Entity Linking for Queries

Yeyao Zhang (yezhang@student.ethz.ch)
Zhichao Han (zhhan@student.ethz.ch)
Lu Chen (luchen@student.ethz.ch)

Group **9** (DataMiners)
25 May 2016

1 Overview

We implemented 3 annotation models to fulfill this entity linking task for queries, namely Fast Entity Linker Model, Probabilistic Bag-of-Hyperlinks Model and Learning to Link Model.

Fast Entity Linker Model and Probabilistic Bag-of-Hyperlinks Model are both probabilistic models, following the paper "Fast and space-efficient entity linking for queries(2015)änd "Probabilistic Bag-Of-Hyperlinks Model for Entity Linking(2016)"respectively.

Learning to Link Model is a supervised learning inspired by the paper "A Piggyback System for Joint Entity Mention Detection and Linking in Web Queries(2016)".

The results(evaluated by relaxed and strict F1) of our approaches are shown in Table1.

2 Fast Entity Linker Model

Instead of separating the entity linking system into two parts, we use a Dynamic Programming Framework to compute the annotation. For each query, we exhaustively take all possible term combinations as mentions, and for each mention, we compute a score with respect to all it's relevant entities, and take the highest one as the mention-entity match. Then for each non-overlapping mention combination, we take the sum of the scores of each mention-entity pair in it, then select the combination which gets the highest sum as the final annotation for this query. The specific approach is illustrated in [1].

Note that we don't have as much context information as presented in the original paper, thus we do something different. Firstly, we simply use commonness to represent $P(e|m)$. When computing $p(t|e)$, we use a pre-trained word2vec dictionary, which is trained using wikipedia corpus. Then, we extract the description of a given entity on Wiki and average the terms on it to give a representation of the entity. However, the result of this model is not very satisfying. We made some changes, and build a PBOH Model, which gains a lot on all metric values.

3 Probabilistic Bag-of-Hyperlinks Model

PBoH Model considers three factors: mention-entity compatibility (commonness), context-entity interactions ($P(e|m)$) and entity-entity coherence (graph-based, relatedness), as shown below.

Also, we notice that misspelling in the queries has huge negative effects on the results. So we introduce BING Spell Check API to gather some extra information about the query. The evaluation results after spelling correction are greatly enhanced.

Tabel 1: Relaxed and Strict F1 of Different Approaches

		mac			mic			std		
		p	r	F	p	r	F	p	r	F
Fast Entity Linker	C2W	0.666	0.436	0.369	0.436	0.413	0.424	0.389	0.441	0.402
	Sa2W	0.653	0.413	0.353	0.424	0.375	0.398	0.392	0.433	0.394
PBoH with Spell Correction	C2W	0.615	0.601	**0.534**	0.552	0.558	**0.555**	0.408	0.404	0.398
	Sa2W	0.593	0.584	**0.517**	0.527	0.532	**0.530**	0.413	0.407	0.401
Learning to Link	C2W	0.659	0.538	0.497	0.574	0.533	0.553	0.410	0.414	0.405
	Sa2W	0.634	0.510	0.472	0.538	0.501	0.519	0.418	0.415	0.405

$$\log P(\mathbf{e}|\mathbf{m},\mathbf{c}) \propto \sum_{i=1}^{n} \log P(e_i|m_i) + \quad \text{mention - entity compatibility}$$

$$\zeta \sum_{i=1}^{n} \sum_{w_j \in c_i} \log P(w_j|e_i) + \quad \text{context - entity interactions}$$

$$\frac{2\tau}{n-1} \sum_{i<j} \log \left(\frac{P(e_i, e_j)}{P(e_i)\, P(e_j)} \right) \quad \text{entity - entity coherence}$$

4 Learning to Link Model

One problem of Baseline is that it only uses *commonness* to disambiguate entities. A more reasonable way is to consider contextual information (see example **??**). Here, we consider disambiguation as a binary classification problem. By adding features of contextual information, we can efficiently distinguish true and false annotations. Details will be explained later.

- **Training data generation:** In the training data, we can get the queries $Q = \{q_1, \ldots, q_n\}$, and true annotation set $\{a_{i1}, \ldots, a_{ij}\}$ for q_i. Each annotation a_{ij} is a *mention-entity* pair, that is $a_{ij} = (m_{ij}, e_{ij})$. For every mention m_{ij} in ground truth, we use the first three entities returned by *WikiSense*[1] as its candidate entity set $\mathcal{E}^c$. Table 2 shows that only using the first three or five entities returned by *WikiSense* would be enough. Then we can label each *mention-entity* pair (m, e) in $\mathcal{E}^c$ as positive class (+1, correct annotation) or negative class (0, false annotation).
- **Feature generation:** For each *mention-entity* pair (m, e) in training set, we construct its three features: (1) the *commonness* of (m, e); (2) contextual similarity of wikipedia title and remaining part of query except for this mention; (3) the contextual similarity of the first paragraph of wikipedia and remaining part of query except for this mention. The contextual similarity could be computed using word embedding[2].
- **Training classifier:** We use a simple binary classifier to learn the training data. Here, quadratic SVM and random forest are used.
- **Entity linking in new data set:** For each query in the test set, we can use *WikiSense* to spot all linkable mentions. Then we construct features of each linkable mention with its 3 top entities returned by *WikiSense*. And we can find the correct entity for this mention using the trained classifier.

Tabel 2: The Rank of Gold Standard Entities in Candidate List returned by *WikiSense*

	1	2	3	4	5	others	totally found	no found
A	256	28	7	6	4	17	318	123
B	279	18	8	1	1	15	323	110
devel	255	28	9	5	3	20	320	99

5 References

[1] Blanco, Roi, Giuseppe Ottaviano, and Edgar Meij. "Fast and space-efficient entity linking for queries."Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. ACM, 2015.
[2] Ganea, Octavian-Eugen, et al. "Probabilistic Bag-Of-Hyperlinks Model for Entity Linking.ärXiv preprint arXiv:1509.02301, 2015.
[3] Cornolti, Marco, et al. "A Piggyback System for Joint Entity Mention Detection and Linking in Web Queries."Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016.

[1] We discard this mention if it is not in *WikiSense*
[2] the pre-trained embedding file could be downloaded from https://github.com/3Top/word2vec-api, https://github.com/stanfordnlp/GloVe.