Iran University of Science and Technology
Deep Learning (Spring 2019)

Deep End-to-End Entity Linking

Mohsen Tabasi
Department of Computer Engineering
Iran University of Science
and Technology
s_tabasi@comp.iust.ac.ir

Abstract

Entity linking is the task of detecting text spans which mention named entities and then linking them to corresponding entries in a knowledge base. Unlike named entity recognition, this cannot be formulated as a sequence labeling (classification) task, because in that case we will have millions of classes (one for each entity). So people generally limit entities to a few candidates and then solve a ranking problem based on the input context and their knowledge of entities. My efforts were focused on embedding our knowledge of entities in dense vectors, in a way that my model can regenerate a very similar vector whenever it hits a related mention.

1 Introduction

Entity linking (EL) includes detecting parts of an input text which mention some entities and disambiguating mentioned entities by choosing the correct one from a provided knowledge base. More formally given a text document as input, an entity linking model should output a list of tuples including: a) Exact boundary of detected mention. b) A link to the correct entity in a valid format like its permanent ID in a knowledge base, a URL to its canonical page or something else based on which knowledge bases are used.

Although the whole process above is needed to perform entity linking, people usually consider the disambiguation part as EL and refer to the whole as end-to-end EL. This is because detecting mentions is very simillar to what we do in NER, but disambiguation has some challenges special to EL. Let's see what are the main questions and challenges of entity linking?

- What is an entity? Is it named entity just like NER? The answer is not very clear! Based on the working dataset, you can see entities in limited named entity classes (like entities in AIDA-YAGO dataset) or you may count each Wikipedia article (including any abstract concept) as an entity.
- You cannot consider EL as a simple sequence labeling or classification. Because there will be thousands or millions of classes (the number of entities). So we need to filter them somehow or deal with the problem in another way.
- Although EL may seem very easy for humans, it can be very hard for machines because of very similar names for multiple entities (usually persons) and several possible ways to mention a single entity (nicknames, temporary titles, etc). In order to address this, your model should have enough knowledge about all of entities, plus the ability of comparing that knowledge with input context (good text understanding).

(LateX template borrowed from NeurIPS 2019)

2 Related work/Background

Most of recent models, trained entity embeddings to compress their knowledge of each entity and use it for disambiguation. For example (Yamada et al., 2016) proposed a method for joint learning of word and entity vectors. It is based on extending skip-gram model (Mikolov et al., 2013) to have entities beside words. They train the skip-gram so that it can predict context words given a single word (its natural behavior), predict context words given an entity (using anchor contexts) and finally predict related entities given a single entity (using graph model of wikipedia as a KB). This is done by three objective functions and makes entity embeddings close to their related words and entities. Then to disambiguate mentions, they encode context by simply averaging word vectors and compare it by candidate entity vectors. in addition they also average vectors of unambiguous entities in context and compare it to ambiguous candidates. finally candidates with better overall score will be the answer.

Another approach by (Gupta et al., 2017) focused on gathering maximum amount of information in entity embeddings. They used information of three different sources to train unified dense vectors: entity description, context around its mentions and fine-grained types from structured sources. Although the performance wasn't better than previous models, I think the idea of embedding multiple types if information led it to EMNLP 2017.

Next year (Raiman et al., 2018) presented a very different method, named DeepType. The idea was to make best use of structured knowledge to disambiguate entities. This is done by designing a type system which gives us optimized fine-grained types of entities. By optimized we mean these types are both useful to disambiguate entities and easy to predict by a classifier. Having the type system with about 120 type axes, they trained a type classifier to predict those types and then disambiguate mentions almost easily. The model inspired me to embed these types in an entity embedding along with extra information to design an end-to-end entity linking model, what I have not managed to do yet!

3 Proposed method

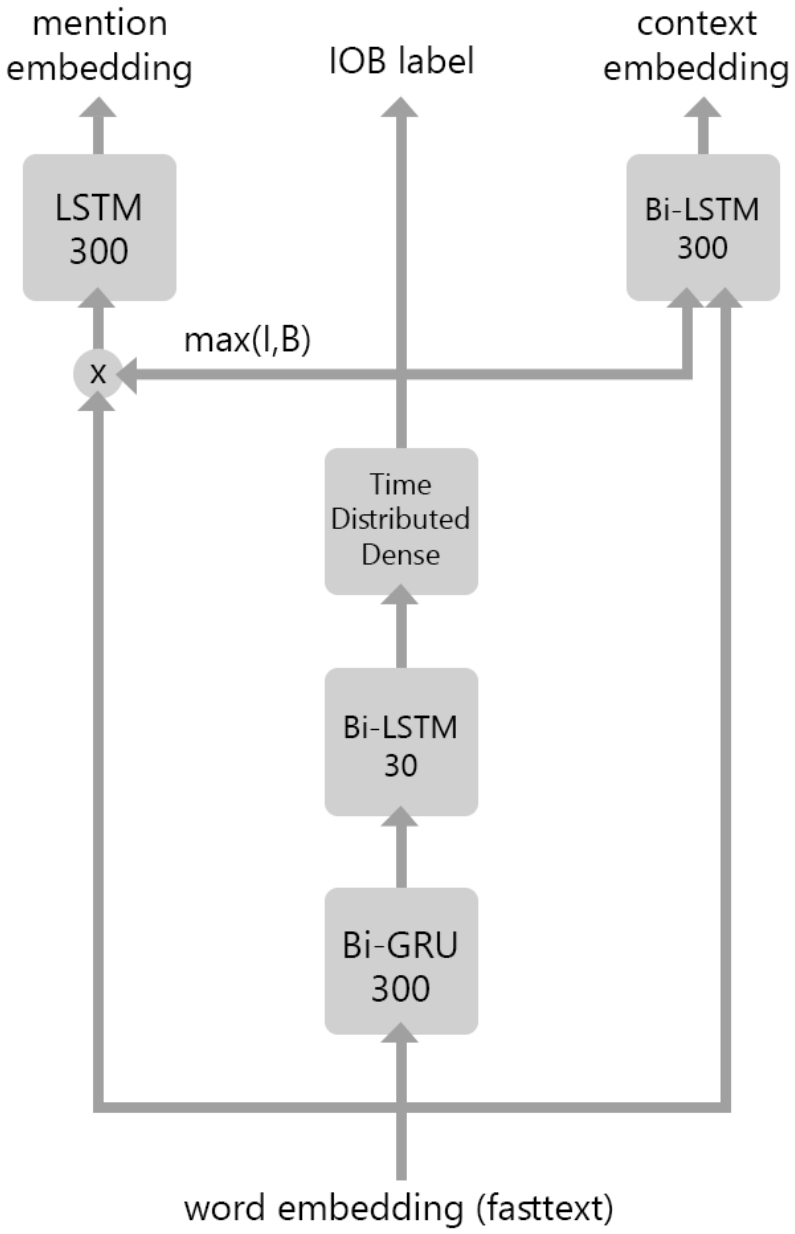
I used AIDA CoNLL-YAGO (no paper found) as one of the most famous entity linking datasets. It has 946 labeled documents for training, 216 for validation (test a) and 231 for test (test b). My Wikipedia dump was a 2010 release because it was the closest available version compared to the dump AIDA tags were based on. FastText word embedding (Bojanowski et al., 2017) is used to get better representation for unknown words in proper nouns and entity titles and abbreviations. It gives embedding of unknown words using smaller n-grams it the word.

First we need to create entity embeddings which consist of two parts: mention embedding v_m and context embedding v_c. For creating mention embedding of an entity e, all text span from Wikipedia which are linked to that particular entity, concatenated, is considered as a single document d_e. Then randomly choosing a large number (say ten thousand) of other entities, we will have a set of mention documents D_m. The mention embedding of entity e will be the average of word embeddings for all words in d_e weighted by their TF-IDF score.

Creating context embedding is similar to previous. The difference is that here d_e will be right and left context of all links of the entity in Wikipedia. We randomly choose other entities to reduce the computational cost and also make equal conditions for new entities. In both mention and context embedding, we can raise TF-IDF scores to the power of $0 < \alpha < 1$ to equalize weights a little after removing scores lower than a threshold t. Finally each entity embedding will be concatenation of v_m and v_c , but we don't really need to join them.

First I had the idea of embedding fine-grained types, extracted by () but I was unable to get them because of lack of space and computational power!

Having entity embeddings, we can train our model on AIDA dataset to jointly learn mention boundaries (in IOB fromat), generate mention embedding where an entity is mentioned and catch context clues in output context embedding. Then after feeding a document to the network, we get mention embedding from where the network detected B label (beginning of mention) and get context embedding from a word before it and a word after mention (after B tag or after last I tag). Now the entity with closest embedding will be the correct one.



The designed network is not mature yet!

4 Results

Unfortunately my model isn't fully implemented yet and initial results are not acceptable (below my simple baseline!). But I will try to do it eventually...

5 Discussion

Having the idea of embedding fine grained types into entity vectors, I thought the most significant information needed to make a perfect entity embedding was simply its name(s). That is because the span used to mention an entity is our greatest evidence to recognize the correct one. Other models

usually use mention itself to prune entities to a few candidates and then solve a ranking problem. But I want my model to predict correct entity by generating a vector that is as similar as possible to its real embedding. This makes the model free of sticking to special cases for special entities and open to new arriving entities.

After implementing first model based on mention embedding, it turned out that detecting the correct mention boundary was a major problem of the model. Since I used a simple 2 layer bidirectional RNN, it wasn't comparable to state-of-the-art models. While disambiguation-only models bypassed this step and assumed it as soleved, it was one of my main problems that paralyzed mention embedding. Besides that, my decision to avoid pruning entities to few number of candidates, made detecting mentions harder, because the model had no idea of which words or phrases can actually be title of an entity. For example if you don't know about a company named Apple, how could you detect lowercased "apple" as a mention?

Considering that, a solution that comes to mind is that the model can work with an external content-addressable read-only memory which has all entity embeddings in it. This way our model can evaluate quality of generated embedding and decide to declare it as an entity or not at the moment. I know this will require a lot of computational power and maybe I should rethink about not pruning entities...

References

[1] Yamada, I., Shindo, H., Takeda, H., & Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343.*

[2] Gupta, N., Singh, S., & Roth, D. (2017). Entity linking via joint encoding of types, descriptions, and context. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2681-2690).

[3] Raiman, J. R., & Raiman, O. M. (2018, April). DeepType: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence.*

[4] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135-146.

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).