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 (Potash et al., 2016). 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

Related work dude.

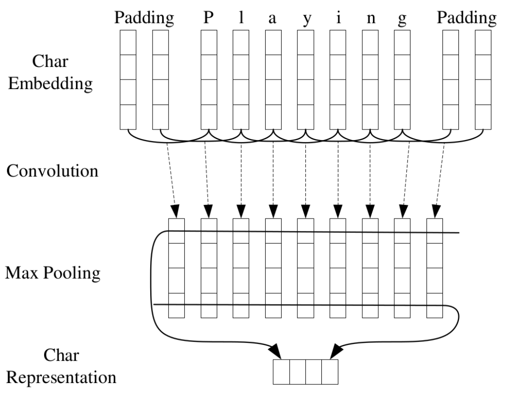
3. Arhitecture

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.

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 (Bengio et al., 1994) and exploding problems (Pascanu et al., 2013).

Figure 1: CNN arhitecture



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 char- acters, 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 vec-

Table 1: This is the caption of the table. Table captions should be placed *above* the table.

Heading1	Heading2
One	First row text
Two	Second row text
Three	Third row text
	Fourth row text

tors are concatenated with the word embedding vectors. Word embeddings are pretrained vectors that model the inter word relatedness. We used twitter Glove embeddings with vector size of 100.

4. Dataset

5. Experiments

5.1. Figures

Here is an example on how to include figures in the paper. Figures are included in LaTeX code immediately *after* the text in which these figures are referenced. Allow LaTeX to place the figure where it believes is best (usually on top of the page of at the position where you would not place the figure). Figures are referenced as follows: "Figure **??** shows ...". Use tilde (˜) to prevent separation between the word "Figure" and its enumeration.

5.2. Tables

There are two types of tables: narrow tables that fit into one column and a wide table that spreads over both columns.

5.2.1. Narrow tables

Table 1 is an example of a narrow table. Do not use vertical lines in tables – vertical tables have no effect and they make tables visually less attractive.

5.3. Wide tables

Table 2 is an example of a wide table that spreads across both columns. The same can be done for wide figures that should spread across the whole width of the page.

6. Experiments

Math expressions and formulas that appear within the sentence should be written inside the so-called *inline* math environment: $2 + 3$, $\sqrt{16}$, $h(x) = \mathbf{1}(\theta_1 x_1 + \theta_0 > 0)$. Larger expressions and formulas (e.g., equations) should be written in the so-called *displayed* math environment:

$$b_k^{(i)} = \begin{cases} 1 & \text{if } k = \text{argmin}_j \|\mathbf{x}^{(i)} - \mu_j\| \\ 0 & \text{otherwise} \end{cases}$$

Math expressions which you reference in the text should be written inside the *equation* environment:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} b_k^{(i)} \|\mathbf{x}^{(i)} - \mu_k\|^2 \qquad (1)$$

Now you can reference equation (1). If the paragraph continues right after the formula

$$f(x) = x^2 + \varepsilon \qquad (2)$$

like this one does, use the command *noindent* after the equation to remove the indentation of the row.

Multi-letter words in the math environment should be written inside the command *mathit*, otherwise LaTeX will insert spacing between the letters to denote the multiplication of values denoted by symbols. For example, compare $Consistent(h, \mathcal{D})$ and
$\mathit{Consistent}(h, \mathcal{D})$.

If you need a math symbol, but you don't know the corresponding LaTeX command that generates it, try *Detexify*.[1]

7. Conclusion

Conclusion is the last enumerated section of the paper. It should not exceed half of a column and is typically split into 2–3 paragraphs. No new information should be presented in the conclusion; this section only summarizes and concludes the paper.

Acknowledgements

Hvala svima u studiju i režiji.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2016. # hashtagwars: Learning a sense of humor. *arXiv preprint arXiv:1612.03216*.

[1]http://detexify.kirelabs.org/

Table 2: Wide-table caption

Heading1	Heading2	Heading3
A	A very long text, longer that the width of a single column	128
B	A very long text, longer that the width of a single column	3123
C	A very long text, longer that the width of a single column	-32