UoR at SemEval-2021 task 7: utilizing pre-trained DistilBert model and multi-scale CNN for humor detection

UoR at SemEval-2021 Task 7: Utilizing Pre-trained DistilBERT Model and Multi-scale CNN for Humor Detection

Zehao Liu, Carl Haines, Huizhi Liang
University of Reading
White Knights, Berkshire, RG6 6AH
United Kingdom
z.liu3@pgr.reading.ac.uk, carl.haines@pgr.reading.ac.uk,
huizhi.liang@reading.ac.uk

Abstract

Humor detection is an interesting but difficult task in NLP. Humor might not be obvious in text because it may be embedded into context, hide behind the literal meaning of the phrase and require prior knowledge to understand. We explored different shallow and deep methods to create a humour detection classifier for task 7-1a. Models like Logistic Regression, LSTM, MLP, CNN were used, and pre-trained models like DistilBERT were introduced to generate accurate vector representation for textual data. We focused on applying a multi-scale strategy on modelling, and compared different models. Our best model is the DistilBERT+MultiScale CNN which used different sizes of CNN kernel to get multiple scales of features. This method achieved 93.7% F1-score and 92.1% accuracy on the test set.

1 Introduction

Humor detection is an interesting but difficult task in Natural language processing (NLP) and requires various techniques to understand the meaning of a sentence and identify humor. For example, humour by sarcasm can mean that a piece of text can have two very different meanings and the NLP algorithm needs to be able to understand which meaning is intended.

The aim of task 7-1a of SemEval 2021 (Meaney et al., 2021) was to address the challenge of classifying humour in text. Provided for this task was a dataset constructed of short phrases in English along with a label classifying whether or not each phrase is intended to be humorous. The labels have been obtained by surveying a group of people that represent a variety of genders, political stances and income levels. The given dataset includes training set with 8000 labeled sentences, a development set of 1000 sentences and a test set of 1000 sentences.

The dataset was made up of English phrases that were labeled by their intent to be humorous. This means the label annotators were not saying whether or not they found the text funny but whether the writer of the text intended it to be funny. The texts were predominantly one sentence long with a small proportion being two or three sentences. Each phrase was labeled for humour intent and, if humour was intended, then a rating was given for how funny it is. Also of the texts intended to be humorous, a label was given for whether it is offensive, and if so, how offensive it is.

The texts covered a range of types of jokes such as puns, sarcasm, dark jokes and "Dad" jokes. One example is "I never finish anything. I have a black belt in partial arts." which contains a pun and is labeled as humorous.

Our best approach is DistilBERT+MultiScale CNN. It introduced a pre-trained DistilBERT model to extract textual features, and created a multi-scale CNN model for humour classification. First, the DistilBERT tokenizer generated a word token vector and an attention mask vector for each sentence. Then, we fed these vectors into a pre-trained DistilBERT model to get hidden features. After that, five CNN layers with different kernel sizes were used to get features of different scales. Each feature vector was subsequently concatenated together. Finally, the fused features were fed into dense layers in order to classify text into humorous or not humorous classes. It achieved 93.66% of F1 score and 92.10% of accuracy in using test set.

2 Related Work

We used four shallow models as a benchmark for our main approach. The first method was K-nearest neightbors (KNN) (Fix, 1951) in which an unlabeled query point is given the label of the majority of the K neighboring points. The second method

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 1179–1184
Bangkok, Thailand (online), August 5–6, 2021.

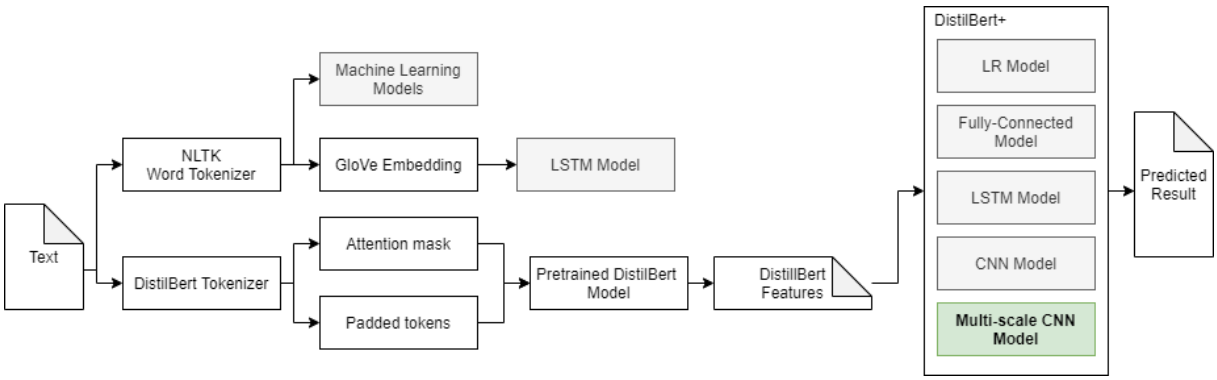

Figure 1: Pipeline of our experimental process

is Naive Bayes (Sammut C., 2011a) which utilizes Bayes rule together with a strong assumption that the attributes are conditionally independent. The third is a random forest (Sammut C., 2011b) which is an ensemble of decision trees trained on a bootstrap sample from the original dataset. The fourth method is the support vector machine (SVM) (Burges, 1998) which transforms the input data to a higher dimensional space and seeks to define a hyper-plane separating the classes. We also implemented voting to combine these methods where the prediction of the voting is simply the majority label predicted by the shallow models.

We also tried deep models. Long short-term memory (LSTM) is a very effective Recurrent neural networks for processing sequence data, which considers long and short term memory over time (Hochreiter and Schmidhuber, 1997). Convolutional Neural Network (CNN) uses a convolution kernel to extract hidden features from input, which is widely used in the field of computer vision (LeCun et al., 2010). 1-dimension CNN layer can fit with sequence data, so it can be used to extract textual features.

Transfer learning methods are widely used in the field of NLP. Pre-trained word embeddings are usually trained on unlabelled dataset and maps each word token into a fixed vector representing the meaning of the word in a hidden space. For example, Global Vectors for Word Representation (GloVe) is trained on a 6 billion word corpus using an unsupervised learning method in order to find word co-occurrence (Pennington et al., 2014).

Unlike conventional embeddings, contextualized embeddings dynamically map a word token into vectors based on the context using a pre-trained encoder. By extracting the features and adapting new data to the model, we can implement general downstream tasks based on the pre-trained model. BERT uses a deep Transformer as its encoder, and trains on language modelling tasks and next sentence prediction, which is often used in NLP with excellent performance (Tenney et al., 2019). DistilBERT uses knowledge distillation method to compress the model, which retains 97% of the performance of original BERT but is 60% faster (Sanh et al., 2020).

3 System Overview

In our BERT-based models, we used DistilBERT which is a light version of BERT to extract textual features. The BERT model is a transformer-based model, which is pretrained on vast amounts of textual data in language modelling tasks (Sanh et al., 2020). Therefore, the weights in the BERT model, which contains semantic information, can be used as contextualized embedding for general purposes. It can better represent textual data than a random tokenizer. Due to hardware limits, we only used the DistilBERT uncased base model in our system. It retains 97% performance of the original BERT but is 60% faster (Sanh et al., 2020). For comparison, we also used GloVe pretrained embedding to represent text and created a 2-layer LSTM model as our baseline. Shallow machine learning models were also explored for comparison. Figure 1 shows the pipeline of our experimental process.

Regarding feature extraction, we first used the DistilBERT tokenizer to vectorize the textual data then padded them into the same size. An attention mask was built to use binary vectors to differentiate padded zeros and word tokens. The vectors and mask were fed into the BERT model, and we stored the output of the last layer as the representation of text. Because the task was a text classification problem, we only used the "[CLS]" value in the text

representation for Logistic Regression (LR) and LSTM model, which was the first vector of each row. The length of each text representation was 768. For example, the sentence "Told my mom I hit 1200 Twitter..." was firstly tokenized into vectors "[101, 2409, 2026, 3566, 1045, 2718, 14840, 10474...]", and the [CLS] vector of DistilBERT output of this sentence was "[6.7492e-02, -1.6599e-01, 1.0417e-01, ...]", which had length of 768. For the Fully Connected (FC) model and all CNN models, we used the full output of the DistilBERT model as features, which were in shape of 136 x 768.

After extracting the features, we applied different models to predict the humor class. First, we implemented the LR model with default parameters. The package scikit-learn was used to build and train linear shallow models, such as LR, KNN, naive Bayes, random forest, and SVM. And then, we created a LSTM model. It consisted of two 32-node LSTM layers and a 32-node fully-connected layer. Also, we built a fully connected network, which had a 128-node dense layer and 64-node dense layer. In addition, we tried a CNN model which contained 2 64-node CNN-1D layer and a 128-node dense layer. For each network, we took the extracted features as input, and used a 1-node dense layer with sigmoid activation to collect output. In addition, dropout layers were used in each model to handle over-fitting problems.

For comparison, we also implemented a GloVe based LSTM. NLTK module was used to tokenize sentences and remove stop words and special characters. GloVe (Pennington et al., 2014) is pre-trained word vectors representation, which was used as the initial weights of the embedding layer. The GloVe+LSTM model used 50-dim embedding which connected to two 32-node LSTM layers. The output of LSTM was flattened and then fed into 16-node fully connected layer. And a 1-node dense layer with sigmoid activation was used to output probability of humorous class. It used the same model compiling parameters as the DistilBERT-based models.

Moreover, previous research proved that multiple scale of CNN layers can capture hidden features in different granularity, which improves performance (Cui et al., 2016; Yuan et al., 2018). Therefore, we build two multi-scale CNN models. The first model is DistilBERT+MultiScale CNN, Figure 2 shows its structure. It used five 64-node CNN-1D layers with different kernel size of [1, 2, 3, 4, 5]. These 5 layers could extract hidden information from the features in various granularity. Each CNN layer was then connected to a GlobalMaxPool1D layer to get a down-sampled representation in the shape of (batch size, 64). Also, dropout layers were applied on each output, and the 5 outputs were concatenated to a single vector in the shape of (batch size, 320). Subsequently, the combined vector was fed into a 512-node fully-connected layer. Rectify Linear Unit (ReLu) activation functions were applied to all CNN and fully-connected layers. Finally, a 1-node fully-connected layer with sigmoid activation function was added at the end of the network to collect the output value. We used 0.5 as a threshold to categorise the probability value into 0 (not humorous) and 1 (humorous) categories. In addition, we used "rmsprop" optimizer and binary cross entropy loss for stochastic gradient descent optimization.

In addition, we tried other multi-scale strategies, inspired by the deep multi-scale fusion hashing model (Nie et al., 2021). To differentiate two multi-scale models, we called this one MultiPool CNN. It used 5 different kernel size of MaxPooling layers to scale input features into different resolution. For each Maxpooling layer we set strides to 1 and used "valid" padding methods, and it connected to a 64-node CNN-1D layer with a kernel size of 1 and ReLu activation. Therefore, 5 CNN layers can extract features from different resolution of input. Similar to the multi-scale CNN, each output subsequently connected to a GlobalMaxPool1D layer and dropout layer. Finally, 5 different outputs were concatenated together and fed into a 512-node fully-connected layer. The output layer and compiling method are the same as the multi-scale CNN model.

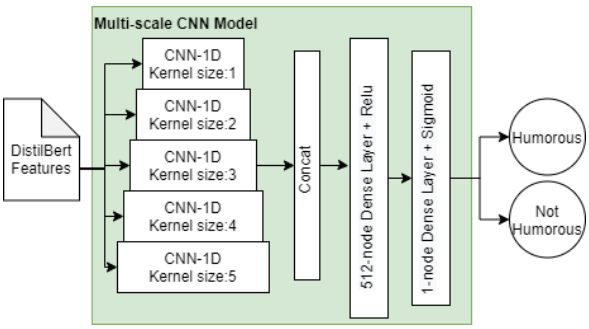

Figure 2: Structure of DistilBERT+Multi-Scale CNN

Model	F1@Dev	Accuracy@Dev	F1@Test	Accuracy@Test
Official Baseline	-	-	*0.857*	*0.884*
KNN	0.808	0.774	0.775	0.742
Naive Bayes	0.831	0.798	0.834	0.758
Random Forest	0.860	0.821	0.836	0.791
SVM	0.870	0.846	0.850	0.817
Voting Ensemble	0.853	0.821	0.817	0.774
GloVe+LSTM	0.888	0.850	0.885	0.853
DistilBERT+LSTM	0.884	0.842	0.896	0.862
DistilBERT+LR	0.898	0.867	0.904	0.880
DistilBERT+FC	0.904	0.877	0.925	0.907
DistilBERT+CNN	0.907	0.883	0.907	0.888
DistilBERT+MultiPool CNN	0.911	0.885	0.931	0.914
DistilBERT+MultiScale CNN	0.913	0.890	**0.937**	**0.921**

Table 1: Evaluation results of various implemented models on the dev set and test set (the gold data)

4 Experimental setup

We implemented our experiments on the university's virtual machine, which has a python environment with a shared 16 GB Tesla v100 GPU. Tensorflow 2.0 and Keras were used to build neural networks. Python "transformers" module was used to import the DistilBERT model. "Pandas" and "Numpy" modules were utilised to manipulate data, and we used "scikit-learn" to import basic machine learning models and evaluation metrics.

Moreover, we used the training set to train our models and took 20% split of training set as validation set to tune hyper-parameter in development stage. The dev set was used to assess performance of models in evaluation stage, and we applied our models on test set in order to submit results. In data preprocessing, we load the data into a pandas data frame, and tried to remove all the tags and special characters in the text. After that, we used the DistilBERT tokenizer to vectorise and encode the text into the format that the DistilBERT model required. In all the models, we set 1 and 2 as random seeds for Numpy and TensorFlow respectively. Also, we used early stopping strategy when training models. All DistilBERT based nerual network models were stopped at 23 epochs. Also, the batch size was set to 64.

Regarding evaluation, we evaluated our model on dev set and test set (gold data). F1 score and accuracy were used to evaluate performance. The accuracy score shows the ratio of correct prediction. Since the F1 score considers both precision and recall, which would be more informative metric, we selected our model based on the F1 score.

5 Results

We found that the features extracted by DistilBERT boosts the model performance significantly and it is difficult to generate a good results without using a pre-trained embedding or model. Due to hardware limits, we only tried the light version of BERT, and only implemented shallow layer models. Our official submission used DistilBERT+FC model, which achieved 92.41% F1-score and ranked 47th in task 7-1a. We modified our model in post-evaluation stage, and our best model is DistilBERT+Multi-scale CNN, which had 93.66% of F1 score and 92.10% of accuracy in using test set. The given official baseline is 85.7% of F1 score and 88.4% of accuracy. All of our DistilBERT based models and one GloVe embedding based model had better performance than the baseline. Detailed evaluation scores on dev set and test set were included in Table 1.

In comparison, the SVM model achieved an F1-score of 85.04% and an accuracy 81.70% on the test set, which is our best shallow model. However, all of these model generated scores lower than the given baseline.

Also, our GloVe-based LSTM model achieved 88.5% F1-score, and DistilBERT-based LSTM achieved 89.6%. It makes sense that transformer-based pretrained representation is better than traditional pretrained embedding. Due to efficiency considerations, we only used 32 nodes in the LSTM layer. So, these two models have poorer performance than other models using more nodes. Even the DistilBERT+LR model has higher F1 score (90.4%) and accuracy (88.0%). Surprisingly, a

	Not Humorous	Humorous
Not Humorous	337	48
Humorous	31	584

Table 2: Confusion matrix of DistilBERT+MultiScale CNN's result

simple 2-layer fully connected network achieved 92.5% F1 on test set. But in the dev set, DistilBERT+CNN outperformed the DistilBERT+FC model. The CNN model had more stable performance, which achieved 90.7% F1 in both dev and test set. Because we used the same epochs for all models, CNN may converge better than the fully-connected model. The multi-scale strategy further improved the performance of the CNN model, which is our best model.

However, our best model still made mistakes on predicting humorous classes of a few sentences. An error analysis is helpful to understand the wrong predictions. In the prediction results of the DistilBERT+MultiScale CNN model, 79 out of 1000 sentences are incorrectly predicted. The Table 2 is a confusion matrix of the result. It shows that 48 predictions were false positive, which assigned a Not-Humorous sentence into the Humorous class. For example, "I think in order to have a great business you have to like the product you're selling more than the money you get.", this sentence is misclassified as humorous. Also, 31 sentences are false negatives. Those sentences are labeled as humorous but ignored by our model. For example, "If alcohol influences short-term memory, what does alcohol do?", and "And then there's my dad...????". Those sentences hide the humor within the context, which is hard for a model to detect. Some of sentences are also difficult for us to understand why they are humorous. Since the humor labels were created based on subjective judgement, even human beings would have diverse opinions and understanding.

6 Conclusions

To conclude, we explored various methods to build humour detection classifiers for task 7-1a. Models like Logistic Regression, LSTM, FC, CNN were used, and pre-trained models like DistilBERT were introduced to generate an accurate vector representation for textual data. Our best model is the DistilBERT+MultiScale CNN, which achieved 93.7% F1-score and 92.1% accuracy on the test set. We focused on applying multi-scale strategy on modelling, and compared different models. And our results shows that CNN are more suitable for this task than LSTM FC and other shallow models. Also, we found that pre-trained embeddings, weights or representations are crucial for our model performance. We only explored multi-scale from the wide dimension, this strategy can also be used in deep dimension. In the future, a deeper network with more nodes can be explored and the full version of BERT model can be exploited.

References

Christopher J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification.

Joseph L. Fix, Evelyn; Hodges. 1951. Discriminatory analysis. nonparametric discrimination: Consistency properties. *USAF School of Aviation Medicine, Randolph Field, Texas.*

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

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

X. Nie, B. Wang, J. Li, F. Hao, M. Jian, and Y. Yin. 2021. Deep multiscale fusion hashing for cross-modal retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(1):401–410.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Webb G.I. Sammut C. 2011a. *Naïve Bayes. In: Encyclopedia of Machine Learning*. Springer.

Webb G.I. Sammut C. 2011b. *Random forests. In: Encyclopedia of Machine Learning*. Springer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations.

Q. Yuan, Y. Wei, X. Meng, H. Shen, and L. Zhang. 2018. A multiscale and multidepth convolutional neural network for remote sensing imagery pan-sharpening. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(3):978–989.