funny3 at SemEval-2020 Task 7: Humor Detection of Edited Headlines with LSTM and TFIDF Neural Network System

Xuefeng Luo, Kuan Tang
Linguistics Department, University of Tuebingen, Germany
firstname.lastname@student.uni-tuebingen.de

Abstract

This paper presents a neural network system where we participate in the first task of SemEval-2020 shared task 7 "Assessing the Funniness of Edited News Headlines". Our target is to create a neural network model that predicts the funniness of edited headlines. We build our model using a combination of LSTM and TF-IDF, then a feed-forward neural network. The system manages to slightly improve RSME scores regarding our mean score baseline.[1]

1 Introduction

How the computer recognizes, generates, and analyzes humor is an interesting but meaningful challenge for artificial intelligence (AI). Not only creativity and sophistication of language are needed for the sense of humor, but also world knowledge, empathy, and cognitive mechanisms which are not so easy for technologists to model theoretically.

Several stable and capacity large enough datasets have been built for training models in computational humor studies. These datasets may contain different contexts. Mihalcea Strapparava (2006) set up funny one-liners which contain more than 24,000 onliner jokes. Filatova (2012) collects regular and sarcastic Amazon product reviews to identify sarcasm on a sentence level or for a specific phrase. Khodak et al. (2017) introduce the *Self-Annotated Reddit Corpus* (SARC) which is a corpus that has 1.3 million sarcastic statements for training and evaluating systems for sarcasm detection. Hossain et al. (2017) set up *Mad Libs*, which is a popular fill-in-the-blank game judged by the human. And then, Hossain et al. (2019) introduce a novel dataset *Humicroedit* that contains original media news headlines from an online posted website *Reddit* and enables several humor tasks for humor understanding, predicting, generating, recommending and ranking. The latest one is *FunLines*, which is a competitive game that not only allows players to edit news headlines but also rate the funniness of headlines edited by others. Each dataset has its unique purpose, but how to understand headlines and what makes a few words in one headline so funny are becoming new problems for AI.

In this paper, we want to test how machines understand humor generated by such short edits rely on Humicroedit to predict the mean funniness of edited headlines.

2 Related work

How to generate humor is not as easy as we think, especially for computers. Binsted et al. (1997) developed a formal model producing riddles and implemented it in *jape*, a computer program which generates simple punning riddles and confirmed that jape's output texts are indeed jokes after evaluating the program by 120 children. Stock and Strapparava (2002) established *HAHAcronym* to realize the irony reanalysis and generation of acronyms in a concentrated but unrestricted context. Petrovic and Matthews (2013) rely on large quantities of unlabeled data to deal with generating a fixed syntactic structure – *[I like my X] like [I like my Y], Z* – where X and Y are nouns and Z describes X and Y, and develop an algorithm called *Libitum* that helps humans generate humor in a Mad Lib, which is a popular fill-in-the-blank game. The algorithm is based on a machine-learned classifier that determines whether a potential fill-in word is

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1013–1018
Barcelona, Spain (Online), December 12, 2020.

funny in the context of the Mad Lib story. Hossain et al. (2020b) use an online game for players to change a single word or entity in a regular headline from a news source and edit regular news headlines and that makes the humor generation process highly constrained.

As for humor detection, Reddit and Davidov et al. (2010) utilize the *Semi-supervised Sarcasm Identification Algorithm* (SASI), which is the first robust algorithm for recognition of sarcasm, to test two different database: 5.9 million tweets from Twitter and 66,000 product reviews from Amazon. Reyes et al. (2012) also use 50,000 texts retrieved from Twitter to evaluate the representativeness and relevance of the humor and irony. Barbieri and Saggion (2014) approach the detection of irony and humor as a classification problem applying supervised machine learning methods to the Twitter corpus. Besides Twitter, Kiddon and Brun (2011) apply methods from metaphor identification to *"that's what she said"* (TWSS) recognition. Khodak et al. (2017) use the *Self-Annotated Reddit Corpus* (SARC), a 1.3 million sarcastic statement corpus, to detect sarcasm.

The analysis of humor can be divided roughly into two different levels: word-level and text-level (or more specifically, sentence-level). In word-level, Engelthaler and Hills (2018) collect 4,997 words to rate humor norms by analyzing several factors that influence the judgments. Based on that, Westbury and Hollis (2019) analyzing the semantic, phonological, orthographic, and frequency of words to predict the original humor rating standard and the score of previously unsettled words. Their reliability is higher than the half reliability in the original standard, which is estimated by segmentation based on gender or age line. While in sentence-level, Shahaf et al. (2015) set up a dataset that contains crowd-sourced captions that are judged by people from *the New Yorker*. Then they formulate a pairwise comparison task by giving a cartoon and two captions based on the data to let the computer determine which one is funnier. They also analyze different variations of the same one-joke and figure out factors affecting the level of perceived humor. Both West and Horvitz (2019) and Hossain et al. (2019) use headlines to analysis what makes headlines funny, but West and Horvitz's work focus on pairs of headlines which is aligned to similar but serious looking, while Hossain et al. (2019) employed editors to create headlines first and then used Web-based game to produce and analyze the modified headlines. And in the latest work, Hossain et al. (2020b) provide instant rating feedback to players when they rate a headline in a humor competition, and that makes the game fair and engaging.

Let us summarize this section. The computational linguistics studies of humor on generation, detection, and analysis are not parallel development, these research directions are related to each other, and we need to develop new algorithms to complete more refined tasks based on the new database.

3 Task Descriptions

This task is based on the publication of Hossain et al. (2019). It studies the humorous of edited headlines and releases its dataset called Humicroedit (Hossain et al., 2019). The task (Hossain et al., 2020a) asks us to predict the funniness of an edited headline, given its original headline and edited headline. Each headline is allowed to only change one word or one named entity. Table 1 gives an example of those original and edited headlines and its funniness scores. The scores are judged by 5 human judges, and the average of judge scores is taken as final funniness score. Each score provided by a single judge is from 0-3, where 0 represents Not Funny while 3 as Funny. The task evaluates the predictions by measuring the Root Mean Square Error (RMSE) of the entire test set. The baseline system takes the mean of true scores.

Original Headlines	**Edited Word**	**Funniness Scores**
Fox's James Murdoch rebukes **Trump** over Charlottesville	grits	0.2
Trump border wall : Texans receiving letters about their **land**	barbecue	2.2
Former presidents raise $ 31 million for **hurricane** relief fund	president	1.4
Royal wedding : Meghan 's **dress** in detail	elbow	2.0
Can Democrat Doug Jones pull off an upset in **Alabama** ?	bed	2.0
Trump Budget Gambles on Having This **Equation** Right	Bet	0.8

Table 1: Examples of original and edited headlines (edited word in **bold text**) and their funniness scores

4 System Overview

We present a two-component neural network system. The first component is a combination of multiple features, included a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) neural network, a TF-IDF (Salton and McGill, 1986) representation and the difference between original headline target words and edited words (subtraction of two vectors). Beyond our LSTM and subtraction, we choose a FastText (Mikolov et al., 2018) model to embed our headlines and words. All features are concatenated together and fed to the second component. The second component is a two-layer feed-forward neural network. Figure 1 shows an overview of our system.

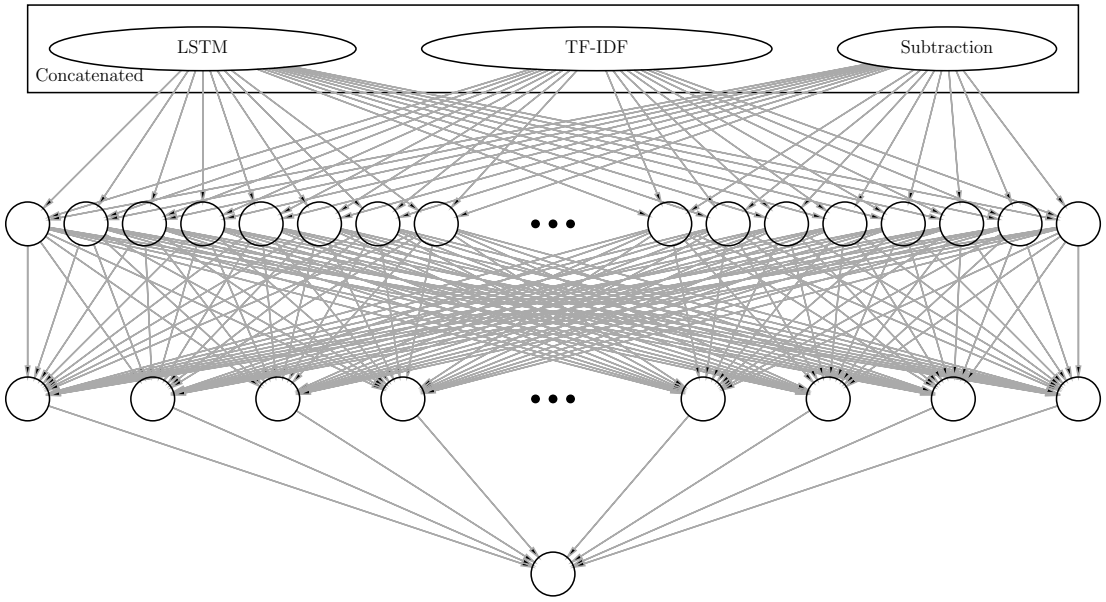

Figure 1: An overview of the model: first concatenate LSTM, TF-IDF and subtraction of original and edited words, then feed it to feed-forward neural network

4.1 Embeddings

Fasttext (Mikolov et al., 2018) is a fast and efficient word-to-vector model. It can be trained with any text corpora and produce vectors for any given words or symbols, even for unseen words, because fasttext not only takes advantage of words but the character level n-grams as well. Skip-gram model is used, in our case, in fasttext training. It takes the target words to produce their contexts.

4.2 LSTM

After we retrieved word embeddings from Fasttext, we feed them into a LSTM (Hochreiter and Schmidhuber, 1997) model. LSTM is a recurrent neural network that is good at processing sequences, like texts. This type of neural network maintains two types of weights: a hidden state and a cell state. For each timeslice, it takes the previous hidden state, previous cell state, and current input (word vector representation). Then, it calculates and feeds the current cell state and current hidden state to the next timeslice. In the end, we can either take the final cell state or take each timeslice cell state and concatenate them together. We conduct a study and find out that taking each timeslice cell state works better. Also, bidirectional recurrent neural networks are good at maintaining information from both directions. We also choose a bidirectional recurrent neural network to keep tracking both direction timeslice information.

4.3 TF-IDF

TF-IDF (Salton and McGill, 1986) reflects the importance of a single word in the corpus. It takes the term frequency (TF) times the inverse document frequency (IDF) so that it reduces the effects of words

that occur more frequently in the entire corpora. This helps adjust the words which are more frequent in general. We use the TF-IDF of each headline in word level, and take them as our second feature.

4.4 Subtraction

Since we want to compare two headlines, we subtract original and edited word vectors where we embed it from Fasttext (Mikolov et al., 2018) to maintain their differences. Since this task allows to modify named entities that may contain multiple words, which may lead to different word length. In these cases, we obtain word vectors from fasttext (Mikolov et al., 2018) for each word, then take the average of all word vectors as combined word vectors.

4.5 Feed-forward Neural Network

After we get the above feature vectors. we feed this into a two-layer feed-forward neural network with single neural output where the upper layer doubles the size of the lower layer. The numeric output value directly becomes our final prediction.

5 Dataset and Experimental Setup

5.1 Dataset

We mainly use Humicroedit dataset (Hossain et al., 2019) to train, evaluate, and test our model. Humicroedit collects original headlines from Reddit (`reddit.com`) and then editing headlines and judging those headlines using Amazon Mechanical Turk (`mturk.com`). Each of the headlines has 3 edited versions, resulting in over 15000 entries of editing (over 5000 original headlines). Data is randomly split into three subsets: training set (64%), development set (16%), and test set (20%). We use additionally FunLines (Hossain et al., 2020b) dataset to train our model as well. FunLines has a similar format to Humicroedit. It allows online users to both edit and judging those headlines.

5.2 Experimental Setup

In our case, we train our embedding model with ABC (Australian Broadcasting Corp.) news headlines (`https://www.kaggle.com/therohk/million-headlines`). This corpus contains published news headlines over seventeen years. After training our embedding model, we vectorize our headlines, original target word, and edited word along with all punctuations and symbols. We set this model with 300 dimension vector, 3-6 character level n-gram, 5 negatives sampled. Here, we use special characters ("|||") to hold the place where the words are edited. Named entities are concatenated together using underscore ("_"), to maintain the same length of edit and original headlines. The LSTM (Hochreiter and Schmidhuber, 1997) layer has 128 cells and the activation function is set to tanh. The feed-forward neural network has 64 and 32 cells and the activation function is set to sigmoid. In order to prevent over-fitting, we apply 0.5 dropouts between the first component and the second component, while we apply 0.2 dropouts between two feed-forward neural network layers.

6 Results

We achieve a score of 0.57237 RMSE in the competition and rank 30th place. Then we later modify our system and further improve the result to 0.56786 RMSE score. Even though we do not reach a higher position, but we conduct several ablation studies to understand which subsystem contributes most.

6.1 Ablation Study

We conduct 3 pair subsystem studies to discover which sub-parts do best (Table 2 shows the results of all subsystems, where concatenation of all three features achieve the best score). The first pair systems are LSTM (Hochreiter and Schmidhuber, 1997) representation whether to use only the final cell state or concatenate all timeslice cell states. Result shows that the concatenated cell state system improves slightly. Then, we compare the LSTM system and the TF-IDF (Salton and McGill, 1986) system with and without subtraction of the original and edited words. TF-IDF with subtraction showed better results. Finally, we add all features together where it further improves slightly to achieve the best scores.

Ablation Subsystem	RMSE
Baseline	0.57469
LSTM with final cell state	0.59105
LSTM with all timeslice hidden state	**0.58611**
LSTM without subtraction	0.58611
LSTM with subtraction	**0.58361**
TF-IDF without subtraction	0.58990
TF-IDF with subtraction	0.58619
Competition System	0.57237
Modified all-together System	**0.56786**

Table 2: RMSE of all ablation systems

6.2 Errors Analysis

Figure 2 indicates the confusion matrix of our system prediction values and the true values. As the figure indicates, our system predicts mostly from 0.34 to 1.47 while the true values are variously from 0 to 2.6. This shows that our system predicts less spread and hardly learns from small differences within the range of 0 to 2.6.

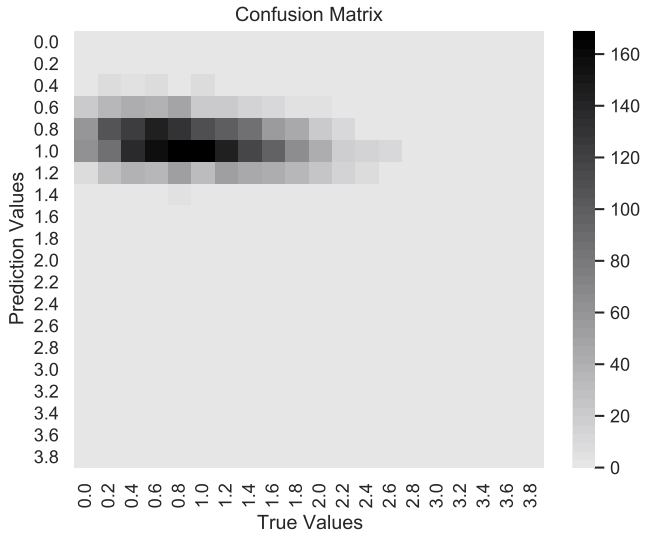

Figure 2: Confusion matrix of errors analysis

7 Conclusion

In this shared task, we present a neural network system combined with LSTM and TF-IDF where it slightly improves RMSE compared to our baselines. Though our system only ranks 30th place of the competition, we still construct a neural network system that predicts funniness score roughly and we study the contributions of our subsystems where we find that the combination of LSTM, TF-IDF and subtraction slightly improved the predictions.

References

Kim Binsted, Helen Pain, and Graeme D Ritchie. 1997. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition*, 5(2):305–354.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Tomas Engelthaler and Thomas T Hills. 2018. Humor norms for 4,997 english words. *Behavior research methods*, 50(3):1116–1124.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *Lrec*, pages 392–398. Citeseer.

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

Nabil Hossain, John Krumm, Lucy Vanderwende, Eric Horvitz, and Henry Kautz. 2017. Filling the blanks (hint: plural noun) for mad libs humor. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 638–647.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. " president vows to cut¡ taxes¿ hair": Dataset and analysis of creative text editing for humorous headlines. *arXiv preprint arXiv:1906.00274*.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020a. Semeval-2020 Task 7: Assessing humor in edited news headlines. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Nabil Hossain, John Krumm, Tanvir Sajed, and Henry Kautz. 2020b. Stimulating creativity with funlines: A case study of humor generation in headlines. *arXiv preprint arXiv:2002.02031*.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 89–94. Association for Computational Linguistics.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1065–1074.

Oliviero Stock and Carlo Strapparava. 2002. Hahacronym: Humorous agents for humorous acronyms. *Stock, Oliviero, Carlo Strapparava, and Anton Nijholt. Eds*, pages 125–135.

Robert West and Eric Horvitz. 2019. Reverse-engineering satire, or "paper on computational humor accepted despite making serious advances". In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7265–7272.

Chris Westbury and Geoff Hollis. 2019. Wriggly, squiffy, lummox, and boobs: What makes some words funny? *Journal of Experimental Psychology: General*, 148(1):97.