CompareNet: Reproduction of Graph Neural Fake News Detection with External Knowledge

Zhiyun Peng, Zhuocheng Guo, Yipeng Liu

Abstract

Our group reproduces the results of the proposed method and one of the baseline methods in Hu et al. (2021). This paper employs external knowledge from Wikipedia (i.e. the knowledge base, KB) and uses Latent Dirichlet allocation to consider topics of news as the components of a graph neural network to improve fake news detection accuracy. The scope of our project is to implement the source code, thus reproducing the main results of the proposed method. Secondly, we implement the basic baseline method, long short-term memory networks (LSTM), in this article, to compare with the proposed method. Also, we carefully examine the author's algorithm and analyze how the algorithm functions – during this process, we have a deep understanding of the model. The datasets we use are the same ones in Hu et al. (2021), and the dataset size as a whole is around 50,000, containing four classes or two classes.

1 Introduction

1.1 Background

Today, society is increasingly turning to digital platforms for their news. In research conducted by the Pew Research Center, half of Americans say they do not make use of print publications or newspapers that much, while two-thirds turn to the Internet and cellphone applications specifically[1].

The role of social media in democratizing media participation, which is being described as the beginning of a new era where all users can share news and commentary, may not be ideal. Especially during the Covid-19 pandemic in the past three years, the fake news spread by social media has brought a huge negative impact. Therefore, developing efficient methods to detect fake news is crucial.

[1] https://pewrsr.ch/2MZqns7

1.2 Problem Statement

We reproduce a state-of-the-art fake news detection approach in Hu et al. (2021). This is a text classification problem, which means it does not need other elements, such as pictures in the news, the publisher or author's information, etc. The input has two parts: 1) the news text such as "Mammograms are not effective at detecting breast tumors ...". The second part: The first paragraph of the main entities of the news in the website Wikipedia: "... The goal of mammography is the early detection of breast cancer." The output is the predicted label of the news, such as "Hoax", and there are 4 classes. Particularly, we first implement the source code, thus reproducing the main results of the proposed method. Secondly, we implement the basic baseline method LSTM in this article, to compare with the proposed method. This problem is of interest and the reasons are stated in Section 1.3. In short, we could learn a lot from this re-implementation, such as an effective code base, and ingenious model design.

Some of the challenges are that Statistically speaking, people usually cannot reproduce the exact results claimed by the authors, not to mention those cases where the codes can't even be run successfully. Also, the author did not provide any source code except their proposed method, so we have to implement them with our efforts.

The research hypothesis of our project is that the results of Hu et al. (2021) are credible, and the CompareNet model does have an evident advantage over baseline method.

1.3 Motivation

For motivation, firstly, we respect Hu et al. (2021) for its success in the top conference of NLP, and that its proposed method outperforms state-of-the-art methods in this field. Moreover, by implement-

ing its approach, we can understand the ingenious design of the approach, the highly-efficient code base, and how and where it is better than the advanced NLP model taught in our course.

1.4 Contribution

Our group reproduces part of the results in Hu et al. (2021), which builds a CompareNet to do the graph neural fake news detection with external knowledge. Our project replicates the findings in this paper. We also implement the LSTM-based model Hochreiter and Schmidhuber (1997) as the baseline method to see the comparison of the performance.

The contributions are as follows:

1. We reproduced the results of the proposed model in Hu et al. (2021) and verified the validity of their results, but also found the shortcomings of the paper.

2. We reproduced the results of the comparative method, LSTM, in Hu et al. (2021), by implementing the model through our own efforts. Thus, we further confirmed the superiority of the proposed method in this paper.

3. We analyzed the function and contributions of different modules in the proposed method.

2 Related Work

2.1 Fake News Detection methods

Fake news detection methods are mainly divided into the following categories: (1) the first category is methods based on content features Conroy et al. (2015). For example, count the number of verbs and nouns of the language used in the news. (2) The second big branch is based on social network methods Zhou and Zafarani (2020). This method includes an assessment of the reputation of the publisher's social media account. (3) The third branch is fake news detection methods based on image analysis Choraś et al. (2021). Many tweets and microblogs contain pictures, which can be additional information to do detection work. (4) The fourth category is based on user characteristics, including users' social influence.

2.2 Related Work Pertaining CompareNet

CompareNet model is the model that our project re-implements. CompareNet belongs to the content-based methods. Before CompareNet model, Conroy et al.'s fake news detector model reply too much on linguistic and semantic features that are manually crafted Conroy et al. (2015). To solve the heavy hand-crafted features issues, Bi-LSTM, convolutional neural networks (CNN), and BERT-based models are developed as the flagship methods of deep learning models Oshikawa et al. (2020). Vaibhav et al. claimed that the deep learning models above fail to consider the sentence interactions in the news text, and proposed a fully connected sentence graph model combined with a graph attention mechanism as the fake news detector Vaibhav and Hovy (2019). However, our selected paper Hu et al. (2021) indicated that although the model in Vaibhav and Hovy (2019) is effective with sentence interactions and a graph attention mechanism, they did not make use of the external knowledge base (KB), which is a powerful helper for discriminating fake news.

2.3 Related Work Pertaining our Implementation

Our baseline method is bidirectional LSTM based on pre-trained word embedding GloVe. Firstly, LSTM Hochreiter and Schmidhuber (1997) is an improved variant of recurrent neural network (RNN) Rumelhart et al. (1985), by turning multiplication into addition. The RNN-based model can reserve and memorize the information in a long sequence, thus outperforming the fake news detector based on CNN. In addition, Schuster and Paliwal (1997) introduced bidirectional LSTM, which further increases the expressive ability of LSTM.

Secondly, we employ pre-trained word embeddings, GloVe, for the word embeddings of each word in news text. GloVe is trained by the co-occurrence of word pairs Pennington et al. (2014), and is a popular trained word embedding in NLP community.

Currently, we found Bahad et al. (2019) adopted the same bidirectional LSTM based on GloVe for fake news detection. However, generally speaking, this journal controversially does not have a good reputation in academia; And this paper uses the already existing methods for fake news detection, which is not novel, though it has more than 100 citations according to the popularity of the LSTM and fake news detection topic.

3 Approach

Firstly, we re-implement the CompareNet whose framework is shown in Fig. 1. There are mainly

three parts. The first part is the directed heterogeneous document graph. It has three elements, including topics, sentences, and entities. The topics and sentences have bidirectional relationships. There is also a bidirectional relationship between every two sentences. The sentences and entities have one-way relationships from sentences to entities. It avoids integrating the correct entity's knowledge into news representation which may lead to the false detection of fake news.

After we obtain the directed heterogeneous document graph, the graph as input will pass into the heterogeneous graph attention network. This is the second part of CompareNet. The heterogeneous convolution layer updates the $(l+1)$-th layer representation of the nodes $\boldsymbol{H}^{(l+1)}$ by aggregating the features of their neighboring nodes $\boldsymbol{H}_{\tau}^{(l)}$ with different types τ. We have three types (denoted as $\mathcal{T}$) of nodes: sentences S, topics T, and entities E. This updating level is defined as dual-level updating. Formally Hu et al. (2021):

$$\boldsymbol{H}^{(l+1)} = \sigma(\sum_{\tau \in \mathcal{T}} B_{\tau} \cdot \boldsymbol{H}_{\tau}^{(l)} \cdot \boldsymbol{W}_{\tau}^{(l)}) \qquad (1)$$

where $\boldsymbol{H}^{(l+1)} \in R^{|\mathcal{V}| \times |M|}$ is the $(l+1)$-th layer representation of the nodes. $\boldsymbol{H}_{\tau}^{(l)} \in R^{|\mathcal{V}_{\tau}|}$ is the features of their (i.e. the nodes in $(l+1)$-th layer's) neighboring nodes with different types τ. $\sigma(\cdot)$ is the activation function. $\boldsymbol{W}_{\tau}^{(l)}$ denotes the different transformation matrix for nodes with different types τ. $B_{\tau} \in R^{|\mathcal{V}| \times |\mathcal{V}_{\tau}|}$ is the attention matrix, whose rows represent all the nodes and column represent their neighboring nodes with the type τ. $\mathcal{V}$ represents the set of nodes in the heterogeneous graph. M is the hidden layer dimension.

This convolution neuron network will output two embeddings. One is sentence embeddings, which is the representation of the sentence nodes $\boldsymbol{H}_S \in R^N$, where N is the node embedding dimension. The other is contextual entity embeddings $\boldsymbol{e}_c \in R^N$. The sentence embeddings will pass into a max-pooling layer and generate the final topic-enriched news document embedding $\boldsymbol{H}_d \in R^N$. The contextual entity embeddings will pass into the entity comparison network.

The third part of CompareNet is the entity comparison network. The entity comparison network needs two kinds of entity embeddings to compare. One is contextual entity embeddings $\boldsymbol{e}_c$ from the heterogeneous graph attention network and the other is KB-based entity embeddings $\boldsymbol{e}_{KB} \in R^M$. To obtain KB-based entity embeddings, the CompareNet needs to get structural embeddings $\boldsymbol{e}_s \in R^M$ and textual embeddings $\boldsymbol{e}_d \in R^M$ from external knowledge base such as Wikipedia. Structural embeddings can be obtained from the relationships between entities via TransE Pan et al. (2018). For textual embeddings, it can be obtained from the entity descriptions via the LSTM network Hochreiter and Schmidhuber (1997). Then CompareNet uses a gating vector $\boldsymbol{g}_e \in R^M$ (w.r.t the entity e) to trade-off information from structural embeddings and textual embeddings and generates KB-based entity embeddings Hu et al. (2021):

$$\boldsymbol{e}_{KB} = \boldsymbol{g}_e \odot \boldsymbol{e}_s (1 - \boldsymbol{g}_e) \odot \boldsymbol{e}_d \qquad (2)$$

where $\odot$ denotes element-wise multiplication.

The KB-based entity embeddings $\boldsymbol{e}_{KB}$ will pass into the entity comparison network with contextual entity embeddings $\boldsymbol{e}_c$. The entity comparison network mainly measures the embedding closeness and relevance between source documents and external knowledge base. The comparison vector $\boldsymbol{a}_i$ can be formulated as follows Hu et al. (2021):

$$\boldsymbol{a}_i = f_{cmp}(\boldsymbol{e}_c, \boldsymbol{W}_e \cdot \boldsymbol{e}_{KB}) \qquad (3)$$

where $\boldsymbol{a}_i \in R^N$ denotes the comparison vector for i_{th} entity ($i = 1, ..., n$). f_{cmp} is the comparison function that measures the embedding closeness and relevance Shen et al. (2018). $\boldsymbol{W}_e \in R^{N \times M}$ is a transformation matrix. The final output comparison feature vector $C \in R^N$ is calculated by max pooling the alignment vectors $A = \boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n$ of all the entities in the news text.

In the end, the final topic-enriched news document embedding $\boldsymbol{H}_d$ will concatenate with the comparison feature vector C as output from the entity comparison network. It will feed into a classifier to decide whether the document is fake Hu et al. (2021):

$$Z = Softmax(\boldsymbol{W}_o[\boldsymbol{H}_d, C] + \boldsymbol{b}_o) \qquad (4)$$

where $\boldsymbol{W}_o$ and $\boldsymbol{b}_o$ denotes the parameter matrix and bias of a linear transformation. Z is the probability of samples being one of the classes. For the training phase, we use the cross-entropy loss over the training data with the L2 regularization Hu et al. (2021):

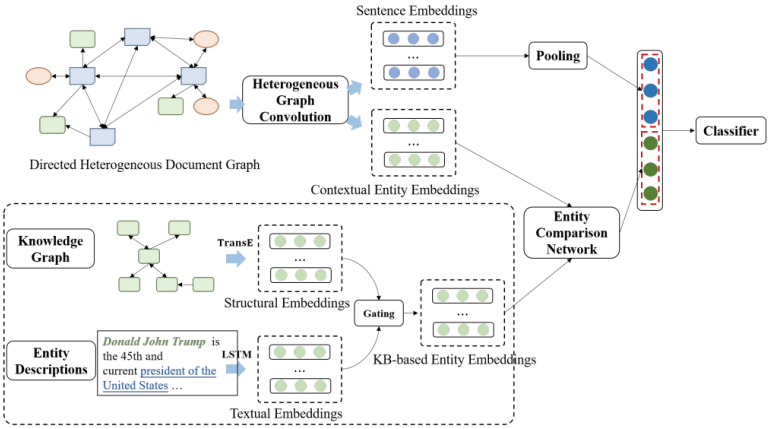

Figure 1: The model architecture of CompareNet (sourced from Hu et al. (2021)).

$$L = -\sum_{i \in D_{train}} \sum_{j=1}^{N} Y_{ij} \cdot log Z_{ij} + \eta ||\Theta||_2 \quad (5)$$

where D_{train} is the news documents in training set. Y is the corresponding label indicator matrix. Θ is the model parameters. η is the L2 regularization factor. We use gradient descent to optimize the model.

After re-implementing the proposed method, we use the bidirectional LSTM network to complete the fake news detection task as our baseline method. Note that considering the pages of this report, we only briefly introduce the mathematical equation of LSTM, and also, LSTM is a baseline method for the paper, which is usually not described much mathematically in a research paper. The basic idea of LSTM can be formulated as:

$$h_t = f(h_{t-1}, x_t) \quad (6)$$

where h_t is a hidden state at time step t, x_t is the input element in the sequence at time step t, $f(\cdot)$ is the updating function for the hidden state, which generalizes the updating functions for input gate, forget gate, output gate, new memory cell, final memory cell, and final hidden cell. In LSTM, it introduces a state called cell state c_t, which stores long-term information.

Firstly, we tokenize all the words in the dataset by GloVe tokenizer. Then we initialize word embeddings for each word in the document by pre-trained word embeddings – GloVe. After that, we push each news text as a sequence of word embeddings into the LSTM network Hochreiter and Schmidhuber (1997). The sequence will pass through an LSTM encoder, a max pooling layer, and a fully connected layer, to get the final label prediction (to determine whether the news is fake).

The existing code is the CompareNet code from the original paper; it mainly uses the Pytorch package. For LSTM, we referred to an open source packaged call "textattack package" Morris et al. (2020), and adapted the modules from this package to complete our implementation of LSTM.

Based on the source code of CompareNet, we changed the setting-ups to generate results for both 2-way classification and 4-way classification. The LSTM model is written by ourselves, only referring to the examples from "textattack package". And the SNE visualization part, and how to take how the hidden state vectors are done by ourselves.

4 Experiments

We first install the environment of the proposed method in Compute Canada (1 GPU, 6 CPU, 32000M RAM) according to the given instructions, as well as download the three datasets, and clone the GitHub repository which already contains the prepared files pertaining to the Wikipedia first paragraph of the corresponding entities. We reproduce the final results claimed in this paper.

The datasets we use are the same ones in Hu et al. (2021), which are "SLN: Satirical and Legitimate News Database" and "LUN: Labeled Unreliable News Dataset". Table 1 shows the statistics. Following this reference paper, we run the models 5 times and report the micro-averaged (Precision = Recall = F1) and macro-averaged scores (Precision, Recall, F1) in all the settings including 2-way and 4-way classification.

Table 1: Statistics of the dataset. GN refers to Gigaword News.

Dataset	Trusted (#Docs)	Satire (#Docs)	Hoax (#Docs)	Propaganda (#Docs)
LUN-train	GN except 'APW' and 'WPB' (9,995)	The Onion (14,047)	American News (6,942)	Activist Report (17,870)
LUN-test	GN only 'APW' and 'WPB' (750)	The Borowitz Report, Clickhole (750)	DC Gazette (750)	The Natural News (750)
SLN	The Toronto Star, The NY Times (180)	The Onion, The Beaverton (180)	-	-

Note that the size of the dataset as a whole is around 50,000, containing four different classes or two different classes. The average length of news is around 450 words.

Specifically, in 2-way classification, we choose the satirical and trusted news articles from 80% of the samples in LUN-train for training and use the left 20% in LUN-train for validation set (or development set). We evaluate the proposed model on the entire LUN-test set and SLN dataset. On the other hand, in the 4-way classification, we split the LUN-train into an 80:20 split for the training set and validation set, and used LUN-test as the testing set.

Afterward, we implement the baseline method: bidirectional LSTM. As stated in this paper, we use LSTM to encode each word in a sample news document with word embeddings from pre-trained word embeddings GloVe and pass the word embeddings sequence to the LSTM model to obtain hidden layer embeddings, which is used for the classification. Thus after the training period, we get all the parameters of the LSTM. Note that we freeze the word embeddings from GloVe, due to the heavy computation load to fine-tune the pre-trained GloVe embeddings.

The existing code is the proposed method's source code. We implement LSTM by ourselves with the help of the package "textAttack" (cited in section 3). We refer to the "textAttack" package for the LSTM as the starting point, adapt it to our experiment, and run it on these datasets.

4.1 Experiment Setting

Following up the best results in the validation set from Hu et al. (2021), the hyperparameters are chosen as follows. We set the number of topics $K = 100$ in LDA, with each sentence assigned to the top $P = 2$ topics with the highest probabilities. The heterogeneous graph convolution's layer number is set as $L = 1$. Also, all the hidden dimensions are set as $M = 100$. The node embedding dimension $N = 32$. The activation function is LeakyRelU with a slope of 0.2. The maximum training epochs is 15, and the optimizer is Adam with a learning rate of 0.001. L2 normalization factor $\eta = 1e - 6$.

For bidirectional LSTM, after analysis and fine-tuning, we choose the hyperparameters as follows. The framework of this method is a dropout layer (first layer) with a dropout rate of 0.3. A Glove embedding layer (second layer) with vocabulary size 400005. Each token (which consists of words and punctuations) 's embeddings dimension is 200. And we only chose the first 128 tokens as the sequence, due to the limited ability to memorize long sequence information. Note that we did not preprocess the original texts. We found that without preprocessing, the LSTM model works well in the first dataset, so we did not try preprocessing.

The next layer is the encoder layer (third layer), i.e. LSTM layer, with input dimension 200, and output dimension 75. Then we concatenate the bidirectional encoding output to get the encoding output with dimension 150. Here, for each news text, we have 128 tokens, and each token has an embedding with 150 dimensions. We do a max pooling: for each token in the 128 tokens corresponding to a new text, we choose the element with the maximum value in the 150 dimensions embeddings. In this way, we flat a news text to only an embedding with 150 dimensions.

Then we feed this encoding output to a fully connected layer (fourth layer) with input dimension 150, and output dimension 2. Note that we only implemented 2-way classification for the interest of time. We set the batch size to 128. We provide the LSTM with 10 training epochs and 20 training epochs. Also, note that we mistakenly set the output layer dimension of the LSTM with 20 training epochs as 4, instead of 2. But we analyzed that during the learning process, it will automatically ignore the useless 2 extra labels slot and work well. According to the experiment results, as expected, this mistake can be fixed by backpropagation automatically.

We run the LSTM model on our workstation with 1 GeForce GTX 1050 Ti GPU, i7-6700 CPU with 8 cores, and RAM 32 GiB. The running time for LSTM with 10 epochs is 20 minutes.

4.2 Results of CompareNet

The performance of CompareNet with different datasets in 2-way classification is shown in Table 2. We report micro F1, macro precision, macro recall, and macro F1. Micro F1 computes a global average F1 score by counting the sums of the True Positives (TP), False Negatives (FN), and False Positives (FP), and the micro-F1, micro-precision, micro-recall, and accuracy share the same value. The Macro F1 score is computed using the arithmetic mean of all the per-class F1 scores. We find that our re-implemented model has 3% higher micro F1 compared to the results of the original paper. This could be a random computing difference between different computation serves. Note that the original paper did not report the accuracy in the LUN training set and LUN test set. Note that the running time is around 2 hours.

Table 2: The performance of CompareNet with different datasets in 2-way classification.

Dataset	Micro		Macro	
	F1	Prec	Recall	F1
LUN development set	96.68	96.84	96.3	96.55
LUN test set	86.67	86.68	86.67	86.67
SLN test set	91.85	91.85	91.86	91.85
SLN test set by Hu et al. (2021)	89.17	89.82	89.17	89.12

The performance of CompareNet with different datasets in 4-way classification is shown in Table 3. We find that our re-implemented model has 8% lower micro F1 compared to the results of the original paper, and the situation is similar for the other three accuracy criteria. This might be a random computing difference between different computation serves, but it is more likely that we had re-implemented all the required hyperparameters just as the same with the author's code. We will look into this issue after the milestone. Note that the original paper did not report the accuracy of the LUN training set. Note that the running time is around 1.5 hours.

Table 3: The performance of CompareNet with different datasets in 4-way classification.

Dataset	Micro		Macro	
	F1	Prec	Recall	F1
LUN training set	95.66	95.58	95.64	95.60
LUN test set	61.30	64.96	61.30	60.07
LUN test set by Hu et al. (2021)	69.05	72.94	69.04	68.26

4.3 Results of LSTM

The performance of LSTM, compared with the CompareNet model, in the LUN validation dataset in 2-way classification is shown in Table 4. And the corresponding results in the LUN test dataset and SLN test dataset are shown in Table 5 and 6, respectively.

Firstly, we found that according to Table 4 and Table 5, the Micro F1 of CompareNet is only 2.2% higher than the counterpart of the LSTM (10 epochs) model. The increasing percentage is similar for the other three criteria.

To some degree, LSTM is an out-of-date model in NLP. And we did not do full fine-tuning to improve the performance of LSTM. Still, the proposed method only has as small as 2.2% margin over LSTM. We can conclude that actually the LUN training dataset and LUN test dataset are not very suitable to fully demonstrate the advantage of the proposed model – making use of outside knowledge to increase accuracy. Unfortunately, the authors chose to not report the results of the comparison in these two datasets for the 2-way classification tasks in their paper. If we were the authors, we would choose some more challenging datasets which also happen to have room for CompareNet to function.

Looking at Table 6, we found that CompareNet has a 17% increase (difference in absolute values) compared to LSTM (10 epochs). Thus, in the SLN dataset, the CompareNet model has a huge advantage over the LSTM model. But still, we can't be sure if the authors intensively choose this SLN dataset to showcase the advantage of CompareNet. Also, the model is trained on the LUN training set, but we found that the SLN dataset has a totally different distribution compared to the LUN training set (from different newspapers). That is one of the reasons that LSTM performs worse. Usually, the training set and testing set should have the same distribution. Thus, in the SLN testing set, it is actually unfair to the baseline model LSTM, which is trained so well in the training set but does not have the ability to transfer to a dataset with new feature distributions of the text.

If we were the reviewers, we would recommend that the authors do their experiments on several popular datasets in the fake news detection community, have a K-fold validation, and take the average for the final results. Then the results would be more persuasive and reliable.

After we first tried the LSTM with 20 epochs, we found that it might be overfitted, because it performs excellently in the LUN validation dataset, but not so well in the LUN test set. So we decreased the training epochs and found that although its performance in the validation set decreased by 1%, it performs much better in both the LUN test set and the SLN test set.

We noticed that in Table 6, the accuracy of our implementation of LSTM (10 epochs) is 9 % lower than the LSTM results reported in Hu et al. (2021). The reason can be that: 1) our implementing details of LSTM are different from the counterpart of the authors. Since the authors did not disclose the details of the implementation of LSTM and did not reply to our emails about requesting the source code of the baseline method LSTM, we could only build this LSTM model to the best of our knowledge. 2) We did not do preprocessing and did not finetune enough for the LSTM model. If we do these, the results in the SLN test dataset can be better.

Table 4: The performance of LSTM, compared with CompareNet model, in LUN validation dataset in 2-way classification.

Model	Micro		Macro	
	F1	Prec	Recall	F1
CompareNet	96.68	96.84	96.3	96.55
LSTM (20 epochs)	95.34	95.07	95.39	95.22
LSTM (10 epochs)	94.61	94.26	94.78	94.48

Note: results of CompareNet model in LUN validation set by Hu et al. (2021) are not given by the authors.

Table 5: The performance of LSTM, compared with CompareNet model, in LUN test dataset in 2-way classification.

Model	Micro		Macro	
	F1	Prec	Recall	F1
CompareNet	86.67	86.68	86.67	86.67
LSTM (20 epochs)	83.39	84.20	83.38	83.29
LSTM (10 epochs)	85.19	86.03	85.19	85.10

Note: 2-way classification results of CompareNet model in LUN test set by Hu et al. (2021) are not given by the authors.

Table 6: The performance of LSTM, compared with CompareNet model, in SLN test dataset in 2-way classification.

Model	Micro		Macro	
	F1	Prec	Recall	F1
CompareNet	89.17	89.82	89.17	89.12
LSTM (20 epochs)	69.17	69.38	69.17	69.08
LSTM (10 epochs)	72.22	73.92	72.22	71.72
LSTM reported in Hu et al. (2021)	81.11	82.12	81.11	80.96

In addition, we demonstrate the expressive ability of LSTM by visualizing the hidden layer of LSTM. The hidden layer vector refers to the output vector of LSTM layer, which has 150 dimensions. We use SNE module [2] to transfer the 150 dimensions vector to 2 dimensions vector, and plot them. In Fig. 2, sub-plot a to c show the hidden state of the LSTM model with 20 epochs in LUN validation set, LUN test set and SLN test set; sub-plot d to f show the counterpart with 10 epochs. The axes are arbitrary units. Each point in the figure represents a news text sample, where the blue point means the ground truth of the sample is "satirical", and the red point "trusted".

According to Fig. 2, LSTM can clearly divide the features of samples in the LUN validation set and LUN test set, but not in SLN test set. Also, the sample numbers in SLN dataset is much smaller than the other two datasets. Moreover, we found that LSTM with 20 epochs is overfitted to some degree. After we decrease the epoch numbers, we can see that in Fig. 2 subplot e, there are fewer red points messed up in the group of blue points, compared with subplot b.

Note that considering the complexity of the proposed method, and the difficulty to take out the hidden states, we did not visualize its hidden states.

4.4 Qualitative example

One news example and the corresponding outside knowledge text are shown in Fig. 3. The entity "Mamograms" is recognized in the news text, and is used to retrieve the knowledge in Wikipedia to help increase the accuracy of fake news detection, and generate the correct class: "Hoax".

[2] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

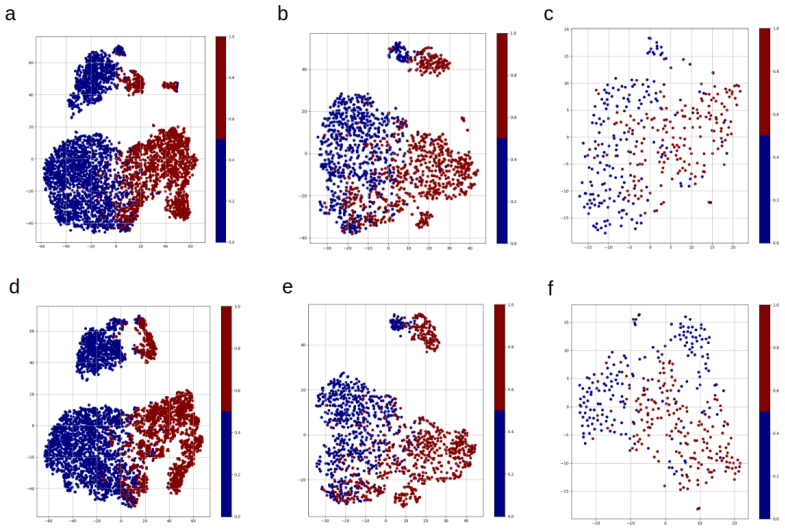

Figure 2: The hidden state of the news samples by LSTM: a-c the LSTM model with 20 epochs in LUN validation set, LUN test set and SLN test set; d-f the counterpart with 10 epochs.

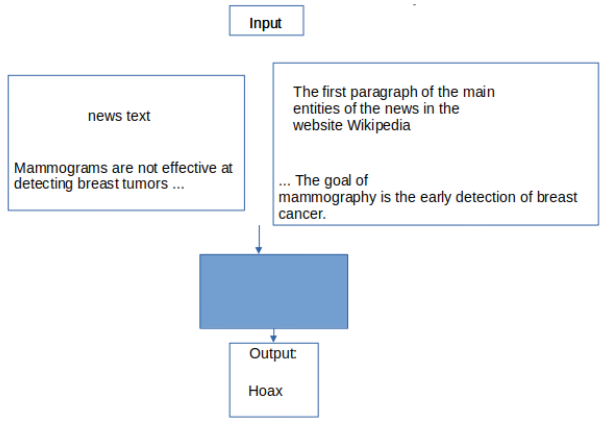

Figure 3: One news example and the corresponding outside knowledge text

5 Conclusion

In this project, we reproduced the results of the proposed model in Hu et al. (2021), and verified the validity of their results. CompareNet has better performance in SLN dataset for 2-way classification, and in LUN test set for 4-way classification. Secondly, we also found the shortcomings of the paper: in LUN validation dataset and LUN test dataset, the proposed method only has as small as 2.2% margin over LSTM for the 2-way classification task, which means these two datasets are not suitable for demonstrating the advantages of CompareNet.

6 Limitations and future directions

One of the limitations of our project is that we did not use text preprocessing for our baseline method (LSTM-GloVe), also we did not conduct full fine-tuning (we only tuned the training epochs) for the baseline method. Because we found that the current baseline model is already good at the dataset. Further processing might not increase the performance too much, and can lead to over-fitting for the unseen dataset (i.e. SLN) with different distributions.

For future works, we plan to find a better dataset that can better makes use of the advantages of CompareNet, and obtain the results for comparison. And we would encompass stronger baseline method, such as BERT-based neural networks.

References

Pritika Bahad, Preeti Saxena, and Raj Kamal. 2019. Fake news detection using bi-directional lstm-recurrent neural network. *Procedia Computer Science*, 165:74–82.

Michał Choraś, Konstantinos Demestichas, Agata Giełczyk, Álvaro Herrero, Paweł Ksieniewicz, Konstantina Remoundou, Daniel Urda, and Michał Woźniak. 2021. Advanced machine learning techniques for fake news (online disinformation) detection: A systematic mapping study. *Applied Soft Computing*, 101:107050.

Nadia K Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for

finding fake news. *Proceedings of the association for information science and technology*, 52(1):1–4.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Linmei Hu, Tianchi Yang, Luhao Zhang, Wanjun Zhong, Duyu Tang, Chuan Shi, Nan Duan, and Ming Zhou. 2021. Compare to the knowledge: Graph neural fake news detection with external knowledge. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 754–763.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Ray Oshikawa, Jing Qian, and William Yang Wang. 2020. A survey on natural language processing for fake news detection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6086–6093.

Jeff Z Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. 2018. Content based fake news detection using knowledge graphs. In *17th International Semantic Web Conference, ISWC 2018*, pages 669–683. Springer Verlag.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Improved semantic-aware network embedding with fine-grained word alignment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1829–1838.

Raghuram Mandyam Annasamy Vaibhav and Eduard Hovy. 2019. Do sentence interactions matter? leveraging sentence level representations for fake news classification. *EMNLP-IJCNLP 2019*, page 134.

Xinyi Zhou and Reza Zafarani. 2020. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53(5):1–40.