Natural Language Processing Coursework

Kevin Landert
Imperial College London
kgl20@imperial.ac.uk

Constantin Eulenstein
Imperial College London
che19@imperial.ac.uk

Daphne Demekas
Imperial College London
ded20@imperial.ac.uk

Abstract

The following report outlines our design methodology, process and results for Imperial College London's Natural Language Processing coursework Task Nr. 1. We designed a variety of models to predict the funniness score of a headline in which a single word has been replaced. We aim to solve this task once by leveraging pretrained corpora and models and once without using any pretrained resources. Our approaches did not lead to any satisfactory results when it comes to solving the task at hand. Nevertheless, we gained valuable insights on the applicability of NLP to such a non-trivial task.

1 Introduction

The analysis of humor is a very complex subject. While in many Natural Language Processing (NLP) applications we can leverage expert knowledge to obtain a ground truth for training an algorithm, the humor level of a passage is strongly dependent on the individual readers perception. For this reason, the application of NLP techniques to determine humor is well suited for exploring various algorithms and analyzing the limitations of such models on an abstract and complex task.
In this coursework we aim to predict the humor score of a headline after a single word was replaced, following the challenge from CodaLab. The project is split into two distinct approaches: First we try to solve this task by leveraging pretrained embeddings and models. We start off by using a `gloVe` [4] embedding combined with a `BiLSTM` [2] model. We then improve on this approach by using a pretrained BERT [1] model. In the second part we aim to work without any pretrained resources. We train our own embedding using the Gutenberg corpus as well as the provided training headlines and then use similarity between word embedding as proxy for humor. Leveraging our embedding, we then aim to improve on this approach by applying the same `BiLSTM` model. Section 2 gives some brief introduction to the data, Sections 3 and 4 highlight our two approaches that leverage pretrained resources. Section 5 shows our two approaches using no pretrained elements. We finish our work in Section 6 with the insights of our work and a discussion on potential improvements and future work. We performed our analysis on the training and validation set and then tested these implementations offline on the CodaLab testset.

2 Dataset

Before implementing our models, we decided to analyze and adjust the data set to make the implementations easier. From the CodaLab website, we found extra data, such as a separate file for dev data, a file for extra train data and a separate test data file with labels. For all of these files, we added a column to the data set which contains the entire edited sentence - so the original sentence with the edited word. We removed the tokens $<$ and $>$ which surround the edited words in the sentences, and we added another column which measures the variance of the grades as well. For space reasons we omit details of our exploration of the dataset.

3 Part 1 - LSTM Model with GloVe

3.1 Preprocessing

Our first version of the `preprocessing()` function for the corpus performed three main tasks: it made all of the letters in the corpus lowercase, it removed some punctuation and it added in special tokens. We decided to leave in question marks because they have potential to impact the headline meaning, especially in the context of humour. The apostrophe was important as well because it was able to create new words out of words like ”don't” - namely do and n't. We also added a special token ??? before and after the words in the sentences which were to be replaced to emphasize the importance of those words and their relation across sentences. After our initial results showed that we were only predicting the mean of the grades, we also added lemmatization and stop word removal. We've additionally explored using stemming, but excluded this from our final submission as we did not see any performance increase using this technique.

3.2 Regression with gloVe embedding

Our first challenge was to figure out the best input to the model, because we needed a way of feeding the network both the original and edited sentences and have it pick up on their relation. We chose to create two corpora, one with the original sentence and one with the edited sentence. Apart from this the two corpora are identical. As we noticed a tendency of our network to overfit, we added dropout of 0.3 on both the LSTM and hidden layer. We also added two extra linear layers with dropout before the output layer. Our experimentation showed that using Leaky RELU activation lead to the best performance. Figure 1 shows the loss of our BiLSTM model for Part 1.

With this first approach using `gloVe` and `BiLSTM` our network achieved both a train RMSE of 0.510, a validation RMSE of 0.561 and a test RMSE of 0.574. We compared this value to that which would be achieved if the network was only predicting the mean of our headline data, which is 0.59.

Figure 2 shows a scatter plot between the predictions and target values. The x-axis shows the predictions and the y-axis the mean grade. Along the exis we can see a histogram of the marginal distribution of the data. Darker values in the plot indicate a higher concentration of values.

4 Part 1 - BERT Model

We then tried to feed the same preprocessed corpora separated by a `SEP` token to our BERT model. However, this resulted in only unsatisfactory results. Upon closer inspection we hypothesized that our preprocessing and inclusion of a ??? token did not match that input form that was expected by BERT. Thus we concluded to feed the entire original and edited sentence into a

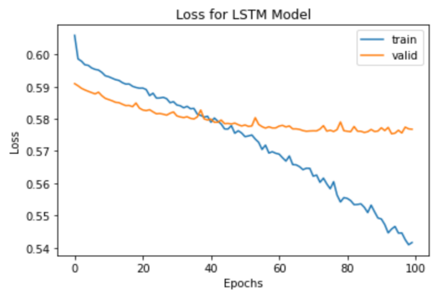

Figure 1: RMSE Loss over Epochs of BiLSTM Model

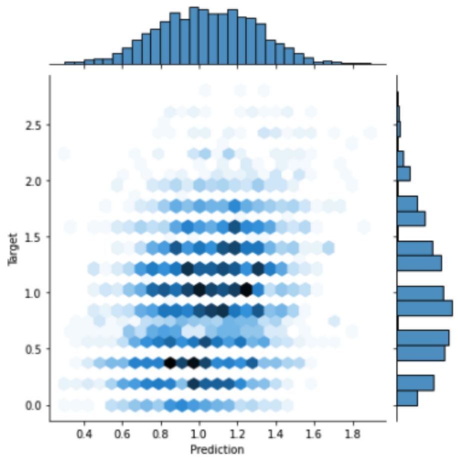

Figure 2: BiLSTM's predictions vs. target grades

`BertTokenizer`, such that this module would preprocess our input into the desired form for BERT. This preprocessing added the `CLS` and `SEP` token and introduced necessary padding.

We then used the BertForSequenceClassification with a single label in order to perform regression on our data. This method gave better results and in much fewer epochs, although the model again had a tendency to overfit.

Figure 3 show the loss curve of our BERT model and Figure 4 shows a scatterplot between our prediction and the target values. This time, the domain of the plot has a much larger range, and the predictions have a more linear trend, so it is clear that BERT was able to learn a better representation of humor than the LSTMs. We can see from this that the model performed better at detecting headlines that were rated not funny than those that were rated very funny. Table 1 shows our performance of both the LSTM and BERT model.

5 Part 2 - Zero Pretraining Approaches

In this part of the project we aim to solve the humor regression task without using any pretrained corpus or models. When it comes to humor

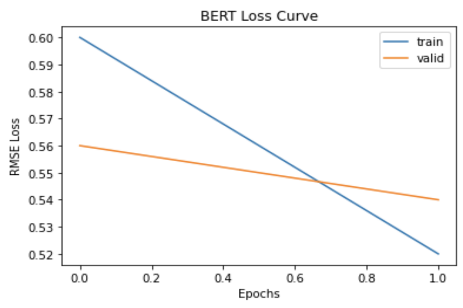

Figure 3: RMSE Loss over Epochs of BERT

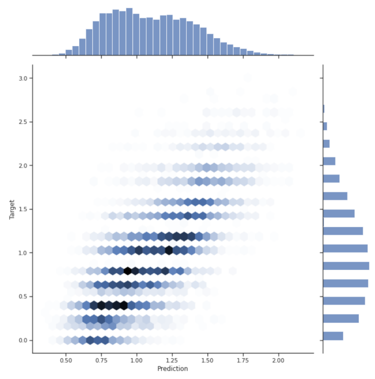

Figure 4: BERT's predictions vs. targets

	LSTM Model	**BERT**
Learning rate	0.008	5e-5
Epochs	80	2
Final Train RMSE	0.510	0.523
Final Val RMSE	0.561	0.541
Test RMSE	0.574	0.570

Table 1: Part 1 - Results: BiLSTM and BERT model

there are several theories of what makes something funny. We decided to follow the idea of incongruity which states that people find fundamentally incompatible concepts or unexpected resolutions funny. Following this concept an edited headline should achieve a high humor score, whenever the edited word does not fit the context in which it is used, leading to incongruity between our expectations and reality. To make use of this idea, our first approach tries to learn the word embedding for both the original and edited word based on the Gutenberg corpus as well as the provided training headlines and then tries to estimate the humor score for the edited headline based on the cosine similarity between the original word and the edited word. Our hypothesis here is that a lower cosine similarity can mathematically capture this concept of incongruity and should should thus determine the level of humor associated with an edit.

Our second approach in Part 2 then uses the same embedding together with a BiLSTM model similar to Part 1 to predict the humor score. We aim to compare these approaches with the results obtained in Part 1. For the implementation of the Skip-Gram we modified the given code from the NLP lab.

5.1 Embedding

To create an embedding we have leverages the Gutenberg corpus consisting of 18 books. We removed punctuation and stop words and added 80% of our original headlines to the corpus and built a vocabulary. The other 20% are used for validation and the embedding is not trained on these tokens. We want to ensure that our embedding is only trained on our training corpus, and headlines used for validation as well as testing have not been used to train the embedding. To build our vocabulary we use a cutoff frequency of one. Additionally, we've experimented with both stemmed and lemmatizated version of the tokens. We then train a Skip-Gram Word2Vec [3] model with a window size of three and an embedding size of 128. This embedding is the basis for both approaches in Part 2.

5.2 Cosine Similarity Model

We aim to compute the cosine similarity as a proxy for incongruity and thus humor. For this we compute the embedding of both the original word and the edited word and look at the cosine similarity between the two embeddings. Based on these similarity scores we train a linear regression model that takes the cosine similarity as features and the mean grade as labels and aims to predict the humor score for the test data.

With this approach we achieve a validation RMSE of 0.594. We managed to bring this down to 0.563 by dropping the OOV tokens from our data, but this does not signify any improvement, because our model was just predicting the mean. After careful analysis of our results we conclude that our approach does not work for the following reasons: 1) Approximately 27% of the words for which we computed the embedding were not in our training corpus and thus had the unknown token assigned. As a result this approach does not work for a significant amount of our data. 2) Incongruity may not necessarily lead to humor. Replacing the word Hitler with candy in: " Saudi crown prince says Iran's Aystollah Khamenei is very much like Hitler" receives a very low humor score even though these two words are highly incongruous according to their cosine similarity. On the other hand replacing words that are very similar, like the word guest with friend in 'Michelle Obama was Jimmy Fallon 's only guest and , no , they did not mom dance' results in a very high humor score even though there is little incongruity between the two. Thus, we conclude that our hypothesis is too simplistic and incongruity alone may not be sufficient to judge the level of humor. Table 2 shows the performance of this approach. The first column shows the performance for the full corpus, the second column excludes all edits that where not present in the training corpus.

	RMSE full	**RMSE known**
Original	0.592	0.573
Lemmatized	0.58	0.596
Stemmed	0.583	0.581

Table 2: Part 2 - Results Cosine Similarity

5.3 BiLSTM Model

To take more context into account than just individual words, we chose to use a BiLSTM model similar to Part 1 leveraging our own embedding. We used the embedding based on the Gutenberg corpus as basis to the architecture that we developed in Part 1. Additionally, we performed stop word removal and lemmatization on the Gutenberg corpus and the headlines. We added this extra preprocessing, because we figured it could reduce the issue of many edited tokens previously not seen by the algorithm as highlighted in the previous approach. We believed lemmatizing the tokens would result in fewer unknown tokens and more frequent realizations of each token when creating and training the embedding. Unfortunately, our model nevertheless performed bad at capturing humor. With this embedding, the model once more deteriorates to predicting the mean humor scores of the training headlines. Comparing these results to the results in Part 1 shows the importance of a well trained embedding or a pretrained model when it comes to predicting humor in this setting. Table 3 shows our final results for Part2 of the project.

	BiLSTM
Learning rate	0.008
Epochs	50
Final Train RMSE	0.593
Final Val RMSE	0.604
Test RMSE	0.594

Table 3: Part 2 - Results of BiLSTM

6 Conclusion and Future Work

After trying several approaches to solve this task, we conclude that this problem is non-trivial. Neither simplistic approaches nor sophisticated state of the art NLP models resulted in significant performance increases in comparison to the baseline. We assume that an algorithm cannot capture a complex concept like humor enough to result in satisfactory results. Upon inspection of the dataset, we can see that expert rating of the same edit had a high variance. This means that potentially not even experts can agree on the score of a specific passage. As a result we experimented with two additional approaches: 1) We excluded the data samples where ground truth ratings from different experts had a high variance, hoping that our algorithm would be able to preform better on this samples. 2) We let our algorithm not only predict the mean score of every sample but also the variance of expert rating. However, both approaches did not lead to satisfactory results.

When comparing our results from part 1 and part 2, we hypothesize that our approaches fail either due to the simplicity of the approach or the quality of the embedding that we have trained. To improve on these approaches we suggest training with a bigger and more task specific corpus. We assume that an embedding trained on a corpus of news headlines could lead to improvements. Additionally, we propose exploring different improvements of BERT that were proposed by the research community. E.g. one could pretrain a language model for text translation on the training headlines and use this embedding for the regression task.

Furthermore, after trying several different approaches and potential solutions to solve this task, we seemed to fail in most cases to get any satisfactory results. We assume that an abstract concept like humor cannot be captured by linguistic analysis only. The lack of unambiguity in the humor score of a passage seems to be a non-trivial problem and requires additional methods and analysis.

7 Link to Colab

Colab Notebook

References

[1] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.