Stance Detection with Attention Based Siamese Text CNN

Yikai Wang
Fudan University

Hanze Dong
Fudan University

Linyang He
Fudan University

Abstract

The Fake News Challenge is an open competition for finding some automatical methods to detect fake news. Our work is based on the first stage of this challenge: stance dectection. There are four different stances between the text body and headline: “agree”, “disagree”, “discuss” and “unrelated”. Our job is to implement text classification based on the bodytext given a certain headline. We developed both machine learning algorithms and Deep Learning algorithms for this task including CNN word embedding model and attention based seq2seq generation model. After lots of experiments across different model architectures and hyperparameters, we determined the optimal model is A-TCSN (Attention based Text Convolutional Siamese Neural Network).The very accuracy of this model (Score: 83.5%) is better than the best one in the competition already. Besides, we will introduce our future work with attend model which we have implemented part of.

1 Introduction

“Fake news” has rapidly become a catch-all term to discredit all kinds of stories recently. It is defined as a made-up story with an intention to deceive. It may cause social unrest in many times. It was even widely considered as a contributing factor to the outcome of the 2016 United States presidential election. Therefore, it’s quite important for us to be smarter at recognising such fabrication. Facebook and other online media outlets began to develop methodology for identifying fake news after people believed that Facebook had an effect on the outcome of the election which Mark Zuckerberg denied. Besides, he admitted that identifying fake news is quite hard, writing, “This is an area where I believe we must proceed very carefully though. Identifying the ‘truth’is complicated.”
Fake News Challenge stage one (FNC-1) was organized in response to this. FNC-1 is a public competition with the stated goal of exploring how artificial intelligence technologies could be leveraged to combat fake news. FNC-1 focuses on an important part of identifying fake news: stance detection. Stance detection involves estimating the relative perspective of two pieces of text relative to a topic, claim or issue. Stances include “agree”, “disagree”, “discuss” and “unrelated”. The goal of FNC-1 is to estimate the stance of an article relative to a headline. More specifically, the headline may agree with, disagree with, discuss, or be unrelated to the bodytext of the article.

In our model, we will seperate the task into two subtasks. The first subtask is to determine wheather the headline and the bodytext is related or not, which means we will consider the “agree”, “disagree” and “discuss” as one “related” stance. This is also what our binary classification implemention doing in the following sections. The second subtask is to determine how there are related. This is exactly the “multi-classication” in our project.

We will review others’ work. A lot of partipants took part in the fake news challenge competition. We choose the top 3 groups’ work as our related work which we will cover in the next section.

After reviewing their work, we find some disadvantages of those work including no deep learning usage and no denoise. We introduce our contribution to solve the problems. First, we will try many kinds of machine learning algorithms of classification(K-Nearest Neighbors, Random Forest, Support Vector Machine, etc.) to build our baseline model with a lot of features extraction.

Then we will focus on our proposed model. We use CNN Siamese neural network and attention based seq2seq model respetively. And at last we will make a comparision between these two models. Besides, for the evaluation, we will consider if the bodytext and the headline is related as 25% of the relative score, while consider how the bodytext and the headline is related (including "agree", "disagree" and "discuss") as 75% of that score as the Figure 1 shows.
Since we cannot get the real test data from the challenge, we split the dataset we can download from the website into training set and test set by ourselves. The ratio between the training set and test set is 0.95:0.05.

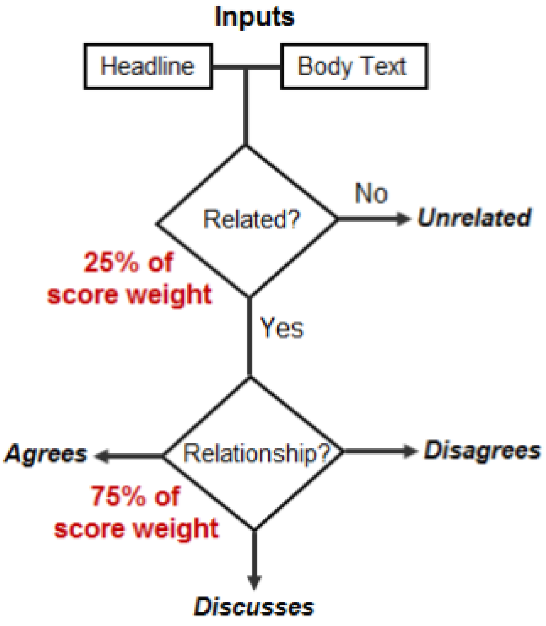

Figure 1: Relative Score

2 Background

2.1 Text Classification

Text classification is a classical problem in NLP, which aims to annotate documents into categaries. For example, we can sort text by topics: "News", "Entertainment", etc. Recent years, CNN has made a great progress in supervised learning, in particular, for text classification, the Text CNN can extract proper features automatically which result a superb results in classification (Kim, 2014). In our setting, different with Text Classification, we need to find the relative relationship between text bodies and headlines, so conventional Text CNN may not suitable in the problem. However, inspired by the power of CNN, we can consider using CNN to extract the features of either text bodies or titles, which include the local information of text content.

2.2 Text Similarity Evaluation

The similarity evaluation have an important role in NLP problems, for example, choice of best answer of a question can based on the evaluation of similarity between question and answer. Different with Text Classification problems, the Text Similarity Evaluation would input two texts rather than just one, which makes conventional model not work as it seperate each text in a model. Attempts by Deep Learning to solve the problem shows the efficiency and power(Yu et al., 2014). (Severyn and Moschitti, 2015) proposed a model to use symmetry CNN network to solve the problem, which shows a amazing performance in Q&A problems. In our setting, the first step, i.e. identity whether the candidate headline is related to the passage, is nearly the same with Text Similarity Evaluation, we can easily regard it as a binary classification. In the second step, as we would like to judge the fine-grain relation between titles and bodies, we can extend our model into a multi-class classification problem.

2.3 Seq2Seq Model

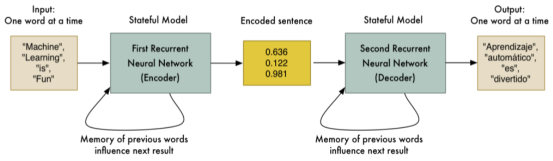

Figure 2: Seq2Seq

Seq2seq is a kind of encode-decode model as the Figure2 shows(Sutskever et al., 2014; **?**). It use one neural network as the encoder and use another as the decoder. Figure2 gives an example in machine translation which both of the encoder and decoder are LSTMs. In our specific task, we will use our bodytext as the input while a sentence generated as the output. That is, we will use the encoder to embedding the bodytext to a code, then we will use the decoder to decode it to a sentence. This generated sentence can be applied in the following steps where we will compare it with the headline to deternmine the relationship between them. Say we are in the i-th step of the decoding stage, we have the probability of the output is:

$$p(y_i|\{y_1, ..., y_{i-1}\}, c) = g(y_{i-1}, s_i, c) \quad (1)$$

And

$$s_i = f(s_{i-1}, y_{i-1}, c) \quad (2)$$

Here the c denotes the final state of embedding vector of the input, and the s_i denotes the hidden state for time i.

2.4 Attention

Attention or attend, is a method in some ways means align. Take the attention mechanism in seq2seq as the example. The naive seq2seq model just consider the final state of the input to decode. This might lead to bias. New approach with attention does not attempt to encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding. Comparing to the naive seq2seq model above, now we have the probability of the output in i-th time as:

$$p(y_i|\{y_1, ..., y_{i-1}\}, X) = g(y_{i-1}, s_i, c_i). \quad (3)$$

Here the s_i is as the same meaning as above. We can find that the c in the naive seq2seq change to c_i, where

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j. \quad (4)$$

Here the h_j is the j-th hidden state in the encoder end while the a_{ij} denotes the align weights between the i-th output and the j-th input. In this way, the output will not decided by the final state of the encoder but all the hidded states of the input end. We have a principal figure as Figure 3 shows.(Bahdanau et al., 2014) Here it uses a bi-RNN, which can get rid of the direction bias with the attention mechanism.

2.5 Related Work

2.5.1 Team SOLAT in the SWEN's Work

The schematic diagram is as Figure 4 shows. In their work, half of the classification choice comes from a decision tree while another half is from a CNN model. For the decision tree model, they used naive count features(BoW), TF-IDF features, SVD features, word2vec features and sentiment features. They use the Sentiment Analyzer in the NLTK package to generate the sentiment features. As for the decision tree itself, they used a kind of gradient-boosted decision tree called XGBoost which is first introduced by Tianqi Chen. For the CNN model, they used 5 convolution layers. The final relative score of their model is 82%.

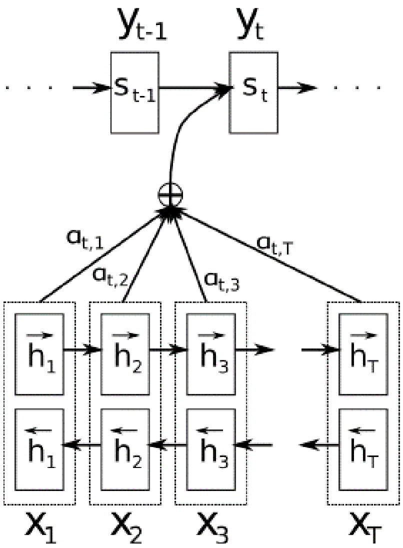
y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 3: Attention mechanism

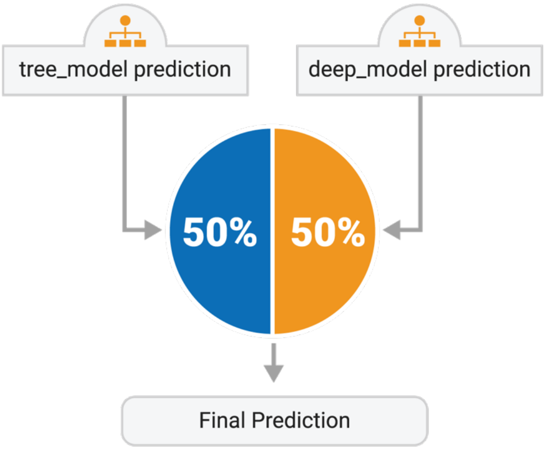

Figure 4: Team SOLAT in the SWEN

2.5.2 Team Darmstadt's Work

Figure 5 shows the schematic diagram of the Darmstadt' work. The baseline of their work is a MLP model. As for the features, they chose Bag of Words (BoW), Non-Negative Matrix Factorization (NNMF), Latent Semantic Indexing (LSI) and Paraphrase Detection based on Word Embeddings (PPDB). With respect to the MLP, they used 7 hidden layers. Their score is 81.97.

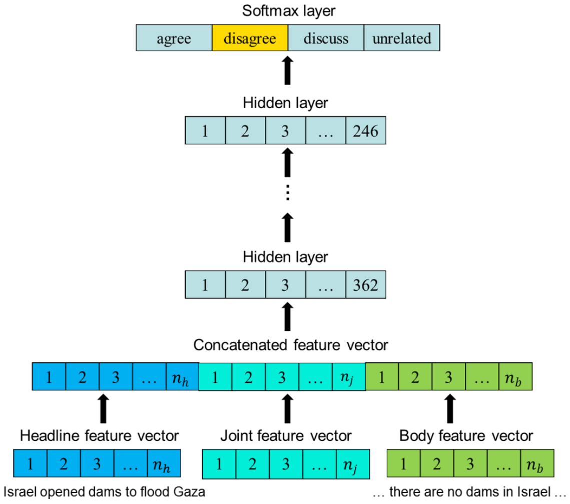

Figure 5: Team Darmstadt

2.5.3 Team UCL's Work

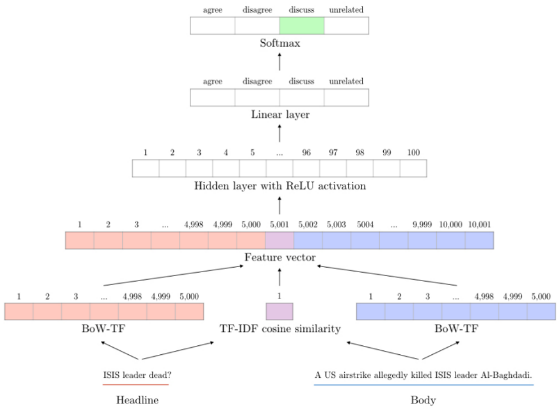

Figure 6: Team UCL

Figure 6 tells the principle of their work which is quite similar to the second group in the baseline. (Riedel et al., 2017) The difference is that they just used the TF-IDF cosine similarity and BoW as their features.Their final relative score on the test data from the challenge competition is 81.72. We will implement on their algorithms on our own. And use our own splitted test data.

Summary of the Related Work

Through review the related work, we can find that their work has such disadvantages:

1. All the related work right now are using the whole texts features, which might include many noisy information. Besides, it's quite hard and low efficient to train the model.

2. They just used models based on machine learning algorithms. We might consider some deep learning methods.

3 Model

3.1 Text CNN and Siamese Network

Convolutional Neural Network (CNN), as we know, has made a great progress in supervised learning, which concentrate on local subtle feature to classify sentense and images. Text CNN was firstly proposed to deal with text classification, which shows an effective performance. However, in our setting, the relationship between headlines and text bodies is not simply a classification problem, we need to understand the relative semantic information between, which makes it a basic inference rather than classification. On the other hand, the inference in the problem is ralative easy becase of the symmetry of defined relation: "unrelated", "agree", "disagree" and "discuss". Ignoring the differences with text sizes between titles and bodies, we can regard it as a symmetry problem, which can be dealt with a Siamese Network to learn types of relative relations.

Siamese Network was firstly proposed to evaluate the similarity between images, which uses a symmetry network for two input data. When we uses a large number of training data to learning the similarity representation, the network can learn a feature representation of images to do a better similarity evaluation.

Moreover, the combination of Text CNN and Siamese Network was firstly proposed by (Severyn and Moschitti, 2015), which aims to learn to rank short text pairs, showing a satisfying result in answer sentence selection problem. Our problem setting, which firstly judge whether titles and bodies are related or unrelated can be seen as a variety of similarity evaluation. To predict the specific relationship between titles and bodies, we can easily make it a multi-class classification.

According to fig.(3.1), we learn a similarity matrix to evaluate the differences of $x_1^{(pooled)}$ and $x_2^{(pooled)}$.

$$S_{12} = (x_1^{(pooled)})^T M x_2^{(pooled)} \tag{5}$$

From above, we can combine text embedding vectors and the similarity of headlines, and the hidden layers forwards help us to classify the exact relationship.

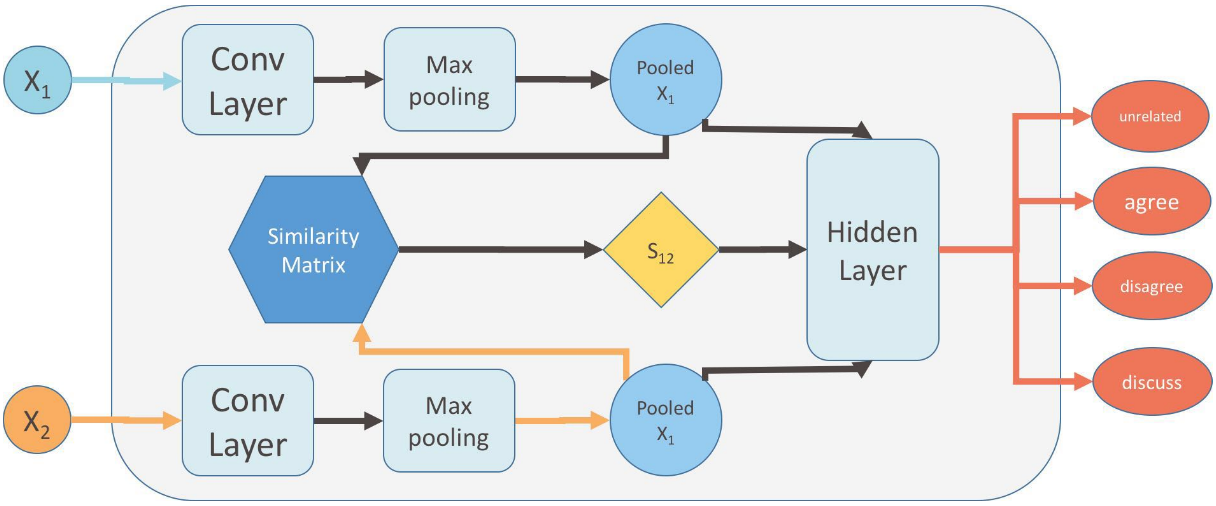

Figure 7: Text CNN Siamese Network

3.2 Attention based model for Abstractive Sentence Summarization

Since the whole text is so redundant, we could use abstractive sentence summarization model to extract one sentence that captures the core meaning of the text. Our model based on the A Neural Attention Model for Abstractive Sentence Summarization(Rush et al., 2015).

To some extent, ”attention” is actually a measure of similarity. In attention based model, if the current input is more similar with the target state, then the weights will be bigger, which means the current output depends more on the current input. The encoder-decoder structure without the attention mechanism usually encodes the last state of the decoder as the input of the decoder (maybe as an initialization or as an input at each moment). However, the encoder state is limited after all and can not store too much information. For the decoder process, each step has nothing to do with the previous input, only with the incoming state. After using the attention mechanism, the decoder can look up information in the addtional input states.

Below are the specific introduction to our model.

Given an input text, our model is aimed at producing an abstract of the text. The input text x is consisted of a sequence of M words $x_1, \dots, x_M$ with a fixed vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = V$. Our target is taking x as input and outputs one sentence y of length $N < M$. Supposing y is consisted of $y_1, \dots, y_N$ and $\forall i, y_i \in \mathcal{V}$. Define the set $\mathcal{Y} \subset (\{0,1\}^V, \dots, \{0,1\}^V)$. We want to find

$$argmax_{y \in \mathcal{Y}} s(x,y) \tag{6}$$

where $s : \mathcal{X} * \mathcal{Y} \longmapsto \mathbb{R}$ is scoring function. We define s as:

$$s(x,y) = \Sigma_{i=0}^{N-1} g(y_{i+1}, x, y_c) \tag{7}$$

where $y_c = y_{[i-C+1,\dots,i]}$ for a window of size C. Considering Bayesian inference, we could regard the score function as the conditional log probability. That is:

$$logp(y|x;\theta) = \Sigma_{i=0}^{N-1} g(y_{i+1}, x, y_c) \tag{8}$$

So our target now is modelling $p(y_{i+1}|x, y_c; \theta)$

From the class of NNLMs described by Bengio et al.(Bengio et al., 2003), we model the conditional logprobability as:

$$p(y_{i+1}|y_c, x; \theta) \propto exp(Vh + Wenc(x, y_c)) \tag{9}$$

$$\tilde{y_c} = [E_{y_{i-C+1}}, \dots, E_{y_i}] \tag{10}$$

$$h = tanh(U\tilde{y_c}) \tag{11}$$

where $E \in \mathbb{R}^{D*V}$ is a word embedding matrix, $U \in \mathbb{R}^{(CD)*H}$,$V \in \mathbb{R}^{V*H}$,$W \in \mathbb{R}^{V*H}$ are weight matrices. h is a hidden layer whose size is H. The word embeddings’s size is D. Function enc is a contextual encoder term. In our model, the function is an attention-based contextual encoder (Bahdanau et al., 2014). More specificly:

$$enc(x, y_c) = p^T \bar{x} \tag{12}$$

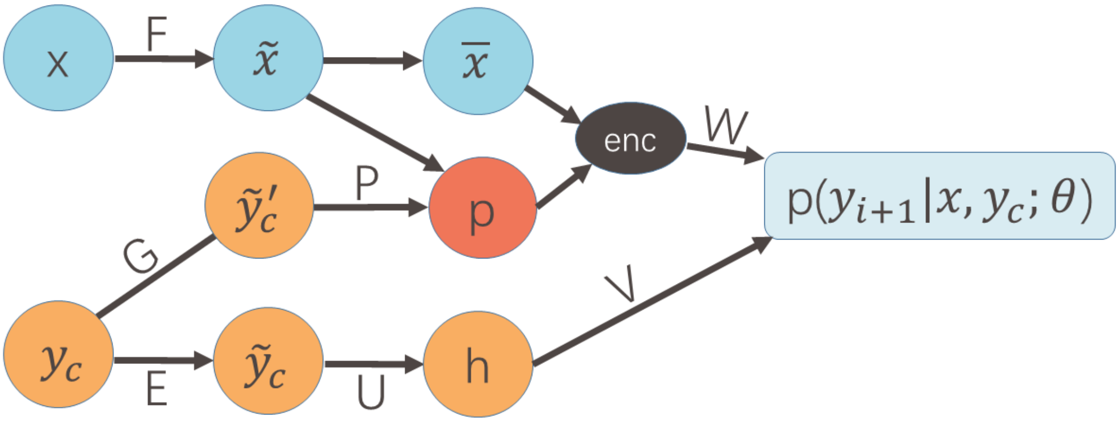

Figure 8: Attention based model for Abstractive Sentence Summarization

$$p \propto exp(\tilde{x}P\tilde{y}_c^{'}) \tag{13}$$

$$\tilde{x} = [Fx_1, \ldots, Fx_M] \tag{14}$$

$$\tilde{y}_c^{'} = [Gy_{i-C+1,\ldots,Gy_i}] \tag{15}$$

$$\forall i, \bar{x}_i = \Sigma_{q=i-Q}^{i+Q} \frac{\tilde{x}_i}{Q} \tag{16}$$

In the equation, $G \in \mathbb{R}^{D*V}$ is an embedding of the text, $P \in \mathbb{R}^{H*(CD)}$ is a weight matrix parameter and Q is a smoothing window. Further more, we use a hybrid pointer-generator network(See et al., 2017) in our model. The network could noticed words from the source text via pointing so it can aid accurate reproduction of information as well as retain the ability to produce novel words through the generator.

Above is the schematic diagram of our model.

Why does attention based model work?

To see whether the attention based model works, we select some results of our text summary model and compare them with the related headlines. Further, we andomly choose some sentences in the body text so see the informance we need has a very little weights in the text. The table 1shows the performance of the model:

Stance	Agree
Title	Small Meteorite Strikes in Nicaragua's Capital City of Managua
Model result	A small meteorite crashed into a wooded area in Nicaragua's capital of Managua overnight .
Body text	... It initially appeared as though Tsarnaev slipped on a discarded banana peel and hit the floor with such force that his head bounced off the hard tile surface, probably 16 or 17 times. ...
Stance	Disagree
Headline	Pope Francis turns out not to have made pets in heaven comment
Model result	Pope Francis , who died in 1978 , once said , reportedly while comforting a child whose dog had died , " one day we will see our animals in the eternity of Christ ."
Body text	... In his Nov. 26 remarks, the pope referred to heaven saying "the Sacred Scripture teaches us that the fulfillment of this marvelous plan cannot but affect all that which surrounds us and which emerged from the thought and the heart of God." He did not refer specifically to animals in his remarks. ...

Table 1: Demo of Attention based text summary.

4 Experiment and Result

Our experiments are based on the Fake News Challenge, we use the same dataset provided:

”fnc_data”. however, the competition didn’t release the label of test result, we have to split orginal data into training and valid data, so that we can compare different models with the same ruler.

As the title candidate set is much bigger than the bodies’, there are many instances share the same body, in case of intersection between training and valid data, we firstly split the text bodies (about 19:1). And in the following experiments, we use the same training and valid data.

4.1 Repeat one of the top models in the FNC

Consdiering the environmental requirements, we choose the UCL group to repeat their work. We split the oringinal training set into training set and test set with the ratio of 0.95:0.05. We build an environment with TensorFlow 0.12.1 and Python 3.5 to train the model. After we train the model and evaluate it with the formula as Figure1 shows, we can get a relative score of 0.82.

4.2 Baseline

Since it is a classification task, we have many traditional machine learning methods to achieve the target. In this section, we test several popular models’s performance in both binary classification and mulit-classification tasks. The specific steps are as follows:

4.2.1 Experiment steps

1. Using tfidf, bow and word2vec as three features extracted from the training data.

2. Using the training data to train the models and find the best parameter of each model.

3. Using the test data to show the performance of the models

Note: we firstly considered use doc2vec(Mikolov et al., 2013) to gernerate the representation of documents, however, as the ambiguous semantic information doc2vec provided, the performance is so bad.

4.2.2 Results

From the figure we could find SVM model has the best performance in binary classsification with word2vec (Mikolov et al., 2014) as the feature. When using tfidf as the feature, Kneighbors has the best result. And RandomForest has the stable performance in three features, but not the best. The results shows that if we want to divide the data into ’related’ and ’unrelated’ in the first step, SVM could be a good choice.

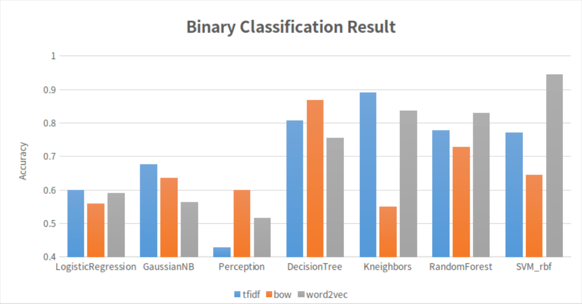

Figure 9: Binary Classification Result

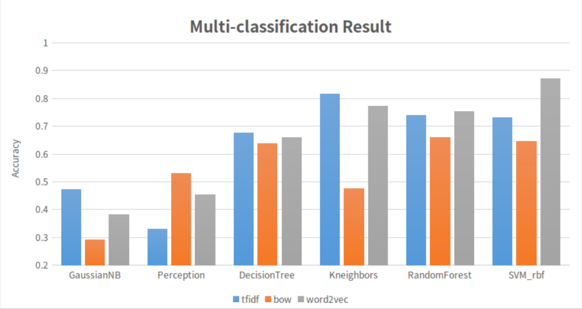

Figure 10: Mulit-classification Result

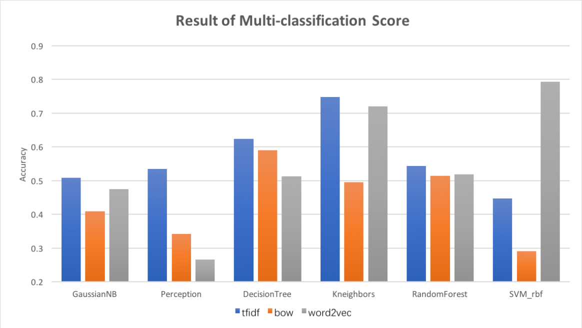

Figure 11: Results of Mulit-Classification Score

In the Figure 10 and Figure 11, you could find the multi-classification results and the penalized multi-classification results, in which the score is computed as 25% of score weight whether or not the headline and body text are related and 75% of score weight which one relationship between headline and body text if they are related. We could find that SVM with word2vec as the feature has the best performance in accuracy but RandomForest with word2vec as the feature has the best score in final result. Therefore our traditional machine learning method’s best result is :

Feature	Model	Score
word2vec	SVM (RBF Kernel)	79.3%

We use the score as our baseline.

4.3 Text CNN and Siamese Network (TCSN)

4.3.1 Full Document (F-TCSN)

We firstly consider using the full documents to as x_1 with word2vec (Mikolov et al., 2014) embedding, and using candidate headlines embedding representation as x_2. We use Text CNN and Siamese Network to train our model. (fig.(3.1)). However, as the full content of documents is so redundant, it takes much times to extract the feature of a text. Meanwhile, as the CNN extracting local information, the long text seems not work well with the model.

4.3.2 Text Summary with Attention model (A-TCSN)

As the computation resources limitation, and the redundant information of full document, we can use the former model, that generate titles (or text summary) to reduce the size of each input document text. In a transfer learning view, the transformation from full document can be regarded as domain adaption process, we embed the document feature into a ”short text space”.

4.3.3 Experiments

We use Tensorflow to build our model, as we expected, the train matrix of x_1 can be reduced a lot by our attention model (from around 2,600 dimension to 100 dimention). More over, the training step can be saved a lot (full document model cost over 5 times than attention based model). Moreover, as the baseline of binary classification indeed shows a amazing performance (about 95%), we concentrate on the fine-grain classification, i.e. the binary classification we use here is based on SVM (RBF Kernel).

We train our model with 10 epochs (as the limitation of computation resources), and we get the result of F-TCSN and A-TCSN below:

As we can see from the tabel, as the characteristic of this problem, the baseline by SVM indeed provide a relative satisfying result, the performance of each just improves about 2-3% with a lot of effort. However, although we only train the model for 10 epochs, our result raises nearly another 2% in such setting. Moreover, the training process of our attention model is much more economical than other methods.

Method	Proposed Score
Baseline: SVM (RBF Kernel)	79.3%
SOLAT in the SWEN	82.0%*
Athene (UKP Lab)	82.0%*
UCL Machine Reading	81.7%*
UCL Machine Reading	82.2%**
F-TCSN	82.9%***
A-TCSN	83.5%***

Table 2: Final Comparison of the Models

*Use the test set privided by the organizers.

**Use the test set splited on our own.

***As the limitation of computation resources, the results may not converge.

5 Conclusion

We propose a method using the CNN to extract sequence information to get a better result. However, as the redundant information in the full documents, the training process is not economical, further, the adjustment of hyper-parameters is difficult with such noise. We use Attention Model to reduce the length of text, which can be seen as a adaption from long text (document) to short text (title) with main information.

In our experiment, we beat all the model in competition. In addition, our model can be extended in other Textual Inference problems, in particular, for the problem whose source domain and target domain are not the same (e.g. title and text body) which can be adapted by attention method.

6 Future work

As we can see from above, all our current work extracts feature from the whole bodytext. Even in the attention model, we will use the whole bodytext to generate a sentence. In our feature work, we will first find some sentences in the text who are most related to the headline. Then we will do the classification task based on the extracted sentences. We might consider use text inference model to detect fake news based on Stanford inference dataset with attend model. Currently, we have implemented the first stage, i.e., most related sentence finding stage.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, L Sutskever, and G Zweig. 2014. word2vec.

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

Benjamin Riedel, Isabelle Augenstein, George Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR* abs/1707.03264. http://arxiv.org/abs/1707.03264.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR* abs/1509.00685. http://arxiv.org/abs/1509.00685.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR* abs/1704.04368. http://arxiv.org/abs/1704.04368.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .