CS 224N Project Final Report
An Ernie-st Attempt at Pun Recognition

Amy Wang
Department of CS + Linguistics
Stanford University
jwang19@stanford.edu

Zheng Yan
Department of CS
Stanford University
yzh@stanford.edu

Custom project. Mentor: Annie Hu

Abstract

This work extends a previous statistical approach based on cognitive science for humor recognition and rating – specifically, classifying English strings as puns or not, and then determining their funniness scores – by using neural methods to estimate distributions that previously relied heavily on human judgements. We integrate various different layers of a pretrained BERT for these distribution estimations, including the attention heads and a the masked language model output. However, we failed to achieve a significant improvement over a baseline end-to-end LSTM classifier.

1 Introduction

Throughout the past decade, artificial intelligence (AI) has made astonishing progress on tasks previously thought untenable by machines. However, there are still large sets of tasks, many of which are regularly done by humans, that AI continues to do extraordinarily poor in. One such task is the classification of humor. Humor is a strange yet important aspect of human cognition, with profound impacts on human behavior. Existing approaches to automatic humor classification consist of end-to-end neural network and naive Bayes approaches, and have not seen much success. In this study, we hope to explore solutions to the problem of automatic humor classification that integrate state-of-the-art NLP and cognitive science theories.

We would like to create a system that accomplishes two tasks: (1) classifies an utterance as a pun, or not a pun; and (2) gives a rating to the funniness of the pun. In doing so, we hope to contribute to the broader progress of bettering AI-human interactions, and present techniques that can make AI companions appear friendlier. We build the first natural language processing system that implements the cognitive science-based model presented by Kao et al. in 2015. This model was used to effectively predict funniness of a pun, based on ambiguity and distinctiveness metrics (explicated in the following section). We plan to use intermediate layers and outputs of BERT to calculate these metrics, which was done manually via crowdsourcing in the original paper. Although our pipeline is not entirely end-to-end (from raw sentence to result), our model automates the slowest step of the process. Future steps may seek to achieve complete automation of the process, with similarly high effectiveness.

1.1 Related Work

Most existing literature on humor recognition is largely based on application of domain knowledge from psychology, sociology, and linguistics. Raskin's theory of humor states that "every joke is compatible with two scripts, and those two scripts oppose each other in some part of the text, usually in the punch line"; Taylor & Mazlack (2004) use this theory as a foundation for an algorithm that uses n-grams to calculate the probability of these "two scripts" [14]. Mihalcea & Pulman (2007) showed that humorous and non-humorous texts were computationally separable using SVMs and Naive Bayes classifiers, and discovered a range of semantic features that distinguished the two, such as negation, polarity of word sentiment, and human-centeredness [15]. Zhang & Liu (2014) used linguistic features including alliteration and rhyme, POS ratio and patterns, lexical ambiguity, and word sentiment to train decision trees on Twitter jokes [16]. Zhang et. al. (2017) introduce a number of new features in addition, including contextual weighting and subjectivity [3].

2 Background

The model defined by Kao et al. is as follows: suppose a given sentence $\vec{w}$ contains a phonetically ambiguous segment h (such as *number*), with a given primary interpretation h_1 and meaning m_1 (algebraic object); as h is phonetically ambiguous, there also exists an approximate homophone h_2 and meaning m_2 (more numb). The authors propose that we can approximate the "funniness" of said sentence by evaluating its ambiguity and distinctiveness with respect to m.

2.1 Ambiguity

Ambiguity is formally quantified as the entropy of the distribution $P(m|\vec{w})$. Intuitively, ambiguity measures the relative fluency of the sentence when either interpretation of h is plugged in. High ambiguity is indicative of funniness, as this means the probability mass of the sentence is not concentrated on any one of the possible meanings; funniness depends on subversion of expectations, but a relatively nonsensical secondary interpretation is not particularly good for subverting any expectations. For instance, “The hair ran rapidly across the field.” is not particularly funny, but "The magician got so mad he pulled his hare out" is, because the meaning $[\![hair]\!]$ is far less fitting in the first context than $[\![hare]\!]$ is in the second. Thus, while phonetically ambiguous words exist in both sentences, only the latter sentence supports both interpretations sensibly, allowing for a humorous incongruency to emerge.

In practice, the above definition $P(m|\vec{w})$ can be calculated using the following:

$$P(m|\vec{w}) = \sum_{\vec{f}} P(m, \vec{f}|\vec{w}) \tag{2}$$

$$\propto \sum_{\vec{f}} P(\vec{w}|m, \vec{f}) P(m) P(\vec{f}) \tag{3}$$

$$= \sum_{\vec{f}} \left(P(m) P(\vec{f}) \prod_i P(w_i|m, f_i) \right) \tag{4}$$

where m and $\vec{w}$ are as described above, and i is the word index. $\vec{f}$ is a sentence-length vector of indicator variables marking whether the word at index i is in semantic focus or not (e.g. if it reflects the topic unique to sentence with ambiguous word h, as opposed to being sampled from random noise as in a unigram). $P(f)$ is constant, since it is a priori assumed that it is equally likely that any word in the sentence is in semantic focus. Given whether a word is in semantic focus, its probability can be determined by Bayes’ law using its probability of appearance given the sentence’s meaning.

2.2 Distinctiveness

Intuitively, distinctiveness is the ability of the sentence to draw attention to ambiguity. For instance, "Look at that hare" is technically ambiguous, but the phrase does little to point it out, and so it is unlikely for anyone to find it comical. However, the utterance

Two fish are in a tank. One says to the other, "You man the gun, and I’ll drive."

can be considered to be fairly funny, due to the uniqueness of the supporting words for one interpretation of the sentence or the other. This idea is derived from the theory that lexicographical priming is necessary to facilitate the disambiguation of sentences, a process which leads to humor.

In practice, this is calculated as the following: Let F_1 denote the distribution $P(f|m_1, \vec{w})$ and F_2 denote the distribution $P(f|m_2, \vec{w})$, where each is the distribution of "relevance" between each word in the sentence and the sentence meaning. Distinctiveness is just the symmetric KL divergence of F_1 and F_2, with $p(f|m, w)$ calculated using Bayes' rule similarly as above using the following:

$$P(\vec{f}|m, \vec{w}) \propto P(\vec{w}|m, \vec{f}) * P(\vec{f}|m)$$

2.3 Calculating Funniness

[1] provided different suggestions of how to integrate the ambiguity and distinctiveness scores into an overall funniness score. Based on the authors' analyses, it appeared that both ambiguity and distinctiveness correlated with funniness, and a non-linear composition of ambiguity and distinctiveness is likely to be most effective for both classifying and rating humor.

3 Approach

To recap, the ambiguity of a sentence $\vec{w}$ was modeled as the entropy of the distribution of the probability of different meanings of the sentence on the words in $\vec{w}$, while distinctiveness of $\vec{w}$ was modeled as the KL-divergence of the distributions of relevance conditioned on the different meanings of $\vec{w}$. Therefore, the problem can be reduced to that of distribution estimation. In the study, various corpus-based procedures were used to estimate these distributions, such as incorporating crowdsourced human estimates of PMI and counting frequency of non-stop-word pairs. We would like to try to improve the performance by using neural methods to estimate them instead.

There are two key distributions to be estimated. One of them is the probability of each word occurring given a context ($P(\vec{w}|m)$), and the other is the probability that a word is semantically relevant in a sentence given a context ($P(\vec{f}|m, \vec{w})$). While the second can be approximated from the first, given a few simplifying assumptions, we believe it is still fruitful to also attempt a direct approach.

3.1 Ambiguity – Entropy of $P(m|\vec{w})$

To calculate ambiguity, we make a simplifying assumption that if one removes a single word from a sentence, not too much is lost of the information needed to encode the sentence's meaning. From this, we can represent the probability of each word w_i conditioned on sentence meaning m as the probability of w_i conditioned on all the other words in the sentence given some interpretation, reducing this problem to that of masked language modeling. For example, we painfully dissect the joke:

> Two fish are in a tank. One says to the other - you man the gun, I'll drive.

This joke derives its humor from two ambiguous interpretations:

> A) Two fish are in an armored vehicle...
> B) Two fish are in a fishtank...

In concordance with the formula presented by the authors of [1], in order to determine ambiguity, we first calculate $P(\text{Armored vehicle | joke}) = P(\text{Two | fish... armored vehicle}) * P(\text{Fish | two.. are.. armored vehicle}) *$ ... and so on and so forth. Then, we calculate $P(\text{Fish}) * P(\text{two})$, .. as unigram frequencies to account for the possibility that these terms were sampled from random noise. We ignore all conjunctions, pronouns, prepositions, and "the", as their likelihoods are unlikely to vary among sentence meanings. At the end, we sum the two aforementioned terms, and multiply them by the unigram frequency of "armored vehicle". We do the same for $P(\text{Fishtank | joke})$. The entropy of this distribution is our ambiguity.

We choose to calculate $P(\text{word | context})$ using a masked language model. This choice of a masked language model is important, as opposed to a traditional directional LM, since we are using the unmasked portions of the sentence as a representation for the sentence's meaning and using an mLM would provide access to more information about the sentence. In terms of data distribution, because we would like to capture semantic dependencies in general English, we chose to use a pretrained BERT masked language model directly as opposed to fine-tuning it on a joke dataset.

The pretrained BERT mLM was a linear layer fit on top of a general BERT model with hidden size 768 and 12 layers and attention heads each.

3.2 Distinctiveness – KL Divergence of $P(\vec{f}|m, \vec{w})$

While [1] outlines a method for computing the relevance distribution directly from $P(w|m)$, we also notice that the basis for the calculation of distinctiveness, $P(\vec{f}|m, \vec{w})$, can be interpreted as the distribution of semantic focus placed on each word, given the sentence meaning m. Since $P(\vec{f}|m, \vec{w}) \propto P(\vec{w}|m, \vec{f}) * P(\vec{f}|m)$ and $P(\vec{f}|m)$ is constant, this distribution can be approximated as the previous $P(\text{word | context})$ distribution.

However, as a secondary approach, we can also approximate this distribution more directly. The focus on a word may be extracted from the attentional distribution from the self-attention layers of a transformer model, such as BERT [4]. BERT learns the semantic and syntactical relationships between context words (as described in their appendix), and these relationships were visualized via observing the self-attention distributions at each level, for each attention head, for each word. Thus, the summed "self-attention" given to each other word across layers and attention heads for the ambiguous term, when encoding a version of the ambiguous sentence, may be used as an estimate for the distribution of semantic relationships. Thus, BERT's attentional outputs may be used to calculate sentence distinctiveness.

Since BERT works on sub-word parts, we average the attention paid for each word from each of the ambiguous word's sub-words to arrive at an attention score.

3.3 Classification and rating

Finally, once the ambiguity and distinctiveness scores are calculated from $P(w|m)$, we constructed and trained a random forest classifier on the two variables to carry out the tasks of classifying whether a sentence is a joke, and rating jokes. The reason for a random forest classifier is that it is a simple nonlinear model, and as an ensemble method, it should be more robust with small sample sizes such as ours. Unlike previous steps, this part was trained on a mixture of puns and an equal amount of sentences that include the same ambiguous keywords in a more general English corpus. Each "null sentence" with the ambiguous term was also replaced with the two possible interpretations as the corresponding puns were in our pun dataset for purposes of calculating ambiguity and distinctiveness. For each test sample, the input would be the two scores, and the output would be a number between zero and one; for the classification task, the output represents a probability, and for the rating task, the output represents a scaling between -3 and 3, with positive scores indicating greater funniness.

3.4 Baseline

As a baseline, we applied the LSTM-based methods in Cai et. al. to the two tasks of classifying and rating jokes. To do so, we modified the encoder code from assignment 4 to interface with a sigmoid layer with cross entropy loss (for classification) or tanh with mean squared error loss (for regression) instead of a decoder. After encoding a sentence, the final combined (backwards and forwards) cell and hidden states are each passed through a hidden linear layer, concatenated, and passed through its respective output layer.

4 Experiments

4.1 Data

Three datasets were used in this project. The first dataset (puns) was created by scraping from Reddit, the second (non-puns) was the free portion of the iWeb corpus [11], and the last (both puns and non-puns) was the same dataset used by Kao et. al., provided by the authors upon request.

For our collection of labeled puns - after filtering for length (<100 characters) and likely troll content (score > 5), we manually processed 7000 potentially eligible jokes from the scraped Reddit data and extracted 400 well-constructed puns. Puns dependent on obscure cultural references or sexual slang were all excluded. Furthermore, all puns were labeled with their key ambiguous word or phrase (if there was more than one, the most relevant was chosen at authors' discretion), along with the two ambiguous meanings in as few words and high clarity as possible. By substituting both potential meaning for the ambiguous word in question and running each word slot through a neural language model, we can calculate $P(w|m_1)$ and $P(w|m_2)$ for each pun.

Some samples from our size 400 pun dataset include:

> Pun: "My 5 yr old son was just imprisoned for skipping naptime. He was resisting a rest."
> Ambiguous word: "a rest".
> Interpretation 1: " a rest". Interpretation 2: "arrest".
>
> Pun: "We should have known communism would fail. In hindsight there were a lot of red flags"
> Ambiguous word: "red flags"
> Interpretation 1: "warning signs". Interpretation 2: "Marxist symbolism".

The iWeb data available included around 16 million words scraped from various web pages. Cleaning involved removing any HTML tags, continuous strings consisting only of punctuation and numbers, and converting all words to lowercase after removing punctuation from them. We separate sentences by characters in the set {. ; ? ! \n}, and group them in pairs of two. To obtain sentences as similar possible as the puns dataset, we filtered the iWeb data to only have sentence pairs that contain the one or more ambiguous words present in the pun dataset. Each of these "ambiguous" words is paired the same "ambiguous" interpretations found in the corresponding entries in the pun dataset. We also filtered the dataset to only contain sentences pairs less than 120 characters. Furthermore, to ensure diversity of "pun words" present, we ensured there was no more than 30 of each pun word in the final dataset, which contained a total of 1600 entries.

Some entries from our size 1600 non-pun dataset include:

> NonPun: "These are not just statistics. They translate into real life stories."
> Ambiguous word: "stories"
> Interpretation 1: "books". Interpretation 2: "floors".
>
> NonPun: "Each grand lodge is sovereign and independent. There is no us or international governing body for freemasonry."
> Ambiguous word: "independent"
> Interpretation 1: "independent". Interpretation 2: "in the pendant".

The dataset used by Kao et al. contains 100 puns and 300 non-puns, all of which are labeled with a crowdsourced "funniness" rating, as well as ambiguous words/interpretations. Funniness ratings are in the range of (-3,3), with mean of 0 and standard deviation of 0.8.

4.2 Experimental details

For evaluation, we used an accuracy metric (fraction of jokes evaluated correctly) for the classification task and a mean squared error metric for the rating task.

As a baseline, we applied the LSTM-based methods in Cai et. al. to the two tasks of classifying and rating jokes. We ran our LSTM on a NV6 for 100 epochs using batch gradient descent. We trained with a train set of 300 non-puns and 300 puns, and validated with a dev set of 100 non-puns and 100 puns, updating the model every time dev set error is decreased with a patience of 3. Early stopping was triggered if we hit the max patience 10 times. Our test set was 140 puns and non-puns from the Kao dataset. For funniness quantification, we only used the Kao dataset (since other datasets did not have reliable measurements of funniness), with a 5:1:1 training:dev:test ratio.

At first, we attempted to train an LSTM-based masked language model from scratch on the iWeb dataset, since most puns involved two sentences, a setup and a punchline, and we feared that a model pretrained on single sentences would not do well on these inputs that were too long. However, when we experimented with a pretrained BERT, we found that the doubled length had little impact on the model's performance. Furthermore, training a language model from scratch would simply have taken far too long. We implemented an LSTM stack, which was shown in a review of neural language models ([7]) to have very good performance. However, the iWeb dataset had over 14 million words and a vocabulary size of about 700,000. As it took one day on an N6 GPU to train on about 320,000 sentences with one word masked out (with severely limited output), it would have taken far too long to train the model to a usable quality.

Thus, we applied the base pre-trained BERT masked language model to predict the probability of each word in the sentence given each interpretation of the ambiguous word. Unigrams for each word were obtained using the $word_frequency$ package [13]. These values were combined to calculate ambiguity, as described previously.
For distinctiveness, we tried two approaches: 1) using Bayes' rule and previous calculations, and 2) using the attentional outputs of BERT. For 2), to obtain the needed distribution of likelihoods that each word in the sentence is semantically relevant to an interpretation of the ambiguous word, we used BERT to encode the sentence and summed the sentence self-attentional distribution when looking at the ambiguous term, across layers and attention heads [12]. These pre-trained models used 12 attention heads in 12 layers.

The same datasets were used as above for testing the final models. We combined our dev set into the train set, since random forest classifiers have built-in leave-one-out validation (out-of-bag/OOB error) and are generally robust against overfitting, and having a dev set is largely unnecessary. We used the randomForest package in R.

4.3 Results

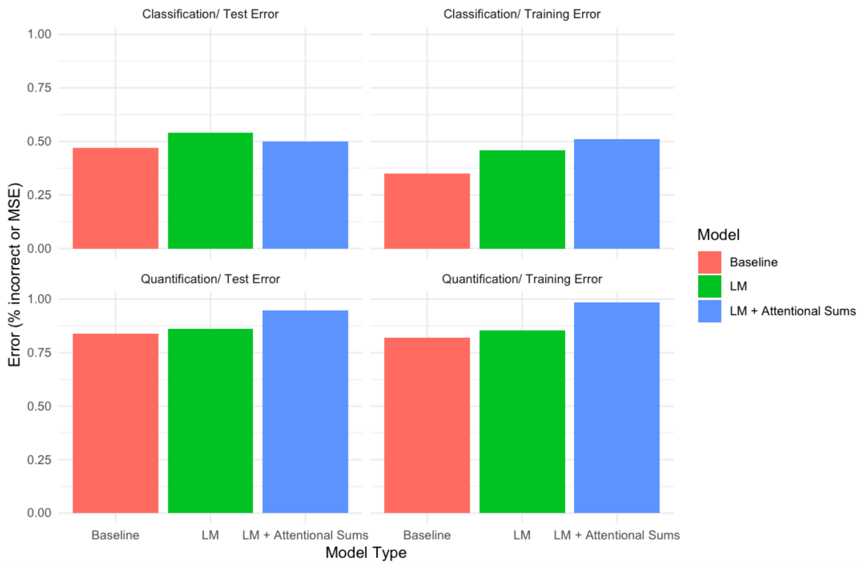
On the training set, our baseline LSTM encoder achieved 53 percent accuracy for puns and 78 percent accuracy for nonpuns. On the test set, it achieved 53 percent accuracy for classifying puns and 57 percent accuracy for classifying nonpuns.

For quantification, we had a final MSE of 0.82 for our train and 0.84 for our test set. This is approximately equal to the standard deviation of funniness ratings.

As a proof of concept, we ran our random forest model on the dataset provided by the authors. We were able to achieve 82 percent validation accuracy on the classification task, and 0.375 validation MSE on the quantification task. Thus, given accurate measurements of ambiguity and distinctiveness, getting good results is fully plausible.

To ensure that our pretrained masked language model was adequate, we tested it on a random sample of five webpages from the iWeb dataset. We found that it had an average perplexity of 75.809.

Unfortunately, our final model was unable to outperform the baselines. A plot of the errors is shown below. Error for puns and nonpuns were not significantly different (+- 5 percent).



We additionally attempted to train a simple neural network model (three linear layers of size 64 each with a sigmoid output for classification, for 20000 epochs) on our entropy/distinctiveness dataset as well, without better results. The model failed to converge and resulted in a final MSE of 0.77 and accuracy of 0.49. Experiments with larger models also failed to show improvement.

4.4 Analysis

At first our baseline performed exceptionally well, with less than 20 percent classification error on both the train and dev set. After investigating, we realized that this was due to the model picking up on the lack of punctuation present in the iWeb dataset from the initial parsing, and recognized that only puns had punctuation. We re-parsed the iWeb dataset upon noticing, with proper punctuation. Notably, our final baseline model did exceptionally worse in the test set non-puns than in the training and dev sets. We believe this is due to the intentional structuring of non-puns to be extremely similar to puns by Kao et al., which made differentiating much more difficult.
We believe the LSTM is unable to achieve good performance due to limited training data and the difficulty of humor to be learned through unstructured processes. Perhaps with more training data, the LSTM could do much better.

We first analyzed the language model outputs for the probabilities of words in various sentences, given two potential interpretations. In addition to amazingly low perplexity, we found that BERT's outputs were often empirically correct. For instance, take the pun "the magician got so mad he pulled his hare out", with the two interpretations of "hare" or "hair".
Masking out previous words, we see that:

"Magician" is more likely given "hare" than "hair" (4.19e-05 vs. 3.91e-05).
"Mad" is more likely given "hair" than "hare" (6.07e-05 vs. 5.00e-05)

Given that this pun has words primed by both definitions with about equal difference in likelihood, we would also suspect that this pun has relatively high ambiguity - and it does, at 0.67 (values ranged from 0.1 to about 0.7).

However, we do see a significant amount of errors as well. Unfortunately, many words used in puns are easy for humans to make semantic connections for, as done in the crowdsourced dataset by Kao et al., but do not occur together often in text. Thus, they are very difficult for BERT to learn and get correctly.

For instance, we observed the pun "An astronaut gets to take a turn as a whirled traveler", with interpretations being "world" and "whirled". BERT sadly predicts that "turn", the only word that brings "whirled" into relevance, is actually more likely given "world" than "whirled". We thus expect incorrectly low ambiguity - which we do see here, at 0.29. Distinctiveness in our first approach was calculated from the same values, and is thus affected by this flaw as well.

Low ambiguity errors due to rare associations were extremely common - other examples include failures to associate the relevant keywords in "aliens are lovable because they're so spatial" or "a pun is its own reword". These errors reflect upon the inherent difficulty in learning deep relationships between words.

Next, we analyzed the semantic relationship distributions produced, which were used to calculate distinctiveness. Unfortunately, across nearly every example that we looked at, it did not appear that BERT's attentional model was able to capture relatedness between words. For instance, in the magician sentence above, both "magician" and "pulled" appeared to have been given a higher proportion of attentional output in the "hair" sentence. The idea that BERT's attentional model, across 12 layers and averaging over attention heads, would be able to identify words more "similar" to the ambiguous word was highly experimental in nature, and based on qualitative observations made in the appendix of the original paper. We are not too surprised that this had a negative outcome.

Another source of error, we believe, comes from the model itself. For certain puns, such as "camping is (intense/in tents)", there is no particular word priming for the word "intense" over the alternative word "in tents". Thus, this pun would be marked as high ambiguity but low distinctiveness, and be lumped together with sentences such as "look at that (hare/hair)" that have the same values, but no humor. In order for this distinction to be made with the current model, "camping" must be separated as two components - one priming for an intense activity, and one priming for an activity done in tents. Future models may look to these examples as potential humor "edge cases" to account for.

Since the model was extremely susceptible to error, the methods we used likely did not capture the semantic connections that were most essential for humor recognition. Because of this, and because the LSTM baseline's performance was better than the experimental model's, we hypothesize that an end-to-end solution may in fact be better suited for the two tasks of classification and rating – a model that can learn semantic features that are more important to these tasks, rather than working off of features important to natural language processing tasks in general, may have better performance. In order for the experimental model to be successful, we expect that some form of artificial intelligence with deeper understanding of natural language must be first developed – and be able to link contextually different but loosely related words such as "alien" and "spatial". Because many of these errors were due to related words being extremely rarely co-occurent, solutions founded on principles of embedding or dimensionality reduction representations of meaning may still be the most promising for this category of problem, as they do not depend on the frequency distribution in the training dataset as much as other solutions.

4.5 Conclusion

We created a neural implementation of a method for humor classification and rating based off of cognitive science research by Kao et. al. in [1]. We used a masked language model and attention outputs from a pretrained BERT model. The results did not reach the accuracy in [1]; in fact, the model did more poorly than a baseline LSTM to predict the scores directly from sequences of word vectors. Errors that contributed to the low accuracy include mispredicting the two factors of ambiguity and relevance, and the frequency distribution of the training data resulting in rare word co-occurrences being misrepresented. The former may be mitigated in future work by fine-tuning models for the predictions of these two factors, and the latter may be improved by using methods that do not depend as much on frequency, such as using a word vector space.

4.6 References

[1] Kao, J. T., Levy, R., & Goodman, N. D. (2016). A computational model of linguistic humor in puns. Cognitive science, 40(5), 1270-1285.
[2] Jim Cai and Nicolas Ehrhardt (2013), "Is This A Joke?". CS224N Winter 2013
[3] Zhang, D., Song, W., Liu, L., Du, C., & Zhao, X. (2017, December). Investigations in Automatic

Humor Recognition. In Computational Intelligence and Design (ISCID), 2017 10th International Symposium on (Vol. 1, pp. 272-275). IEEE.
[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008).
[5] Raskin, V. (2012). Semantic mechanisms of humor (Vol. 24). Springer Science & Business Media.
[6] Martin, R. A., & Ford, T. (2018). The psychology of humor: An integrative approach. Academic press.
[7] Shi, Dengliang (2017). A Study on Neural Network Language Modeling. Arxiv.
[8] Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014). GloVe: Global Vectors for Word Representation.
[9] taivop (2017). joke-dataset: A dataset of 200k English plaintext jokes. GitHub.
[10] Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012, July). Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1 (pp. 873-882). Association for Computational Linguistics.
[11] Davies, Mark. (2018). The 14 Billion Word iWeb Corpus. Available online at https://corpus.byu.edu/iWeb/.
[12] huggingface (2019). PyTorch Pretrained BERT: The Big & Extending Repository of pretrained Transformers. Github.
[13] Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, & Lance Nathan. (2018, October 3). LuminosoInsight/wordfreq: v2.2. Zenodo.
[14] Taylor, Julia M. & Mazlack, Lawrence J. (2004). Computationally recognizing wordplay in jokes. Proceedings of Cognitive Science Conference.
[15] Mihalcea R & Pulman S. (2009). Characterizing Humour: An Exploration of Features in Humorous Texts. International Conference on Computational Linguistics and Intelligent Text Processing.
[16] Zhang, Renxian & Liu, Naishi (2014). Recognizing Humor on Twitter. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management.