Team Project Final Report

Zerui Tian and Yixing Zheng
{zeruit2, yixingz3}@illinois.edu

Group ID: 64, Paper ID: 79
Code Link: https://github.com/yixingz3/DL4H_team_project
YouTube Video Link:
Please check GitHub repository for presentation slides

1 Introduction

This work aims to provide a new effective disease inference method utilizing symptoms extracted from Electronic Medical Records (EMR) data. The relationship between symptoms and diseases is represented by the term frequency-inverse document frequency (TF-IDF) model. And a bidirectional recurrent neural network (Bi-LSTM) is utilized to model the symptom sequences in EMR data. This combination of models shows a significant improvement in disease inference - 4% to 10% on average improvement from the two baseline models (Guo et al., 2018).

2 Scope of reproducibility

This paper introduced a new disease inference method utilizing the combined model of TF-IDF and Bi-LSTM that outperforms two similar existed models on the task of disease inference based on symptoms. Utilizing the TF-IDF and Bi-LSTM combination will provide higher accuracy than utilizing DeepLabeler and WordVec + Bi-LSTM with the same extracted symptoms data. Besides the TF-IDF, WordVec is also a method of embedding symptoms and converting them into vectors. This method is also considered in this paper, but it does not perform as well as TF-IDF in terms of accuracy.

We decide to choose this claim as it is the major contribution of the paper and we would like to verify its validity with our own implementation of the combined model. In addition, in the course Deep Learning for Healthcare, we learned the word embedding method of WordVec, but this paper uses TF-IDF to achieve better results, and we have a strong interest in it. We wanted to try out different word embedding methods and compare them or combine them to achieve better results.

In the meanwhile, to replicate this article we need the relevant knowledge of natural language processing. We are passionate about this field and we want to apply the knowledge we learn in the course to solve some relevant problems.

2.1 Addressed claims from the original paper

Clearly itemize the claims you are testing:

- **TF-IDF + Bi-LSTM** provides higher accuracy on disease inference than **GloVe + Bi-LSTM**, **WordVec + Bi-LSTM**, and **Embedding** given the same extracted symptoms input data.

Since the baseline model DeepLabeler was originated from the authors' other work, we have decided to replace it with the **GloVe + Bi-LSTM** model along with **Embedding Layer**.

3 Methodology

We couldn't locate any code and the original data set that was extracted from EMR records using MetaMap by the authors and used to train the proposed and baseline models. Therefore, we will re-implement the proposed model based on the paper's description as well as the baseline models. So that we can compare the results and validate the claim with our implementation and a different data set that is available to us.

3.1 Model descriptions

We have re-implemented the TF-IDF + Bi-LSTM model. TF-IDF is proposed to represent text documents as vectors of identifiers. We can use TF-IDF to model the relationship between symptoms and diseases. Now each symptoms is transformed to vector representation. For symptoms S_i, it can be represented as follows:

$$S_i = (W_{i,1}, W_{i,2}, \ldots, W_{i,d})$$

And $W_{i,j}$ is the strength of the association between symptoms i and diseases j used TF-IDF method.

$$W_{i,j} = TF_{i,j} * \mathbf{log}\frac{N}{D_i}$$

Here N is the number of all diseases we take into consideration and D_i denotes how many diseases are associated with the symptoms i. $TF_{i,j}$ is the number of symptom i in the discharge summaries correlated with disease j.

Since RNNs have the problem of vanishing gradients, it is difficult to handle long sequences of data. The special case of RNN, LSTM (Long Short-Term Memory), is improved by RNN. It can avoid the gradient disappearance of conventional RNN, so it has been widely used in the industry. LSTM has three gates, namely forget gate, input gate and output gate. These three gates cooperate with each other and can solve the problem of gradient disappearance to a certain extent. And bidirectional LSTM can better extract information. So the model adopts bidirectional LSTM as the neural network structure.

About the evaluation metrics, the measurements are defined as follows:

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

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

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

Here y_i^j stands for the true label i of sample j, $\hat{y}_i^j$ stands for the predicting label i of sample j.

In general, the structure of this paper is to first extract symptoms, then use TF-IDF to represent symptoms as vectors, and then input them into bidirectional LSTM for prediction and evaluation.

We also attempted different embedding method including GloVe. GloVe method will first construct a co-occurrence Matrix X. And X_{ij} represents the number of co-occurrences of word i and context word j within a context window of a certain size. Then it builds an approximate relationship between word vectors and co-occurrence matrices using this formula:

$$w_i^T \hat{w}_j + b_i + \hat{b}_j = \mathbf{log}(X_{ij})$$

And then use this loss function to train the model:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \hat{w}_j + b_i + \hat{b}_j - \mathbf{log}(X_{ij}))^2$$

Word2Vec is also a popular embedding method. And we have learned it in this course. It is used to find embedding vectors such that similar words are close together and dissimilar words are far apart.

To sum up, we implemented different embedding methods including **TF-IDF**, **GloVe**, **Word2Vec** and regular embedding layer. Then we combined these embedding method with Bi-LSTM model to do the classification job.

3.2 Data descriptions

We use the IMDB data set, which contains 50,000 IMDB movie reviews for sentiment analysis. We use 25,000 as training data and another 25,000 as test data for training. In the training and test sets, half are positive reviews and half are negative reviews.

At the due time of our final submission, we have not received any response from the original authors for the code and data that we requested. Although we were granted the access to use MetaMap, the result produced by the software was different from our expectations and thus would not be usable for training and evaluating our implementation of the models. Additionally, we have tried to extract the symptoms using a Python script, but the data produced was not competent to be consumed by the models. Therefore, we decided to use the IMDB data set, which is less than ideal but should be sufficient to confirm that the proposed model performs better than the baseline models.

3.3 Hyperparameters

The learning rate for our four models is 0.001. The batch size is 64. The hidden size is 100. And the dropout rate is 0.8.

3.4 Implementation

We have re-implemented the model ourselves and the code can be found from the following Repo. Our GitHub repository has a README file that states the files and data set used for our implementation. In the DL4H_project file that contains all the model implementations, we also included inline comments describing the process we use for training the model and validating the result.

3.5 Computational requirements

We use the Google Colab as our hardware to perform the calculation with the basic version that provides up to 12.68 GB RAM and 78.19 GB Disk space. While we don't have the specific statistics of RAM and Disk usage for each model's computation, we know that this amount of resource is abundant to perform the training and evaluation for our models. We have used 10 training epochs for each of our implementations of the models, and the average runtime for each epoch is around 1 minute.

Model	Hardware	Runtime	Epochs
TF-IDF	Colab	∼10 Mins	10
GloVe	Colab	∼10 Mins	10
WordVec	Colab	∼10 Mins	10
Embedding	Colab	∼10 Mins	10

4 Results

4.1 TF-IDF + Bi-LSTM provides higher average accuracy on test set and WordVec + Bi-LSTM provides higher accuracy after 10 epochs training

We have implemented four models: TF-IDF+Bi-LSTM, GloVe+Bi-LSTM, WordVec+Bi-LSTM, Embedding Layer+Bi-LSTM. We trained them separately and analyze the results.

Table 1: Mean results:

Model	Loss	Train Acc	Test Acc
TF-IDF	**0.0738**	**0.9108**	**0.8345**
GloVe	0.1836	0.6814	0.6776
WordVec	0.1528	0.7929	0.8042
Embedding	0.1429	0.8331	0.7682

Table 2: Last epoch result:

Model	Loss	Train Acc	Test Acc
TF-IDF	0.0208	0.917	0.827
GloVe	0.0363	0.843	0.840
WordVec	0.0285	0.885	**0.881**
Embedding	**0.0128**	**0.955**	0.811

4.2 Analyses

We decided to implement the Glove + Bi-LSTM model, as a replacement of the DeepLabeler model, since there is more information available than the DeepLabeler model that was also created by the authors. Along with it, we have the WordVec + Bi-LSTM and embedding models as the baseline models.

The accuracy result we have from our implementation of the TF-IDF + Bi-LSTM model shows a similar result to the paper (0.8345). We believe that the variance of the data set would be mainly responsible for the gaps here. Additionally, even with different data, our implementation of the proposed model performed consistently better than the three baseline models, which is the main claim that we have now verified after building and training all the models using the same data set.

We can see that TF-IDF achieves the highest average accuracy. This is because the TF-IDF method considers the label of the data when performing word embedding, so the accuracy of the initial training will be high. However, WordVec is better than other models in the final result, which shows that other models including TF-IDF may cause over-fitting problems. Among them, although the regular embedding layer achieves the highest accuracy on the training set, the accuracy in the test set is the lowest, which indicates that over-fitting is indeed easy to occur. Of course, different methods may have different performances for different data sets, but this also shows that TF-IDF and WordVec are both excellent word embedding methods.

5 Discussion

We were able to reproduce a similar result as the work indicated and verified the main claim. However, we did encounter some issues. The two major ones are: 1) without the details of the original implementation, we are not able to fine-tune the model to achieve optimized performance; 2) without the original processed data set and detailed guidance on how to obtain such data set, we have to train and evaluate our implementation of the proposed model on an alternative data set, which might lead to even further performance decrement and less accurate results.

5.1 What was easy

The implementation of the proposed model along with the baseline models was relatively easy in the whole process of reproducing the work. For the proposed model, the original authors provided a detailed description of the structure, so it was comprehensible for us. Since the proposed model is the composition of two existed models, we were able

to obtain more information about these two models, and eventually, combine them to form the proposed model. On the other hand, besides the DeepLabeler model that was developed by the same authors, all other baseline models were existed, which enabled us to gather information from different sources and reproduce them.

5.2 What was difficult

We have encountered two major difficulties: the code and the data.

On the code side, the paper described the proposed model with clear language and indicated that the performance enhancement between the proposed model and the baseline models. However, the implementation of the models is not provided. Ideally, we would expect to have the implementation with a pre-trained model and a smaller sample data set for verifying the performance of the proposed model. With none of the above provided, we had to implement the model ourselves with the instruction stated in the paper. And we suspect that we might have the most optimized implementation as the authors did, due to missing details of hyperparameters and potentially other optimizations that were not captured in the writing of the paper. This might lead to an under-performed implementation of the proposed model, thus making the performance comparison between the proposed and baseline models less accurate.

On the data side, the main issue is that we are not able to process the data to obtain a valid input form for our implementation of the model. We followed the instruction and requested access to the MIMIC-III data set and the data mapping tool called MetaMap, both accesses were granted in a timely manner. However, MetaMap was not working as expected when we attempted to extract the symptoms from the 'Discharge Summary' section in the EVENTNOTES.csv file in MIMIC-III. We were expecting extracted and highlighted symptoms with their corresponding ICD-9 codes so that we could perform the mapping and find the correlations. In the results we obtained, the symptoms were extracted and assigned an integer number less than 1000, which is different from the ICD-9 coding system. With this setback, we decide to find the ICD-9 codes for patents from the MIMIC-III data set. But the issue with this approach is that each patent has multiple 'sequence' records within each visit (some patients might have multiple visits) and there is no indication of how and why those different records were assigned with different ICD-9 codes. To fix this, we have two approaches: 1) we pick the first ICD-9 among the many and attempt to map the symptoms to it; 2) we take the Cartesian product of the symptoms and ICD-9 codes as our input. Neither approach provided the expected data that we could use as input, so we decided to use the IMDB sample data set for training and evaluating all of our implementation of the models.

5.3 Recommendations for reproducibility

Based on the two major difficulties we have encountered, we believe that having the implementation of, at least, the proposed model with a small subset of the processed data set would be critical for improving the reproducibility. Since we have only had very little to work with, i.e. the concept of the proposed model and the description of the original and processed data set, we would not be able to provide more specific recommendations such as implementing the code utilizing Jupyter Notebook to simplify the setup process, providing a pre-trained model, or having more detailed documentation either in the code or supplemented files.

Additionally, it would be very helpful if the original authors were more responsive to our help inquiries. We would, at least, have a better idea of the direction we are going for the model implementation and data processing, or ideally, the code and original data set used to train and evaluate the models.

5.4 Additional results not present in the original paper

As previously mentioned, we replaced the DeepLabler model from the paper with the GloVe model mainly due to the complicity of reading through and implementing another original model, while the GloVe model has more available information on the Internet. Additionally, we have added the embedding model as another baseline model because both GloVe and Word2Vec are powerful word embedding methods, and their method of word embedding is different from TF-IDF and Word2Vec. So we choose it as another baseline model. We also added an embedding layer that did not load any pre-trained weights to see if the pre-trained word vectors had an effect on the results.

6 Communication with original authors

We have tried to contact the original authors of the paper in hope of receiving help for the code and data used, but as of now, we have not received any responses.
Please access this Google Drive with Illinois credentials to see the email sent to the original authors:
https://drive.google.com/drive/folders/1_p4CNHAMX1lt7tltUyxu0U6N3rptSqK8?usp=sharing

References

Donglin Guo, Min Li, Ying Yu, Yaohang Li, Guihua Duan, Fang-Xiang Wu, and Jianxin Wang. 2018. Disease inference with symptom extraction and bidirectional recurrent neural network. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 864–868. IEEE.