Wikification: Entity annotation with Wikipedia

Jie Tang
jietang@stanford.edu

Abstract

Wikification(entity annotation) is one of challenging NLP tasks. In this project, we built an end-to-end system to annotate document with wiki links. At first we implemented baseline model with LESK algorithm/Bayesian model. Then we applied Siamese neural network in this model, it improves similarity accuracy a lot, meanwhile it improves annotation precision from 84% to 88%.

1 Introduction

Nowadays, people usually links text to Wikipedia link at webpage so that readers could get extra information from Wikipedia, meanwhile publishers don't need to supply background information. In natural language processing, this is called entity disambiguation, entity linking or entity annotation. It's a fundamental component for understanding text semantically since natural language by nature is ambiguous. Linking text to wiki entity is aimed to eliminate wording disambiguation, also this is also good moving from unstructured words to structured/semi-structured data. E.g. for a sentence

"One-night President Obama and his wife Michelle decided to do something out of routine and go for a casual dinner at a restaurant that wasn't too luxurious"

We will link "President Obama" to US president "Barack Obama", "Michelle" to "Michelle Obama", also since we know the underground entities, by looking up freebase, we know "Barack Obama's wife is "Michelle Obama", we may also link wife to "Michelle Obama".

This is very attractive since we could understand real entity of text and even also extend more information from entities' relationship/properties. But entity disambiguation is very challenging problem, it's mainly due to following issues:

Words/phrases in different contexts show different meanings(senses) or link to different entities. E.g. "I like apple." "apple" could be linked to "Apple Inc." or apple fruit. Correctly identify different contexts is the key to link correct entity to text.

One entity could be linked to different aliases, e.g. Microsoft has aliases like MSFT, MS, etc. It's not easy to know all those aliases, if considering different languages, it's much more difficult.

In this project, I plan to implement one baseline model purely based on unsupervised learning/statistical model. In original plan I was going to train a LSTM model which consider wiki link annotation as a translation problem, but when I started working on it, I realized this is not a doable project since English wiki entities are more than 7M, it's impossible to load so large vocab into LSTM model, so I switch to training a Siamese Network to improve baseline model's core component.

2 Background and related work

There are lots of research papers in this area, [1][2] use lesk algorithm which considers wiki page as dictionary gloss, words in the same sentence as context, so wiki pages which share biggest overlap with context would be linked to the text. [1][2][3] also implemented supervised word disambiguation system, which extract features like surrounding words and their POS tags in a context window, co-occurred words in the same sentence, local collocations which are common expressions containing the word to be disambiguated, and also the verb and noun before and after the ambiguous word. [1] also merge two methods by some voting mechanism.

[4] defines commonness(number of times the text is used as a destination in Wiki) and relatedness(semantic similarity between two wiki pages, by comparing their incoming/outgoing links). The disambiguation process is trying to balance commonness and relatedness.Besides commonness and relatedness, [5] also introduces text/entity compatibility score, redefine relatedness as entity coherence to model compatibility across entities, then optimize a model to balance between local/global optimality. [6] built a system named TAGME, which annotates short texts with wiki links.For each phrase, TAGME defines collective agreement between one of its possible entity and all possible entities of other phrases, collective agreement is still based on relatedness and commonness under the hood.[7][8] proposed some graph-based algorithm to solve WSD and entity linking.

3 Approach and Models

3.1 Baseline model

With LESK algorithm, we consider each wiki page as dictionary gloss, words in annotating document as context, so for each n-gram in document, we will annotate it with the wiki page which shares biggest overlap with current document. We combine this algorithm with Bayesian model to build a baseline model.

3.1.1 Model description

$$bestlink = \max_{link} p(link|context, text)$$

Where context is sentences/words around the text we are going to annotate with wiki link. If we assume context and text are independent, (this is usually not true, to make computation simple, we make this assumption here), so we have

$$p(link|context, text) = \frac{p(context|link)p(text|link)p(link)}{p(context,text)} = \frac{p(context|link)p(link|text)p(text)}{p(context)p(text)} = \frac{\frac{p(context|link)p(link|text)}{p(context)}}{}$$

$p(link|text)$ could be easily calculated as

$$p(link|text) = \frac{doc\_count(link, text)}{doc\_count(text)}$$

where $doc\_count(link, text)$ is number of documents, text is labeled by the link, $doc\_count(text)$ is number of documents contain the text.

For $p(context|link)$, we may model it as a norm distribution for a given link,

$$p(context|link) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{dist^2}{2\sigma^2}}$$

where $dist = 1 - similarity\ (context, link) \in [0,1]$, $\sigma = \sqrt{var(dist)}$ is calculated over all contextual texts we found in Wikipedia pages for a given wiki links, in our experiment the contextual sentence window is 1.

To calculate $similarity\ (context, link)$, for each wiki link, we extract outgoing anchors from its own wiki page, look up word vector for each word in these anchor texts, average all these word vectors as this wiki link's embedding vector. The reason we do so is that usually anchors in wiki page are highly related with the wiki page's main topic, could represent the wiki page well.

We also average word vector for all words in the context, but we weight each word by $weight = \frac{D_{link}}{D_w}$, where $D_{link}$is number of document where the word is in anchor text, D_w is number of document where the word is occurred. This way we could reduce impact of the words are not frequently occurred in anchor text, here we hold the same assumption, words in anchor text would be more representative.

Finally $similarity(context, link)$ is calculated by cosine similarity of two average vectors. And now the criterial to select best link is like this

$$bestlink = \max_{link} p(context|link)^{w_{context}} p(link|text)^{w_{prior}}$$

where $w_{context}, w_{prior} \in [0,1]$ and $w_{context} + w_{prior} = 1$, they are here for balancing impact of prior knowledge/context over final decision, in our experiment.

3.1.2 Final algorithm

The final algorithm is implemented like this

1. For each document, break down it into sentences.
2. For each sentence $S(i)$, create a context window $C = \{S[i-c], .. S[i-1], S[i+1] \ ... S[i+c]\}$
    a. Extract n-grams from $S[i]$, $n = 1, .., len(S[i])$
    b. Lookup each n-gram in anchor text dictionary, if there is no corresponding entry, drop it. Otherwise keep it and its all possible corresponding wiki links as potential candidates.
    c. For each remaining n-gram, at first we remove words in n-gram from its context window, then select best link as we described in 3.1.1, here $context = C \cup S(i)$ (words in n-gram are also removed from $S(i)$), and also store the best link's score. So after this step, for each n-gram, we keep only one wiki link for it.
    d. At this step, we resolve overlapped n-grams by maximizing likelihood $L = \prod p(link|context, text)$ over the whole sentence:
    $L(i) = \max_j score(i,j) * L(pre(j))$ where $score(i,j)$ is score of j-th ngram which ends at i-th word(named as $ngram(i,j)$, more than one ngram may end at i-th word), $pre(j)$ is word index of nearest forward n-gram which is not overlapped with $ngram(i,j)$, $L(i)$ is largest likelihood ends at i-th word.

So finally in each sentence, we assign wiki links to non-overlapped n-grams, each is attached with an annotation score. We could set some threshold to filter unconfident annotation.

3.2 LSTM model/seq2seq model

My initial plan was train a LSTM model which could translate text to entities, I expect LSTM model may capture contextual information better. But in real practice, I found this is not doable, given following reasons:

- Number of entity is far more than number of words, e.g. there are more than 7M English entities. It's impossible to load such a large vocab into LSTM model, at least till now I can't figure out a way to do so.
- I may reduce vocab by filter not frequent entities, or split entities to words(so that we learn translation model between raw text and wiki link at word level), but this would reduce coverage/precision a lot, it's hopeless to beat baseline model.

Based on these 2 reasons, I switch to improving base line model with Siamese network.

3.3 Siamese network

In baseline model, the key component is $similarity\,(context, link)$, in some cases, we underestimate this similarity, so that link is wrongly annotated as mostly frequent wiki link. There are a few things we may tune to improve precision:

1. Better prior weight, with proper weight we could reduce prior probability's impact over final decision.
2. Better features for calculating similarity, in baseline model we simply average selected words' embedding vector as features.
3. Better way to calculate similarity, in baseline model, we simply apply cosine similarity over feature vectors.

We decide to find a better way to calculate similarity between textual context and wiki links, while keep other parts as the same with baseline model. As [11] suggested, we built following neural network to better model context/wiki link similarity:

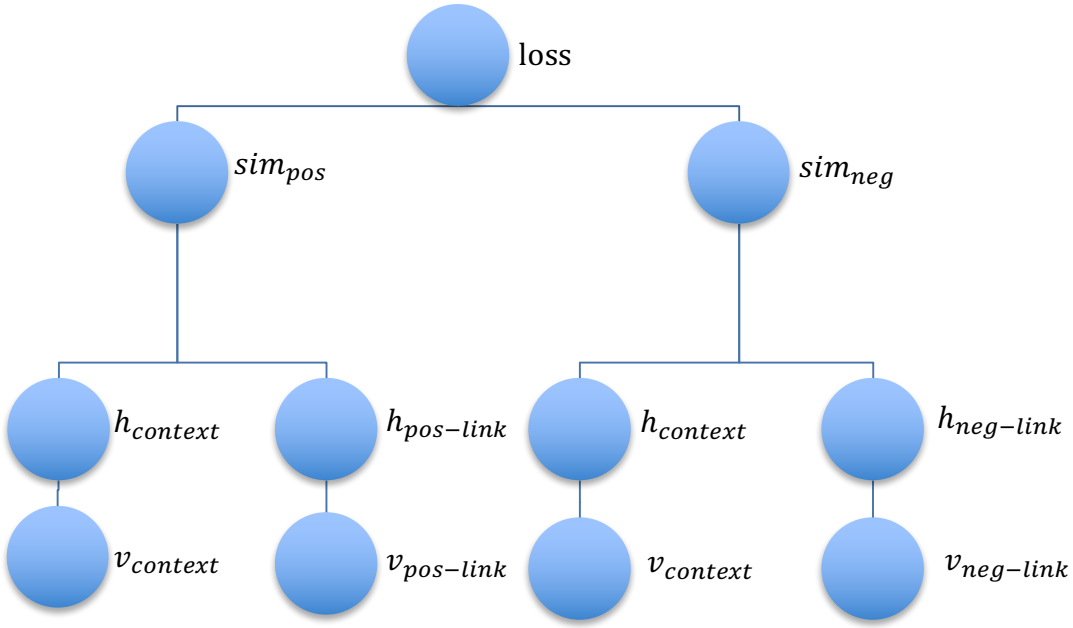

Figure 1 Siamese network

where

$$h_{context} = \sigma(v_{context}W + b)$$

$$h_{neg-link} = \sigma(v_{neg-link}W + b)$$

$$h_{pos-link} = \sigma(v_{pos-link}W + b)$$

$$sim_{pos} = \frac{h_{context} \cdot h_{pos-link}}{|h_{context}||h_{pos-link}|}$$

$$sim_{neg} = \frac{h_{context} \cdot h_{neg-link}}{|h_{context}||h_{neg-link}|}$$

$$loss = \log\,(1 + e^{-\lambda(sim_{pos} - sim_{neg})})$$

$v_{context}$ $v_{pos-link}$ $v_{neg-link}$ $\in R^{1\times n}$, they are embedding vectors of contextual text/correct/incorrect wiki links of the text. $W \in R^{n\times m}, b \in R^{1\times m}$, $h_{pos-link}$ $h_{neg-link}$ $h_{context-link} \in R^{1\times m}$. n is embedding vector size, m is hidden layer size. σ is sigmoid function, λ is the weight to scale difference between sim_{pos} and sim_{neg}.

By minimizing $loss$, we will make correct wiki links yield higher similarity score than incorrect wiki links.

4 Experiment

4.1 Training Data

We downloaded Wikipedia dump on May 8th 2016, only keep English wiki pages, filtered out disambiguation/list/redirection pages. For both baseline model and Siamese Network based model, we preprocess data as following steps:

1. Inside each page, outgoing wiki anchors are extracted, with those anchors, we build a table which maps anchor texts to wiki links, this table is used to calculate $p(link|text)$, and also acts as dictionary during annotation process.
2. We also create a table maps wiki link to outgoing links from its own page, this table acts as wiki link's dictionary gloss.
3. Create a table to map Wikipedia link to its redirection destination Wikipedia page URL, this is useful in evaluation since some wiki links in evaluation data set are redirection pages.
4. Also we create another table to map each wiki link to its similarity variance.
5. Finally we create a table to map each word to its link weight $\frac{D_{link}}{D_w}$.

For Siamese Network based mode, we prepare training samples this way:

1. For each Wikipedia page, break document into a few sentences.
2. For each sentence, pick up previous/next sentence as contextual text, then iterate each link inside the sentence:
    a. Merge words in contextual text and current sentence as context, remove words in link text from context.
    b. Generate average embedding vector for context and current link (correct link) as we described in 3.1.1, we also look up link text in dictionary, so that we could get other candidate links for the link text, we call these links as incorrect links, embedding vectors are also generated for these links.
    c. Calculate similarity between context and correct/incorrect links.
    d. Generate following examples for current link
        i. {correct link, context, top incorrect link}, top incorrect link is the incorrect one with highest score. If {correct link, context} similarity score > {top incorrect link, context} score, it's a positive example, otherwise it's a negative example.
        ii. random sample a few {correct link, context, non-top incorrect link}, they could be either positive or negative example.
3. Finally, we sample 1M training examples, 40K dev examples, 40K testing examples, in this sampled dataset, by apply cosine similarity, the precision would be 63%.

4.2 Evaluation Data

There are a few data sets available for evaluation:

1. Wiki-disamb30 and Wiki-annot30 in [6]
2. IITB dataset in [5]
3. ERD-14 challenge data set, which is annotated by freebase id.
4. Annotated web pages in [10]

2 and 4 are annotated by one or two people manually, the results are subjective and biased, since when people label webpages, they usually choose annotate text they are familiar with, or for unknown terms, they do short research to make decision, this is sometime error-prone especially for some professional term or not famous person. Wiki data is edited by lots of people, it's much cleaner, so I finally choose wiki-annot30 as my evaluation data set. Wiki-Annot30 contains a list of 186K Wikipedia page fragments, which are drawn from Wikipedia snapshot of November 6, 2009. Fragments are composed by about 30 words, and they contain about 20 *non-stopword* on average. I filtered text contained in wiki-annot30 from my

wiki training data, since wiki-annot30 was snapshotted 6 years ago, lots of texts are changed a lot after that, it's not easy to filter all of them, there may be still some few overlap.

4.3 Evaluation

4.3.1 Baseline model

Besides the algorithm we described above, we select baseline model weight parameters as $w_{prior} = 0.1, w_{context} = 0.9$. And we use glove.6B[12] embedding vector (200d) in our system.

There are 835,757 valid entities in Wiki-annot30, and another 18,281 invalid entities whose Wikipedia page ids can't be found anymore. My baseline model generates in total 1,497,147 entities, among them:

- 473,980 entities are aligned with Wiki-annot30
- 88,970 entities are annotated by different wiki links from the links in Wiki-annot30.
- 6,173 entities are missed from our output, that means for these phrases/words, we fail to annotate them by wiki link.
- We still output other 934,197 entities, which are not overlapped with Wiki-annot30's output. For these 934,197 entities, it's difficult to tell whether they are correct or not, since they are not annotated in evaluation data set. So here we have to ignore them.

We follow [6] to define topic-based notion of precision and recall:

Let $G(T)$ be the wiki links associated to the anchors of text T in the ground truth, and let $S(T)$ be the wiki links identified by the tested system over text T, so

$$P_{topics} = \frac{\sum_{T \in \mathcal{F}} |G(T) \cap S(T)|}{\sum_{T \in \mathcal{F}} |S(T)|}$$

$$R_{topics} = \frac{\sum_{T \in \mathcal{F}} |G(T) \cap S(T)|}{\sum_{T \in \mathcal{F}} |G(T)|}$$

So for baseline mode, P_{topics}=$\frac{473{,}980}{473{,}980+88{,}970}$ =84.2%, $R_{topics} = \frac{473{,}890}{835{,}757}$=56.7%

4.3.2 Siamese Network based model

We create a Siamese network with hidden layer size is 256 as we described in Figure 1. The parameter is $l2 = 1e-6, \lambda = 100, batch\ size = 32, learning\ rate = 0.01,\ dropout = 0.95$ max $epoc = 20$. As we said in 4.1, cosine similarity's precision over testing set is around 63%, the trained network outperforms cosine similarity a lot, achieve 78% precision over the same testing set.

For overall annotation system end-to-end test, by replacing cosine similarity with Siamese network, P_{topics} is increased to 88%, R_{topics} is increased to 58%.

4.3.3Errors analysis and examples

To demonstrate performance of our system, we also tried over some toy examples to show the system doesn't always annotate text with most frequent link, e.g.

1. Let's have some beer in bar.
   We correctly link "bar" with "/wiki/Bar", instead of most frequent link "/wiki/Bar_(law)"
2. Fishing to get some bass
   "bass" is linked with "/wiki/Bass_(fish)" by our system, instead of more frequent link "/wiki/Bass_guitar".

By checking the evaluation results, we found errors mainly come from following parts

1. POS mismatch
   Our system doesn't consider POS alignment, so for example "(born 11 January 1984) is a Bulgarian footballer currently playing for Lokomotiv Sofia as a striker.", we wrongly link "Bulgarian" with /wiki/Bulgarian_language, in fact /wiki/Bulgaria is a better choice considering POS alignment.
2. Coreference
   Our system annotates each text segments independently, so if first mention is full name, second mention is first name, we may wrongly link second mention to another entity with higher prior probability.
3. Incorrect contextual similarity
   We represent context/link by averaging word vectors, this is an efficient way, but not accurate. In some case, it can't capture contextual information correctly. E.g. "may limit the power output of microwave frequency transmitters in spacecraft and high-altitude aircraft applications. Waveguide components in such applications are sometimes pressurized to overcome the limitation imposed by", our system links "pressurized" to "/wiki/Cabin_pressurization", actually it should be linked to /wiki/Atmospheric_pressure, we are just biased by the word "spacecraft".
4. Evaluation data error
   There are also some errors in evaluation data set, e.g. "He has played for the Northern Virginia Royals of the United Soccer League's", is evaluation data set, "United Soccer League" is linked with "/wiki/United_Soccer_League_(1984%E2%80%9385)", but in latest Wikipedia page, it's already fixed to "/wiki/United_Soccer_League".
5. Different word breaking ways
   E.g. for "Canadian province of Quebec", our system will annotate "Canadian province", and "Quebec" separately, but in evaluation set, only "province of Quebec" is annotated.
6. Our system suggests better links
   In some cases, our system annotates text with more specific wiki link, e.g. "as the governor of the state of Washington", we will link "governor" to "/wiki/Governor_(United_States)", while in evaluation data set, it's linked with more general concept "/wiki/Governor".

Also since entity annotation somehow is subjective, even different human beings may generate different annotation results, evaluating system by comparing to a fixed labeled data set may not be a good idea. A better way may be asking some human raters to check whether annotated wiki links are correctly linked with texts. But this is time consuming and will cost too much money.

5 Conclusion and future work

In this project, we successfully built an entity annotation system based on Wikipedia database. By applying Siamese neural network, we improved the system precision and recall. But there are a few directions we may want to continue exploring:

- Apply POS signal/Coreference in to the system, this is known drawback of current system.
- Better features to represent wiki link and contextual information, right now we simply average word vector. This way, we weight each word equally, but actually some word may be more meaningful. Also parsing tree structure of context may contribute more useful features like which term is more meaningful.
- Deep learning shows big improvement in high level NLP tasks like named entity recognition, sentiment analysis, etc. But for some low level NLP tasks like Wikification/entity annotation, seems there is still no good way to apply deep learning directly into this field, we have shown Siamese neural network could improve precision a little bit, but the whole system still depends on good data statistics a lot. We may explore more on this direction. e.g. We may try to train a

model directly capture context/link similarity, instead of making some independency assumption.

References

[1]Rada Mihalcea, Andras Csomai. Wikify! Linking Documents to Encyclopedic Knowledge,CIKM'07

[2]Rada Mihalcea, Using Wikipedia for Automatic Word Sense Disambiguation,Proceedings of NAACL HLT 2007

[3]Hwee Tou Ng,Hian Beng Lee, Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach, IN PROCEEDINGS OF THE 34TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS

[4] David Milne, Ian H. Witten, Learning to Link with Wikipedia,CIKM'08

[5] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti, Collective Annotation of Wikipedia Entities in Web Text, KDD'09

[6] Paolo Ferragina, Ugo Scaiella, TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities), CIKM'10

[7] Andrea Moro, Alessandro Raganato, Roberto Navigli, Entity Linking meets Word Sense Disambiguation: a Unified Approach, Transactions of the Association for Computational Linguistics, 2 (2014) 231–244

[8] Johannes Hoffart, Mohamed Amir Yosef , Ilaria Bordino, etc, Robust Disambiguation of Named Entities in Text,Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing

[9]Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to Sequence Learning with Neural Networks, NIPS 2015

[10]Silviu Cucerzan, Large Scale Named Entity Disambiguation Based on Wikipedia Data

The EMNLP-CoNLL Joint Conference, Prague, 2007

[11] Wen-tau Yih Kristina Toutanova John C. Platt Christopher Meek, Learning Discriminative Projections for Text Similarity Measures, 2011

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation