Research of Text Classification Based on TF-IDF and CNN-LSTM

Research of Text Classification Based on TF-IDF and CNN-LSTM

Hai Zhou[1,*]

[1]School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214000, China

[*]email: 6181611035@stu.jiangnan.edu.cn

Abstract. With the rapid development of deep learning, many deep learning models have been widely used in Natural Language Processing(NLP). The Long-Short Term memory network(LSTM) model and convolutional neural network(CNN) model can achieve high classification accuracy in text classification tasks. However, the high input dimension of text features and the need to train a large number of parameters in the deep learning model often take a lot of time. This paper uses Term Frequency-inverse Document Frequency(TF-IDF) to remove features with lower weights, extract key features in the text, extract the corresponding word vector through the Word2Vec model, and then input it into the CNN-LSTM model. We compared the model with CNN, LSTM, and LSTM-attention methods and found that the model can significantly reduce model parameters and training time in short and long text data sets. The model hardly loses accuracy in the long text, but the model will lose a certain amount of accuracy in short texts. This paper also proposes fusing original text features to make up for the accuracy loss caused by the TF-IDF feature extraction method.

1. Introduction

Text classification is one of the hot research topics in Natural Language Processing (NLP), and many experts and scholars have conducted in-depth research on it. Text classification technology is widespread in our lives, such as commodity evaluation information classification, movie type classification, and spam identification applications, which can help people make quick choices or decisions. Traditional text classification methods are mainly based on dictionary methods, but such methods have high labor costs, low accuracy, and other indicators. In recent years, machine learning methods and deep learning methods have continued to develop. These models are often used in NLP and have also achieved good results in text classification tasks [1].

In machine learning and deep learning, text classification-oriented models mainly need to perform the following three steps: text vectorization, input model, model classification:

(1) Text vectorization: The text is displayed in the form of variable length and inconsistent format. How to properly process the text is very important. The quality of text vectorization will directly affect the effectiveness of the target task. The currently commonly used method is Count Vectorization, TF-IDF Vectorization, Word2Vec, GloVe, FastText, etc. The first two better retain the overall word frequency characteristics of the text but ignore the connection between the context and the semantics of each word. The latter three better retain the semantics of the word and are commonly used methods in deep learning.

(2) Input model: The input model is generally a machine learning or deep learning model. After the text is vectorized, it is organized into features suitable for model input and then input into the corresponding model. Generally, in deep learning, standard models are LSTM, CNN, etc.

(3) Model classification: The model classification is relatively simple, the sigmoid method is commonly used for two categories, and the softmax method is widely used for multiple classifications.

The above three steps constitute the primary method of model training. Generally speaking, the first two steps have excellent research value. How to extract effective features as much as possible for text vectorization, how to compress text expression, how the model learns text features, how to pay attention to context and global features are all very worthy of research.

In this paper, we combine the TF-IDF algorithm to extract the most important key features in the text, extract the word vector through the Word2Vec model, Then use the CNN model to learn local information and convert it into high-dimensional features for the LSTM model to learn the overall information. Such an architecture has significantly filtered the text features with less information at the input level, allowing the CNN-LSTM model to learn the most useful knowledge in the fastest and most complete way. We compared the model with CNN, LSTM, and LSTM-attention in terms of long and short text, accuracy, parameter amount, and time-consuming experiments. We found that the model can significantly reduce model parameters in short and long text data sets. However, in short texts, the model will lose a certain accuracy. This paper also proposes a method that fuses the original text's features to compensate for the loss of accuracy of the method.

2. Related work

In the field of text classification, the architecture of deep learning models is different from traditional machine learning in two aspects: feature extraction and classifier principles[2].

Feature extraction refers to a type of technology that transforms text into vectors. Feature extraction in machine learning generally ignores the connection between words, the semantics are relatively weak, and the Count Vectorization[3] and TF-IDF Vectorization[4] are constructed. The feature dimension is rather high, and it isn't easy to apply to deep learning models with many parameters. As the computing power of computer GPUs continues to increase, the time cost of model training decreases, and the iteration speed increases. Therefore, training a fixed-dimensional word vector model has become one of the crucial ways to extract text features. Bengio et al. first proposed in 2003 Neural Network Language Model (NNLM). NNLM uses a three-layer feedforward neural network to model the corpus[5]. The NNLM model lays the foundation for the study of language neural models. Many scholars have proposed improved models based on NNLM. Pennington et al. proposed a global word vector model (Global Vector, GloVe). GloVe pays more attention to global text information, forming an association matrix through the word segmentation results of the text and decomposing it to obtain a set of distributed word vectors[6]. This type of model can be trained unsupervised through an unlabeled corpus and generate a fixed-dimensional vector space. Each word can find its mapped vector in the vector space. Such a vector has high semantics. And the dimensionality is low, suitable for deep learning models.

Recurrent neural network (RNN)[7] and variant Long-Short Term Memory (LSTM)[8], Gated Recurrent Unit (GRU)[9], this type of deep learning model related to timing is often used in text classification. They regard the contextual relationship of the text as a temporal relationship and classify and predict the text by learning the characteristics of the context. In addition to the perspective of the timing relationship, Yoon Kim et al. proposed to use the Convolutional Neural Networks (CNN) model to extract and learn the global features of the text, thereby reducing the number of parameters of the classification model and accelerating the training and prediction speed of text classification[10].

Feature extraction and classifiers are not entirely separated. Take the C-LSTM model proposed by Chunting Zhou et al. combining CNN and LSTM as an example. This model uses the advantages of the two technologies to construct text vectors through CNN and use RNN loops. The structure captures contextual information to build a text classification model. C-LSTM uses CNN in feature extraction and LSTM as the essential classification model, achieving good results.

Similarly, Bao Guo et al. improved the effect of multi-channel textCNN based on weighted word embeddings. Gang Liu et al. used Bi-LSTM as the feature extractor of the context. They used the attention mechanism to give different attention to the information in it, capture the local and global features of the phrase, and then classify the text based on the fully connected layer. Good results have been achieved on the data set.

3. Text classification based on TF-IDF and CNN-LSTM

The text classification model based on TF-IDF and CNN-LSTM proposed in this paper can be mainly divided into TF-IDF, Wor2Vec, CNN, LSTM, and the model structure shown in Figure 1. Among them, TF-IDF and word2vec are collectively referred to as feature extraction processes, and CNN and LSTM are collectively referred to as the CNN-LSTM model. In the feature extraction stage, TF-IDF will extract keywords from the preprocessed text and then extract word vectors by using the pre-trained model Word2Vec. In the CNN-LSTM stage, we use CNN to learn the local key features in the word vector and input the processing results into the LSTM layer. The LSTM layer learns the global features of the entire text through the forget gate, update gate, and output gate mechanism and Output the learned results to the hidden layer, and then the sigmoid is used to predict the two classifications.

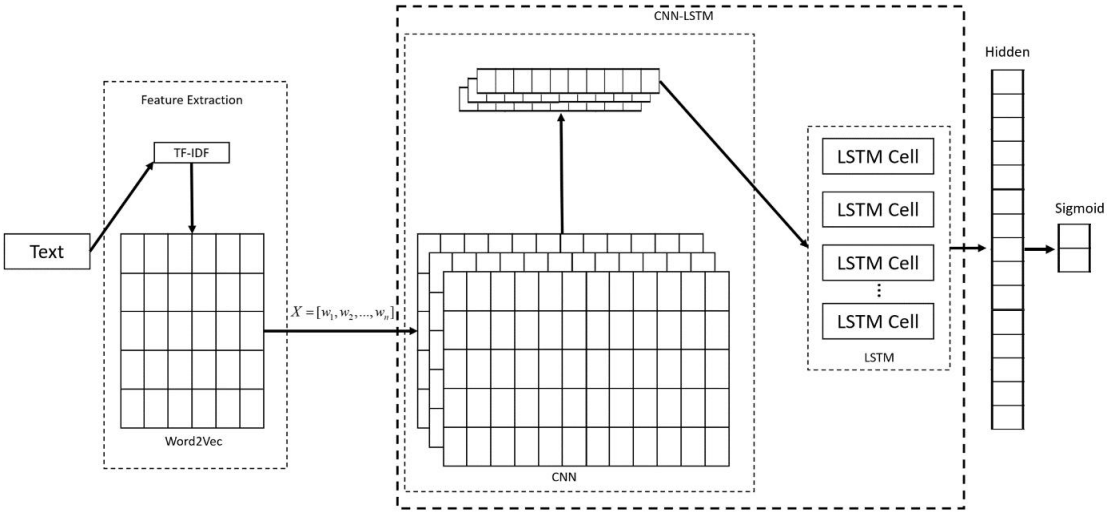

Fig. 1 model structure diagram

3.1. Feature extraction

In the preprocessing stage, we use word segmentation and stop word removal and use TF-IDF to extract the keyword information of the text as the text feature. TF-IDF is a statistical method used to evaluate the importance of a word in a text. The importance of a word increases in proportion to the number of times it appears in the document, but at the same time, it decreases in inverse proportion to the frequency of its appearance in the corpus. TF-IDF can better recognize some distinguishable words. These words appear more frequently in a certain text but appear less frequently in other texts.

We use equation 1 to calculate the IDF score of each word in a certain text in a data set:

$$idf(w_i) = \log \frac{|D|}{|\{j \mid w_i \in D_j\}|} \tag{1}$$

In equation 1 D represents the set of text samples, $|D|$ represents the total number of text samples, D_j represents the *j*-th text in the text samples, w_i represents the *i*-th word in the vocabulary based on the construction, and $\{j \mid w_i \in D_j\}$ represents the total number of text samples containing words w_i.

We use equation 2 to calculate the word frequency in each text sample. Without considering IDF, if a word appears in a large number of text samples, then the word may have a higher degree of discrimination and contribution to this text sample:

$$tf(k,i) = \frac{count(w_{k,i})}{\sum_{j} count(w_{k,i})} \tag{2}$$

Among them $w_{k,i}$ is the *i*-th word frequency of the vocabulary in the *k*-th text, $count(w_{k,i})$ which means the frequency of the *i*-th word in the *k*-th text, and $\sum_{j} count(w_{k,i})$ means the sum of the word frequencies of all words in the vocabulary in the *k*-th text sample. According to the above two equations, we use equation 3 to calculate the TF-IDF value of each word:

$$TFIDF_{k,i} = idf(w_{k,i})\ tf(k,i) \tag{3}$$

This equation extracts the different scores of each word in the text. Words with higher scores have a higher degree of discrimination and are most capable of distinguishing text categories. These words retain the words that contribute to the classification to the greatest extent, Which compresses the number of texts and the number of features that need to be calculated.

TF-IDF sorts different words according to their importance in different texts, from large to small. Words with higher weights in a text will have less weight in other texts, so for each text, The distinguishing ability of words with a higher degree of importance is higher. We use TF-IDF to explicitly extract the most important text features, filter out invalid features, and then use the Word2Vec word vector model that retains semantic information to map to get the corresponding word vector. To preserve the semantic information of the text as much as possible, we input the text features extracted by TF-IDF into the trained Word2Vec model, extract the corresponding word vector, and then input it into the CNN-LSTM model for learning.

3.2. CNN-LSTM Model

CNN is divided into a convolutional layer and a pooling layer. We use the convolutional layer as the feature extractor of the context and then use the pooling layer to compress the data and the number of features to improve the model's fault tolerance. The convolutional layer learns the local features through its neurons' receptive field and fuses the text's local features to form a more global overall feature. It is processed into nonlinear features through the activation function tanh and then the nonlinear characteristics of the input to the pooling layer. We use the maximum pooling method to extract the most significant features and remove relatively less obvious features, thus simplifying the network complexity and reducing the amount of calculation. When the CNN processes the text features appropriately, we pass the output into the LSTM model. In the LSTM model, each neuron cell has three mechanisms: forget gate, update gate, and output gate, as shown in the figure below:

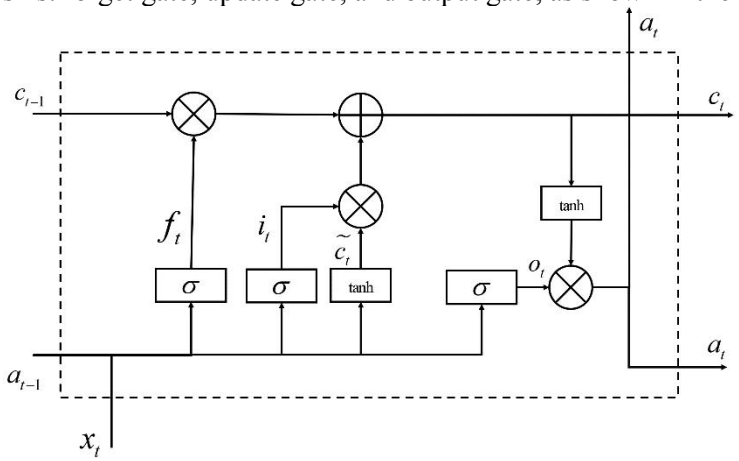

Fig. 2 LSTM structure diagram

The learning process of LSTM is as follows:

Let t represent the time sequence, the cell state output by the previous sequential unit, and the hidden state a will be used as the input of the current sequential unit. In the current time sequence t, the unit mainly performs the following operations on the input information:

(1) Forget irrelevant information: After the current input is spliced with the hidden state of the $t-1$ time sequence output, the information that needs to be retained in the last time sequence c_{t-1} determined by the forgetting gate deletes irrelevant or secondary information:

$$f_t = \sigma(W_f\ [a_{t-1}, x_t] + b_f) \tag{4}$$

(2) Obtain new information: The update gate extracts meaningful information from the current input and adds it, and determines which features are introduced through the sigmoid function:

$$i_t = \sigma(W_f\ [a_{t-1}, x_t] + b_i) \tag{5}$$

Then create a feature vector containing all the information in the current input:

$$c_t = \tanh(W_c\ [a_{t-1}, x_t] + b_c) \tag{6}$$

$i_t\ c_t$ represents the feature vector formed by extracting useful features from the current input and $f_t\ c_{t-1}$ represents the features retained after filtering out irrelevant features in the previous time sequence. Add them to get the current cell state c_t:

$$c_t = f_t\ c_{t-1} + i_t\ c_t \tag{7}$$

(3) Output the hidden state of the current sequence: After calculating the state c_t of the current cell, the output gate will filter once to select the feature information that needs to be retained to the hidden state:

$$o_t = \sigma(W_o\ [a_{t-1}, x_t] + b_o) \tag{8}$$

Finally, normalize the cell state to (-1, 1) through the tanh function, and perform operations with the vector o_t to obtain the hidden state a_t of the time sequence t:

$$a_t = o_t\ \tanh(c_t) \tag{9}$$

4. Experiments and Results

4.1. Dataset and Experiment Setup

Two Dataset: This article uses THUCNews and crawled Taobao review. Each data set selects 20000 data, of which THUCNews has a total of 10 categories. To verify the effect of the model proposed in this article on the short text and long text, we only take two of them for experimentation, and the two data sets have two categories. The ratio is 1:1.

Word2Vec pre-training model: This paper uses the open-source Word2Vec model to extract word vectors. The model uses Chinese Wikipedia as a corpus and is trained based on the Skip-Gram training mode. The vector dimension obtained is 300 dimensions.

Model training, verification, and testing process: The training process is the same for any model M. In each epoch, the training set is input to model M. After the training is completed once. The model is allowed to make predictions on the data in the validation set. Calculate the accuracy of the prediction results and record them. When the accuracy on the verification set is no longer improved after multiple rounds of epoch training, take the best model to predict the results of the test set, and calculate each item. The indicator serves as the final performance of the model.

4.2. Evaluation

Since there is no imbalance in the data set selected in this article, the evaluation indicators selected in this article are mainly accuracy and F1. For a sample, we use the model to classify it. There are four situations:

1. True Positive (TP), the true category of the sample is a, and the model correctly predicts that the sample is category a;

2. False Negative (FN), the true category of the sample is a, and the model incorrectly predicts that the sample is another category;

3. False Positive (FP), the true category of the sample is other categories, and the model incorrectly predicts that the sample is a;

4. True Negative (TN), the true category of the sample is other categories, and the model incorrectly predicts that the sample is also other categories.

According to the above basic definition, for a certain class a, its precision, recall, F1, and accuracy calculation equations are as shown in equation 10:

$$\begin{cases} P = \dfrac{TP}{TP + FP} \\ R = \dfrac{TP}{TP + FN} \\ F1 = \dfrac{2 \times P \times R}{P + R} \\ accuracy = \dfrac{TP + TN}{TP + TN + FP + FN} \end{cases} \tag{10}$$

4.3. Experiment 1: Model training time and parameter comparison

The two data sets are long text and short text, respectively. We compare the text classification effects based on these two data sets. The base method means that TF-IDF is not used, only Word2Vec is used to extract word vectors, and text classification is performed based on the selected model. The TF-IDF method means that TF-IDF keyword extraction is performed on the text first, and the number of keywords extracted is 20, then Word2Vec is used to extract word vectors, and then text classification is performed based on the selected model. We chose the following four models for experiments: LSTM,bi-LSTM, CNN-LSTM, and LSTM+attention. To prevent over-fitting, we use the early stop mechanism to stop the iteration when the model effect on the validation set produces over-fitting and calculate acc, F1, using time and parameters based on the optimal model generated on the validation set before. The experimental results are shown in Table 1 and Table 2.

Tab. 1 Experiment of THUCNews

model	method	*acc*uracy	*F*1	using time	parameters
LSTM	base	0.922	0.92183	70	15,399,425
	TF-IDF	0.989	0.988991	**9**	**2,573,825**
bi- LSTM	base	0.998	0.998	156	15,924,993
	TF-IDF	**0.99**	**0.989986**	16	3,099,393
CNN -LSTM	base	**0.999**	**0.999**	14	15,465,217
	TF-IDF	0.988	0.987984	**2**	2,639,617
LSTM +attention	base	0.997	0.99699	63	15,465,217
	TF-IDF	0.981	0.980977	9	2,639,617

Tab. 2 Experiment of Taobao review

model	method	*acc*uracy	*F*1	using time	parameters
LSTM	base	0.91901	0.91898	73	4,193,025
	TF-IDF	0.8713673	0.8709525	9	**3,323,137**
bi-LSTM	base	0.91663	0.91663	180	4,718,593

CNN-LSTM	TF-IDF	0.867556	0.8669384	18	3,848,705
	base	**0.93187**	**0.93187**	**8**	4,258,817
	TF-IDF	**0.8808957**	**0.8808253**	**3**	3,388,929
LSTM	base	0.91853	0.91853	74	4,258,817
+attention	TF-IDF	0.8637446	0.8636703	12	3,388,929

Although all models are based on LSTM, there is still a certain gap in the performance of different models in different data sets. According to the experimental results, the number of parameters and training time is significantly reduced after using TF-IDF to process the data. The number of parameters in the long text is reduced by about 80%, and the calculation time is only 10% of the original. However, the model effect (accuracy, F1) decreased by about 1%. In the short text, the parameter decline is not obvious, about 25%, but the training time is significantly reduced by 90%, and the effect of the model is reduced by 2-5%.

In the experiment, we found that regardless of whether TF-IDF optimization is used or not, the effect of the CNN-LSTM model can achieve the best and the effect of bi-LSTM is the second. The accuracy of LSTM and LSTM+attention is not as accurate as CNN-LSTM, and bi- LSTM is high. The reason is that bi-LSTM has more neurons and parameters to learn features than LSTM. Although the attention layer can extract local information, whether long or short, the attention layer does not significantly increase the number of parameters. And attention may filter out some meaningful features when extracting information, resulting in a loss of accuracy. Although the CNN layer will also filter features, it still "enlarges" the expression of the features after convolution and maximum pooling so that there is a certain effect of optimization. In addition, since the convolutional layer and the pooling layer of the CNN layer are constantly reducing parameters, the CNN-LSTM model only adds a small number of parameters compared to the LSTM, but the effect is somewhat improved.

4.4. Experiment 2: Introduce original text features into short text

In the THUCNews data set, the models based on TF-IDF and CNN-LSTM have significantly optimized in terms of computational efficiency and the number of parameters, and the effect has not decreased much. However, in the short text, when the number of features of the short text itself is small, TF-IDF can extract the most important text features, but some information is lost, so the model accuracy is lost by about 2-5%. In this experiment, based on the taobao review data set, we added original text features in short text classification to supplement the information lost by TF-IDF. The experimental results are shown in Figure 3.

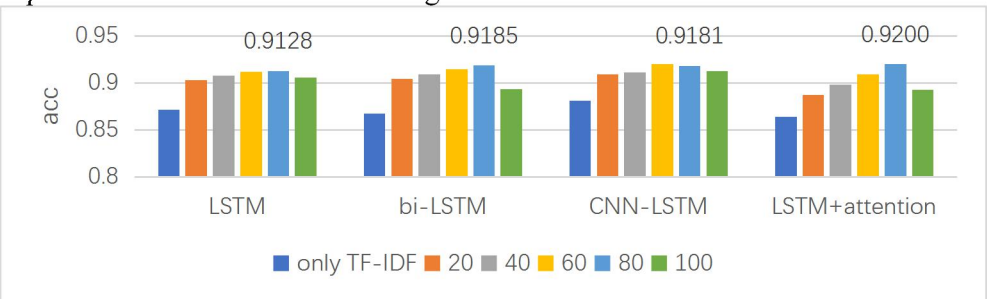

Fig. 3 experiment of TF-IDF feature with origin feature

Among the various models, only using TF-IDF as the feature extraction method has the worst effect, but the effect significantly improves after the original text features are introduced. When the number of text features is 80, other than the CNN-LSTM model, The effect of the model is the best, and the acc of LSTM+attention can reach 92.00%. The reason is that TF-IDF provides the most important features but still loses some meaningful features. After supplementing the original text features, Attention will find distinguishable features for learning, so the model's accuracy will be greatly improved. It can also be seen from the figure that when the text features are introduced at 20, 40, 60, and 80, the effect of LSTM+attention is steadily improved, indicating that when the original text features are increased from 20 to 80, the attention can still be Continuously obtained

distinguishing features from it. However, when the original text features increase to 100, the effect of each model has a certain decline, which means that the number of original text features is too much. Still, it covers some distinguishing features, making the model unable to extract from TF-IDF effectively. The most effective knowledge is learned from the characteristics. This experiment can also prove that the features extracted by TF-IDF have a high degree of discrimination. However, a certain number of original text features are still needed to supplement it so that the training speed and the reduction of parameters can be significantly optimized. Reach or exceed the level of effectiveness of original text classification.

5. Conclusion

We describe a text classification model based on TF-IDF and CNN-LSTM. TF-IDF can explicitly extract the most important features of text, CNN focuses on capturing local information, and LSTM can capture overall information. We conducted experiments on short text data sets and long text data sets, respectively, by combining these three. We verified that the model could achieve training acceleration and parameter reduction effects on both short text and long text, and the model is in There is almost no loss of model effect on long texts. Although some accuracy will be lost on short text, the model effect can be improved by supplementing the original text features.

The current calculation method of TF-IDF is calculated in the current data set, which cannot achieve higher accuracy in feature extraction. The next step of this paper is to optimize the feature extraction step of TF-IDF, introduce domain knowledge data, and acquire corresponding knowledge according to different tasks to extract the features of the text better and optimize the model effect.

Acknowledgments

This work was supported in part by the Jiangsu Science and Technology Plan Funded Project, grant number BE2018056.

References

[1] Minar M R and Naher J 2018 Recent advances in deep learning: An overview *Preprint* gr-qc/180708169
[2] Otter D W, Medina J R and Kalita J K 2020 A survey of the usages of deep learning for natural language processing *IEEE Trans. Neural Networks and Learning Systems* **32** pp 604-624
[3] Gupta G and Malhotra S 2015 *Text Document Tokenization for Word Frequency Count Using Rapid Miner (taking resume as an example)* **975** p 8887
[4] Dadgar S M H, Araghi M S and Farahani M M 2016 *2016 IEEE International Conference on Engineering and Technology(Coimbatore)* vol 1 pp 112-116
[5] Bengio Y, Ducharme R and Vincent P 2003 A neural probabilistic language model *The journal of machine learning research* **3** pp 1137-1155
[6] Pennington J, Socher R and Manning C D 2014 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing(Qatar)* vol 2 pp 1532-1543
[7] Cho K, Van Merriënboer B, Gulcehre C and et al 2014 Learning phrase representations using RNN encoder-decoder for statistical machine translation *Preprint* gr-qc/14061078
[8] Graves A and Schmidhuber J 2005 *Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures* **18** pp 602-610
[9] Chung J, Gulcehre C, Cho K H and et al 2014 Empirical evaluation of gated recurrent neural networks on sequence modeling *Preprint* gr-qc/14123555
[10] Kim Y 2014 Convolutional Neural Networks for Sentence Classification *Preprint* gr-qc/14085882