ENTITY LINKING AND TOPIC MODEL

Yifan SU, Hantian ZHANG

Department of Computer Science
ETH Zürich
Zürich, Switzerland

ABSTRACT

We designed and implemented an entity-linking system to link short and poorly composed search queries to corresponding Wikipedia pages. The system strives for enriching query contents as document for each query and utillizing web search results (contents of Wikipedia Pages) as candidate documents for all possible entities. After text preprocess, unsupervised LDA topic model is used to extract useful information through Gibbs sampling iterations. After the last iteration, all topics are represented as global word map probability distribution, all documents as topic statistic distribution. Then we sort the cos-similarity values between query document and entity documents of each query. Finally we apply rules for query-entity matching.

1. INTRODUCTION

Short query annotation seems straightforward and simple at the first glance. However, after a careful analysis, many limitations arose without noticing. Short queries carry less information compared with longer paragraphs and documents, especially when they are abbreviated or misspelled. Moreover, Wiki entities face this problem as well. It's indirect and difficult for us to catch the basic idea of the page just given the corresponding page name. So how to enrich our current knowledge and understanding of the query as well as entities is of great importance. We decide to find ways to augment short query and entities to longer documents.

2. SOURCES AND PRE-PROCESSING

Our documents come from three sources:

1. Query Extension: Here we use first page Bing search result for each query (title and brief intro for each snippet). We concatenate all the contents into one document to represent corresponding query document.

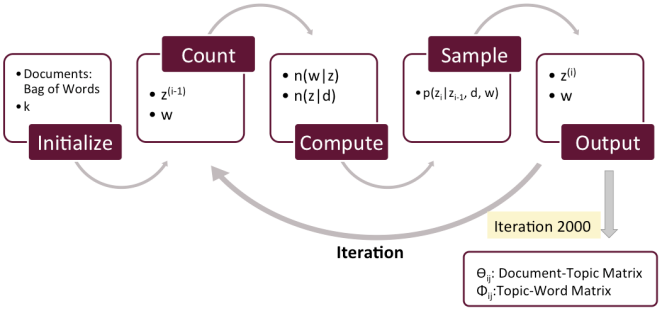
2. Candidate Entities Source 1: Here we use every Wiki page appearing in the first page of Bing search result as candidate entities.

3. Candidate Entities Source 2: We also use our naive baseline entity results as candidates. Together with the first entity sources, we copy the first 1800 characters of each candidate wiki page to represent each entity document.

Before applying topic model to our dataset, text pre-processing, which has significant effects on modeling results, need to be done first. There're several steps for this job: 1. Candidate Duplicate Entity Elimination 2. HTML CSS Special Symbol Filter 3. Stop Words Filter and Stemmer 4. Low Frequency and Short Word Filter 5. Global Word Map Construction (word to index)

3. LDA TOPIC MODEL AND GIBBS SAMPLING

After pre-process, topic modeling process then uses unsupervised LDA model to add a topic layer between document and word layers. LDA model can automatically extract topics which have obvious implications and distinction. The two unknown matrix(document-topic and topic-word) based on a collection of documents is required to estimate. We first choose topic number 60 for training set and 20 for develop and test dataset. Then we use Gibbs Sampling algorithm to sample each random variable in turn, iterate 2000 times after the convergence of the Latent Dirichlet Allocation model variables and estimate θ and ϕ matrix distributions. The work flow is as follows:



d	Document
z	Topic
w	Word
k	Number of Topics
θ_{ij}	Weight of Topic z_i in Document d_j
ϕ_{ij}	Weight of Word w_i in Topic z_j

After the last iteration, all topics can be represented as word-probability matrix, and all documents as topic weight matrix, which is exactly what we need to calculate the similarity in the next step.

4. ENTITY SORTING

Since each document can be represented as a weighted topic vector as in matrix ϕ, we need to calculate topic vectors' similarity between each query and corresponding entities. Then we sort the entities according to their similarity with the query vector.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

5. QUERY-ENTITY MATCHING

We also want to take in to account the edit distance between the query and potential entities. We rearrange the entities according to a weighted score of its similarity from the topic model and its Levenshtein distance with the query. Formally, for each query q and entity e, we have

$$S_{q,e} = \alpha * similarity(q,e) + (1-\alpha) * (1 - distance_L(q,e)) \quad (1)$$

After several tests, we chose $\alpha = 0.5$. We sort the entities from the highest score to the lowest score.

Having the final sorting of entities, we use a naive way to match each entity to mentions. We start with the entity with the highest score. We would match each entity to the longest substring of the query where each word in the substring appears in the entity. If no such substring exists, we choose the word that has the shortest Levenshtein distance with the entity. Also, we assume that there is no overlap between mentions, so if the current mention intersects with any mention that was chosen before, we would discard the current mention and the corresponding entity.

6. EXPERIMENT RESULT

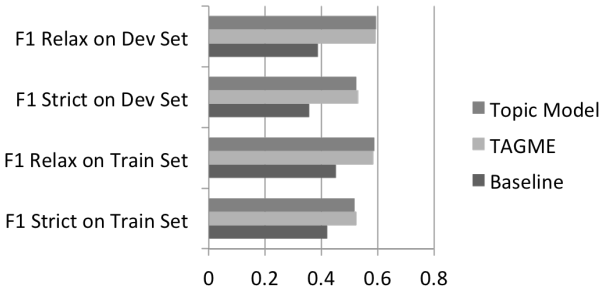
We show our experiment result in two parts.

Sorting Part. The following table lists part of our sorting result.

Query	Entity 1	Entity 2	Entity 3	...
Google youtube	Youtube: 0.48	Google: 0.29	History_of_YouTube: 0.085	...
Best but	Best_Buy : 0.82	Pete_Best: 0.27	The_Best_(song): 0.11	...
obama family tree	Michelle_Obama : 0.95	Family_tree: 0.91	Family_of_Barack_Obama : 0.049	...

As it shows, the candidate order seems to be reasonable in large.

Matching Part. We carried out two experiments calculate the three sources on the fully-annotated training and developing dataset, also un-annotated test set. We use F1 strict score and F1 relax score as metrics for measurement. Results are listed in the table.



There exists a big jump comparing to our original baseline and a little improvement corresponding to TAGME.

7. FUTURE WORK

However, because our team has only two members and time limited, the results can still be improved if we have time to try different parameters, sorting or matching methods. Here we provide some further tips.

- Try different parameters in topic model.
- Try complicated matching method instead of current naive way.
- Use longer document instead of only the first page of search result.
- Refer to and combine other state-of-art idea for query-entity linking.