CS 512: Data Mining Principle
Spring 2019
Assignment 2 Final Report
Due: April 23, 2019
Name: Shuyue Lai (shuyuel2)

Entity Linking (EL)

Part I
Introduction

1 My Understanding

1.1 EL's Importance

Entity Linking is used to map the name entity mentions in sentences or articles to related entity entry in a knowledge base(KB). There are mainly two reasons why Entity Linking is and will play a crucial role in Data Science. First, unstructured and ambiguous online text is a major component of modern information. Thus, one of the crucial tasks is to identify those disambiguous context of mentions. By using knowledge base such as Wikipedia, name entity mentions can be better recognized with provided external background contexts. Second, in the era of information explosion, unstructured text grows rapidly every moment and the relation between name entity mentions also changes dynamically. Therefore, both name entity mentions' attributes and relationship between name entity mentions provided by knowledge base can contribute to name entity recognition.

1.2 EL's Downstream Applications

Entity Linking can be useful in searching system and recommendation system. Take "Michael Jordan" mentioned in assignment 2 as an example, although "Michel Jordan", the former basketball player, has a higher frequency compared to the professor at Berkeley, due to Entity Linking, search system may give the professor as the right result which improves users' experience. Another possible downstream application is using Entity Linking to detect fake news on social medias. Since nowadays, there are thousands of bots generating fake news and fake information in Facebook, Twitter and other social media. Entity Linking can be used to automatically analysing posts and users' comments which is the foundation of fake news detection.

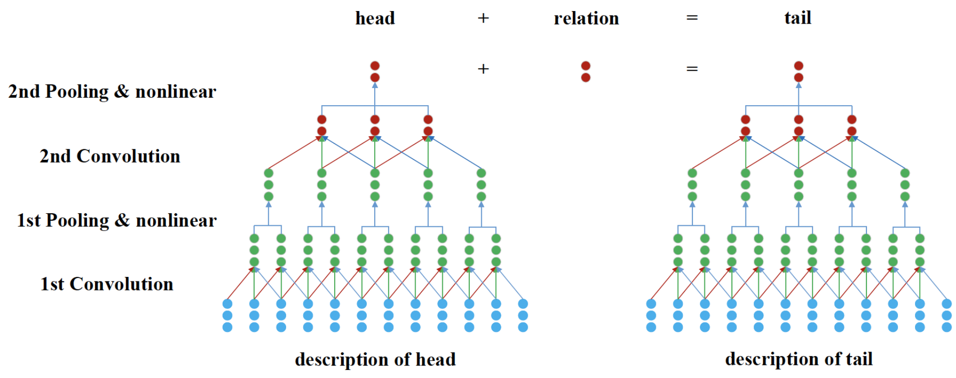
2 Existing Works

I read two papers about ”knowledge representation” according to this Piazza post https://piazza.com/class/jr0v8ds1yee5wy?cid=121 for assignment 2 based on the given data under ’entity’ folder.

2.1 Representation Learning of Knowledge Graphs with Entity Descriptions [1]

In this paper, the team proposes a new model called DKRL model for representation learning of knowledge graphs. The major contribution of this paper compared to its pioneers who concentrate on learning representations with relations between entities is that taking advantage of entity descriptions. The paper introduces two encoders to encode semantics of entity descriptions, one is continuous bag-of-words and the other is deep convolution neural models.

The paper defines two representations of name entity mention. One is structure-based representations which can directly represent entity mentions. The other one is description-based representations which are built from entity descriptions. By combining those two kinds of representations, the team applied two methods. One is Continuous Bag-of-words Encoder(CBOW). From each short description, the team generates a set of keywords which are capable of capturing the main ideas of entities but hard to detect through structural information. Then sum up the embedding of keywords to get the entity embedding ignoring word orders. The other method is using a convolutional neural network encoder which is called DKRL.



However, there is limitation of this work. This paper only considers structural information and entity descriptions for representation learning, but there are various of information like categories can be useful and should be considered.

In our assignment, structure-based Representations are attributes and relationship between mentions and candidates in train/test datasets, while Description-based Representations are attributes of contexts and each candidate's wiki's description.

2.2 Knowledge Representation Learning with Entities, Attributes and Relations [2]

Existing KR models regard all relations equally and usually suffer, while in this paper. The team comes up with a new KR model with entities, attributes and relations (KR-EAR). The main improvement is treating attributes and relations separately since attributes and relations exhibit distinct characteristics. For relations, both head and tail entities are usually of a large entity set. Each entity only builds a specific relation with a limited number of entities. However, for attributes, they usually from a small set of entries. Thus, it is reasonable to treat attributes and relations separately.

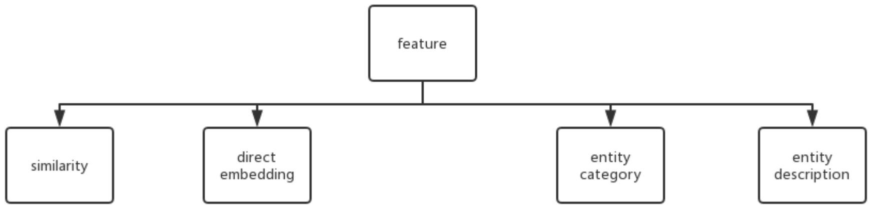
The limitation of this paper is that entities, relations and attributes are handled independently, while their correlations should be considered.

In our assignment, there is also a similar topic. When handling entity's description and category information, their independence and correlations should be under consideration.

Part II
Methodology

3 New Model

My methodology is an extension based on step 2, step 3 and papers mentioned above. First, using attributes which are similarity features from step 2 and semantic features after doing embedding from step 3. Then, I combine information from wiki entities, the given json files, as knowledge base. Then, do continuous bag-of-words as mentioned in paper 1 of entity description and add additional features from entity category. In the end, use all features mentioned above and random forest tree to train the model, 'NewModel'.



3.1 Similarity Module

This module is based on step 2, using a prior probability and text similarities between the surface name of the mention and the title of a candidate entity.

```
# feature1: prob
x.append(candidate.prob)

# feature2: similarity
x.append(textdistance.hamming.normalized_similarity(surface,
                                                  candidate.name))
```

```
# feature3: similarity
x.append(textdistance.hamming.normalized_distance(surface,
                                candidate.name))
```

3.2 Direct Embedding Module

This module is based on step 3, using entity embeddings. For each candidate entity, there are three attributes. (1) the entity embedding of the candidate; (2) the sum of the context word embeddings of the mention; (3) the cosine similarity between the two embeddings.

```
# surface embedding
try:
    if " " in surface:
        temp_surface = surface.replace(" ", "_")
        surface_embedding = self.ent2embed[temp_surface]
    else:
        surface_embedding = self.word2embed[surface]
except:
    continue

# sum of contexts embedding
sum_contexts = np.zeros(300)
for word in mention.contexts:
    try:
        sum_contexts += np.array(self.ent2embed[word])
    except:
        try:
            sum_contexts += np.array(self.word2embed[word])
        except:
            continue

# feature6: the cosine similarity between the two embeddings
x.append(scipy.spatial.distance.cosine(surface_embedding,
                                candidate_embedding))
```

3.3 Entity Category Module

This module is an extension of paper1's methodology. Using entity category's information as extra information from the knowledge base to improve performance. After experiment, this attribute does not contribute a lot of performance. One possible reason is that the categories can be very sparse.

The other possible reason is that the name of category is not trained in the same embedding field.

```
sum_category = np.zeros(300)
for word in entity[candidate.id]["categories"]:
    try:
        sum_category += np.array(self.ent2embed[word])
    except:
        try:
            sum_category += np.array(self.word2embed[word])
        except:
            continue
sum_category_list = list(sum_category)
for i in sum_category_list:
    x.append(i)
```

3.4 Entity Description Module

This module is based on paper1's methodology. Though entity's description can be an important component of attributes. The words in description can also be sparse an the cost to calculate every word's embedding is high. Thus, I use continuous bag-of-words to choose top ranking keywords to represent entity's description. In the end, sum up their embedding value to ignore the order of these keywords.

```
sum_desc = np.zeros(300)
vectorizer = TfidfVectorizer()
vectorizer.fit_transform([entity[candidate.id]["sections"][0]
                                        [1]])
count = min(20, len(vectorizer.get_feature_names()))
for word in vectorizer.get_feature_names():
    count -= 1
    if count == 0:
        break
    try:
        sum_desc += np.array(self.ent2embed[word])
    except:
        try:
            sum_desc += np.array(self.word2embed[word])
        except:
            continue
sum_desc_list = list(sum_desc)
for i in sum_desc_list:
    x.append(i)
```

```
# feature9
x.append(scipy.spatial.distance.cosine(sum_contexts, sum_desc
                              ))
```

Part III
Results

4 F1 Score

4.1 Random Model

aidaA	aidaB	msnbc	ace	aquaint
0.358	0.342	0.428	0.257	0.330

4.2 Prior Model

aidaA	aidaB	msnbc	ace	aquaint
0.738	0.719	0.898	0.873	0.844

4.3 SupModel

aidaA	aidaB	msnbc	ace	aquaint
0.864	0.812	0.907	0.845	0.806

4.4 Embedding Model

aidaA	aidaB	msnbc	ace	aquaint
0.860	0.806	0.907	0.865	0.847

4.5 New Model

aidaA	aidaB	msnbc	ace	aquaint
0.861	0.808	0.904	0.865	0.846

5 Error analysis

	entity	ground truth	prediction
1	Michael Collins	Michael Collins (film)	Michael Collins (Irish leader)
2	Kurdish	Kurdistan	Kurdish languages
3	New Hampshire	New Hampshire, Ohio	New Hampshire

Typically there are two kinds of errors. In the above table, both example 2 and 3 are caused by non-existence of ground truth in candidates list. In this case, there is no possibility to predict the right answer. Another reason caused the error is similar to example 1. Since prior probability plays an important role in my model, once surface and candidates are similar to each other, it's hard to predict them right.

According to my result and paper2 mentioned above, there are two possible methods to improve. One is split attributes to different models with different weight just similar to the method of dealing with attributes and relations. The other possible solution is to use Deep Learning to train the model, rely more on semantic attributes rather than structural attributes.

References

[1] J. J. H. L. M. S. Ruobing Xie, Zhiyuan Liu, “Representation learning of knowledge graphs with entity descriptions,” *AAAI’16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[2] M. S. Yankai Lin, Zhiyuan Liu, “Knowledge representation learning with entities, attributes and relations,” *IJCAI’16 Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.