Entity Linking & Entity Disambiguation Survey

Group 4

Task Introduction and Survey

A sentence “Apple released the latest product iPhone13”, one would want to only retrieve all mentions of “Apple (technical company)” and exclude mentions of other objects with the same name such as “Apple(fruit)”. However, the main obstacle in this setup is the lexical ambiguity of entities.

This is the reason use Entity Linking (EL) to match a mention, e.g. “Apple”, in a textual context to a database record (e.g. “technical company” or “fruit”) fitting the context, which is the key technology enabling various semantic applications. Thus, Entity linking is the task of identifying a mention in the (unstructured) text and establishing a link to an entity in a (structured) knowledge graph. EL is an essential component of many information extraction and natural language understanding pipelines since it resolves the lexical ambiguity of entity mentions and determines their meanings in context.

EL technique can be widely used. It is an essential building block for various Natural Language Processing tasks as information extraction, biomedical text processing, or semantic parsing and question answer. This wide range of direct applications is the reason why entity linking is enjoying great interest from both academy and industry for more than two decades.

An EL system typically performs two tasks which are Mention Detection (MD) or Named Entity Recognition (NER) which extracts entity references in a raw textual input, The second task is Entity Disambiguation (ED) based on entity linking is to map a mention in context to corresponding entity in the database. Until recently, the biggest difficulty in EL occurs around entity disambiguation because of the ambiguity of entities. The result of Mention Detection is difficult to attach directly to the knowledge graph.

Key to solving entity-related tasks is a model to learn the effective representations of

entities. Conventional entity representations assign each entity a fixed embedding vector that stores information regarding the entity in a knowledge base (KB) (Bordes et al., 2013; Trouillon et al., 2016; Yamada et al., 2017). One of the first surveys on Entity linking was prepared by Shen et al. in 2015. They cover the main approaches to entity linking, its applications, evaluation methods, and future directions. The work of Martinez-Rodriguez et al. in 2020, involves information extraction models and semantic web technologies. They consider many tasks such as named entity recognition, entity linking, terminology extraction, keyphrase extraction, topic modeling, topic labeling, relation extraction tasks. Similarly, the work of Al-Moslmi et al. (2020), overviews the research in named entity recognition, named entity disambiguation, and entity linking.

The task of entity disambiguation based on entity linking is to map a mention in context to corresponding entity in the database. A natural approach is to learn entity representations that enable this mapping.

Since entities are usually stored in a database and represented as a structured text record. One of the solutions of ED task is to directly calculate and rank the similarity score of detected mentions and corresponding candidates. Recent work uses a Universal Sentence Encoder (USE) model (Cer et al., 2018) for encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks. We improved the original USE model to achieve better performance. It is a structured text-based entity disambiguation.

The traditional popular works separately address MD and ED stages of EL, without leveraging their mutual dependency, which caused by MD will propagate to ED without possibility of recovery. Recent study jointly performs two task achieved impressive performance gains (Kolitsas et al., 2018*)*. The challenges in joint MD and ED modeling arise from their different nature. Few previous methods tackle the joint task, where errors in one stage can be recovered by the next stage. Hence, our second model is a neural end-to-end EL system that jointly discovers and links entities in a text document. The model computes span embeddings that combine context-dependent boundary representations with a head finding attention mechanism. It is trained to maximize the marginal likelihood of gold antecedent spans from coreference clusters and is factored to enable aggressive pruning of potential mentions.

Furthermore, to solve the NIL prediction problem, a BERT based model which introduce a global coherence mechanism (Yamada et al., 2020). Conventional entity

representations assign each entity a fixed embedding vector that stores information regarding the entity in a knowledge base. Document level's information will be beneficial. And this model provides a novel idea to add global information for ED, and it achieved well results.

Methodology

Dataset Introduction

The dataset we used is the bench mark datasets of this ED task. And we further process the dataset the form a well-structured one. It consists of three Python classes ($Document$,$Mention$, $Candidate$). Document take Document Id, tokenized results of raw text as $words$, and all tagged $mentions$ as it third attributes. And the element of the Document's mention is another class. Besides the mention itself, this class also store the context and position information of the mention, and the generated candidates of the mention. And the last class is candidate, which is the sub elements of mentions attributes candidates, contains the candidate content and its prior probability

Document	Mention	Candidate
Attribute	Attribute	Attribute
1. Id	1. Text	1. Title
2. Words	2. Start	2. Prior prob
3. Mentions	3. End	
	4. Title	
	5. Candidates	

Figure 1 Three Python classes of Dataset

Model 1 – fine-tuned USE Entity Linking

Universal Sentence Encoder is a sentence level encoder, it was trained based on Transformer's structure on large datasets. The model will take in raw text and output a fix size vector representation of the origin text. We fine-tuned the model by adding an additional Fully connected layer upon the encoder. And then borrowed the idea from $ResNet$ (skip-connection) to stabilized model performance. And then, after we obtain the fix-size representation of both mention and candidate entities, we are then able to calculate the cosine similarity score for mentions and their candidates. The candidate with highest score will be picked as the output result.

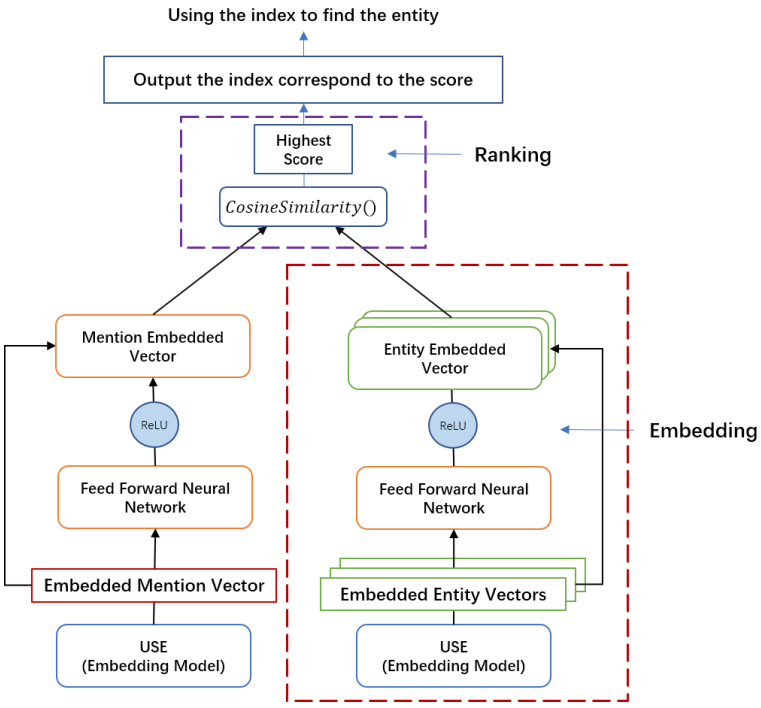

Figure 2 Model 1 Architecture

In training phrase, we set the loss function to be $L1$ loss (reduction=" sum"). For each iteration, we calculate the L1 loss between the predicted vector and the target vector (The one that input mention should be predicted as). And then using the SGD optimization method to update the parameters.

Model 2 – End-to-End Neural Entity Linking

This model is end to end type, which means just input the raw text and get the linking output directly, it includes mention detection as well as the entity disambiguation. The process is equivalent to a black box, we do not need to process the features anymore.

Basic Flow:

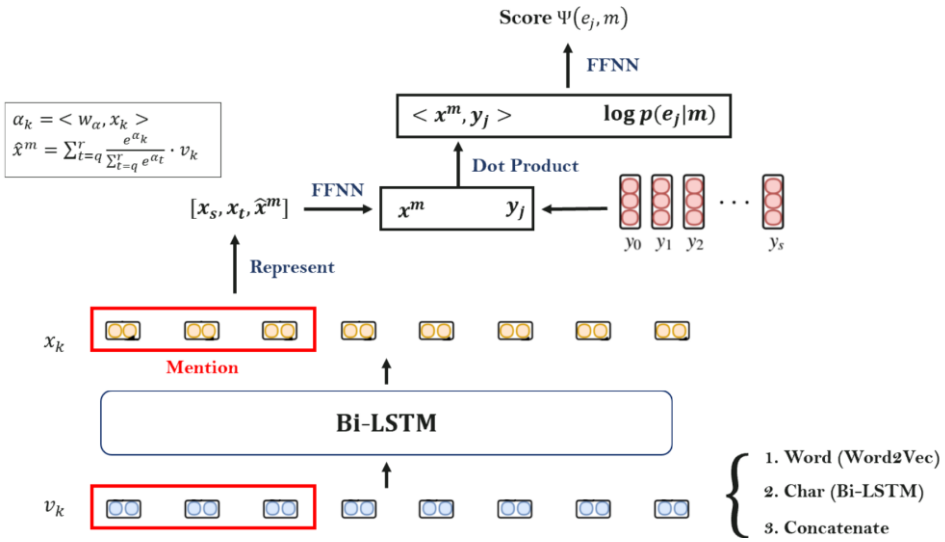

Figure 3 Model 2 Architecture

1. The input of the model is a sequence. For each word in this sentence, do the embedding in two levels. In words level, it uses *word2vec* model to embed the word in 300 dimensions. For the first and the last of this word, it trains a bidirectional long short-term memory (bi-LSTM) to embed it with 100 dimensions. Then concatenate the two-levels embedding as a input result.
   The known sequence model recurrent neural network (RNN) adds the hidden state of the previous block to the propagation in the basic neural network. On this basis, LSTM adds several parameters to the hidden layer, which can also be said to be "gates", separately users to adjust how much previous layer information is forgotten, how much input information is added and update. The details as follows:

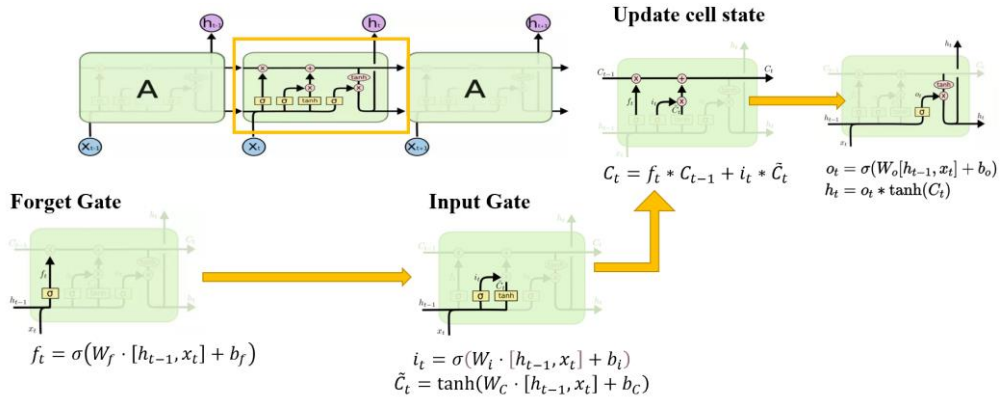

Figure 4 LSTM Structure Details

The input sequence is input to the two LSTM neural networks in forward and backward direction respectively for feature extraction. And then concatenate the two output vectors (the extracted feature vectors). The formed word vector is

expressed as the final feature of the word.

2. Then the word vectors should pass by another Bi-LSTM to find the word most likely to be mention in this sentence. The dimension of the output vector is also 500.
3. After mention detection, it uses a special form to represent the mention, including the first and the last word embedding of the mention, as well as the soft head $\hat{x}^m$ which can calculate by this formula:

$$\alpha_k = \ < w_\alpha, x_k > \ , \quad \hat{x}^m = \sum_{t=q}^{r} \frac{e^{\alpha_k}}{\sum_{t=q}^{r} e^{\alpha_t}} \cdot v_k$$

Where w_α is trainable parameter. After some operations such as normalization, we can get the soft head. Thus, we use $g_m =< x_s, x_t, \hat{x}^m >$ to represent mentions. Then just pass by a feed forward neural network to make sure the embedding of the representation to be same.

4. Candidate embedding use a pretrained model similar to word2vec, the embedding of it do the dot product with the mention representation, and then combine (concatenate) with prior probability from the dataset. Then this combined information is passed through a layer of feed-forward neural network to obtain the final score. Finally the loss function is calculated with this score to back-propagate the network.

Advantage & For Reference

1. Use Bi-LSTM model to detect the mention. In general, the span of the words in a mention is not large. In this case, we don't need to worry about the vanishing gradient problem that may happen to the lstm model, which means that it is very likely that he has almost forgotten the information in the initial hidden state if the span of the word is large. Furthermore, Bi-LSTM can also get the context information since we know that it has both forward and backward direction to update the hidden state.
2. Use innovative form to represent the mention. As mentioned in the basic flow, the form includes the first and the last word embedding of the mention, as well as the soft head. Over all, it contains not only global information (soft head's global attention mechanism), but also boundary information. We think the boundary information is the key and innovation point. For example, the first and the last word of mention “New York Times Newspaper” are “New” and “Newspaper”, but the “New York” “Times Newspaper” and “New York Times” can also be mentions. Thus, adding boundary information to the information on the global features of the mention may have a great impact on the accuracy of the results. Of course, it may not matter for mentions consisting of only one or two words.

Model 3 – Global Coherence BERT Based Model

This model is constructed based on Bert tokenizer and transformer block. What different from previous work are the induction of global coherence. Besides the origin text, the model also takes in the candidates of the mentions.

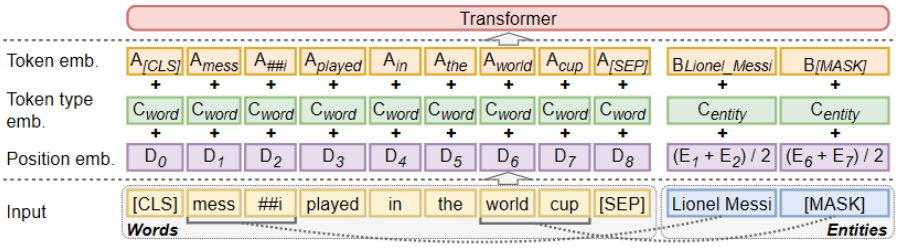

Figure 5 Model 3 Architecture

Take an example, when the input text is as the figure shows. Since there are two mentions in this sentence, the model will initialize the position for the possible entities with the value as "[MASK]". And in the first-time forward propagation, the model will make prediction on what these "[MASK]" really are. And then replace the "[MASK]" token with the prediction. And start second round, executing iteratively until some confidence criteria is met.

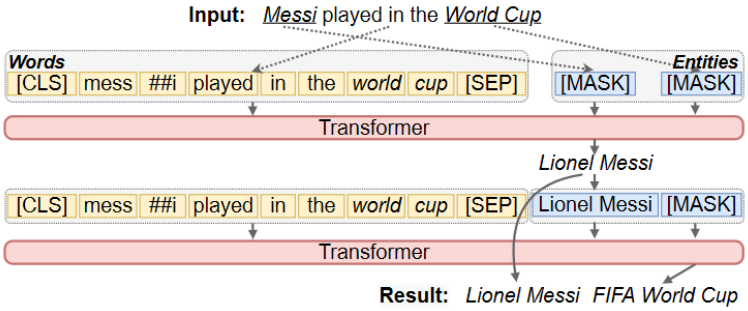

Figure 5 Model 3 Application Example

Experimental Results

Model	AIDA_A	AIDA_B	ACE2004	AQUAINT	MSNBC
USE (base)	63.4	64.3	70.9	74.5	75
USE (tunning)	71.3	71.3	80.6	65.9	70.1
EndtoEnd	88.8	82.7	83.5	79.7	82.5
LUKE	92.4	94.6	91.9	93.5	96.3

F1-score

Table 1 Result: F1-score of Four Models on Five Datasets

Here we display the F1-score of four different model on five different datasets. Generally. Fine-tuned USE model improves the performance significantly (compare with the origin one) on nearly all dataset except **AQUAINT.** And for the two deep learning model. They both reach the state-of-art performance. However, since the End-to-End model perform two tasks together, the model performance degrades.

Conclusion

We explore the task objectives, overall process and methodology of entity linking and entity disambiguation. In addition, we experimented the effect of three models on disambiguation, and we can find that the transformer model based on BERT will perform better than the other two models in terms of results. Although Bi-LSTM-based models may not be as good at processing contextual information, incorporating boundary information and special encoding representations can contribute to achieve good results. Finally, based on the pre-trained USE word embedding model, the vector information is directly compared to the similarity to get the result. As a result, the performance was of course not as good as the former. By fine-tuning the model by adding a layer of fully connected layers, the effect can also be greatly improved.

Future work

In this project, we focus on entity disambiguation. But for the entity linking task, another important step is mention detection. Although it has been mentioned in the end-to-end type (model2), we have not compared it with other models and experimented. Thus, this is a job that can still be explored. More importantly, this dataset has already helped us select candidate entities. But in most areas of practice and application, it is up to us to identify candidate entities. Therefore, this is also the key work that needs to be paid attention to in the future. In addition, entity linking is usually combined with related fields of knowledge graph, so to understand graph-related neural networks such as (CGN, GAT) can better combine the structural information of graph when linking entities to knowledge graph in the future.

Reference

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas *Hofmann. 2018. End-to-End Neural Entity Linking. In Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 519–529*, Brussels, Belgium. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. *End-to-end Neural Coreference Resolution.* In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto GarciaDuran, Jason Weston, and Oksana Yakhnenko. 2013. *Translating Embeddings for Modeling Multirelational Data*. In Advances in Neural Information Processing Systems 26, pages 2787–2795.

Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric ´ Gaussier, and Guillaume Bouchard. 2016. *Complex Embeddings for Simple Link Prediction*. In Proceedings of The 33rd International Conference on Machine Learning, volume 48, pages 2071–2080.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. *Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation*. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. *Learning Distributed Representations of Texts and Entities from Knowledge Base*. Transactions of the Association for Computational Linguistics, 5:397–411.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. *Universal Sentence Encoder*. arXiv preprint arXiv:1803.11175.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto.

2020. *LUKE: Deep contextualized entity representations with entityaware self-attention*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics.

W. Shen, J. Wang and J. Han, *Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions*, IEEE Transactions on Knowledge & Data Engineering 27(02) (2015), 443– 460. doi:10.1109/TKDE.2014.2327028.

J.L. Martínez-Rodríguez, A. Hogan and I. López-Arévalo, Information extraction meets the Semantic Web: *A survey, Semantic Web* **11**(2) (2020), 255–335. doi:10.3233/SW180333. https://content.iospress.com/articles/semantic-web/sw180333.

T. Al-Moslmi, M. Gallofré Ocaña, A.L. Opdahl and C. Veres, *Named Entity Extraction for Knowledge Graphs: A Literature Overview*, IEEE Access 8 (2020), 32862–32881. doi:10.1109/ACCESS.2020.2973928.