Statistical NLP Project Proposal: 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 these mathematically. We can benefit from the latest advances in Deep Learning applied to Natural Language Processing to hopefully learn the latent factors that form the essence of what makes a joke "funny". Our target variable is itself nontrivial, and most progress in this field has been limited due to the evaluation metric - how do we measure success? Traditionally, a linear scale of "not funny" to "funny" with human evaluators is used. Understandably, this method is time consuming and subjective. This project aims to automate this process further, offering a novel framework for training these models. We first look to train a scoring model to predict the "score" of a joke, and use this model explicitly in the objective function of our generative network, which will generate punchlines conditional on the "setup" of the joke.

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". Early work in this field amounted to the formulation of mathematical models and graphical networks that were based on some underlying or predetermined assumptions about either the structure of the joke or aspects that make it funny. These papers are explored in Section 2.1. More recently, the rise of Deep Learning has enabled models that do not require these assumptions (Ren and Yang, 2017; Chippada and Saha, 2018). This field poses a number of challenges especially in the evaluation stage of the model. Much of the prior work relies on human categorisation - evaluating jokes on a linear classification scale. This process greatly impedes progress in the field, as this evaluation methodology is both time consuming and sometimes costly.

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 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.

2 Literature Review

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" (humor is perceived at the moment of realisation of incongruity, i.e intrinsic to humour is intentional ambiguity, popularised by prolific thought leader Emmanuel Kant). We are most interested in the incongruity theory, and it suggests that there are latent factors underlying the success (or otherwise) of a joke that can be learned from all words. These theories motivate a machine learning based approach to learn these factors.

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).

2.1 Discussion of existing sentence classification work

The scoring model, which aims to predict scores of jokes posted to Reddit, can be though of as a sentence classification/regression problem - a sentence is provided to the model and the purpose of the model is to predict a class/value respectively.

Sentence classification is a well established field in Natural Language Processing. 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 (according to some distance metric). 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).

Kim (2014) trained a simple CNN with just one convolutional layer, having multiple filter widths and feature maps, on top of the word embedding 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. Padding is added to the shorter sentences. 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 (for example, a sentiment analysis model would output a distribution over sentiment labels: "very positive", "positive", "neutral", "negative", "very negative"). Dropout is applied on the penultimate layer, l_2-norms of weight vectors are applied and early stopping on the development set is used for regularisation. The network is trained by AdaDelta stochastic gradient descent over shuffled mini-batches.

If we preferred to work in the regression setting, rather than binning the joke ratings and classifying, we could easily change the final layer to do so. For example, we could replace the softmax layer with another linear layer. We would also change our loss function accordingly, replacing multiclass crossentropy with (R)MSE or similar.

The most important feature of this model for our purposes is that it is fully differentiable. This will allow us to include its output in the objective function of our generative model and propagate gradients all the way back through the network to train our generative model.

2.2 Discussion of existing sequence generation work

There have been many research efforts to create models that generate humorous text. Much of the early work constrains the problem by introducing templates and building models that fill the gaps in the sentences. 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. Interestingly, they do not use jokes of this format to train their model. Instead they use large amounts of unannotated data 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

They construct a factor model, with X, Y and Z as variables and the four encoded assumptions as factors. They fix one of the nouns and perform exact inference to find the other two words that are most probable conditioned on it.

(Winters et al., 2018) continued the work on "I like my X like I like my Y, Z" jokes. However, among other additions, they have a factor of word "sexiness" - comparison of frequency of word in corpus of sexual domain and normal domain. This work is particularly interesting because 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). In our work we look to put this assertion to the test. We believe that although the unequal exposure may cause the information to be imperfect, it is still likely to have some useful information content. We also believe that, since a typical characteristic of humour is large disagreement between raters, a biased model may still be useful. It is clear that any generative model for humour is unlikely to cater to all cultural communities, so solving this sub-problem of maximising humour with respect to a biased audience is nevertheless worthwhile. Additionally, the abundance of data that can be gathered easily from these forums is a strength that should not be overlooked.

This research conforming to predefined structure is inflexible and often lacks the subtlety required to generate novel jokes and as such we would like to move beyond it.

Work has since been done in the context of homographic pun generation (Yu et al., 2018). Homographic puns exploit distinct meanings of the same written word (e.g *rain* vs *reign*). This worked in a similarly unsupervised manner with a general text corpus (Wikipedia) and generated puns using an intricate decoding algorithm. The challenge is to generate a sentence containing a word with two specific senses. They do this using an improved language model that uses a novel joint beam search algorithm. For our purposes, the most interesting thing about this work is the methods that they apply to automatically evaluate the generated jokes. They 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 second (tri-gram) language model on the development set and use this to estimate perplexity scores for model evaluation after learning. Similarly, we plan to train a separate model to estimate the quality of a joke and go one step further by introducing the output of this into the objective function of the generative model during learning as feedback.

Recurrent neural networks have become well established in the field of language generation. Long short-term memory (LSTM) networks have been found to be successful in character level language generation (Sutskever et al., 2011) and both LSTMs and GRUs have been shown to outperform standard RNN architectures on a number of tasks (Karpathy et al., 2015). Character level constructions often struggle to generate meaningful content, so word level encoder-decoder architectures have been proposed (Sutskever et al., 2014). An encoder-decoder LSTM architecture has proven to be very successful in image captioning (Vinyals et al., 2015). Though RNNs have improved the ability of neural methods of sequence generation to generate meaningful content, they have often struggled to capture very long distance relationships in sequences. Attention-mechanisms have proven to be very helpful in this domain (Serban et al., 2016).

2.3 Critical analysis of key generative models

The primary paper that we will be building on, using some of the ideas discussed above, is Ren and Yang (2017). In this work they generate relevant and comprehensible jokes given user specified topic words. They use two data sets - short jokes of Conan O'Brien and news data. The motivation for including news data is that jokes often have themes from current affairs underlying them, and including them supplements training with other relevant input data. They evaluate their jokes using ratings from five English speaking humans. According to the paper, their methods outperformed the current state of the art (Petrovic and Matthews, 2013), without the fixed sentence structure that this probabilistic model required.

We are dubious about this claim, since the evaluation metric they choose to use here seems abstruse. While Ren and Yang (2017) achieves a higher Human Evaluation Score, Petrovic and Matthews (2013) outperforms them on percentage of "good jokes" (defined as being scored 2 or 3 on the linear scale of 0,1,2,3). They claim that their "neural joke generator has lower percentage of good jokes than the [Petrovic] baseline model, ... probably because [their] model generates the whole sentence rather than filling in a few slots of words" which we agree with. They further go on to say that their higher average "indicates that among the good (somewhat funny or funny) neural-generated jokes, the percentage of funny is higher, compared with the baseline". While it is true they are exploring a much richer space of joke possibilities, the results displayed in the paper seem rather marginal.

They used pre-trained `GloVe` embeddings (finding an embedding length of 300 to work best) to represent input words because they decided that the training corpus was not large enough to learn the embedding matrix. A Part-of-Speech (PoS) Tagger was then used to extract topic words (proper nouns) from training data. The bag of these topic words was then averaged to encode the extracted proper nouns. They tried concatenation to avoid information loss, but this leads to variable length output and dependence on order, which complicates the modelling.

The neural network architecture began with a linear sigmoid layer to increase dimension from word embedding size to some larger size in order to enable a larger RNN. Next they had an RNN (LSTM with attention-mechanism) decoder, which generates the joke conditional on topic representation. The attention-mechanism receives individual topic word embeddings, which allows it to attend to different topic words at different time steps in generation. The output of the RNN is a sequence of vectors, each one is the confidence of each word coming next in the sequence. They use a weighted-pick strategy to choose each word in proportion to the model's confidence. This strategy is thought to introduce an additional element of randomness, which the authors hope will lead to additional incongruity and therefore "funniness".

Early on, they trained the decoder-only model alone. This allowed them to tune the hyperparameters, such as number of layers and hidden state length, such that the output was the most syntactically correct generation. Then they include the encoder initialization of hidden state (which uses the averaged proper noun features) and finally they add the attention-mechanism. The attention mechanism interestingly caused a decrease in fluency. This is why they initialised the RNN parameters in the architecture with attention, with the RNN parameters from the architecture without attention.

Ren and Yang (2017) stated that they may be able to improve the encoder with a sequence model (which is could be a separate RNN) and have sequential input of proper nouns. This suggested extension of their work is what we expect to do in order to feed the model the entire set up sequence - motivating elements of the Project Roadmap as discussed in Section 3.2, and identifies a key limitation of their work.

The dataset of 7699 jokes was supplemented with news data (the quantity of which was unspecified) which seems relatively small, and could be a weakness of their model. Understandably, they didn't learn the embedding matrix, which may have proved useful in learning "joke specific" embeddings for this particular context, although it is unclear whether or not this would have any performance bonuses over the `GloVe` embeddings. The paper used word level RNNs over character level RNNs, citing that these often give better sentence cohesion, yielding particular benefit since one of their ratings of jokes in the human evaluation stage is *0: Nonsense (syntactically)*.

This work was followed by (Chippada and Saha, 2018), who trains controlled LSTM on jokes, quotes and tweets together, with the category as an extra input. They had a different number of samples for each category, so used a weighted sample strategy to ensure that the contribution of each category to the loss was equal. They also added a one-hot encoding to the `GloVe` encoding as input. Interestingly they also chose to use quite a different architecture, which did not include attention. The motivation for this was not made clear, which is a weakness of the paper, so we may need to investigate the performance of the two models ourselves. A few different LSTM variants were tested in the Chippada and Saha (2018) work, which is a key strength as there was an element of architecture exploration and optimisation. They extended the weighted-pick strategy using an exploration factor (0.1), which determines the proportion of the time that the model picks the maximum probability word or uses the weighted-pick. This exploration factor is advantageous specifically in the joke generation setting since we have established "incongruity" as correlating with the success of a joke, and this mechanism mimics this.

In addition to human evaluation, Chippada and Saha (2018) use some automatic evaluation metrics. They use a similarity metric to see whether the generated jokes are different from those in the training data and show that the model can generalise to novel jokes. They also use a Link Grammar Parser to evaluate syntactic correctness (Grinberg et al., 1995). This automatic evaluation is what we intend to improve on.

Both papers provide credible and well documented starting points for our project, exploring a variety of possible algorithms to tackle a difficult problem. The key limitation of both works that we suggest improving upon is the change to our objective function in the generative model, by including a term to directly measure "funniness".

3 Project Proposal

3.1 Research hypothesis

We hypothesise that joke generation models may be improved by introducing constant feedback on the quality of the generated joke during training.
We suggest using state-of-the-art word embeddings (`word2vec`, `GloVe`) to:
(a) Create a classification/regression model that measures how funny a joke is;
(b) Utilise this classifier to train/evaluate a generative model for "punchline" generation conditional on the "setup" of the joke.

3.2 Data acquisition

The data we plan on using for this project is a set of approximately 195,000 jokes found on Github (Pungas, 2017). To supplement this data, we may also scrape jokes through the Reddit API, and we have already built a working scraper to do this.

3.3 Project roadmap

Firstly, we plan to build the data processing pipelines. This will include restricting our attention to jokes of a certain length and throwing away anything where the Reddit joke punchline includes images or non standard characters, amongst other things. From initial tests this data set will have a size of 100-150k jokes. We will likely threshold the score (as determined by Reddit) between suitable values to constrain our regression output space, since the scores vary massively (see Section 4.1), supplementing our data via scraping if necessary. We will then encode the jokes initially using `word2vec` or `GloVe`. The generative model and scoring model can be worked on in parallel.

For the scoring model we will build and test a similar model to Kim (2014), which will predict the scores of our Reddit jokes. This is a small and simple model that can be used for benchmarking purposes. We can iterate on this model by experimenting with deeper architectures if performance is inadequate.

For the generative model research, we will start by implementing one of the current state of the art models as a benchmark - both Ren and Yang (2017) and Chippada and Saha (2018) have made their source code public and these will serve as a good starting point for this part of the project.

At this point, we will begin work on the novel element of our research. We will include the output of our scoring model in the objective function of the generative model and study how this effects the generated sequences. We will need to ensure that the hyperparameter balancing the two contributions to the loss, the sequence loss and score loss (as determined by our scoring model), balances the joke reproduction and "funniness" sufficiently well. This will involve hyperparameter tuning.

We will evaluate the relative performance of the benchmark generative model (i.e without our novel loss metric) and our prototype generative model (with the additional factor in the loss) using both the scoring model and human evaluation; the human evaluation serving to validate the success of our scoring model.

Human evaluation would involve holding about 100 jokes as a test set. We would run each of these jokes through our generative model, as well as our benchmark model(s), thereby creating (at least) three sets of jokes: a benchmark set(s), a set that was machine generated with a corresponding score (as determined by our scoring model) and a set that is the ground truth (with the score being determined by Reddit). Each of these sets will have the same setup. These three sets will be randomly selected and rated by humans. Human assessors would assign relative scores to the jokes based on their perceived comedic value, on a scale as yet to be determined, but would probably be similar to that of Ren and Yang (2017) - a linear scale from 0 to 3 where the correspondence of each number is as follows: *0 - nonsense; 1 - not funny; 2 - somewhat funny; 3 - funny*. We could then compare the performance of jokes generated by our models and the actual jokes from the data.

Depending on how much time we have, we may be able to proceed with further research. The metrics obtained from this modelling phase may give us some insight as to the direction we should take. One possibility is to look to improve the underlying benchmark models, which would involve tuning hyperparameter settings and possibly testing other architectures as well as word embeddings.

This project already has a dedicated Github repository, which will prove helpful in working on different sections of the project simultaneously.

3.4 Existing models and frameworks in this area

A common thread that we observe in the generative joke models literature is the issue of how best to evaluate the model's performance without having human's reading and subjectively assessing the model output. Yu et al. (2018) use metrics that compare the diversity of the generated sentences in order evaluate how varied the generated jokes are. They also train a separate 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 to evaluate syntactic correctness of the generated text.

None of the methods above attempt to assessing comedic value, they either assess syntactic correctness or novelty. We aim to 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 generative models that have been proposed previously in the literature.

For the scoring model we have found a similar project on Github (badsines, 2018) that uses Deep Learning to build a joke classifier. It does this by thresholding the joke score at a certain level and performing binary classification. It achieves good performance, with a 0.7 F1-score. We will be building on this by including ideas from Kim (2014), who applies a convolutional layer, followed by a linear layer and then a softmax layer above the word embeddings. We may try both predicting score in the regression setting and binning the scores and utilising the classification setting as in the first piece of work. The loss function that will be used to evaluate this model in each scenario would be the (Root-)Mean-Square Error ((R)MSE) in the continuous score setting and the cross-entropy loss in the multiclass setting.

For the generative model both Ren and Yang (2017) and Chippada and Saha (2018) have made their source code public and we will use this as a starting point. These models both utilise LSTMs, which have proven successful in many areas of natural language processing, with Chippada and Saha (2018) in particular trying a few variants of LSTM including Stacked LSTMs and Bidirectional LSTMs. In the case of the generative model, the input will be an embedding derived from the joke setup only and the output will be the generated punchline. The full concatenated joke is then fed into the scoring module. The resulting score and the sequence loss will be combined and propagated back through the network during training.

Kim (2014) used `word2vec` word embeddings whereas Ren and Yang (2017) and Chippada and Saha (2018) used `GloVe`. We will likely use `GloVe`, because we believe that the biggest challenges lie in the generative model side of the project, so following proven methodologies there will be more important. Time and data permitting, we may consider unfreezing the parameters of the embedding matrix later in training (Kim, 2014) in order for the model to learn task specific embeddings.

Much of the existing code base utilises Python and TensorFlow to build and test the models and we will continue this trend. We plan on using the Keras API as much as possible for quick prototyping and once a general architecture has been chosen, lower level APIs may be used for more granular model development. We have set up a dedicated Github repository for team members to share code and collaborate in a controlled manner.

A high level illustration of the flow of information through the two models during training is illustrated in Figure 1 below.

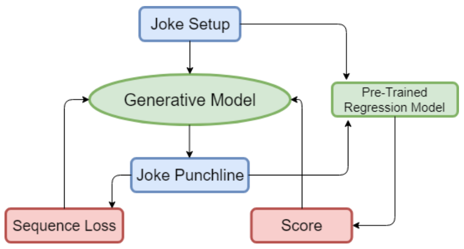

Figure 1: Training Loop

4 Feasibility Study

4.1 Exploratory Data Analysis and Summary Statistics

id	title	body	score
5tn84z	Breaking News: Bill Gates has agreed to pay for Trump's wall	On the condition he gets to install windows.	48526
4y28jy	How many Feminists does it take to screw in a lightbulb?	One. Men can be Feminists, too.	20353
2kdoye	What do you call a fat psychic?	A four-chin teller.	4850
4s1fdz	What's made of leather and sounds like a sneeze?	A shoe.	429
41itwh	Soap.	Radio!!!	0

Table 1: Sample of jokes from the data - jokes range from topical political commentary to absurdist comedy and many could be regarded as offensive by some.

The primary data set we plan to use for this project is a corpus of nearly 195,000 jokes scraped from the popular *Jokes* subreddit. This data set is popular in the literature and has been used for similar purposes in previous research projects. The data comes as a .json file and consists of four fields; *id*, *title*, *body* and *score* (A number calculated based on how many up-votes and down-votes the post received). Some examples are shown in Table 1.

To better understand the data, we performed some exploratory data analysis. Descriptive summary statistics and plots are shown below. Clearly, the data is heavily skewed with 50% of jokes scoring just 3 or less. This skew may make it difficult for our scoring model to learn to discriminate between low and high scoring jokes. Consequently, it may be judicious to attain more joke data with higher scores to counter this imbalance.

From Figure 2, we can gain a number of insights about the data. For instance, jokes with a very high number of words in the body do not score well. It is likely these jokes are either spam or simply take to long to read to engage users. We see a similar trend with respect to title lengths. We can also see from the third plot that there is a greater range of body lengths at smaller values of title length.

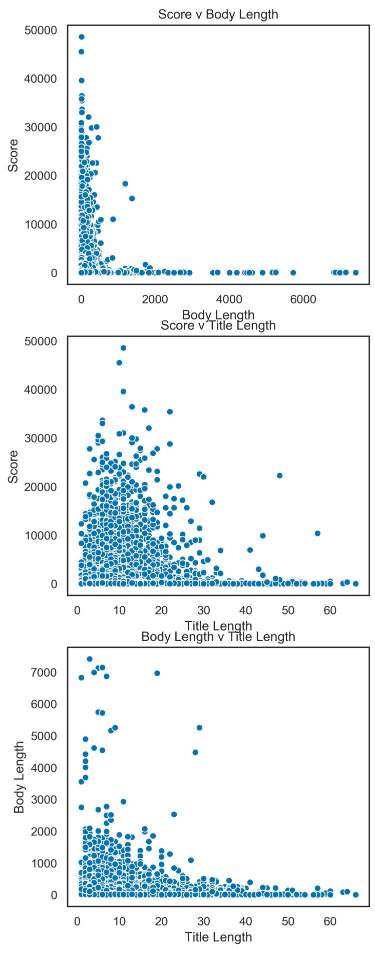

Figure 2: Scatter Plots of the relationships between Score, Body Length and Title Length, where length is the number of words.

	Score	Body Length	Title Length
Count	194553	194553	194553
Mean	118.22	37.66	8.59
S.D.	936.23	103.02	4.46
Min	0	1	1
25%	0	4	6
50%	3	8	8
75%	16	26	11
Max	48526	7408	66

Table 2: Summary Statistics of Score, Body Length and Title Length, where length is the number of words.

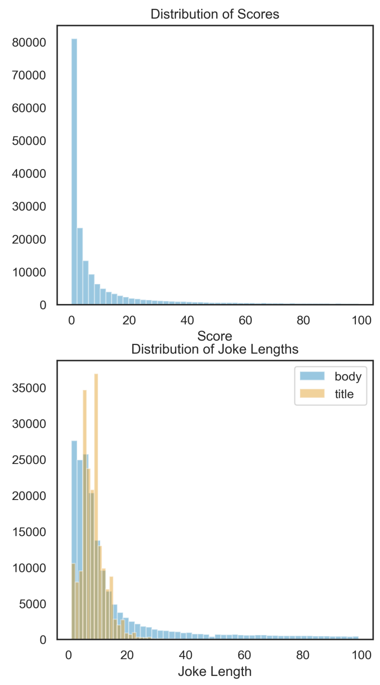

Figure 3: Histograms of Scores and Joke Length, where length is the number of words. The x-axes have been truncated due to the long tails for display purposes.

The data is already in a usable format. In most cases the *title* field corresponds to a joke setup or a question and the *body* field corresponds to the punchline. We also decided to set some constraints on the project scope by only using jokes we considered to be "well structured" i.e consisting of clearly separable "setups" and "punchlines" and grammatically correct sentences (some jokes made use of surreal or abstract humour that would likely prove difficult to learn) as well as preprocessing our data to only include ascii characters. Additionally, we set a limit on the length of the jokes as shorter jokes should in theory be easier to learn for the generative model. We may also need to consider omitting jokes that make use of a very specific vocabulary that is unlikely to be found in the pre-trained embeddings.

To circumvent extreme skewness, as evidenced in Figure 3 which shows the distribution of joke scores decaying very rapidly (which may pose a problem for the scoring model), we plan to threshold the score so that it only takes values in a certain range (which will require careful thought). Thresholding the score allows us to overcome popularity and age biases of the Reddit posts in the data. As mentioned above, we additionally would want to threshold the length of the body and the title, restricting our attention to more classical jokes with a shorter punchline. This will also help us remove the unusual "spam"-like jokes from our dataset. This preprocessing *without* score thresholding leaves us with around 100-150k jokes.

These constraints will reduce the size of the data set. We see that the score distribution of this dataset is very long tailed (as seen in Figure 3) with Table 2 indicating that 75% of our data has a score of 16 or less. This means that if we wanted to bound our scores inside some suitable range, we might have large regions of the score domain being unoccupied with jokes (i.e we have a few jokes in the hundreds, a few in the thousands with not much in between). Since we believe that the lower scoring jokes aren't as funny, we can augment the original data set with additional joke data scraped from Reddit through the Reddit API to supplement key ranges of the scores that we are interested in. This additional data would be useful for "smoothing" our distribution of scores by acquiring jokes with higher scores. This may be necessary due to the heavily skewed nature of the score distributions in the figures and tables.

4.2 Motivations from related work(s)

In terms of the feasibility of our proposed scoring model, one project found online (badsines, 2018) achieved good results on a similar task. Instead of a regression model, they posed the problem as a binary classification task and assigned labels of *funny* and *not funny* based on some threshold score. Using a very simple single hidden layer neural network, they achieved a F1-score of roughly 70%, suggesting that our task should be achievable. If the performance of our regression model is insufficient, we may consider binning the score response and re-framing the problem as multiclass classification, as described in Section 3.4.

The Github repository relating to (Chippada and Saha, 2018), found under the reference of (Chippada, 2018), contains useful Jupyter Notebooks with varying LSTM architectures (with attention-mechanisms) in Keras and Tensorflow. These will provide us with relevant code that we can implement and build upon for the second part of our project. As mentioned throughout, one key thing we will need to build upon is the implementation of our novel objective function by utilising the outputs of the scoring model, but this code will certainly provide a starting point.

References

badsines. 2018. A nlp dnn joke classifier using reddit data. https://github.com/badsines/JokeNLP.

Bhargav Chippada. 2018. Generate humor using lstm variant. https://github.com/bhargavchippada/humor-generation.

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

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*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks.

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*.

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.

Iulian Vlad Serban, Alberto Garca-Durn, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1017–1024, USA. Omnipress.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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.