Humor Detection and Ranking

Bartol Freškura, Filip Gulan, Damir Kopljar

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{bartol.freskura, filip.gulan, damir.kopljar}@fer.hr

Abstract

In this paper, we consider the task of comparative humor ranking in two manner: detecting which tweet of two is more humorous and ranking the given tweets by how humorous they are in three classes. We opted for different approaches based on recent deep neural models in order to eschew manual feature engineering. In evaluation section we experimented with bi-directional LSTMs and CNNs, in combination and separately. For constructing feature vectors we also experimented with *GloVe* word embedding and character embedding. The system was tuned, trained and evaluated on SemEval-2017 Task 6 dataset for which it gives representative results.

1. Introduction

Understanding humor expressed in the text is a challenging natural language problem which has not yet been addressed extensively in the current AI research. Humor is often subjective and relies on the vast knowledge base, which is sometimes hard to reason, even for humans. It is also important to say that what is humorous today might not be humorous tomorrow due to the fact that humor can be trend dependent. In this paper, we describe a system for humor detection and ranking. Our system is designed to solve two tasks. For the first task, the system is given two tweets and it should predict which tweet is more humorous. For the second task, the system is given a set of tweets and it should rank them in three categories (2 for the most humorous tweet, 1 for top ten humorous tweets and 0 otherwise). To learn and test our model we used a novel dataset that was given in SemEval-2017 Task 6 (**?**). Dataset consists of tweets that viewers sent as part of the Comedy Central show @midnight. For every episode topic for that show was defined and viewers were asked to send a humorous tweet about given topic.

2. Related Work

In the last few years there were a lot of approaches in humor detection (**?**; **?**; **?**; **?**; **?**). Some of those works (**?**; **?**; **?**), have also acquired humor dataset from Twitter.

Most of those related works separates the apprehension of humor into two groups: humor and non-humor, basically a binary classification. Using this representation ignores continuous nature of humor, while also not accounting for the subjectivity in perceiving humor (**?**).

3. Arhitecture

In this section, we describe architecture of our system. Our most complex architecture that we tested in this paper consists of bi-directional *Long Short Term Memory*, further referred as Bi-LSTM, convolutional neural network, further referred as CNN, and fully connected neural network.

The whole pipeline was built using the open source frameworks Tensorflow(**?**)[1] and scikit-learn(**?**)[2].

3.1. Recurrent Neural Networks

The main idea behind RNNs lies in retaining information from "history". In the context of NLP, history refers to observing the context of the sentence up to the currently processed word. Despite the promising results in short sentences, RNN losses its performance dramatically with the increasing sentence length due to the gradient vanishing (**?**) and exploding problems (**?**).

LSTMs were designed with the purpose of correcting the RNNs shortcomings. Although LSTM can successfully capture the past context, it is sometimes good to have an insight at the future sentence context. Bi-LSTMs model this by adding an extra LSTM layer which has a reversed information flow meaning that the information is propagated from the end of the sentence towards the beginning. Output of a Bi-LSTM is a concatenated vector of the two opposite LSTM layers.

3.2. Convolutional Neural Networks

CNN networks are famous for their appliance in the *Computer Vision* domain but have also demonstrated an ability to extract morphological information from word characters, encoding them into neural representations. We first create a hash map of all characters that appear in the dataset where values are arbitrarily assigned integer values. All sentence characters are then represented using their mapped integer values but the padding is also applied on the word level as shown in Figure 1. Encoded sentence represents an input which is fed into a trainable character embedding layer of $C_e \times V$ dimensions, where C_e is the character embedding size, and V is the number of unique characters in the dataset.

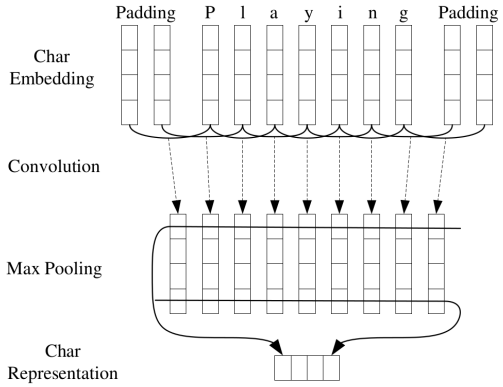
One dimensional convolution is applied after the dropout which yields character feature vectors. Generated vectors are concatenated with the word embedding vectors. Word embeddings are pretrained vectors that model the inter word relatedness. We used Twitter Glove(**?**)[3] embeddings with vector size of 100.

[1] https://www.tensorflow.org/

[2] http://scikit-learn.org/stable/index.html

[3] https://nlp.stanford.edu/projects/glove/

Figure 1: CNN arhitecture



4. Dataset

In this paper, we trained and evaluated models on the dataset that was created from tweets of viewers who watched TV show @midnight. As part of this "game-show" viewers were asked to write a humorous message about a topic that was announced in the show. The day of the ensuing episode, @midnight would create a post that announced the top-10 tweets from the previous episode. The whole dataset consists of 11,685 tweets about 106 different topics. Every tweet in the dataset is labeled in one of the three labels. Label 2 indicates winning tweet, label 1 indicates that tweet was selected as top-10 and label 0 intended for all other tweets. A number of tweets per topic vary among different topics, but 71% of topics contain at least 90 tweets. Topics with the lowest number of tweets have 20 tweets, and topics with the highest number of tweets have 180 tweets. It is also important to note that for some topics like "FastFoodBooks" external knowledge is required to understand the humor, while for other like "IfIWerePresident" it isn't.

5. Experiments

We divide this section into two parts: the former one reporting results on the non-evaluation data, and the latter results on the official evaluation data.

5.1. Non-evaluation data results

During the model training and evaluation we used a k-fold cross-validation technique ($k = 35$) to properly optimize our models hyper-parameters. Grid search method was not feasible in our case due to the large parameters search space and slow training time. Our method was to change each parameter value from the low to high extremes and see how it affects the model performance, in the end interpolating to find near optimal parameters. In addition to using dropout for regularization, we also employed the early-stopping(**?**) method to achieve the best possible validation performance. Table 1 shows the final hyper-parameters used in training and validation. For the optimization algorithm we used Adam(**?**).

Our model is trained to maximize the class probability $p(y|x)$ using cross-entropy as the loss function. Output 1 means that the first tweet was funnier and 0 otherwise. In our results, we report results in form of four metrics: accuracy, precision, recall and F1 score. All entries represent 95% confidence intervals calculated from the 35 validation runs.

Table 1: Final network hyper-parameters

Hyperparameter	Value
Dropout rate	0.5
BLSTM hidden state	128
Timestep	30
Learning rate	0.0002
CNN filter size	60
FC layer size	256
Character embedding layer size	50

Table 2: 95% confidence scores on the non-evaluation data for all models (in $\%$)

	Baseline	BLSTM	CNN	CNN + BLSTM
Accuracy	50.1 ± 0.2	$\mathbf{67.8 \pm 1.7}$	52.9 ± 1.4	67.0 ± 1.7
Precision	49.8 ± 0.3	$\mathbf{68.2 \pm 1.7}$	54.1 ± 1.4	67.4 ± 1.7
Recall	50.3 ± 0.2	$\mathbf{67.8 \pm 1.7}$	52.9 ± 1.4	67.0 ± 1.7
F1	50.0 ± 0.2	$\mathbf{67.8 \pm 1.7}$	53.2 ± 1.4	67.0 ± 1.7

In Table 2 are the results from the non-evaluation data. Baseline model randomly guesses the funnier tweet and is expected to have metrics around 50%.

5.2. Evaluation data results

In this section we demonstrate how our best model compares with other solutions on the official evaluation data. Note that the accuracy and distance measurements listed in Table 3 and Table 4 are defined by the task organizers(**?**).

Write something about comparison.

6. Conclusion

We proposed three different models for solving comparative humor ranking task: pairwise comparison task and direct ranking classification task. All three models use deep learning architecture combining approaches of recurrent and convolutional neural networks.

For pairwise comparison task best results were achieved using MODELMODELNAME and were XX% on develop-

Table 3: Official Task A results in comparison with our model

Team	Accuracy
HumorHawk	67.5
CNN + Bi-LSTM	66.7
TakeLab	64.1
HumorHawk	63.7
DataStories	63.2
Duluth	62.7

Table 4: Official Task B results in comparison with our model

Team	Distance
Duluth	0.872
TakLab	0.908
Our model	0.9
QUB	0.924
QUB	0.924
SVNIT@SemEval	0.938

ment set and YY% on unseen evaluation data, and for direct ranking classification task best results were achieved using MODELMODELNAME and were XX% on development set and YY% on unseen evaluation data. Model evaluation on final unseen data is done using official evaluation scripts given in SemEval-2017 Task 6.

We have compared our results with the results of other task participants resulting in our model ranking X. out of Y on the Task A, and Z out of K on the Task B. The main distinction between our model and competitive models is in the lack of hand engineered features which indicates that automatic feature extraction using deep learning framework has a great prospect in this task and requires further work.

For the next step we would experiment with specially adapted word embeddings trained only on the humor containing corpus. We believe it is crucial for word vectors to learn semantic meaning from the domain specific data because of the complex humor structure. TODO: One short sentence about number of parameters and overfitting.

Acknowledgements

Hvala svima u studiju i režiji.

References