Course Project Report for Introduction to Machine Learning
Fake News Detection

Chuizheng Meng[*]

06-11-2017

Table of Contents

[*]Student ID: 2013010952, Email: mengcz95thu@gmail.com, Phone: 17600739085.

1 Introduction

The problem stems from the competition Fake News Challenge[1]. The goal of the competition is to use artificial intelligence to improve the automation of detecting whether a message is a rumor, and to reduce the burden on human examiners. This challenge does not intend to automate all testing steps once and for all. Instead, try to complete one of the most important steps - Stance Detection, which detects the attitude of different media to the same topic. Obviously, if their attitudes are very different or negative attitudes, the credibility of the message will be suspicious. This step will become an important part of the future automatic rumor detection system.

2 Related Work

The problem that needs to be solved this time belongs to the stance detection problem, that is, two paragraphs of text are given to judge whether the opinions expressed by them are consistent. Similar problems have been proposed several times before and as topics of some competitions. For example, SemEval-Task 6 in 2016 requires competitors to judge whether the given twitter text supports the subject to which it belongs, and to complete the tasks under supervision/unsupervised settings[1]. In this game, the model of deep learning has been widely applied. The two teams winning the first and second prizes used RNN[2] and CNN[3] to solve the problem.

Stance detection is a text classification problem, and many papers have tried solve it with RNN, such as bi-directional RNN[4], a mixed model of CNN and RNN[5] and RNN with attention mechanism[6][7].

Compared with traditional feature engineering based methods, deep learning models require a mapping from words to vector space (word embedding). Word2vec [8] from Mikolov, GloVe[9] from Pennington, and Fasttext[10]from Bojanowski are some successful word embedding models.

3 Tasks

The tasks that the experiment needs to accomplish are: building a model based on the annotated data, and judging the relationship between the message body and the message title, including ”agrees”, ”disagrees”, ”discusses” and ”unrelated”:

- Input: a title and a message body (may come from the same message or two different messages).
- Output: the relationship between the message body and the message title:
  - Agrees: the message body agrees with the title;
  - Disagrees: the message body disagrees with the title;
  - Discusses: the message body and the title discuss the same topic, but not clear stance is shown;
  - Unrelated: the message body and the title don’t have the same topic.

Annotated data is provided[2], including about 50k messages, composed of titles, bodies and their relations.

4 Design of Models

4.1 Baseline

We use the baseline model provided by the competition[3]. The model manually extracts features and use a Gradient Tree Boosting Classifier to finish the classification.

Extracted features are listed as follows:

- word overlap features: Jaccard similarity between the vocabulary of a title and a body[11].

[1]http://www.fakenewschallenge.org/
[2]https://github.com/FakeNewsChallenge/fnc-1
[3]https://github.com/FakeNewsChallenge/fnc-1-baseline

- refuting features: a 0-1 vector showing the existence of refuting words. Refuting words are: 'fake', 'fraud', 'hoax', 'false', 'deny', 'denies', 'not', 'despite', 'nope', 'doubt', 'doubts', 'bogus', 'debunk', 'pranks', 'retract'. If some word exists, the corresponding element is 1, else is 0.
- polarity features: The "polarity" of the attitude expressed by the title and the body. The calculation of the "polarity" of a piece of text is to count the parity of the number of occurrences of the aforementioned refuting words. An even number of negatives reflects a positive attitude, and an odd number of negatives still shows a negative attitude.
- binary co-occurrence: The sum of the number of times the vocabulary in the title appears in the body.
- binary co-occurrence stops: The sum of the number of times the vocabulary in the title appears in the body with stop words (some words with no actual meaning, e.g. a, the, is...) removed.
- count grams: n con The number of times n consecutive letters/n consecutive words appear in the body in the title. For letters, the model chooses n as 2,4,8,16; for words, the model chooses n as 2,3,4,5,6.

The classifier uses the decision tree as the base classifier and uses the Gradient Boosting method [12] as the enhanced learning method. The Gradient Boosting method is similar to the Boosting method execution process, which is serial execution, but the former uses the loss function for each update. Train the new base classifier on the gradient of the classification result instead of the real result, and determine a step size that minimizes the loss result after the update. The updated result of this round is the result of the previous round + step length * this round base classification The result of Gradient Boosting is that the prediction ability is strong, and the disadvantage is that it is not convenient to parallelize.

4.2 Word Embedding

Most machine learning methods are better at processing numeric types of input data, so it is necessary to use word embedding to map words in the input text to n-dimensional vectors. A good word embedding should map semantically similar words to similar positions in n-dimensional spaces. Position, and the positional relationship between word vectors should be able to reflect the relationship between the original vocabulary. Considering the little amount of available data in this experiment, it is better to use the pre-trained word embedding directly.

Considering the size of each pre-training model, the feature dimension and the size of the dataset used, this experiment selects the GLoVe model that Stanford NLP Group trains on the twitter dataset[4] as word embedding. The model contains 1.2 million words and the result is a set of 25/50/100/200-dimensional word vectors.

4.3 Loss Function

We use the cross entropy function as the loss function:

$$L(X,Y) = -\sum_n \sum_i y_i^{(n)} \log\left(f(x^{(n)})_i\right)$$

$y_i^{(n)}$ is a binary value, showing whether the nth sample belongs to category i, $f(x^{(n)})_i$ is the predicted probability of that the nth sample belongs to category i. $e^{-L(X,Y)}$ is the predicted probability of that X belongs to Y, and minimizing the loss function is equivalent to finish a maximum likelihood estimation.

[4]http://nlp.stanford.edu/data/ glove.twitter.27B.zip

4.4 Models Based on CNN

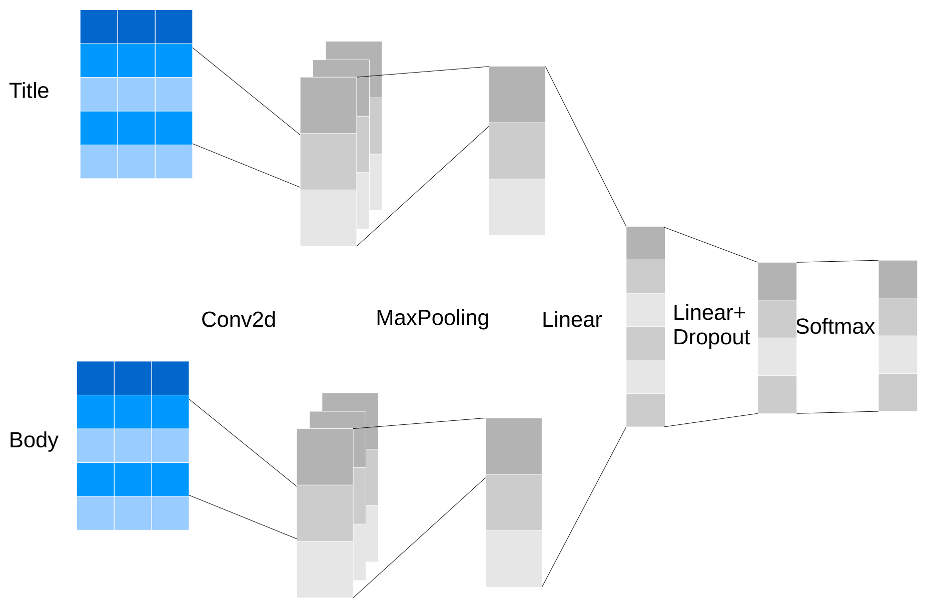

Figure 1: The CNN model used for classification, including convolution layers, pooling layers, linear layers and softmax layers

CNN(convolutional neural network) can extract local features and reduce the number of model parameters. In the text classification task, the convolution kernel can capture the vocabulary relationship between contexts to some extent. The main parts of the CNN model used in the experiment includes:

- word2vec: Convert the title/body text to the input matrix through the pre-trained word2vec model, the size is the text length multiplied by word vector dimension.
- Convolutional layer: The convolution kernel size is the window size multiplied by the word vector dimension, stride is 1, no padding. The title text convolution kernel number is the same as the word vector dimension. For the title text window size is 3, for the body window size is 5.
- Pooling layer: MaxPooling, output the maximum of the output of each convolution kernel.
- Linear layer: taking the concatenation of title hidden vectors and body hidden vectors as input.
- Linear+Dropout: get the output of 4 categories. Dropout can help reduce the chance of overfitting[13].
- Softmax: output the probabilities of 4 categories.

4.5 Models Based on RNN

RNN(recurrent neural network) is capable of ”memorizing” previous input data and is suitable for serial data, thus models based on RNN is widely used in natural language processing. The memory of RNN can fade out if the input sequence is too long, but LSTM[14] and GRU[15] both can solve the problem. All RNN units used in our models are LSTM/GRU.

The main differences of RNN models listed lie in the RNN part. The classifier is the same: 1 linear network with 1 hidden layer and a softmax function. Dropout is used in the hidden layer to reduce overfitting.

4.5.1 Bi-directional RNNs

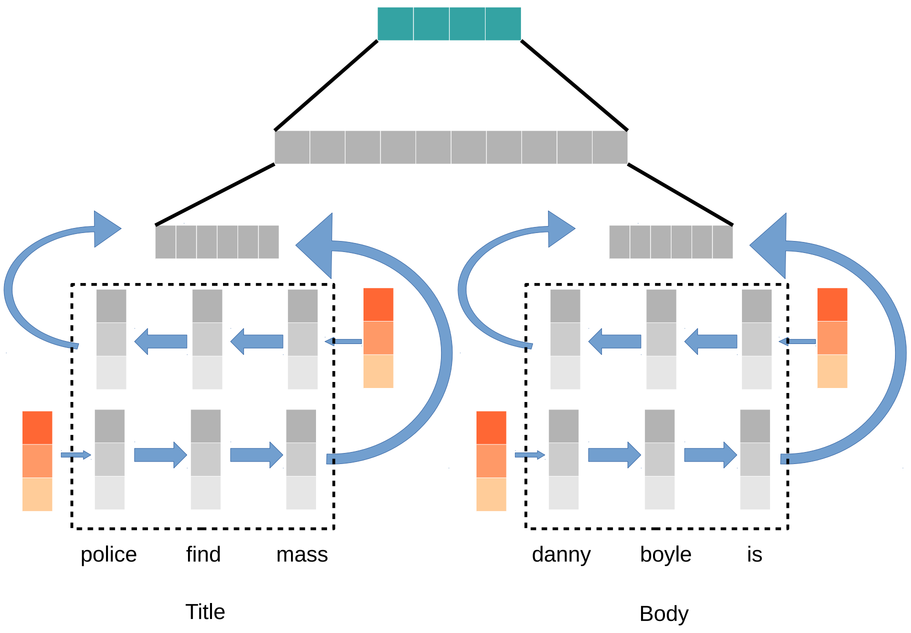

Figure 2: Unconditional Bidirectional RNN

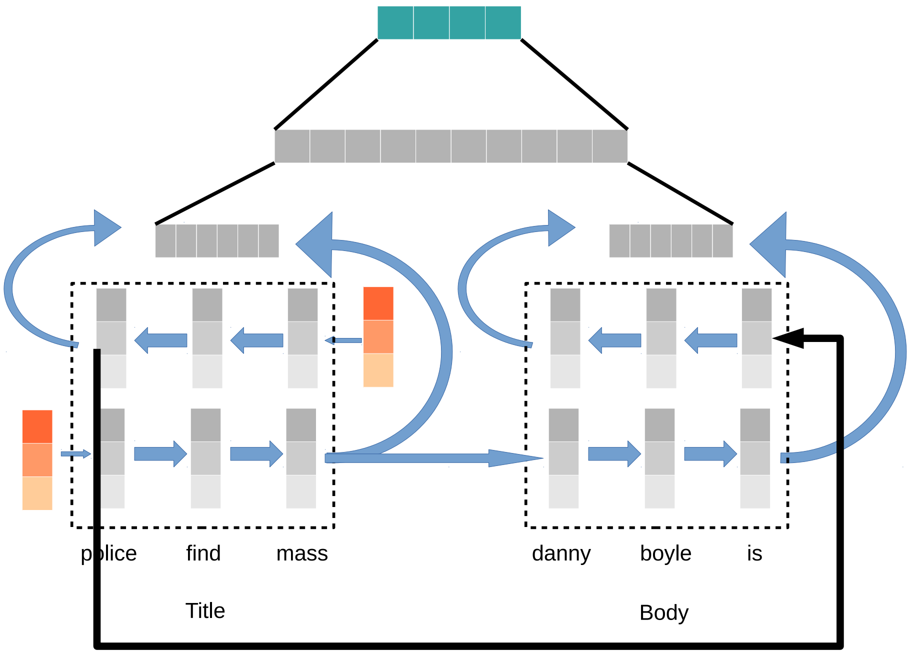

Figure 3: Conditional Bidirectional RNN

The bidirectional RNN model inputs the input sequences in order and in reverse order. The final vectors obtained in the two sequences are combined as the output vector of the RNN. The RNN output vectors of the title text and the body text are concatenated and sent to the classification network for classification.

Two different bidirectional RNN models can be constructed based on the type of initial vector used by the RNN that processes the body text:

- Unconditional Bidirectional RNN: 0-vector is used as the initial hidden vector.
- Conditional Bidirectional RNN: the hidden vector from the final step of RNN processing titles is used as the initial hidden vector, which enhances connections between titles and body texts and may have better performance.

4.5.2 RNN with Attention Mechanism

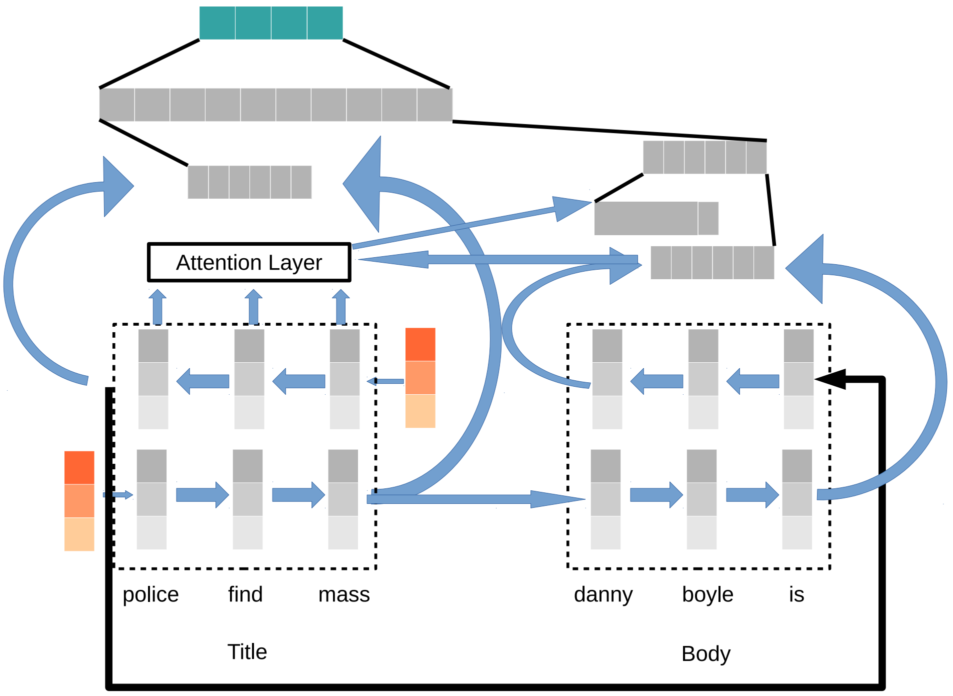

Figure 4: Type-1 Conditional Bidirectional RNN with Attention Mechanism

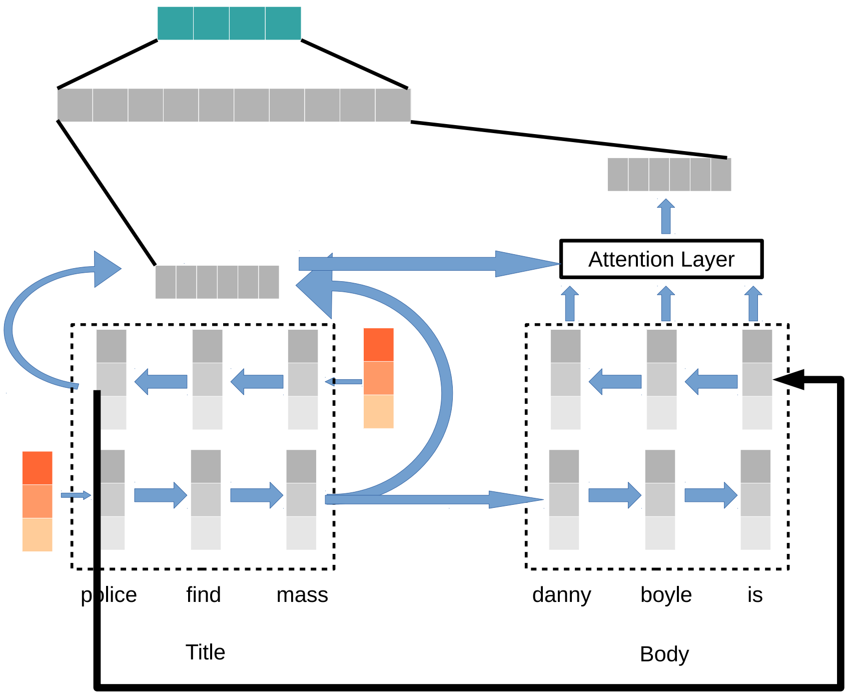

Figure 5: Type-2 Conditional Bidirectional RNN with Attention Mechanism

There are many variants based on the RNN model, one of which introduces the so-called attention mechanism [6]. The principle of the Attention mechanism is to assign different weights to the output vectors of all units according to the output vector of the last unit of the RNN. The unit output vector is weighted and concatenated with the output vector of the last unit, and output after a nonlinear layer. According to previous research and experimental verification, the better weight calculation method is: The output of the last unit is h_T and each The output of the unit h_t is bilinearly transformed. The result of $h_T^T W h_t + b$ is processed by the softmax function as the weight of h_t. The function of the nonlinear layer can be selected as tanh.

Specific to the problem of this experiment, since the title and body are processed in 2 RNNs, the output vector of the body RNN can be used to make an attention with all the output vectors of the header RNN, and the result is concatenated with the last output vector of the title RNN and enters the classification network.

The literature [7] simplifies the attention model. The main difference is that the nonlinear layer is eliminated and the weighted result is directly used as the output vector. The simplified model has also been modified and applied to this experiment.

4.6 Mixture Models

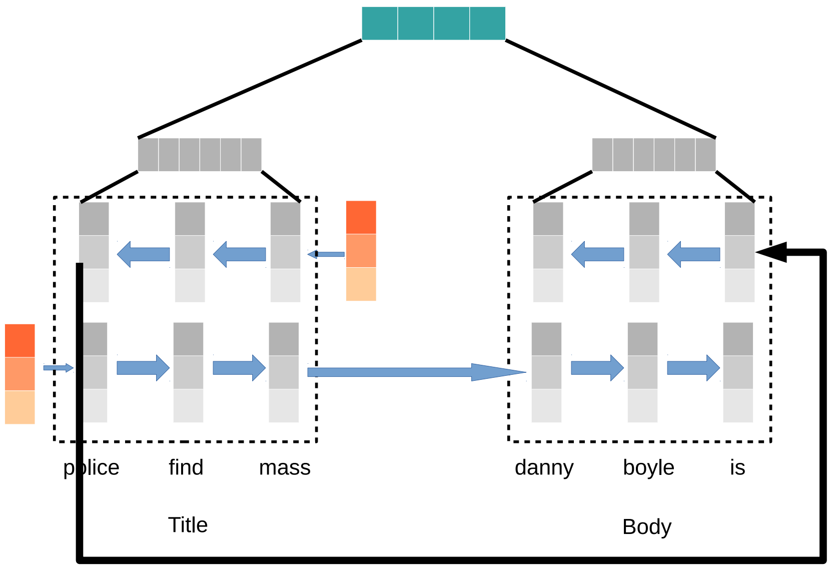

Figure 6: CNN-RNN Mixture Model

This model combines RNN and CNN[5]:

- ”convolution” layer: use the output bidirectional RNN at each step as the output of ”convolution”.
- non-linear layer: a linear layer with tanh as the activation function.
- pooling layer: maxpooling along the step dimension.
- classifier: a linear layer and a softmax function.

5 Experiments

5.1 Dataset

The dataset is from Fake News Challenge, including about 50k messages:

- train_bodies.csv: text bodies and corresponding IDs;
- train_stances.csv: stance, titles and coresponding IDs.

In the experiment, 80% of the data was used as the training set and 20% of the data was used as the test set. For the Baseline method, 10% of the training set was selected as the verification set for 10-fold cross-validation; for the neural network model, it was run once. It takes a long time and the amount of data is limited, so you can use all training set data directly for network training.

After a simple analysis, the data provided is very unbalanced. Up to 73.13% of the examples are unrelated, while the examples of discus, agree, disagree only account for 17.83%, 7.36% and 1.68%. If used directly, the classification will be made. The result is a serious deviation, and the classifier can easily judge all the samples as unrelated.

Common methods for dealing with unbalanced data include over sampling, under sampling, and artificially increasing the weight of the lower category of the loss function. Over sampling has put back all the data of the non-maximum type, until all categories The number of samples is equal; under sampling does not put back all the data of the non-minimum type, so that the number of samples of all categories is equal; for some classification problems commonly used loss function (such as cross entropy function), It is possible to artificially adjust the weights of different categories. Obviously, the data set obtained by using under sampling in experiments is too small, which is not conducive to the processing of deep learning methods, and the method of adjusting weights cannot solve the problem that SGD class optimization method is likely to appear when dividing batch. A situation in which a very small percentage does not appear in a batch, resulting in a decrease in the optimization speed, so the over sampling is used to preprocess the data. It should be noted that over sampling must be performed on the training set after dividing the training set and the test set. Otherwise, the information of the test set category distribution will be "leaked" into the training set, making the model perform better on the test set preferences.

Another problem with the data is that the length of the title and the length of the text differ greatly. The maximum length of the title is <50, and the maximum length of the text is >2000. When humans judge the relationship between the title and the body, they often read the beginning of the body. The long text is probably not meaningful for the correct classification of neural network models. For the purpose of speeding up training and saving memory, only the beginning of the body is used when using the neural network model. In order to verify the hypothesis, the experiment compares the classification effects when used with texts of different lengths.

5.2 Metrics

Metrics used are listed as follows:

- Accuracy: $\frac{\text{\# samples correctly classified}}{\text{\# all samples}}$.
- Evaluation method of the contest: for each example, if the related (including 3 categories other than unrelated)/unrelated is successfully distinguished, 0.25 points can be obtained. If it is classified correctly and belongs to 3 categories other than unrelated, then the model can continue to get 0.75 points.

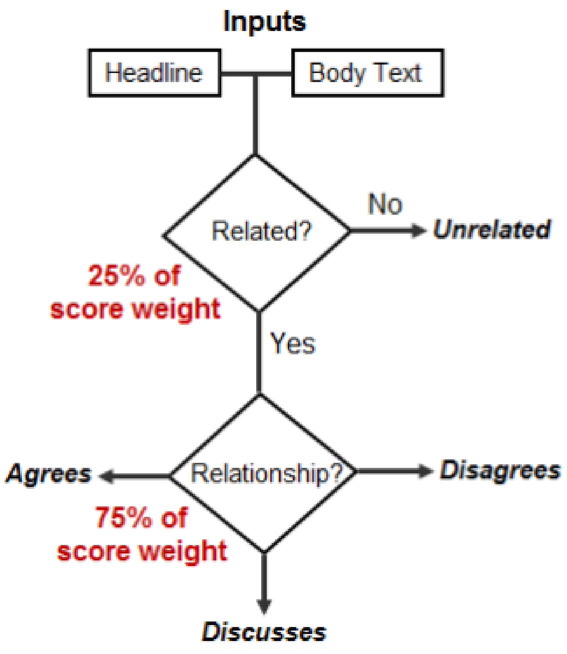

Figure 7: Metric of Fake News Challenge

- Decompose the problem to multiple binary classification problems, and evaluate precision, recall and F1-score on each category

5.3 Implementation

Baseline is from the contest and is based on the GBDT classifier from scikit-learn. The rest of the neural network model is based on the PyTorch framework [5]. The PyTorch framework provides common CNN, RNN units, as well as auto gradient solving, GPU acceleration and multiple optimizers. It is convenient for users to build flexible network structure and use GPU to accelerate training.

The parameter settings are shown in the appendix. This experiment is not intended to explore the influence of many model hyperparameters on the results, so only the differences of a few hyperparameters and the different model structures are compared to the classification results.

Baseline model running platform hardware configuration is Intel i7-6600u@3.0GHz, 16GB RAM, software configuration is Ubuntu 16.04 LTS Desktop, Python 3.6.0. Neural network model running platform is p2.xlarge instance of Amazon Web Services, hardware configuration is 4 Core CPU, 64GB RAM, Nvidia K80 GPU (graphics memory 12GB) x1, software configuration for Ubuntu 16.04 LTS Server, Python 3.6.0.

5.4 Results

5.4.1 Effect of Dimensions of Word Vectors

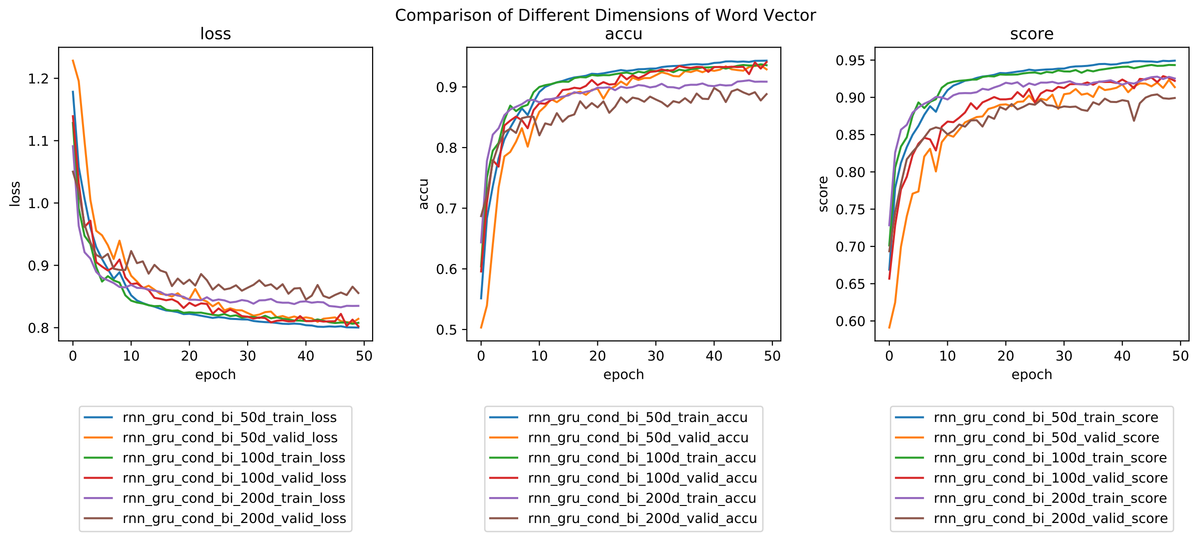

Figure 8: Effect of Dimensions of Word Vectors

Table 1: Performance on Test Set of Different Word Embedding Dimensions

Dim of Word Vector	Accuracy	Score
50	0.9356	0.9235
100	**0.9406**	**0.9285**
200	0.8872	0.9040

The GLoVe model provides word vectors of various dimensions. If the word vector dimension is too low, the relationship between words will not be fully reflected. If the word vector dimension is too high, the number of model parameters will be greatly increased, and over-fitting will occur, and the training speed will drop. In the experiment, the word vectors of lengths 50, 100 and 200 were selected to compare their effects on the Conditional Bidirectional RNN model. It can be found that the word vectors of length 50 and 100 behave similarly and are better than the words of length 200. Vector. The reason may be that the amount of data used for training is not large, and the higher expression of

[5]http://pytorch.org/

high-dimensional word vectors is difficult to express on small-scale data sets, but it is difficult to optimize.

For the consideration, follow-up experiments select 50-dimensional word vectors.

5.4.2 Effect of Different Word Embedding Training Methods

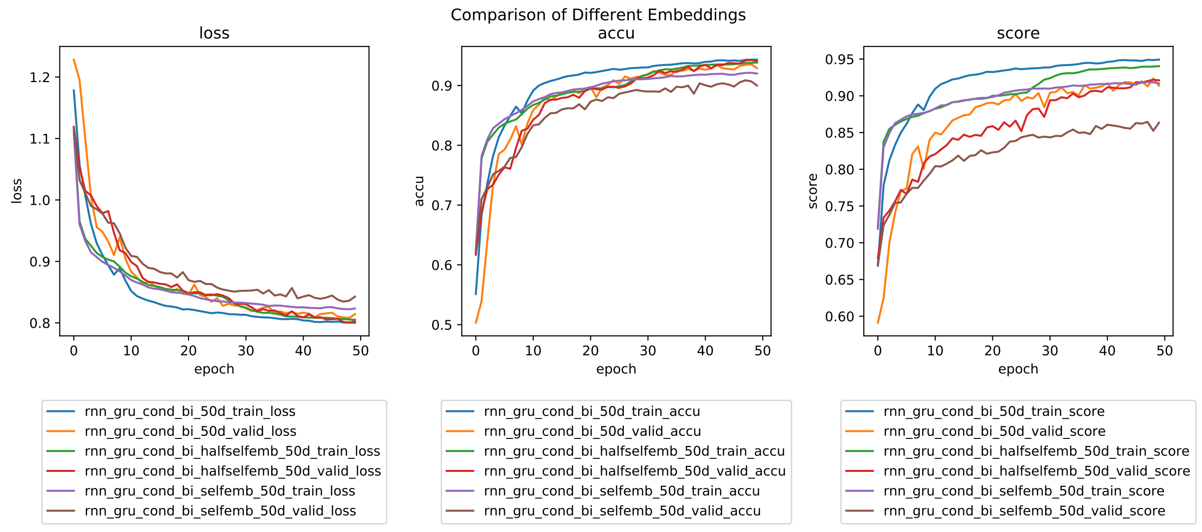

Figure 9: Effect of Different Word Embedding Training Methods

Table 2: Performance on Test Set of Different Word Embedding Training Methods

Embedding	**Accuracy**	**Score**
GLoVe Fixed	0.9356	**0.9235**
GLoVe Not Fixed	**0.9426**	0.9219
Self Embedding	0.9087	0.8645

This experiment compares the training methods of three word vectors: directly use the pre-trained model (GLoVe Fixed), fine tune (GLoVe Not Fixed) and self-training (Self Embedding) based on the pre-training model. The pre-training model works the best. Fine tuning the word vector based on the pre-training model does not improve the result, and the result of training the word vector from zero is the worst. The reason is that the amount of text used for training is too small to correctly reflect the relationship between words.

5.4.3 Effect of Different Lengths of Body Text

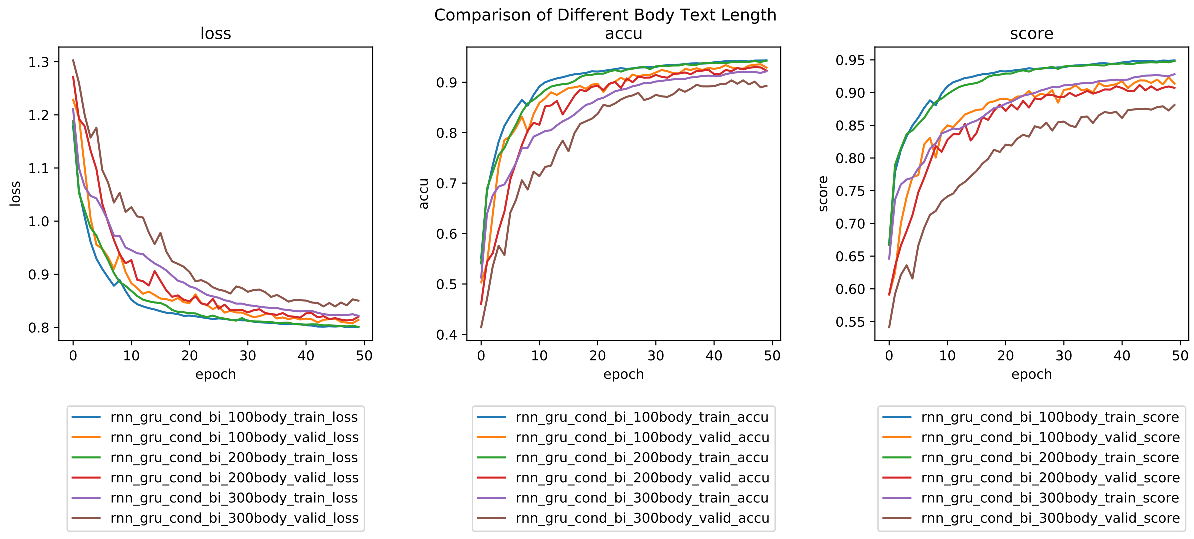

Figure 10: Effect of Different Lengths of Body Text

Table 3: Performance on Test Set of Different Lengths of Body Text

Body Text Length	**Accuracy**	**Score**
100	**0.9356**	**0.9235**
200	0.9220	0.9116
300	0.8930	0.8811

If the length of the text interception is too short, the attitude of the text cannot be fully extracted. If the length of the interception is too long, the resource consumption and training time of the model will be greatly increased. The experiment compares the performance of Conditional Bidirectional RNN model with the lengths of the texts of 100, 200 and 300 respectively. It can be found that the models have similar performance with body text lengths of 100 and 200, and are better than that with body text length of 300. The reason may be that the length of the text makes the model more difficult to train, and the difference between the body and the title length can weaken the ability to extract title features.

In order to improve the training speed, the follow-up experiment selects the length of the body text as 100.

5.4.4 Effect of Different RNN Units

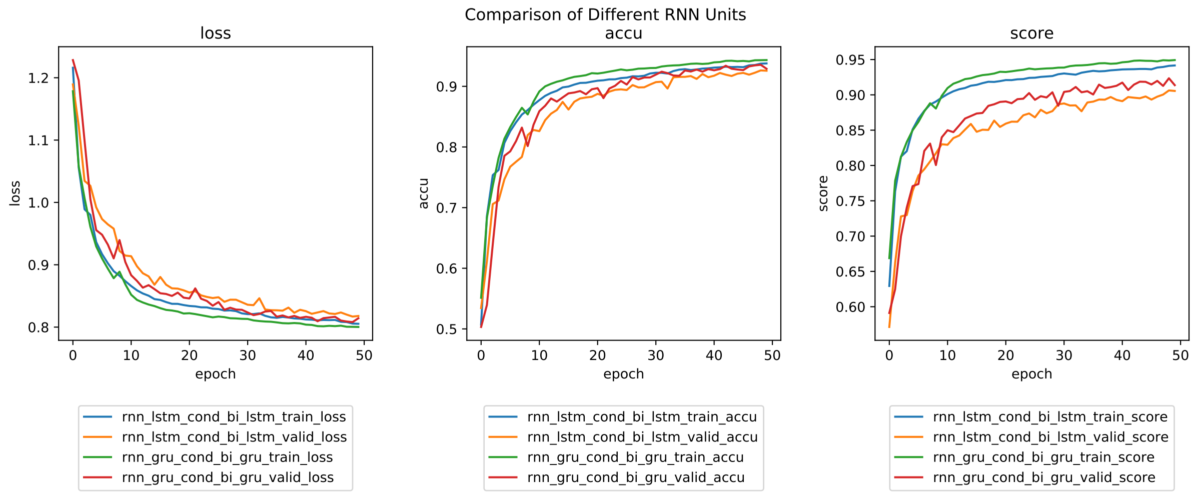

Figure 11: Effect of Different RNN Units

Table 4: Performance on Test Set of Different RNN Units

Type of RNN	Accuracy	Score
GRU	**0.9356**	**0.9235**
LSTM	0.9263	0.9063

The performance of the two RNN units on the test set is very close. In view of the fact that the GRU training speed is slightly faster, the subsequent experiments use GRU as the RNN unit.

5.4.5 Comparison of Models

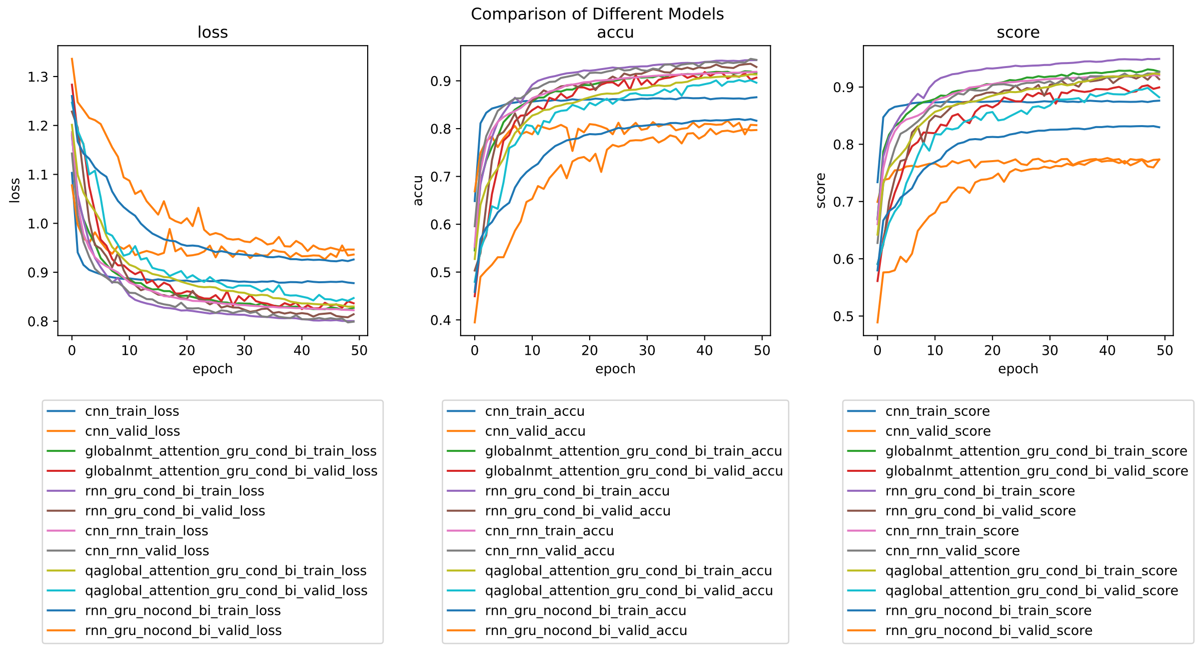

Figure 12: Comparison of Models

Table 5: Performance on Test Set of Models

Model	Train		Test	
	Accuracy	Score	Accuracy	Score
Baseline	-	-	0.8774	0.7953
CNN	0.8613	0.8741	0.8122	0.7747
Unconditional Bidirectional RNN	0.8175	0.8311	0.7860	0.7760
Conditional Bidirectional RNN	0.9431	0.9486	0.9356	0.9235
Attention RNN I	0.9200	0.9308	0.9156	0.9030
Attention RNN II	0.9112	0.9211	0.8982	0.8986
CNN-RNN	0.9188	0.9203	**0.9438**	**0.9262**

Table 6: Binary Classification on "Agree"

Model	Train			Test		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	-	-	-	0.6051	0.1549	0.2466
CNN	0.8946	0.8654	0.8797	0.4887	0.7203	0.5823
Unconditional Bidirectional RNN	0.8569	0.8143	0.8351	0.4718	0.6913	0.5608
Conditional Bidirectional RNN	0.9361	0.9186	0.9273	0.7507	0.829	0.7879
Attention RNN I	0.8865	0.9136	0.8999	0.5902	0.8391	0.693
Attention RNN II	0.8657	0.9406	0.9016	0.6181	**0.8609**	0.7196
CNN-RNN	0.9216	0.9048	0.9131	**0.8211**	0.7783	**0.7991**

Table 7: Binary Classification on "Disagree"

Model	Train			Test		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	-	-	-	0.25	0.0185	0.0345
CNN	0.9186	0.9677	0.9425	0.4595	0.7553	0.5714
Unconditional Bidirectional RNN	0.9141	0.8065	0.857	0.3746	0.6596	0.4778
Conditional Bidirectional RNN	0.95	0.9188	0.9342	**0.6857**	**0.766**	**0.7236**
Attention RNN I	0.9412	0.8598	0.8987	0.6536	0.6223	0.6376
Attention RNN II	0.9691	0.8113	0.8832	0.6269	0.6702	0.6478
CNN-RNN	0.9343	0.8032	0.8638	0.6413	0.6277	0.6344

Table 8: Binary Classification on "Discuss"

Model	Train			Test		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	-	-	-	0.6608	0.8483	0.7429
CNN	0.8853	0.7601	0.818	0.6859	0.6957	0.6907
Unconditional Bidirectional RNN	0.8057	0.8071	0.8064	0.6551	0.7183	0.6852
Conditional Bidirectional RNN	0.9366	0.968	0.952	0.8644	0.9206	0.8916
Attention RNN I	0.9041	0.9517	0.9273	0.8286	0.8931	0.8596
Attention RNN II	0.9045	0.954	0.9286	0.7937	0.9035	0.8451
CNN-RNN	0.8909	0.9728	0.93	**0.8725**	**0.9245**	**0.8978**

Table 9: Binary Classification on "Unrelated"

Model	**Train**			**Test**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	-	-	-	0.9564	**0.9849**	**0.9704**
CNN	0.7603	0.8521	0.8036	0.9104	0.8513	0.8798
Unconditional Bidirectional RNN	0.7243	0.8443	0.7797	0.9124	0.8325	0.8706
Conditional Bidirectional RNN	0.95	0.9671	0.9585	**0.9822**	0.9537	0.9678
Attention RNN I	0.9364	0.9394	0.9379	0.9805	0.9237	0.9513
Attention RNN II	0.919	0.9388	0.9288	0.978	0.9063	0.9408
CNN-RNN	0.9182	0.9793	0.9478	0.9747	0.9656	0.9701

The experiment compares the effects of the aforementioned model on the test set, summarized as follows:

Although the Baseline model has a general overall performance on the test set, it works best when related/unrelated classifications, because many of the features in the Baseline model directly compare the similarity of the text from the word/phrase level, so it works well. Judging the relevance of the two texts. However, the Baseline model has very few characteristics about attitudes, so it performs poorly when distinguishing from agree/disagree/discuss.

The CNN model and the Unconditional Bidirectional RNN model perform poorly, even worse than the Baseline model. The reason is that only the part of the two models that deal with the title text and the body text separately, the connection between the title and the body can only be based on the last layer. The classifier to extract, greatly affects the classification effect.

The Conditional Bidirectional RNN model establishes the connection between the title and the body by using the output of the RNN that processes the title as the input to the RNN that processes the body, thus achieving a higher degree of precision on the test set.

The performance of the RNN model with the Attention mechanism has not improved or even decreased compared with the conventional RNN. The observation results show that the model performs poorly on the test set when predicting the related relationship, which may be the effect of over sampling.

The CNN-RNN model performs best on the test set. The introduction of the convolutional layer enables the model to make full use of the information of each step in the RNN, which improves the model's ability to capture text features.

Observing the classification performance of each model in four kinds of relationships, we can find that although over sampling and other methods can avoid the model from premature convergence to some extent, the imbalance of the original data still has a great impact on the classification. In the original data. When there are few types of agree and disagree, the performance of each model on the test set is significantly worse than that of the training set; when categorizing on a little more dicuss, the difference between the performance of the test set and the performance of the training set will be reduced; When the most unrelated types are classified, the test set performs even slightly better than the training set.

6 Conclusion

This experiment attempts to use different depth learning models on the stance detection problem. The Conditional Bidirectional RNN model and the CNN-RNN model have achieved good classification results. The deep learning model does not require manual extraction of features, and can also be compared on this issue. Good results, but the flaw is that the model is poorly interpretable. It should also be noted that the traditional method based on feature engineering can also achieve good results on specific problems (such as unrelated classification) at the time of feature selection. Both the feature engineering method and the deep learning model require an in-depth understanding of the problem itself in order to extract the appropriate features/construct a reasonable network structure to better solve the problem.

References

[1] Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. *Proceedings of SemEval*, 16, 2016.

[2] Guido Zarrella and Amy Marsh. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*, 2016.

[3] Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. *Proceedings of SemEval*, pages 384–388, 2016.

[4] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[5] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[6] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[7] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[11] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. Using of jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, page 6, 2013.

[12] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

A Running Configurations

Unless otherwise specified, the operating parameters used by the program default to the following:

Table 10: Default running parameters

Parameter	Meaning	Default Value
datapath	path to the folder having pre-trained word2vec models	"./data"
word2vecmodelfile	files of pre-trained word2vec models	"glove.twitter.27B.50d.txt"
fix_embedding	no fine-tuning of word embeddings	False
train_ratio	ratio of training data/whole dataset	0.8
remove_stopwords	remove stopwords	False
batch_size	the size of one batch	384
epoch	number of epoches to run	50
cuda	enable CUDA (needs NVIDIA GPUs)	False
lr	learning rate	0.001
weight_decay	L2 loss weight	1e-6
title_trunc	length of truncated titles	40
body_trunc	length of truncated body text	100
seed	random seed	0
wechat	enable sending notifications via WeChat	False
model	name of the model	compulsory
modelpath	folder to store checkpoints	compulsory
optimizer	optimization algorithm	"Adam"
selfemb	training word embeddings from scratch	None
resume	resume training from the latest checkpoint	False

B Commands

- Run a single model: python main.py model modelpath [other parameters]. Parameter settings can be found in the table 10 or using the python main.py -h command to view help.
- Run all the models in this experiment: Run python worker.py. This maintains a process pool of size 2, which can train 2 models in parallel.