abstract*, *long abstract*, *links* to other Wikipedia entities, and Wikipedia *categories*.

4.2 Feature Set

Each instance in our ranking process is a combination of a query and an entity, represented by a set of features. We employ two groups of features, as shown in Table 2. The first group considers the entity mention, while the second one measures the similarity between the query and the entity.

Name-based features (the top block in Table 2) include two binary features, indicating whether the query equals (QET) or contains (QCT) the entity's name. QET and QCT have been used in [7] and were shown to be effective. In addition, *LenRatio* defines the query to n-gram length ratio, where the n-gram matches a surface form of the entity.

To calculate the text-based similarity between the entity and the query (the bottom block in Table 2), we use language modeling techniques [6]. In this approach, we estimate the probability of an entity being relevant to the query q, based on a textual representation of the entity, d, built from on one or more fields. After performing Bayes's rule and the usual algebraic steps, the entity's score is in proportion to $p(q|d)$. To calculate this probability, we use the Jelinek-Mercer smoothing method with λ set to 0.1:

$$p(q|\theta_d) = \prod_{t \in q} \big((1-\lambda) \cdot p(t|d) + \lambda \cdot p(t|C)\big)^{n(t,q)}.$$

4.3 Machine Learning Methods

We employed Gradient Boosted Regression Trees (GBRT) [2], the state-of-art in web search ranking, to rank candidate

Feature	Description
QET(q, e)	Does the query equal the name of the entity?
QCT(q, e)	Does the query contain the name of the entity?
$LenRatio(q,s) = \frac{\|q\|}{\|s\|}$	Query length to n-gram length ratio
T-Sim(q,e)	LM similarity between the title of the entity and the query
Abs-Sim(q,e)	LM similarity between the short abstract of the entity and the query
T-Abs-Sim(q,e)	LM similarity between the title & short abstract of the entity and the query
LAbs-Sim(q,e)	LM similarity between the long abstract of the entity and the query
WLinks-Sim(q,e)	LM similarity between Wikipedia links of the entity and the query
WP-Sim(q,e)	LM similarity between Wikipedia categories of the entity and the query
CatchAll-Sim(q,e)	LM similarity between all fields of the entity and the query

Table 2: Feature set used for ranking entities, given query q, entity e, and n-gram s.

entities. GBRT is a point-wise learning-to-rank algorithm that performs gradient descent to minimize a loss function. During each iteration, a new tree of pre-defined depth, so called weak learner, will be created. The final learning model is an ensemble of weak learners constructed using the boosting approach.

GBRT employs three parameters: number of trees (t), number of leaves for each tree (l), and learning rate α. In our experiments, we set $\alpha = 0.1$, and we tuned t and l for our dataset.

5. INTERPRETATION FINDING

After getting a score for each candidate entity, the last step is to find all valid interpretations of the query. There are zero or more interpretation sets associated with the query, where each set consists of one or more entities. Our approach to get interpretation sets is based on some simple but effective heuristics, controlled by two threshold parameters that are set manually: entity scores (T_s) and number of entities (T_n).

Our interpretation finding algorithm (shown in Algorithm 1) takes a list of ranked entities as input (the scoring is detailed in Section 4). We hypothesize that there are only a few entities relevant to the query; we first find these top entities based on the thresholds T_s and T_n. Function `FindTopRankedEntities` outputs up to T_n number of entities with scores higher than T_s.

In the next step, the algorithm identifies the interpretation(s) of the query. We make use of the query examples in Table 3 to describe this part. For *Q1*, two entities are identified as top ranked entities. The algorithm creates a set consisting of "Total Recall (1990 film)." Since the n-gram related to this entity ("total recall") does not overlap with "arnold schwarzenegger," the current *setID* is kept. As a result, the next entity is added to the same set. Hence, this query has a single interpretation. Whenever the algorithm finds an overlap between the current n-gram and the next one, a new set will be created. For example, both entities in query *Q2* have the same surface form "total recall." Therefore, two interpretations are assigned to this query.

6. EVALUATION AND RESULTS

The performance evaluation of our system is based on precision and recall, as defined at the ERD challenge [3]. Given a query, with labelled interpretations $\hat{A} = \{\hat{E}_1, ..., \hat{E}_n\}$, where each interpretation consists of a set of mentioned entities

Algorithm 1 Interpretation Finding

Require: Ranked list of entities, T_s, T_n
Ensure: Interpretation set $\hat{A} = \{\hat{E}_1, ..., \hat{E}_n\}$

begin
1: $topEns \leftarrow$ FindTopRankedEntities(T_s, T_n)
2: $setID \leftarrow 1$
3: **for** en In $topEns$ **do**
4: $\quad \hat{E}_{setID}.add(en)$
5: $\quad$ **if** $en.surface$ overlaps next entity surface **then**
6: $\quad\quad setID += 1$
7: $\quad\quad \hat{A}.add(\hat{E}_{sestID})$
8: $\quad$ **end if**
9: **end for**
end

1: **function** FindTopRankedEntities(T_s, T_n)
2: $\quad topEns \leftarrow \emptyset$
3: $\quad i \leftarrow 1$
4: $\quad$ **while** $i <= T_n$ **do**
5: $\quad\quad en \leftarrow rankedEntities[i]$
6: $\quad\quad$ **if** $en.score < T_s$ **then**
7: $\quad\quad\quad$ topEns.add(en)
8: $\quad\quad$ **end if**
9: $\quad\quad i += 1$
10: $\quad$ **end while**
11: **end function**

Entity	n-gram	SetID	Score
Q1: "total recall arnold schwarzenegger"			
Total_Recall_(1990_film)	total recall	0	0.373
Arnold_Schwarzenegger	arnold schwarzenegger	0	0.296
Q2: "total recall movie"			
Total_Recall_(2012_film)	total recall	0	0.359
Total_Recall_(1990_film)	total recall	1	0.356

Table 3: Example queries with recognized interpretation sets. *Q1* has one only interpretation, while *Q2* has two interpretations, represented by different *setIDs*

.

Run ID	F-Measure	Diff
Test1	0.4826	-
Test2	0.4830	+0.001
Test3	**0.4946**	+0.01
Test4	0.4943	-0.0003
Test5	0.4553	-0.04
Test9	0.2461	-0.209

Table 4: Results for the ERD short text track. "Diff" represents the F-measure difference compared the previous run.

$E = \{e_1, ..., e_l\}$, precision and recall of the hypothesized interpretations $A = \{E_1, ..., E_n\}$ are:

$$Precision = \frac{|\hat{A} \cap A|}{|A|}, \qquad Recall = \frac{|\hat{A} \cap A|}{|\hat{A}|}.$$

The average F-measure of the evaluation set is simply the unweighted average of the F-measure for each query:

$$\text{AverageF} - \text{Measure} = \frac{1}{N} \sum_{i=1}^{n} F - measure(q^i).$$

6.1 Runs

The following runs were submitted to the ERD challenge.

Test1 The baseline run, which uses *LenRtio*, *QET*, *QCT*, *T-Sem*, *Abs-Sim*, and *T-Abs-Sim* as features. GBRT parameters are $t = 5$ and $l = 5$. ERD query annotations are used for training. Rank and score thresholds are $T_n = 3$ and $T_s = 0.1$.

Test2 Identical to `Test1`, except that all features mentioned in Table 2 are used.

Test3 Identical to `Test2`, but GBRT uses different parameter settings: $t = 5$ and $l = 5$.

Test4 Identical to `Test3`, except that the number of trees in GBRT is set to $t = 20$.

Test5 The same settings as in `Test3`, but the training set is extended with data from Yahoo! Webscope [1].

Test9 Identical to `Test3`, but the score threshold is changed to $T_s = 0.18$. The entity names lookup table has been extended to contain more name variants (specifically, <*foaf:name*> was added).

6.2 Results and Discussion

We tested the performance of our system by varying the features used, thresholds (T_s and T_n), GBRT parameters, and employing different training sets. Table 4 presents our experimental results on the official test set, as provided by the challenge.

As shown in Table 4, `Test3` is our best performing run. It uses all features described in Table 2, which confirms that adding new features improves the end-to-end performance of the system. However, even more can be gained from optimizing the learning model (cf. `Test2` vs. `Test3`). GBRT has a number of tunable parameters; out of these, the number of trees and the number of nodes have the highest impact in our setting. We tuned GBRT parameters in `Test2`, `Test3`, and `Test4` and found that having 10 trees, with 10 nodes each, improves the accuracy of the model, without overfitting the training data.

In the `Test5` run we extended the training set with the Yahoo! webscope data [1]. Contrary to our expectations, this resulted in a lower F-measure. This can be due to two reasons: 1) the Yahoo! webscope data does not consider different interpretations of a given query and links the single most relevant entity to each query; 2) by adding this dataset, too many negative instances are inserted to the training set. This results in an imbalanced training set (which should be addressed, e.g., by random sampling of negative instances).

In the last run, `Test9`, we changed the score threshold of the interpretation finding algorithm to $T_s = 0.18$ and improved the candidate entity detection step by adding more name variants. The results show drastic decrease of system performance; supposedly, this is due to a technical bug in our implementation. It was a last minute submission and the submission system was closed before we could investigate this further; a detailed examination of this problem is left for the future.

7. CONCLUSIONS

We described our participation in the short text track of Entity Recognition and Disambiguation (ERD) challenge. We employed a three-stage pipeline, where we first detect candidate entities, then score them, and finally find interpretation sets for the input query. Detection is based on known entity surface forms in DBpedia. Scoring employs a learning-to-rank method, with features revolving around entity mentions in the query and text-based similarities between the query and entity predicates. Interpretation sets are discerned using a greedy algorithm that considers the top ranked entities. Our results show that parameter settings of the learning-to-rank method and of the interpretation finding algorithm have the most impact on end-to-end system performance.

References

[1] Yahoo! webscope dataset, ydata-search-query-log-to-entities-v1-0. URL http://webscope.sandbox.yahoo.com/.

[2] C. J. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *Journal of Machine Learning Research: Workshop and Conference Proceedings*, 14:25–35, 2011.

[3] 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).

[4] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194(0):130 – 150, 2013.

[5] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: A graph-based method. In *Proc. of SIGIR '11*, pages 765–774, 2011.

[6] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[7] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *Proc. of WSDM'12*, pages 563–572, 2012.

[8] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proc. of CIKM'08*, pages 509–518, 2008.