Statistical NLP Final Report: Does this sound like a joke to you?!

Ryan Jenkinson
UCL CSML
ucabrcj@ucl.ac.uk

Éanna Morley
UCL CSML
ucaborl@ucl.ac.uk

Angus Brayne
UCL CSML
ucabar5@ucl.ac.uk

Abstract

Human:	*What do we want!?*
Computer:	*Natural language processing!*
Human:	*When do we want it!?*
Computer:	*When do we want what?*

The field of computational humour is cross disciplinary, bringing together sociological and psychological theories and trying to model them mathematically. Determining whether a piece of text is funny is nontrivial and the progress of generating funny text has been limited because automating the evaluation process has proved to be very challenging. Traditionally human evaluators have been used, but this is time consuming and costly. We propose a model for automating this evaluation process - predicting the "score" of jokes posted on a joke sub-Reddit as a proxy for how funny a joke is. We demonstrate that basic baseline models can outperform random prediction and we improve on these results using recent advances like `BERT`. We visualised our model predictions using t-SNE in order to gain a greater understanding of the important prediction features. Our champion model followed BERT (Devlin et al., 2018), with only one additional fully connected fine-tuned layer. It achieved 46.4% accuracy on a balanced test set with 3 classes. We also contribute a large dataset of jokes, which we hope will aid progress in computational humour[1].

1 Introduction

What makes something funny? This question is incredibly complex. The most common theory in the field of computational humour posits "incongruity" as a key mechanism to the success of puns and jokes more generally. Kao et al. (2013) defines incongruity as "perceiving a situation from different viewpoints and finding the resulting interpretations to be incompatible". These ideas suggest that there are underlying features of humour that could be modelled in order to improve our understanding of humour and generate humour more reliably.

Understanding this field is directly applicable to advancing social robotics, "chatbots" and more interactive artificially intelligent assistants, because humour is a key aspect of the human condition - it is heavily utilised in forming relationships. Research into humour may also open the door for systems which can generate other traits often found in human interactions, such as inspirational or empathetic language. Additionally, it is a multidisciplinary field where advancements in the humour domain can direct research in sociology, psychology, computational neuroscience as well as many others.

As human-machine interaction becomes more ubiquitous, the ability for machine agents to engage in humorous conversation with human interlocutors is clearly of great utility, making conversation more natural and engaging. This will greatly improve the user experience of many services, encouraging increased usage. However, it is also an area fraught with difficulty due the contextual and subjective nature of humour. The risk and cost of offending end users may be high, as illustrated by Microsoft's experimental twitter chatbot "Tay", and consequently learning a representation of humour that is both universal and inoffensive is likely to be extremely important. Clearly, there is much work left to be done on the subject of computational humour. Indeed, this area of research is no joke.

Much of the prior work on generative joke models (Ren and Yang, 2017; Chippada and Saha, 2018) relies on human evaluation - categorising jokes on a linear classification scale. This process greatly impedes progress in the field, as this evaluation methodology is subjective, time consuming and sometimes costly.

[1] `github.com/RyanJenkinson/nlp-jcag`

In this paper we investigate models for predicting Reddit "scores" on a joke sub-Reddit as a proxy for quality of joke. A model with such capabilities could be used for automatically evaluating the quality of generative joke models or as a discriminator for these generative models during training. We present basic baseline models that outperform random prediction and we improve on these using more advanced models. Our champion model followed BERT (Devlin et al., 2018) and achieved 46.4% accuracy on a balanced test set with 3 classes. We investigated our model predictions using t-SNE and illustrate important prediction features.

2 Related Work

2.1 Theories of humour

There are various social and psychological theories of humour, as outlined in (Mulder and Nijholt, 2002). The three most prominent theories are that of "Relief Theory" (laughter is a homeostatic mechanism for the reduction of psychological tension), "Superiority Theory" (laughter arises by laughing at the misfortune of others, attributed to the ideology of Platonism) and "Incongruity juxtaposition/resolution theory" (humour is perceived at the moment of realisation of incongruity - intrinsic to humour is intentional ambiguity, popularised by prolific thought leader Emmanuel Kant). We are most interested in the incongruity theory - it suggests that there are latent factors underlying the success (or otherwise) of a joke that can be learned.

Advances have been made in understanding the foundations of humour, both mathematically and computationally (Paulos, 2008). Research leveraging probabilistic sentence comprehension models has suggested that ambiguity and distinctiveness are important features of humour (Kao et al., 2013).

It is instructive to consider the generative joke model literature due to the overlap in modelling humour, as well as the fact it would be a key place to apply such a scoring module. For example, Petrovic and Matthews (2013) build a probabalistic graphical model to generate jokes with the fixed "I like my X like I like my Y, Z" structure, where X, Y are assumed to be nouns and Z is an adjective. They use large amounts of typical text data (not jokes) to find the probability distributions that encode their four assumptions that jokes are funnier if:

1. the attribute, Z, is used to describe both nouns more often;
2. the attribute Z is less common;
3. the attribute Z is more ambiguous;
4. the two nouns X and Y are more dissimilar

This work was continued (Winters et al., 2018), with the addition of word "sexiness" factor - comparison of frequency of word in corpus of sexual domain and normal domain. These are factors that our neural models may be able to exploit to detect the quality of humour.

An additional interesting feature of the work of Winters et al. (2018) is that they use a corpus of *rated* example jokes. They created an application that allowed volunteers to submit jokes and rate other user's jokes. Their study included 203 volunteers, 9034 ratings and 534 jokes, of which 100 were generated jokes. They created this application under the assertion that the ratings of jokes scraped from Twitter and Reddit wouldn't reflect the quality of joke sufficiently. They argued that the ratings are skewed by unequal exposure (there is correlation between exposure and number of followers, as well as number of hashtags) and that the audience is biased (viewers tend to have correlated humour with websites that they visit). We believe that there is likely to be some useful information content although the unequal exposure may cause the information to be imperfect. We assess the plausibility of learning and detecting humour based on this noisy signal.

2.2 Sentence classification

The generative joke model literature consistently cites the issue of difficulty in evaluating model performance automatically and, without exception, relies heavily on human evaluation. Yu et al. (2018) compare the diversity of generated sentences following Li et al. (2016) under the premise that if the generated jokes are more diverse then the model is likely to be more creative. They also train a separate (tri-gram) language model to estimate perplexity scores and judge whether the generated sentence likely to occur in some corpus. Chippada and Saha (2018) used a similarity metric to see whether the generated jokes are different from those found in the training data, in order to show that the model can generate novel jokes. They also use a Link Grammar Parser (Grinberg

et al., 1995) to evaluate the syntactic correctness of the generated text.

None of the methods above attempt to assess comedic value, they either assess syntactic correctness or novelty. We propose the first method that seeks to assess comedic value directly. We hope that this will not only prove useful for evaluating generative models, but also improve the output quality of generative models that have been proposed previously in the literature by incorporating the predictions of our scoring model into the loss function for the generative model.

The scoring model that we present, which aims to predict scores of jokes posted to Reddit, utilises ideas from the well established field of sentence classification. The combination of pre-trained word embeddings and convolutional neural networks (CNNs) have been very successful in this domain (Kim, 2014) giving competitive results against more sophisticated models that do not utilize pre-trained embeddings (Kalchbrenner et al., 2014; Socher et al., 2013). The use of pre-trained embeddings is a form of transfer learning. Word embeddings reduce the dimensionality of the representation of a word from a sparse representation, with vectors having length of the vocabulary size, onto a lower dimensional dense representation. In this low dimensional space, semantically similar words are close together. Examples of such word embeddings are `word2vec` (Mikolov et al., 2013) and `GloVe` (Pennington et al., 2014). CNNs leverage the discrete convolution operation in some of their layers (Lecun et al., 1998) which in theory could mimic n-grams of various lengths.

Recently contextual language representation models have been applied to a range of natural language processing tasks with with great success. Bidirectional Encoder Representations from Transformers (`BERT` - Devlin et al. (2018)) obtained state-of-the-art results on eleven tasks, such as language inference and question answering, outperforming human performance in some cases. `BERT` uses a pre-trained deep bidirectional Transformer that conditions on the entire context. Its contextual nature makes it very powerful on many tasks, requiring the training of just one additional task specific output layer. For pre-training `BERT` uses the "masked language model", randomly masking tokens and predicting the original word based on its context, to enable the inclusion of both left and right context. They also use the "next sentence prediction" task, which jointly trains text-pair representations.

The key advantage of `BERT`, over pre-trained word representations such as `GloVe` followed by a neural architecture trained on the jokes, is that it learns contextual representations during pre-training. The dataset used for pre-training is in our case, and often more generally, much larger than the available task specific data set. Pre-training contextual embeddings relieves significant pressure on the task specific architecture to learn this contextual information.

3 Methods

3.1 GloVe - a handy word embedding

In order to represent jokes in a format that is suitable for modelling, we transformed each word from the raw text into a 300 dimensional numerical vector using pre-trained GloVe embeddings (Pennington et al., 2014), trained using text from Wikipedia and Gigaword. Using pre-trained embeddings was preferable due to the size of our data set.

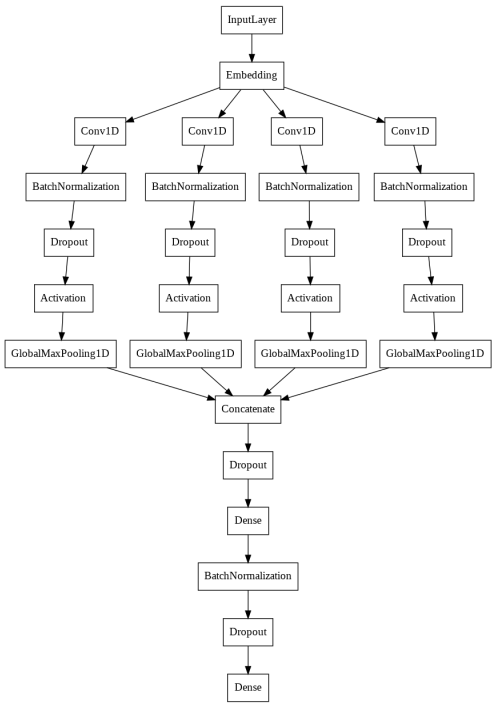

Figure 1: Model Architecture for variant of Kim (2014)

Once each word is represented by a numerical vector, it can be fed into a model in a number of ways. The vectors can be concatenated to form a single vector padded with zeros so that each joke is of the same length. Alternatively, the mean can

be taken across each dimension in the embedding space to produce an average joke vector. Word vectors can also be stacked to form a matrix (with padding) that can then be processed by a convolutional model. We experimented with all of these techniques and assess their relative performance.

We use one architecture (see Figure 1), similar to that of Kim (2014) on top of the word embedding. It uses a simple CNN with just one convolutional layer, having multiple filter widths and feature maps, and achieves excellent results on a number of benchmarks. The input to the convolutional layer is $k \times n$-dimensional, for maximum sentence length n and k-dimensional word embeddings. After the convolution a max-over time pooling operation is applied. This is followed by a fully connected softmax layer, which returns the probability distribution over the labels. Dropout, l_2-norm penalty on weight vectors and early stopping are used for regularisation.

It is important to note that GloVe embeddings are not contextual. This means that a single word will be represented by a single vector in the embedding space regardless of how it is used in a sentence. For instance, the word "bank" in the phrase "river bank" will be represented by the same vector as "bank" in the phrase "bank vault" despite possessing very different meanings. As context is important for humour content, we also experiment with a contextual model `BERT`, which eases the burden on the latter layers of the model (seen in Kim (2014)) to learn contextual features.

3.2 t-SNE

The popular dimensionality reduction technique, t-SNE (Laurens van der Maaten, 2008), was used to visualize the GloVe embeddings. Due to the high dimensionality of the embedding space, reducing the dimensions of this space and visualizing it can be helpful in understanding the data and the models that learn from it.

To aid our understanding we created an interactive visualization using the `Plotly` library. This allowed us to annotate points with their corresponding joke and score, greatly enhancing our understanding of the clusters found in the embedding space which helped us to attribute meaning to the emergent substructures that the scoring model may take advantage of.

This technique is not without its pitfalls and it is easy to misinterpret results or find structure in the data where there isn't any. Some caveats to bear in mind when interpreting the results of the visualization are that cluster sizes and distances between clusters are not necessarily meaningful.

3.3 BERT wait, there's more...

We initially encoded the individual words of jokes using `GloVe`, which is often used in a generative joke context (Ren and Yang, 2017; Chippada and Saha, 2018). However, we hypothesised that key performance gains would be exhibited by using contextual word embeddings, since they are able to learn contextual features during pre-training on the larger non-task specific dataset. This relieves a significant amount of the pressure that is on the task specific architecture. Context is likely to be extremely important for jokes, because incongruity juxtaposition resolution theory relies on it heavily; we need to know about the context of the words in the joke since this is what makes it funny (e.g words/phrases about nouns used in unconventional ways).

`BERT` (Devlin et al., 2018) uses a pre-trained deep bidirectional Transformer architecture, that conditions on the entire context, and has become ubiquitous. It is almost identical to the original implementation in Vaswani et al. (2017). The data is first formatted to match the data that `BERT` was pre-trained on, it is lowercased, the sentence is split into tokens, each word is then broken into WordPieces (i.e. "buying" $\rightarrow$ ["buy", "##ing"]), these are then mapped to the `BERT` vocab, "CLS" tokens are added to the beginning of sentence and "SEP" to the end and finally the input embedding is the sum of token, segment (only relevant for pairs of input texts - e.g. semantic similarity tasks) and position embeddings. Position embeddings are required because Transformers do not encode the sequential nature of their inputs.

`BERT` can be applied to give "pooled outputs" for classification tasks on an entire sequence, which is relevant to our use case, or it can provide "sequence outputs", which are at token level and are useful for sequence modelling. The fixed-dimensional pooled representation takes the final hidden state for the first token of the input - the [CLS] word embedding. The Google researchers behind `BERT` have released a number of pre-trained models, we use `BERT-Base` which is a model for uncased text and contains 12 layers (Transformer blocks), hidden size H of 768 and 12

self-attention heads. The feed-forward filter size is $4H$ and this model has 110M parameters in total.

We apply dropout (dropout probability 0.1) to the `BERT` output, to reduce overfitting, and add a single new task specific layer, which adapts it to our joke classification task. This form of transfer learning, using a mostly pre-trained model, is referred to as fine-tuning. This additional layer is a fully connected and maps from the `BERT` output layer of dimension $H = 768$ to the number of classes $K = 3$ of the classification problem, so $W \in R^{K \times H}$. We apply a softmax activation, such that the output can be interpreted as the probability of each class, and minimize the cross entropy (negative log-likelihood) loss. Under these time constraints, we were unable to perform task specific hyperparameter tuning, so we used the suggested parameters in a Google demonstration[2]. These include a batch size of 32, 3 epochs, learning rate of 2×10^{-5} and a warm-up proportion of 0.1 (proportion of training iterations to gradually increase the learning rate and stabilise training). The model took one hour to train and evaluate on a `colab` GPU over 50277 example jokes (20% of which were held out for testing).

4 Experiments

4.1 Data

Our analysis was performed on a dataset of approximately 195,000 jokes (Pungas, 2017) from the "Jokes" section of the website Reddit. Our data pre-processing restricted the joke setup (title) length to 50 words and punchline (body) to 20 words and we concatenated these. Jokes with a very high number of words in the title or body tend to be spam and do not score well. We discarded any jokes that contained images and non standard characters. After this we had 146,000 jokes. We then converted characters such that all were lower case and we removed punctuation.

Each joke had a corresponding score which is a calculation based on the number of "upvotes" and "downvotes" the post received. The exact method Reddit uses to calculate this score is unknown. We initially investigated the regression setting and looked at various transformations of the score, using increasing one-to-one functions, to reduce the skew of our distributions (50% of jokes score 3 or less). We found the correlation of our predictions to the transformation of true scores to be low (≈ 0.1).

We simplified the problem and applied thresholds to the scores to form a classification problem. We adopted a similar evaluation methodology to that of (Ren and Yang, 2017), where they defined a linear scale for evaluation as `Score: 0 - nonsense; 1 - not funny; 2 - somewhat funny; 3 - funny.` This setting is representative, because a human evaluator is unlikely to consistently distinguish joke quality much more precisely than these classes. Developments in this space could be transferred to regression by changing the last layer of our neural network architecture appropriately.

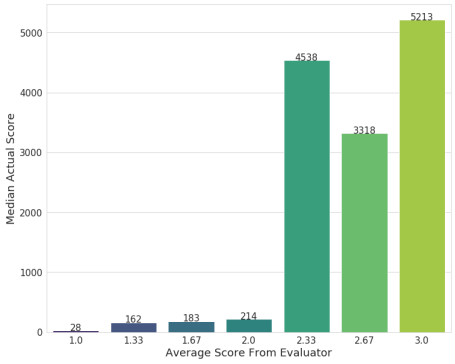

Figure 2: Median actual scores against average enlisted human evaluator.

Ren and Yang (2017) use this scale to evaluate their generative model, which was imperfect and sometimes generated nonsense. Our joke corpus contained very few incoherent jokes, so we used only the categories from 1-3, where category 1 would also include nonsense jokes.

We enlisted three human evaluators to rate 300 jokes from our dataset, so that we could calibrate the score threshold such that the joke partition we used aligned with these descriptions. We took a logarithmically representative subsample of the data, such that there were 60 jokes randomly sampled from each of 5 score ranges with width increasing logarithmically. The mean rating of the jokes as scored by our team is plotted against the median Reddit score of the jokes receiving that rating in Figure 2. We choose the median Reddit score because this metric is more robust to outliers, which are common in this noisy domain. We chose to split the scores at thresholds of 2 and 50 - scores less than or equal to 2 were assigned to

[2] `colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb`

class 1, greater than 2 and less than or equal to 50 were class 2 and above 50 were class 3. This was a compromise between the thresholds of ≈ 50 and ≈ 200, suggested by the graph, and enabling larger amounts of data for the third class, because higher scores occur less frequently. It is quite possible that our human evaluators were harder to impress than the average Reddit user.

It is also instructive to consider the results displayed in Figure 2 in the context of whether the score is a reasonable proxy for quality of joke. It is clear that there is a positive correlation, which is encouraging, however the correlation is not strong. This emphasises the incredibly challenging nature of this task. The jokes deemed to be funny have much larger median scores than those deemed to be somewhat funny, while the scores of jokes that are not funny and somewhat funny are quite similar. This supports our hypothesis that the Reddit score should not be interpreted as a strictly linear measure of "funniness". This may be due in part due to the "winner takes all" phenomenon, whereby more popular jokes have higher visibility and thus attract more upvotes.

4.2 Evaluation

After applying these thresholds to the scores, we randomly select a balanced number of jokes from each class and are left with 50277 example jokes. The models were trained on a random selection of 80% of these jokes and evaluated on the other 20%. The comparison metric used is accuracy. The process of training and testing was repeated ten times for each model to assess the stability of performance and ensure a reliable estimate of the accuracy. While we also looked at other metrics such as F1 scores, we did not deem these important due there being no class imbalance in the training set or use case specific preference towards correctly predicting a particular class. Hyperparameter tuning for models was performed on the training set using cross-validation.

5 Results and Discussion

5.1 Models

We model the three class classification problem - a sub-problem of the regression on scores. We implement some simple baseline models (LSTMs have been ommited from the results because they showed less promise than CNNs in initial testing) in addition to two more advanced models based on CNNs (Kim, 2014) and `BERT` (Devlin et al., 2018) respectively. These authors achieved very strong results on a wide range of tasks.

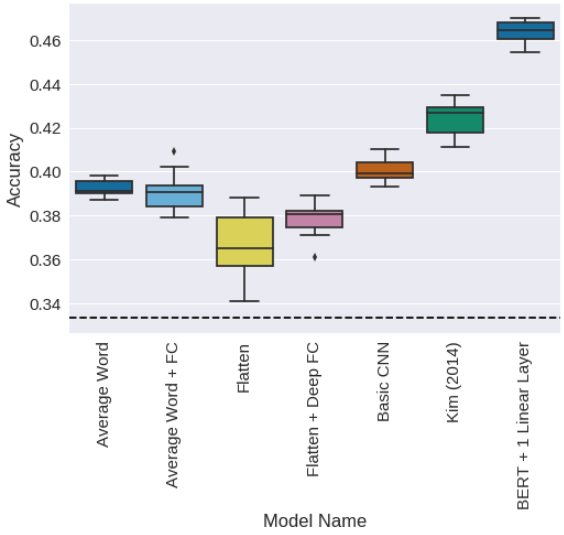

Figure 3: Accuracies over 10 runs of the baseline models, Kim's model and `BERT`. Note that Kim (2014) and `BERT` were applied here with the authors hyperparameter choices. The black dashed line shows random guessing accuracy.

It can be seen in Figure 3 that the average word model alone performs 6% better than random prediction. The Basic CNN model was the most promising baseline model outperforming random by 7%, so we investigated a more advanced CNN. The models in Figure 3 were averaged over 10 runs in order to understand the variance and consistency of the models.

Before we tried BERT, we performed a hyperparameter grid search on the model of Kim (2014) in an attempt to increase performance. The parameters that we searched over are:

(a) Kernel sizes - for models labelled $m \in \{0, 1, \ldots, 6\}$ we have $m + 1$ parallel CNN processes with kernel size $\{3, 4, \ldots, m + 3\}$ (see Figure 1).

(b) Learning Rate

(c) Dropout Rate

(d) Number of filters for each kernel size.

(e) Number of hidden units in the final fully connected layer.

For each setting of the hyperparameters we named the model `Kim (2014)_(a)_(b)_(c)_(d)_(e)`. The top

5 performing and lowest 5 performing models (mean accuracy) are shown in Figure 4. We were unable to achieve accuracies greater than 44%. Interestingly, we see that the performance is not increased by larger filters sizes than 3/4. This suggests that capturing longer range dependencies was not necessary, however with low maximum prediction accuracy of 44% and similar results for various different models we cannot read into this too much.

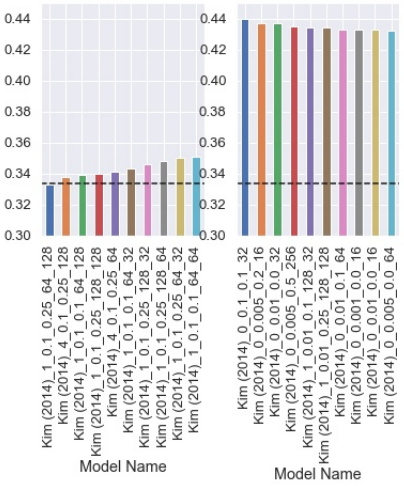

Figure 4: Top 5 best performing and top 5 worse performing (mean accuracy) model variants of (Kim, 2014). The black dashed line shows random guessing accuracy.

As mentioned in Section 3.3, we also hypothesised that contextual word embeddings would improve the performance by reducing the burden of learning contextual features on the task specific layers. BERT outperformed all previous models, with an accuracy of 46%. Due to time constraints, we did not perform a task specific hyperparameter search and opted to use some recommended values, but fine tuning could yield higher results. From this we can draw the clear conclusion that leveraging pre-training to learn contextual features is useful on this task.

5.2 Visualization

To understand the limited success of our initial benchmark models, we undertook analysis on what the model was learning in the basic cases, for example the average word model. We hoped to determine the attributes that the model was learning, since it was performing better than random prediction.

Since the baselines were using mean GloVe embedding vectors as inputs, it was decided to visualize these 300 dimensional vectors in 2D using t-SNE dimensionality reduction. This visualization (Figure 5) yielded some interesting findings. One axis consistently seemed to strongly correspond to joke length. This was surprising due to the fact that mean vectors were used and thus there was no explicit information about joke length contained in the model inputs. We hypothesize that this is due to longer jokes containing a higher proportion of frequently occurring words and thus tending towards the same region in the embedding space.

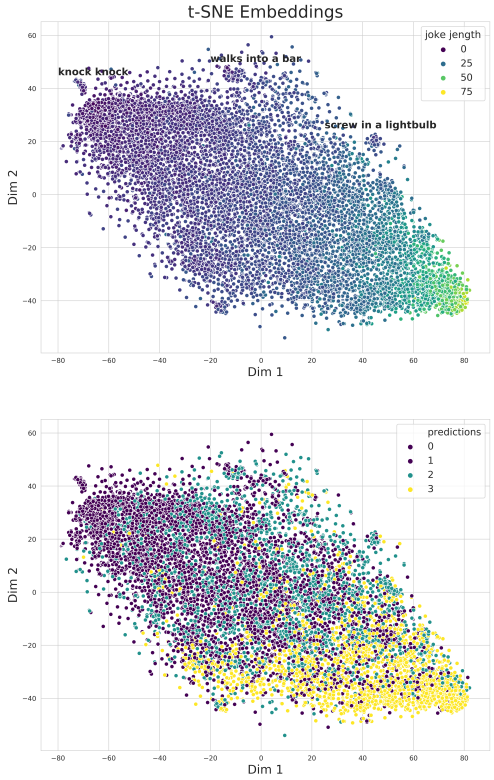

Figure 5: These plots show a 2-D t-SNE embedding of the mean GloVe vectors for each joke in the test set. The top plot is coloured by joke length and the bottom by baseline model prediction.

When colouring the points on the t-SNE scatterplot by the predictions of a baseline model, we found that the baseline model had indeed learned a relatively weak correlation between score and joke length. We contrasted this with a more complex and better performing convolutional model proposed by Kim (2014). We can see from Figure 6 that the model is not simply predicting on the basis of joke length and has learned something more subtle about what constitutes a funny joke.

Other findings of interest from this visualization

were that there emerged clear clusters of conceptually similar jokes in the 2D representation of the embedding space. Some of these are highlighted in Figure 5. For instance, we found clusters of "A man walks into a bar..." jokes, "How many X does it take to screw in a light bulb?" jokes and several others. This suggests that if a particular "class" of joke tends to score better on average, our models could in theory learn this.

Figure 6: t-SNE embeddings colored firstly by joke length and secondly by the class predicted by the Kim (2014) model. In contrast to the baseline, it is not simply predicting on the basis of joke length.

6 Conclusion

In this paper we propose the first automatic evaluation method that seeks to assess the comedic value of a joke directly, rather than restricting assessment to syntactic correctness or novelty. We aim to reduce the human evaluation requirement that is so often found in the generative models literature, in order to progress towards more humorous chatbots and assistants.

We began by building a series of baseline models and found that the predictions of the average word model, which performed marginally better than random prediction, were correlated with the trivial feature of joke length. Our accuracy improved when looking at CNN architectures, motivating a model similar to Kim (2014), which has seen success over a wide range of classification tasks. We also hypothesised that a contextual embedding model would improve the prediction performance by reducing the burden of learning contextual features on the task specific layers. We chose to apply `BERT` because we believe that the bi-directionality would increase the model's capacity to recognise key features from incongruity juxtaposition resolution theory (Mulder and Nijholt, 2002). These hypotheses were supported by our results. `BERT` applied in this novel setting, without hyperparameter tuning, achieved a mean accuracy of 46.4% - 2.5% higher than a hyperparameter optimised version of Kim (2014).

The Reddit score signal, used as a proxy for funniness, is noisy; for further improvements principled data analysis and preprocessing could be undertaken to smooth this signal. We believe that this classification setting is representative, because a human evaluator is unlikely to consistently distinguish joke quality much more precisely than the not funny, somewhat funny and funny classes. However, remaining in the regression setting may enhance the ability of models to learn, due to the information lost in transferring to the classification setting. The utilisation of text parsers could prove useful for removing nonsense jokes.

There is lots of scope for interesting work in this area and clearly there is plenty of room for improvement. Of course, improvements may be made by performing a hyperparameter search over the `BERT` model and exploring other architectures like `ELMO` and `ULM-FiT`. For future work, we have contributed a much larger dataset of scraped jokes[1]. Additional progress may be made in examining a much smaller (9034 ratings and 534 jokes), but much cleaner, dataset provided by Winters et al. (2018), who created an application that allowed volunteers to submit jokes and rate other user's jokes. We also believe that it may be possible to introduce these scoring models in the generative model context, as a discriminator or as an additional objective - sequence-to-sequence learning combined with score maximisation in a multi-task manner.

Humour detection and generation poses an incredibly challenging problem to the natural language community and with that come many opportunities for exiting research.

References

Bhargav Chippada and Shubajit Saha. 2018. Knowledge amalgam: Generating jokes and quotes together. *CoRR*, abs/1806.04387.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dennis Grinberg, John D. Lafferty, and Daniel Sleator. 1995. A robust parsing algorithm for link grammars. *CoRR*, abs/cmp-lg/9508003.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Justine T. Kao, Roger Levy, and Noah D. Goodman. 2013. The funny thing about incongruity: A computational model of humor in puns. In *Proceedings of the 35th Annual Meeting of the Cognitive Science Society, CogSci 2013, Berlin, Germany, July 31 - August 3, 2013*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Geoffrey Hinton Laurens van der Maaten. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Matthijs P Mulder and Antinus Nijholt. 2002. *Humour research: State of the art*. Citeseer.

John Allen Paulos. 2008. *Mathematics and humor: A study of the logic of humor*. University of Chicago Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Sasa Petrovic and David Matthews. 2013. Unsupervised joke generation from big data. In *ACL*.

Taivo Pungas. 2017. A dataset of english plaintext jokes.

He Ren and Quan Sheng Yang. 2017. Neural joke generation.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Thomas Winters, Vincent Nys, and Daniel De Schreye. 2018. Automatic joke generation: Learning humor from examples. In *Distributed, Ambient and Pervasive Interactions: Technologies and Contexts*, pages 360–377, Cham. Springer International Publishing.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1660. Association for Computational Linguistics.