Word Embeddings on Chemical-Protein Dataset

Training and testing NER model with

SEOKHWAN SONG

Iowa State University

Through text mining chemical-protein interactions, it will be able to detect relations between chemical compounds and genes [1]. In this paper, we train and test NER(Named-Entity Recognition) model on a chemical-protein dataset with Spacy. The methodology is introduced and through some experiments, the results such as precision and error are discussed for better performance.

CCS CONCEPTS • Word Embeddings • Named-Entity Recognition • Text mining

Additional Keywords and Phrases: Machine Learing, Word2vec, FastText

ACM Reference Format:

SeokHwan Song. 2018. Word Embeddings on Chemical-Protein Dataset: In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 10 pages. NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

Word embedding gives very significant helps in NLP field to make word vectors with their documents in a huge corpus, and we use its information of words to measure the word similarity [2]. In this paper, we use the dataset about chemical-protein to detect the relations between chemicals and genes, and before train and test the dataset, we train word embedding modes to use for retraining the NER model that we conducted with Assignment 1 and compare the results. Therefore, we can figure out the differences between three embedding methods and NER model without embedding methods.

The natural language procession library called, Spacy is used for the NER model in python, and three different word embedding methods: FastText - Skipgram, FastText – CBOW, and Word2vec – Skipgram are used. Moreover, we are using a data, PubMed to get one more different embedding to compare.

2 METHODOLOGY

To train and test the dataset, data preprocess, word embedding modeling, and implementing word embedding models are applied.

2.1 Data Preprocess

As long as, Spacy model is used, for the NER model, we firstly need to get the word embedding models to implement dataset into the blank Spacy model to use a certain embedding model. To train word embedding

models, we use library Gensim which has Word2vec and FastText methods, and for the package datasets should be preprocessed in a certain format. The train dataset of chemical protein dataset is used.

There are some steps to do preprocess. Firstly, when the datasets are imported with pandas, the chemical compound, 'NA' is changed into NaN, so all 'NA' has to be changed to str(NA). To train the text, each entity has to be connected with a sentence which the entity is located in. Therefore, the merged abstracts and titles have to be tokenized and connected to the entity mentions. NLTK tokenizer is used to conduct the task. Lastly, the tokenized words should be in a list with brackets.

2.2 Model

The library Gensim is used for unsupervised word embedding methods. Three different methods are applied: FastText - Skipgram, FastText – CBOW, and Word2vec – Skipgram. Word2vec coverts words into vectors based on distribution hypothesis. When the size of window (size = 5) is selected, it will detect neighboring words and update their vectors. FastText is similar with Word2vec but it checks words as 'Bag of Characters' and uses characters of 'n-gram' [5]. We use 100 size of dimension for word embedding methods.

2.3 Implementing Word Embedding Model

We use the terminal to convert downloaded text file into vector file (.vec) to apply the word embedding models into Spacy NER model. After the files have been converted, they are linked to Spacy and we can call linked model when we make a pipeline for NER [3].

3 EXPERIMENT

We retrain the same model we used for Assignment 1 with the different embedding models above. We used the best model which we could make for Assignment 1. We used 30 of iterations and batches (1.0, 4.0, 1.001) are applied. And we compared the best result from the Assignment 1 and three different word embedding methods: FastText - Skipgram, FastText – CBOW, and Word2vec – Skipgram. Every method is conducted twice with two different size of dimensions (100, 200). Finally, with the best method, we train 'pubmed' dataset for word embedding model and retrain the NER model with the embedding result, and compare the results together.

4 ERROR ANALYSIS

4.1 Scores

The values of precision, recall, and F scores are compared, and F scores are calculated by labels as well.

4.2 Tables

Firstly, three different methods and the best result from Assignment 1 are compared. Using FastText – Skipgram with 100 size of dimension gets the best score with 78.4563% testing model precision. The precision is higher than the precision of the best model from the first assignment (78.2201). Moreover, FastText -CBOW gets the higher score (78.5337) than the model from the previous assignment. However, this model has lower training model precision and F scores than the Skipgram model. Therefore, we can say that FastText – Skipgram has the best score.

Table 1: Scores of Train and Test with blank Spacy model (best result) and FastText -Skipgram (dim =100)

Train (Best result Assignment 1)	Test (Best result Assignment 1)	Train (FastText – SG (dim=100))	Test (FastText – SG (dim=100))
Model Precision	Model Precision	Model Precision	Model Precision
96.0053	78.2201	96.6467	78.4563
Model Recall	Model Recall	Model Recall	Model Recall
96.8872	74.5934	97.3965	74.0771
Model F Score	Model F Score	Model F Score	Model F Score
96.4442	76.3637	97.0201	76.2039
Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score
95.8032	74.1087	96.5264	73.9758
Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score
97.0105	78.3260	97.4596	78.2293

Table 2: Scores of Train and Test with FastText – CBOW (dim =200) and Word2vec -Skipgram (dim = 200)

Train (FastText – CB (dim=200))	Test (FastText – CB (dim=200))	Train (FastText – CB (dim=100))	Test (FastText – CB (dim=100))
Model Precision	Model Precision	Model Precision	Model Precision
96.3569	78.5337	95.8392	77.4238
Model Recall	Model Recall	Model Recall	Model Recall
96.6825	72.5789	96.6968	72.5598
Model F Score	Model F Score	Model F Score	Model F Score
96.5194	75.4390	96.2661	74.9129
Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score
95.9405	73.5576	95.7204	72.4290
Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score
97.0327	77.1243	96.7469	77.0754

Secondly, we also compared the model with the model with PubMed dataset. We used the same model of the best model. The dataset is a larger corpus, so we expected to receive the better result, but the precision and F scores slightly decrease. The scores are better than the rest of scores of models with other methods, but the result is still not better than the result from the previous assignment.

Table 3: Scores of Train and Test with Word2vec -Skipgram (dim = 200) and PubMed dataset

Train (Word2vec (dim=200))	Test (Word2vec (dim=200))	Train (PubMed dataset)	Test (PubMed dataset)
Model Precision	Model Precision	Model Precision	Model Precision
96.6399	77.7529	96.3322	78.1365
Model Recall	Model Recall	Model Recall	Model Recall
97.4678	73.9842	96.8824	74.1795
Model F Score	Model F Score	Model F Score	Model F Score
97.0521	75.8217	96.6065	76.1066
Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score	Label 'GENE' F Score
96.4446	73.1720	96.2325	73.8447
Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score	Label 'CHEMICAL' F Score
97.5906	78.1800	96.9398	78.1741

5 DISCUSSION

With Spacy, training NER models on chemical-protein dataset with word embedding methods perform well. Through some experiments with different size of dimensions and methods, the performances of the model change a little bit but not dramatically. However, the performances improve comparing to NER models without word embedding methods. We trained with dataset without stop words before the final trainings and models without stop words performed slightly better. With a different way to preprocess the dataset or a bigger amount of training set will help this model to improve the accuracy. However, it does not improve its performance dramatically with the larger number of datasets for word embeddings. Moreover, setting a different value of dropout rates is not applied in this paper, but it would help to improve the performance as well.

ACKNOWLEDGMENTS

I had some discussions with Yun Qi Bang.

REFERENCES

[1] The BioCreAtIvE (Critical Assessment of Information Extraction systems in Biology). Retrieved from https://biocreative.bioinformatics.udel.edu/tasks/biocreative-vi/track-5/

[2] Yang Lie, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical Word Embeddings. Retrieved from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.699.6286&rep=rep1&type=pdf

[3] RASSA: Loading fastText vectors with Spacy https://medium.com/@expanded_blue_elk_810/rasa-loading-fasttext-vectors-with-spacy-1aad2a2431b

[4] Python for NLP: Working with Facebook FastText Library. Retrieved from https://stackabuse.com/python-for-nlp-working-with-facebook-fasttext-library/#disqus_thread

[5] 1.word2vec,fastText. Retrieved from https://velog.io/@happykimnh/%EC%A0%95%EB%A7%90-%EA%B0%84%EB%8B%A8%ED%95%9C-FastText-%EC%A0%95%EB%A6%AC