Project Final Report for CS598 DL4H in Spring 2023

Binbin Weng
binbinw2@illinois.edu

Group ID: 43
Paper ID: 145
Presentation link: https://youtu.be/ZYbRTobqKLc
Code link: https://github.com/binbinweng/CS598FinalProject/tree/main/code

1 Introduction

The method I am implementing in this project is from the paper, *A disease inference method based on symptom extraction and bidirectional Long Short Term Memory networks*, which introduces a method of multi-label classifier for disease inference with clinical text data. More specifically, the authors of the paper first extract symptoms from the clinical text data, and then use a deep learning model, bidirectional Long Short Term Memory network (BiLSTM), combined with two different representations of the extracted symptoms, representations with TF-IDF (term frequency-inverse document frequency) and Word2Vec (an embedding method), to improve the performance of the classifier.

2 Scope of reproducibility

The data used in the original paper are clinical text data. As mentioned above, the authors first extract the symptoms from the clinical data. Then they introduce a multiple diseases inference method of two BiLSTMs with two representations of the extracted symptom data, one is the representation with TF-IDF and the other is the representation with Word2Vec, that outperforms one BiLSTM with only one of the two representations of the extracted symptom data. The reason they involves the two representations of the symptoms is that the representation of TF-IDF can reflect the relation between symptom and disease, and the representation of Word2Vec can reflect the relation between symptom and symptom, so involving this two representations means involving much more information in the model than using traditional methods such as multi-hot encoding.

More specifically, the method the original paper proposes has better Precision, Recall, AUC and F1 score than the other methods that it is compared with.

2.1 Addressed claims from the original paper

In this project, I will test the following points:

- The method of using the two representations of the extracted symptoms, one with TF-IDF and the other with Word2Vec, in the BiLSTMs will outperform the method of using only the representation of the extracted symptoms with TF-IDF in the BiLSTM

- The method of using the two representations of the extracted symptoms, one with TF-IDF and the other with Word2Vec, in the BiLSTMs will outperform the method of using only the representation of the extracted symptoms with Word2Vec in the BiLSTM

- In the original paper, the two BiLSTMs are trained separately first and then the outputs of the two BiLSTMs are added together with weight to get the final results. I want to test if training the two BiLSTMs together would have better performance than training them separately.

- I will test if training the two BiLSTMs together would have better performance than training two BiGRUs together with the same hyperparameters as the BiLSTMs. BiGRU is simpler than BiLSTM, so I want to compare the performance of these two methods to test if BiLSTMs would have better performance due to its complexity and test if the difference is worthy compared with the computation time difference.

In addition, the authors of the original paper also show that this method outperforms one BiLSTM with the representation of the full original text data using Word2Vec, and another method called DeepLabeler. However, in the original paper, the methods are not described in details, so in this project, these two methods will not be implemented.

3 Methodology

3.1 Model descriptions

The data used in the paper are clinical text data. The method of processing the clinical text data involves the following steps:

- Filter out the contents in the negated contexts, so that the symptoms in the negated contexts would not be extracted in the later step.
- Identify the symptoms in the clinical text data with the techniques of National Language Toolkit (NLTK) and MetaMap which is used to recognize Unified Medical Language System (UMLS) concepts in the text. After the MetaMap, symptoms are generated. Per the orginal paper, symptoms in the following categories, sosy, dsyn, neop, fngs, bact, virs, cgab, acab, lbtr, inpo, mobd, comd, anab, are kept, and the rest are dropped.
- The numbers of symptoms extracted from each text are different. The maximum number of symptoms one text has is 228, while the minimum number of symptoms one text has is 1. To make the later data processing easier, if the number of symptoms is larger than 50, truncate the part that exceeds 50; if the number of symptoms is smaller than 2, delete the observation.
- Since the numbers of disease codes for each observation are different, to make the later data processing easier, the authors keep the most common 50 disease codes, and the observations without the most common 50 disease codes are deleted. Doing so is based on the analysis that observations with the most common 50 disease codes take over 97% of the total observations. Those 50 disease code will be used as target in the model. In the original paper, a version of the most common 100 disease codes is also implemented.
- Represent the symptoms with TF-IDF.

  Each symptom i will be represented as vector $S_i = (W_{i,1}, W_{i,2}, ..., W_{i,d})$, where $W_{i,j}$ is the strength of the association between symptom i and disease j. More specifically, $W_{i,j} = TF_{i,j} * log\frac{N}{D_i}$, where $TF_{i,j}$ is the number of symptom i in the clinical texts data correlated with disease j, N is the number of all diseases, D_i is the number of diseases associated with symptom i.
- Represent the symptoms with embedding method of Word2Vec with skip-gram, in which the window size is set to be 5 and the vector dimension is set to be 128. In the original paper, the Word2Vec is trained with the raw clinical text data after stop words are removed. However, I train the Word2Vec with the extracted symptoms, because after doing symptom extraction, some extra words about symptoms may be added, and if we train the Word2Vec with raw data, when we try to represent the words in the symptoms, some words may be missing. But training the Word2Vec with the extracted symptoms would not have such an issue.

After the data is processed, the structure of the BiLSTM model is built as the follows:

- Two unidirectional LSTM layers with hidden node size of 100, one for forward input data and the other for backward input data. The outputs of the forward LSTM and backward LSTM will be concatenated as the input for the later layers.
- A dropout layer with drop-out rate of 0.8.
- A fully connected layer.
- A Sigmoid layer.

Two of such BiLSTM models are trained separately with loss function of binary cross entropy and optimizer of Adam with learning rate of 0.001, one for the TF-IDF representation of the symptoms and the other for the Word2Vec representation of the symptoms. The final output will be the weighted sum of the outputs of the two BiLSTM models with weight of 0.5.

The number of parameters in the LSTM layers of the first BiLSTM model is $2 * 4 * ((50 + 100) * 100 + 100) = 120,800$.

The number of parameters in the LSTM layers of the second BiLSTM model is $2 * 4 * ((128 + 100) * 100 + 100) = 183,200$.

3.2 Data descriptions

The data used in the original paper is the table NOTEEVENTS from MIMIC-III, which involves the discharge summaries for patients' admissions. The purpose of the model introduced in the original paper is to predict the diseases a patient has based on his/her discharge summaries, so the table of DIAGNOSES-ICD from MIMIC-III, which contains the diagnose information of a patient's admission, is also being used. These tables are accessed through physionet.org.

In the table of NOTEEVENTS, there are over 2,000,000 observations, which not only includes discharge summaries but also other categories such as nursing notes, nutrition notes etc. Since the data used in the original paper are the discharge summaries, that part of data are extracted, with about 60,000 observations. Among the 60,000 observations, there are over 46,000 unique patients with over 58,000 unique admissions, meaning that some admissions have more than 1 discharge summary. So the discharge summaries with the same admission are combined. Then the discharge summaries needs to merge with diagnoses ICDs in the DIAGNOSES-ICD table. Finally, over 52,000 observations with discharge summaries and diagnoses ICDs are achieved.

3.3 Hyperparameters

The original paper releases the hyperparameters for building the model:

- The vector dimension of the Word2Vec is set to 128, and the window size is set to 5.
- The hidden size of the LSTM layer is set to 100. The drop-out rate of the drop out layer is set to 0.8.
- The weight to sum the output of the two BiLSTMs is 0.5.
- The learning rate of the optimizer is set to 0.001.
- The batch size is set to 400.
- The threshold value used to calculate the metrics when comparing the models is set to 0.2.

3.4 Implementation

I write my own code to implement the methods introduced in the original paper. The github link is `https://github.com/binbinweng/CS598FinalProject/tree/main/code`.

3.5 Computational requirements

The step of identifying the symptoms from the discharge summaries with the MetaMap on python takes much more time than I expected. I was expecting that the time for processing data may need 1 CPU hour. However, extracting the symptoms from 1,000 discharge summaries took me more than 1 day, more specifically 25 CPU hours. The reason why it takes so much time is because the function to extract symptoms provided by MetaMap will print out all the processing contents as well as the information about the package every time when it is called, which takes much of the time. As suggested in the documentation of MetaMap, 2 GB memory and 16 GB disk space are needed to use the MetaMap, because in the process of extracting symptoms, it has a lot of contents to display on the screen.

Luckily, the website of National Library of Medicine provides the Batch MetaMap service, where you can submit the texts and it will help extract symptoms from their side and return result files when the job is done. It takes over 80 hours to finish the symptom extraction.

Since there are some errors in the process of extracting symptoms, the final number of observations is about 47,000, which are split into training set (80%) and testing set(20%).

Total time of the running program from processing original datasets to building the models but excluding the time of getting symptoms from BatchMap is about 6.5 CPU hours.

The whole project involves 4 trials, including building 3 BiLSTM models and 1 BiGRU model.

Training one BiLSTM with one representation of the symptoms takes around 0.8 CPU hour with 100 training epochs, so the average runtime for each epoch is around 0.008 CPU hour. Training two BiLSTMs together with two representations of the symptoms takes around 1.6 CPU hour with 100 training epochs, so the time is doubling that with one BiLSTM, which is making sense.

Training two BiGRUs together with two representations of the symptoms takes around 1.5 CPU hour with 100 training epochs, so average runtime for each epoch is around 0.015 CPU hour. Since BiGRU is simpler than BiLSTM, this runtime is also making sense.

4 Results

Per the original paper, the method of using the two representations of the extracted symptoms, one with TF-IDF and the other with Word2Vec, in the BiLSTMs would outperform the method of using only one of the two representations of the extracted symptoms in the BiLSTM.

The metrics used to compare the methods are four micro-averaging measurements: Precison (MiP), Recall (MiR), F1-score (MiF1) and AUC (area under the receiver operating characteristic curve). The measurements are defined as the follows:

$$MiP = \frac{\sum_{i,j} y_i^j \hat{y}_i^j}{\sum_{i,j} \hat{y}_i^j}$$

$$MiR = \frac{\sum_{i,j} y_i^j \hat{y}_i^j}{\sum_{i,j} y_i^j}$$

$$MiF1 = \frac{2*MiP*MiR}{MiP+MiR}$$

The results are showed in Table 1 and Table 2.

Table 1: The performance of different models (cutoff: 0.2) with the most common 50 ICD codes

Model	MiP	MiR	MiF1	Micro Auc
BiLSTM with TF-IDF	0.418	0.596	0.492	0.804
BiLSTM with Word2Vec	0.460	0.584	0.515	0.814
BiLSTMs with TF-IDF and Word2Vec	0.456	0.611	0.522	0.829
Combined training of BiLSTMs	0.486	0.643	0.554	0.842
Combined training of BiGRUs	0.508	0.659	0.573	0.856

Table 2: The performance of different models (cutoff: 0.2) with the most common 100 ICD codes

Model	MiP	MiR	MiF1	Micro Auc
BiLSTM with TF-IDF	0.442	0.491	0.465	0.820
BiLSTM with Word2Vec	0.459	0.504	0.480	0.826
BiLSTMs with TF-IDF and Word2Vec	0.463	0.508	0.484	0.835
Combined training of BiLSTMs	0.475	0.535	0.503	0.843
Combined training of BiGRUs	0.495	0.568	0.529	0.857

4.1 Result 1

Based on Table 1 and Table 2, MiP, MiR, MiF1, and AUC of the BiLSTMs with the two representations are all higher than that of BiLSTM with only the representation of TF-IDF, which fully supports the claim in the original paper.

4.2 Result 2

Based on Table 1, MiR, MiF1 and AUC of the BiLSTMs with the two representations are higher than that of BiLSTM with only the representation of Word2Vec, which does not fully support the claim in the original paper.

Based on Table 2, MiP, MiR, MiF1 and AUC of the BiLSTMs with the two representations are all higher than that of BiLSTM with only the representation of Word2Vec, which fully supports the claim in the original paper.

4.3 Additional results not present in the original paper

4.3.1 Result 3

In the original paper, the two BiLSTMs are trained separately first and then the outputs of the two BiLSTMs are added together with weight to get the final results. I want to test if training the two BiLSTMs together would have better performance. Based on the results in Table 1 and Table 2, training the BiLSTMs together has higher MiP, MiR, MiF1, and AUC than the model that simply sums up the results from the two separately trained BiLSTMs with weight. So training the two BiLSTMs together has better performance.

4.3.2 Result 4

In addition, suprisingly, training two BiGRUs together has better performance than training the two BiLSTMs together. Since BiGRU is simpler than BiLSTM, it was expected that BiLSTM should have better performance. However, the results in the Table 1 and Table 2 show that the model of BiGRUs outperforms the model of BiLSTMs with all the measurements.

The reason might be that since the symptoms are finally used in modeling instead of the original full texts, the long term memory is not important in predicting the results, while the short term memory or more specifically the memory from last time step is more important in predicting the results. The reason still needs to be tested with further experiments.

Based on the runtime of training the BiLSTMs and BiGRUs and performance of these two methods, the method with BiGRUs is better.

5 Discussion

The results I got support almost all the claims of the original paper, that the method introduced in the paper outperform the other methods with almost all the measurements. After adjusting the method of modeling a little bit (training two BiLSTMs togeteger), all the claims can be supported. In the other words, the experiments from the paper are reproducible.

5.1 What was easy

After preparing the data, the modeling part is not too difficult. Since the models used in the original paper do not have complex structures, as long as you are familiar with the pytorch and contents from class and homework, it would be easy to build the models.

5.2 What was difficult

The process of dealing with the text data is difficult. The texts in the dataset are messy, so it takes me much time to fully understand that text data and try different ways to process data for modeling use. The original paper introduces MetaMap to extract syptoms from text data, but it does not describe the methods in details. So it also takes me a lot of time to do research about the method.

5.3 Recommendations for reproducibility

Before working on the whole datasets, it is better to extract a subset of the datasets to process the data, build the models and compare the results to see if the methods are reproducible. If the program you build works, put it on the whole datasets. It takes much more time to process the whole datasets than just a little subset, but the results reflected from the little subset is close to that of the whole datasets. So the little subsets is enough to help you understand the data, build the program and save you a lot of time.

6 Communication with original authors

The original paper introduces the model of two BiLSTMs with TF-IDF representation of symptoms and Word2Vec representation of symptoms has better performance than the model of one BiLSTM with just one of the representations. It also introduces that the model is better than the model of one BiLSTM with WordSeq, but the method of generating WordSeq is not described in details which is hard for the readers to implement.

In addition, the original paper trains the Word2Vec model with raw clinical text data, but since some extra words are added to symptoms after the MetaMap does symptom extraction, these extra words would be missing in the Word2Vec model. This issue and solution are not described in the original paper.

References

[1]Donglin Guo, Guihua Duan, Ying Yu, Yaohang Li, Fang-Xiang Wu, Min Li: A disease inference method based on symptom extraction and bidirectional Long Short Term Memory networks

[2]Parikshit Sondhi, Jimeng Sun, Hanghang Tong, ChengXiang Zhai: SympGraph: A Framework for Mining Clinical Notes through Symptom Relation Graphs

[3]https://pytorch.org/

[4]https://numpy.org/

[5]https://pandas.pydata.org/

[6]https://stackoverflow.com/

[7]https://lhncbc.nlm.nih.gov/ii/tools/MetaMap/documentation/Installation.html

[8]https://scikit-learn.org/

[9]https://physionet.org/content/mimiciii/1.4/