Word representation in biomedical domain

Liu Beini[1], Chen Yiting[1], Yan Zikai[1], Shao Yinghui[1], He Xinling[1], Liu Haoyang[1]

[1] Imperial College London

[2] Data Science Summer School Team 5

Abstract: Nowadays, in spite of the overwhelming amount of scientific articles in the field of natural language processing (NLP), there still exits an urgent call for developing more effective language models that target at tackling a wide range of biomedical tasks applying NLP methods. In this specific field, high-quality and meaningful biomedical words enable doctors to obtain the gist of information and knowledge in a short time to make quicker clinical decisions. In this report, we have been through four major parts training distinct tackling models confronting related NLP problems in biomedical domain. The first part is to parse relevant biomedical articles from the given 247236 JSON files and the second part is to do word tokenization with the standard split method, NLTK and BPE applied to segment the text. The next part is about word representations, where the tokenization results are further processed. Three commonly used word embedding models, N-gram, Skip-gram and MLM(masked language model) are trained in this part in sequence to capture useful semantic properties and linguistic relationships between words, obtaining word representations. Finally, we are to do some exploration based on the word representation results in the last part, where t-SNE and K-means are introduced to reduce the vector dimension and to conduct clustering correspondingly. After all, Medterms medical dictionary and Stopwords list are referred to in an attempt to select biomedical entities and to find frequent co-occurrence words with the word 'coronavirus' as well as biomedical entities with the closest semantic similarity.

Keywords: NLP; Biomedical; NLTK; BPE; N-gram; Skip-gram; MLM; T-SNE

1 Introduction

The application of Natural Language Processing (NLP), a significant explorable and valuable subfield of linguistic and artificial intelligence dealing with gigantic natural language data, is currently playing an increasingly important role in various areas, biomedical researches included. Typically, numerous clinical medical information is stored in the system in the form of unstructured (or semi-structured) text, while NLP can be the key technology used to extract useful information from complex medical text by transforming the data into structured models. Thus, it is possible for us to realize word representation in biomedical domain using NLP.

According to the given instructions, to develop appliable algorithms, it is suggested that we follow the steps of parsing the data, tokenization, building word representations, exploring word representations and finally mining biomedical knowledge. From the papers we have looked through concerning biomedical word representations, after importing the data, semantic resources could be utilized in the initial tokenization and building process. As is demonstrated in Distributional Semantics Resources for Biomedical Text Processing[1], S Moen and others shows a way to build a dataset for BioNLP. To be specific, extracted from biomedical articles, text is processed, preserving only the title, abstract and main body; Unicode characters are transformed to ASCII; the whole text is partitioned into single sentences and tokenized. Next, data from 1-gram to 5-gram are collected. Then the word vectors are built. Moreover, Word2vec (Mikolov et al., 2013) could be used to calculate vector representations, and to induce word clusters. Finally, the outcome is assessed using extrinsic evaluation means. Aside from applying

semantic resources, distributed phrase representation is also considered for text processing. PMCVec, a model proposed by Zelalem Gero et al.[2] is able to generate effective phrases from corpus, and to build a distributed representation that contains both single words and multi-word phrases by treating both as a specific unit using unsupervised method. By establishing the model, not only could useful phrases get generated from corpus, but also a new criterion can be introduced to rank the generated phrases to avoid incorporating all the phrases and achieve a better embedding for both single words and multi-word phrases. (However, they were unable to train different models in large dimensions and window sizes and only compare them in smaller dimensions.)

As for the stage further building and exploring word representations, clinical abbreviation disambiguation could be introduced to better the algorithm. As the challenge handling abbreviation widely used in clinical notes is never neglectable, a word embedding method is tailored to medical domain (Yonghui Wu et al., 2015) based on datasets from hospital and Health Service. According to SBE feature (Li et al.,2014), its variation LR_SBE and MAX_SBE are generated to better describe word semantics.[3] Nevertheless, the dataset only focuses on abbreviations that have enough samples to train; the size of the window is limited, so some semantically important words might be missed.

Another topic universally referred to when refining the models is word-bedding. Di Zhao et al suggests a local linear embedding framework based on manifold learning to solve the problem that distributed word representation ignores the influence of the word embedding geometric structure obtained through calculation on the word semantic information.[4] The method theoretically founded in metric recovery paradigm is proved to improve the accuracy of classifications and similarities. Word-bedding is worth working on also in that it encourages the future studies to undertake the model in two meaningful directions, both reducing the computational requirements and exploring the impact on model performance of using vocabulary built on specific pretrained corpus.

Additionally, we have also noticed a pre-trained language representation model dealing with relevant NLP problems. When applying progress of NLP technology directly to biomedical text mining, transferring vocabulary distribution from general domain corpus to biomedical corpus is always unavoidable. Jinhyuk Lee et al thus proposes the BioBERT model which is pre-trained on large-scale biomedical corpora.[5] The new model performs better than previous BERT model in biomedical named entity recognition, biomedical relation extraction and biomedical question answering. What is more, scholars also tend to focus on designing models using more efficient architectures to reduce computing costs. Thus Giacomo Miolo et al. proposes a pretrained domain-specific model called ELECTRAMed, which takes the advantage of computing of ELECTRA model and is specifically suitable in biomedical field.[6] The results show that ELECTRAMed obtains valuable results for common NLP task in the biomedical field and is efficient for the use of ELECTRA's architecture in pre-training stage.

In a nutshell, to develop feasible algorithms for word representations, semantic resources and distributed phrase representations are ideal references for the tokenization and establishing stage while clinical abbreviation disambiguation and word-bedding are expected to be introduced when further building and exploring the model, and above all, a pre-trained language representation model could be taken into consideration in an attempt to solve some commonly related NLP problems.

2 Methodology

2.1 Parse the data

Before we start to design the algorithm, we first need to scan all the JSON files and extract texts from the relevant fields including title, abstracts and body text. The variable all_json is a list to store the path of all JSON files from the given folders. Then in the class $FileReader$ we extract the parts we want from the file and put them in the list. For example, if we want to get the first file in the given file list, we can use $FileReader(all\_json[0])$ then we can get all the three parts desired. And

$FileReader(all\_json[0]).abstract$ can return the abstract of the first file in the list. Applying the methods given above, we can get the sting type of the content in JSON files.

2.2 Tokenization

Tokenization plays an important part in natural language processing, which is traversing the extracted text and segment the text into words(tokens). But splitting a text into smaller chunks is a task that is harder than it looks due to different rule-based tokenization methods. In this report, we applied three different methods to realize this algorithm —— the standard **split** method in Python, **NLTK** and **Byte-Pair Encoding(BPE**).

2.2.1 Standard split() in Python

The simplest method to segment the text into single word is to use the standard $split()$ method by Python. But we not only want to split by and delete the space, but also to do segmentations by punctuation. So if we just simply use $split()$, we need to write many lines, each consists of one separator. So we tend to the extension of this $split()$ method: $re.split()$ method, which can split and deleted more than one separators at the same time. Firstly, $row[i]$ in "$rows= [FileReader(all\_json[i])$ for i in $range(len(all\_json))$]" is the article we get consists of three parts mentioned in part one. Then we use $row[i].abstract$ and $row[i].body\_text$ to get the contents in each article, which is String type. Then, $re.split('[\s;,.:()]\s*', abs)$ is used to tokenize the abstract or body text by space and punctuation. This can generate a list consists of the word in the text.

2.2.2 NLTK

However, space and punctuation is not that satisfying. For example, some times we don't want to delete the " ' " between some special words like "o'clock", so we seek for other methods. There are many methods by NLTK. In this report we choose $word\_tokenize$ in $nltk$, which split every words as well as punctuation in the passages, to conduct tokenization. We first use $nltk.word\_tokenize$ to segment the text based on the established rules, and this remains all the punctuation. Then , we use a for loop to delete all separators besides letters and numbers. The method $isalnum()$ is contained in the loop. At last we transform all the uppercase letters into lowercase.

2.2.3 BPE

BPE relies on a pre-tokenizer that splits the training data into words. We put all the text in to a list called "files". And we use the NLTK method above to get the segmented words. After pre-tokenization, a set of unique words has been created and the frequency of each word occurred in the training data has been counted. We use a dictionary called "vocab" to realize this(key is the word and value is the frequency it occurred). Its form looks like this: $vocab$ = { 'l o w ' : 5, 'l o w e r ' : 2, 'n e w e s t ':6, 'w i d e s t ':3 }. Next, BPE creates a base vocabulary consisting of all letters that occur in the set of unique words and learns merge rules to form a new symbol that is combined by two symbols of the base vocabulary. It does so until the vocabulary has attained the desired vocabulary size. After this we put the keys in the dictionary in to a new list called "keys". But each element looks like this: "Tay lor", "according ly". So we use $re.split()$ again, to segment each element as "Tay", "lor" and "according", "ly". And store them into "output" list, so in this way the output may not be every single word but some familiar symbol pair which is part of the word like "ing" or "ly".

2.2.4 New BPE

In 2.2.3, a BPE model is built to realize tokenization by creating a vocabulary base presenting every letter in one unique word and forming into new symbols, but this method is not specially designed for

biomedical domain. To better the algorithms, a new BPE model is introduced in this track particularly. Words are additionally filtered so that only terms of biomedical domain would be put out into the list.

To build the new BPE, it is necessary to decide what the biomedical words are. *MedicineNet* offers an ideal list of all biomedical terms which could be used to distinguish oriented words. Therefore, all that need to be done is to add a conditional statement telling apart whether the processed words belong to biomedical domain based on the list of terms collected from *MedicineNet*.

2.3 Word representation

Distributed word representation, usually obtained through calculation from large corpora, has been widely used in text because of its effectiveness in representing word semantic information. This part mainly contains two models: **N-gram** and **Skip-gram**. The former one use 'n' number of previous background words to predict the next word, while the latter one use the central word to predict the background words. Besides, a **Masked Language Model(MLM)** is used to contextualize word representations considering different conditions.

2.3.1 N-gram

N-gram Language Modeling is to predict a target word by using n words from previous context. If expressed in conditional probability, it can be written as $P(Wi|W_{i-1}, W_{i-2}, \ldots, W_{i-n+1})$. We first establish a training set and traverse the whole *test_sentence*, separating into three groups of words, the first two as input and the last as the result of prediction. Then encode each word and use numbers to represent the word. Only in this way can we pass in *nn.embedding* to get the word vector.

Now we can build the N-gram model. We pass in three parameters: the total number of words, the number of words on which the predicted word depends, and the dimension. We use the tokenization results from 100 papers, set '$n = 2$' and iterate 100 times, input the word and label, transport forward and backward to train the model.

In the last we can get the predicted word as output as well as the word representation.

2.3.2 Skip-gram

In skip-gram, we use a central word to predict its context. If expressed in conditional probability, its $P(W_{c-m}, \ldots, W_{c-1}, W_{x+1}, \ldots, W_{c+m}|W_c)$. First of all, we set the training parameters as: $\{learning\_rate = 0.05, batch\_size = 128, total\_steps = 1000000, display\_step = 10000, evaluation\_step = 100000\}$. Then we set the evaluation parameters using five words: "disease", "virus", "symptoms", "coronavirus", "health" and "medical". Next, we set the Word2Vec Parameters to be: "$count = [('UNK', -1)]$" to replace rare words with UNK token. $min\_occurrence = 1000$ to retrieve the most 1000 words which have the largest occurence frequency, and remove samples with less than *min_occurrence*, in this report we set it as 10. "*word2id*" is a dictionary to assign an id to each word. Finally we can generate training batch for the skip-gram model. "$skip\_window = 3$" indicates the number of words to consider left and right, so the window size "span" is "$2 \times skip\_window + 1$". "embedding" is the embedding variable, each row represents a word embedding vector. The number of negative sampling to sample is defined as 64.

During the model training, the cosine similarity between input data embedding and every embedding vector is used for evaluation. Then the algorithm will output the nearest background word of the central words.

2.3.3 MLM

A defection of our model is that the above two methods do not include conditions to contextualize word representations, so the results may seem strange without contexts. For example, the word representation

of "stick" is fixed regardless of its context under N-gram language modeling and word2vec. However, the representation of "stick" is shown dynamatically based on context using MLM. To refine the model, Masked Language Model, which was proposed by BERT, could be introduced to contextualize the word representations in the pre-training stage.

MLM model is easy to be set up based on related references, and repeating similar training process realized in N-gram language modeling and word2vec could help get the model running. To present the outcome, corresponding contexts are to be listed beside the word representations, telling the conditions defining the results.

2.4 Explore the word representation

2.4.1 The word representations by t-SNE

In this part we need to visualize high-dimensional vectors, and we choose t-SNE algorithm to realize this. T-SNE is a dimension reduction algorithm which transform high-dimensional vectors into 2-dimension. The words we choose is the first 1000 high frequency word acquired in Skip-gram model, exclusing the stopwords. After reducing the dimension we use K-means to cluster the words. We classify the words into 6 parts, which can be visualized. Then we can plot the figure.

2.4.2 Visualize the Word Representations of Biomedical Entities by t-SNE

In 2.4.1 we focus on all the words that frequently occurred, but in this part we only focus on biomedical entities. To get all the professional biomedical words, we read form $MedTerms$ medical dictionary, and we tokenized the text we get. To make our word list more accurately, we read the file "stopwords.txt" to delete those *stopwprds*. Then we search those biomedical entities in $word2vec$, also visualize and cluster the words that has occurred in $word2vec$.

2.4.3 Co-occurrence

Then, to present the word that occurs with the word Covid-19, first we count the words appeared in $MedTerms$ medical dictionary and sort the frequency. Then the frequency distribution histogram is drawn to visualize the data. Next, we look for the dictionary again, count the words (3 before and 3 afterwards) around the "coronavirus", and delete the stopwords in them.

We count the co-occurrence frequency of the words in the texts that read from JSON file, and put them in the 100×100 matrix, which is then standardized for calculating the conditional probability. Last, we print out the first 20 high-frequency biomedical words and their conditional probabilities after deleting stopwords.

2.4.4 Semantic similarity

This part is to find the words that have semantic similarity with "coronavirus", like "Covid-19". Simply calculate the cosine similarity with "coronavirus" can generate the output. The method used here is $cosine\_similarity()$.

3 Result

3.1 Prase the data

Considering the computation ability of CPU and the max RAM, we load 100 papers' abstract to conduct the latter n-gram and load abstract of all papers(the total number of papers are 247236 while about 172068 papers whose abstract is not empty). The loading process of all papers is shown in Fig.1.

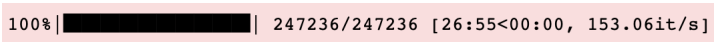

Figure 1: load abstract of all papers

3.2 Tokenization

The advantages and disadvantages of four tokenization methods are shown in Tab.1. Comparing among these method, we finnaly choose the result of NLTK and pass them to the next part.

Table 1: Compare among four tokenization methods

Compare among four tokenization methods		
Method	**Advantage**	**Disadvantage**
Split()	easy to understand and code	difficult to handle punctuations
NLTK	can use package functions	still remain some punctuations
BPE	reduce vocabulary size	rely on corpus and inaccuracy
New BPE	consider multiple meanings	based on BPE's performance

3.3 Word representation

Here we compare the training results from N-gram model and Skip-gram model.

The first is N-gram. Atfer 100 epochs' training, the value of loss function of N-gram remain stable at around 2.0(shown in Fig.2), which basically satisfy the standard of an effective training result. Especially, we randomly choose {'the', 'science'} at position [733] to test the result, and the predicted word is shown to be 'curriculum', which is same to the real word behind these two words(shown in Fig.3).

```
epoch: 20, Loss: 1.928090
epoch: 40, Loss: 2.175143
epoch: 60, Loss: 2.318204
epoch: 80, Loss: 2.182490
epoch: 100, Loss: 2.062860
```

Figure 2: the loss function value of N-gram's training

```
input: ('the', 'science')
label: curriculum

real word is curriculum, predicted word is curriculum
```

Figure 3: test the training result of N-gram model

Here comes the Skip-gram. After the total learning steps of 1000000, the value of loss function of Skip-gram remain stable at around 4.4(shown in Fig.4), a little larger than that of N-gram. And the evaluation result of five evaluation parameters are shown below. Take 'coronavirus' for instance, its nearest eight neighbors are 'epidemic', 'as', 'using', 'disease', 'with', 'evaluated', 'background', 'from'.

```
Step 1000000, Average Loss= 4.4226
Evaluation...
"disease" nearest neighbors: has, risk, the, background, is, infection, review, have,
"virus" nearest neighbors: for, of, by, on, health, samples, with, that,
"symptoms" nearest neighbors: but, that, parameters, than, infection, these, positive, models,
"coronavirus" nearest neighbors: epidemic, as, using, disease, with, evaluated, background, from,
"health" nearest neighbors: for, on, of, UNK, using, from, current, and,
"medical" nearest neighbors: 1, based, p, human, by, all, first, areas,
```

Figure 4: The training result of Skip-gram model including loss function value and validation results

Comparing between N-gram and Skip-gram, although the loss function of N-gram is smaller than that of Skip-gram, but it only uses 100 papers as the material due to the limit of computation ability, while Skip-gram uses all the 247236 papers. Besides, the dimention of embeddings of N-gram is 10 while that of Skip-gram is 200, 19 times more than N-gram. Consequently, we choose the result from Skip-gram and pass it to the nex part.

3.4 Visualization

After deleting the stopwords, 717 words remain in the vocabulary. the visualization result of vocabulary after t-SNE and K-means is shown in Fig.5. Next, Fig.6 shows visualization result of biomedical vocabulary after t-SNE and K-means.

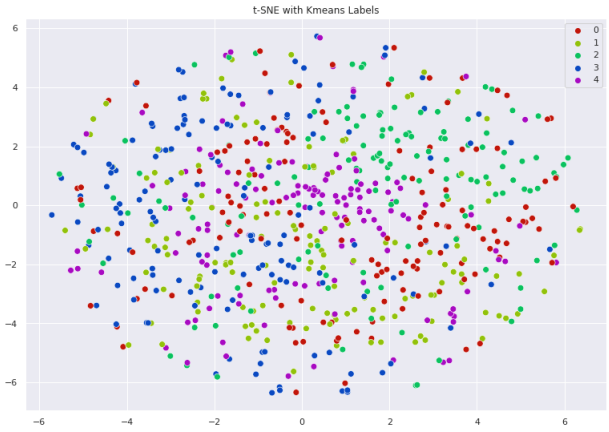

Figure 5: vocabulary t-SNE with K-means labels

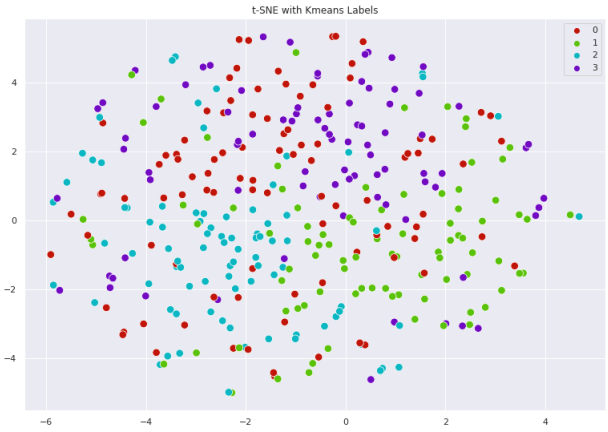

Figure 6: biomedical vocabulary t-SNE with K-means labels

3.5 Co-occurrence and semantic similar words

The result shows that the most frequent co-occurrence biomedical entities with 'coronavirus' is 'disease', which co-occur with it for almost 17982 times, with the conditional probability of 9.1740%. The frequencies of top 10 words are shown in a histogram in Fig.7, and the conditional probabilities of top 15 words are shown in Fig.8.

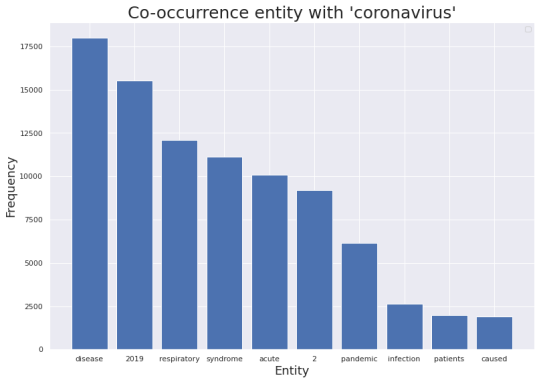

Figure 7: co-occurrence biomedical entities with 'coronavirus'

Rank:	Word:	Probability:
1:	disease	9.1740%
2:	2019	7.9195%
3:	respiratory	6.1711%
4:	syndrome	5.6788%
5:	acute	5.1533%
6:	2	4.6967%
7:	pandemic	3.1452%
8:	infection	1.3458%
9:	patients	1.0183%
10:	caused	0.9770%
11:	outbreak	0.9489%
12:	sars-cov-2	0.8270%
13:	severe	0.8168%
14:	human	0.7724%
15:	east	0.7178%

Figure 8: probability of co-occurrence word with 'coronavirus'

Fig.9 shows the top 10 words which have the largest cosine similarities with word 'coronavirus'. It illustrates that word 'virus' has the largest cosine similarity(about 0.45) with it. And the result is consistent with commom cognition.

Rank:	Word:	Cosine Similarity:
1:	coronavirus	1.000000
2:	virus	0.449346
3:	domain	0.413443
4:	protein	0.399445
5:	disease	0.389628
6:	activity	0.385477
7:	ci	0.383529
8:	infection	0.371769
9:	quality	0.370868
10:	heart	0.365938

Figure 9: semantic similar words of 'coronavirus'

4 Discussion

From the above methodology we use and the results we get, we can conclude that the there are some parts which we conduct quite well. Here are some points that we consider to perform well in this project.

- In part 1, we load all the passages given to train the model, so our training result is relatively accurate.
- In part 2, we conduct all the tokenization methods that are required and they all perform well. Besides, we successfully built a new BPE model based on the traditional one, which takes into consideration of biomedial entities.
- In part 3, for both the N-gram and Skip gram, the loss function is controlled in range between 2 to 4, which is reasonable. To be specific, for skip gram, we set the total steps to be 1000000, and the loss value begin to remain stable at around 600000 step, indicating that there are a lot more space for training a larger dataset.
- In part 4, we calculate the frequency of the words and present the frequency distribution histogram, which can show the result more directly. Besides, we create a co-occurrence matrix to store every words' frequency of co-occurrence with other words as well as their conditional probabilities. Moreover, the biomedical words or entities we choose are from an authority medical dictionary called $MedTerms$, and we also take English stopwords into considerarion and delete them from the final result to make both the tokens and visualization result more effective.

However, there are still some shortcuts in our projects.

- For the tokenization part, we choose the simplest method in NLTK, there are still many other complicated methods that can tokenize the text more accurately and help to complete the following task better. But due to our lack of time, we just tend to the word_tokenize method. After summer school, if we have time we will use those better method to refine our project.
- For the word representation part, N-gram model can predict with considerable accuracy. However, if we increase the number of passages as input, the run time will be much slower, we tried to improve our algorithm and change the "N", but the result is still unsatisfactory.
- For the visualizing part, the points of the same color is not particularly concentrated, but relatively scattered. We guess it may because the word that have similar meaning may have great difference in form, such as prefix and suffix. So their word representation would not be similar. As a result, after reducing the dimension, the word may not be presented together.

5 Conclusion

The current study builds and showcases effective language models dealing with related natural language processing tasks in a certain subfield. Despite some inherent defections of the models, the overall algorithm design well meets the need for practitioners in biomedical domain to obtain core information required in clinical practice as soon as possible. To be more specific, data are tokenized using standard split(), NLTK, BPE and advanced BPE models in sequence and put out into word representations through N-gram, skip-gram and MLM training models separately. Moreover, t-SNE is also run in our system to demonstrate the outcome with co-occurrence and semantic similarity results shown visually and vividly for a better understanding of the word data. Beyond all, further research and development are still in need to better extract intelligence from biomedical text corpus automatically and efficiently.

References

[1] Pyysalo S, Ginter F, Moen H, et al. Distributional semantics resources for biomedical text processing. 2013.

[2] Gero Z , Ho J . PMCVec: Distributed Phrase Representation for Biomedical Text Processing[J]. Journal of Biomedical Informatics X, 2019, 3:100047.

[3] Wu Y , Xu J , Zhang Y , et al. Clinical Abbreviation Disambiguation Using Neural Word Embeddings[C]. Proceedings of BioNLP 15. 2015.

[4] Zhao D , Wang J , Chu Y, et al. Improving Biomedical Word Representation with Locally Linear Embedding[J]. Neurocomputing, 2021.

[5] Lee J , Yoon W , Kim S , et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining[J]. Bioinformatics, 2019.

[6] Miolo G, Mantoan G, Orsenigo C. ELECTRAMed: a new pre-trained language representation model for biomedical NLP[J]. 2021.

Acknowledgments

First, we would like to show our deepest gratitude to our supervisor, Dr. Zhang Jingqing, who has provided us with valuable guidance in every stage of this project, especially the advice on optimizing the visualization result and the better method to calculate co-occurrence. Without his enlightening instruction, impressive kindness and patience, we couldn't have completed this project.

Then we would like to show our sincere appreciation for DS summer school and all the teachers who have imparted knowledge to us. During this summer school we learned a lot from our erudite teachers, learning a lot of cutting-edge knowledge about data science.

The three-week summer school will be a very unforgettable experience in our university life.