What is Humor for Deep Learning Models?

Shubham Annadate, Nupur Baghel, Nikhil Motwani and Natasha Menon
University of Pennsylvania
{annadate,nbaghel,nmotwani,natmenon}@seas.upenn.edu

1 INTRODUCTION

Humans have a sense of finding things funny. Why do we laugh at such things? How we detect humor in a sentence and find joy in the humor? As simple as it seems to us, the answer is more complicated when it comes to training Deep Learning Models for identifying the same. This project is an attempt to demonstrate this complexity. Given a dataset containing carefully curated samples of jokes and non-jokes, we make an attempt to train various Deep Learning models to learn patterns to discriminate between the two. Following that, we bring novelty to the project by making use of Deep Learning models that add interpretability to their classification, conducting ablation studies, and employing statistical techniques to gain insight into what our models have learnt, and understand the notion of humor for these models.

2 RELATED WORK

A number of studies have been conducted to identify humor in text. Peng-Yu Chen et al. [5] built a humor recognizer using CNNs with extensive filter size on Chinese and English Datasets, achieving equivalent performance on both of these languages. Luke de Oliveira et al. [6] used Shallow models such as Random Forest and Linear Classifiers on Bag of Words model. They also experimented with Feedforward Neural Networks, CNNs and RNNs on word vector features to identify humor in a text.

Interpreting what these Deep Learning models learn is also extensively studied. Jiwei Li et al. [10] used several methods to visualize and interpret neural models by using RNN, LSTMs and Bi-Directional LSTM autoencoder models to understand how neural models convert the natural language into semantic space. They also studied how the neural models compose meanings and are able to detect irregularities of negation of the sentences. Andrej Karpathy et al. [8] used character-level language models for testing the interpretability of LSTMs in learning long range dependencies. They analysed that certain neurons within the network identify interpretable high level patterns such as quotes, length and brackets.

In addition, several statistical techniques have also been used to make machine learning models more interpretable. Riberio et al. [13] developed LIME, an explanatory technique that helps to determine words that have influenced the prediction of the model. This technique can expose models whose predictions are due to peculiarity of the data rather than the generalization of the model. Leila et al. [4] applied a technique called as Layer-wise Relevance Propagation (LRP), which assesses the contribution of each individual word in a text to the prediction of the model by backpropagating its output. Similar work was done by Jiwei Li et al. [10] where they used the cue of backpropagation and calculated the first order derivative of loss function with respect to the word-embeddings of the words in the sentence to know their contributions. They observed that higher the value of derivative, higher the contribution of the word in the prediction of the model. Ákos Kádár et al. [7] studied a technique that can be used to interpret RNNs. They compared the activation vector of a full sentence, produced by the neural network and the activation vector when a word from the sentence is removed. They took the difference between the two vectors and observed that higher the difference, higher the contribution of the word in the prediction of the model. Similar work was done by Jiwei Li et al. [10], where instead of focusing on the activation vector, they focused on final Log-Likelihood that the model assigns to the correct label for each of the two inputs.

3 DATASET

3.1 Dataset Curation

To accomplish this learning task of identifying humor, we curated a dataset consisting of jokes and non-jokes using four different data sources. The joke samples were from a combination of 'The Reddit Question-Answer Joke' [14] and 'The Short Jokes' [12] datasets, whereas the non-joke samples were from a combination of 'The SQuAD 2.0' [3] and 'The Reddit QA Corpus' [1] datasets. For our investigation we only focused on samples with the specific structure of Setup-Response format (i.e. Builtup-Punchline for jokes and Question-Answer for non-jokes). The combined dataset consisted of 1,031,549 non-jokes and 113,855 jokes.

3.2 Exploratory Data Analysis

As part of Exploratory Data Analysis, we looked at the distribution of various features and descriptive statistics that could be extracted from the samples. This involved analyzing the lengths of Setup and Response (combined and individually), finding the proportion of samples consisting of a Wh-word (i.e. what, when, where, who, whom, which, whose, why and how), multiple question marks, multiple full-stops, weird symbols or digits and empty setup/responses for both jokes and non-jokes. The statistics for the same have been summarized in Table 1. It is clear from this table that there is discrepancy between the various statistics for jokes and non-jokes with the proportion being relatively higher on the non-jokes side for all the statistics.

3.3 Data Cleaning based on EDA

The Exploratory Data Analysis revealed that there are features in the data that are significantly different for both jokes and non-jokes. It is important that we get rid of such discrepancies so that our model learns the actual pattern for a sample being a joke or non-joke and does not fixate on learning these discrepancies. Therefore, we performed the following tasks to maintain uniformity between the jokes and non-jokes samples:

Statistic	Jokes	Non-Jokes
% sentences with Digits	7.9	18.2
% sentences with Weird Symbols	7.9	28.2
% sentences with More than 3 full stops	2.3	10.6
% sentences with More than 1 question mark	0.2	3.9
% sentences with Wh-words in setup	78.9	79.7
Median length of Setup	9	10
Median length of Response	5	8
Median length of Combination	14	18

Table 1: Statistical analysis

- Removed any empty setup or response fields
- Filtered out duplicates and any NaN values
- Removed any samples that have digits or any weird symbols
- Removed any samples that have greater than 3 full stops or multiple questions marks in setup
- Removed any samples where the response contains a question mark, as well as if the setup does not start with a Wh-word
- Removed any samples where the setup or response length is more than 25
- Carefully extracting samples such that the vocabulary match between jokes and non-jokes is maximal.

The percentage of jokes containing words that are common for both jokes and non-jokes matched to 99.3%, as well as the same process for non-jokes matched to 99.6%, hence showing that the vocabulary used in both categories is very similar.

3.4 Data Splits

We then used the final dataset to create two different data splits where each split had a train, validation and test set in the 7:2:1 ratio. In the first data split, each held-out set had an imbalanced distribution of jokes and non-jokes, with the number of non-jokes being in majority. This distribution resembled real world data and was used to interpret whether the model is able to identify jokes amongst many non-jokes. In the second data split, we reserved one joke dataset completely as the test set (Reddit Jokes) and created similar imbalanced data distributions for train and validation using the remaining sources. This helped to check the ability of the model to generalize on a different joke domain on which it was not trained. It also helped interpret whether the model is able to identify the structure of a joke rather than basing its decisions on some other pattern. Table 2 gives the distribution of jokes in the different splits.

Data Split	Percentage of Jokes		
	Training	Validation	Testing
1	21.2	21.0	21.2
2	15.2	15.1	100.0

Table 2: Data Distribution

4 JOKE RATIONALE

Following we list some properties that we believe hold when a given sample is a joke:

(1) The setup and response both play an important role in determining whether a given sentence is a joke.
(2) The order of the setup and the response matters for a sentence to be a joke.
(3) The setup should act as an appropriate context to the response for the sentence to be a joke.
(4) The order of words in the response of a joke matters for a sentence to be a joke.
(5) A sentence becomes funny because of the presence of certain keywords in either the setup or the response.

5 METHODS

5.1 Non Deep Learning

For our baseline models we used three non deep-learning methods which include Logistic Regression, Gaussian Naïve Bayes and XGBoost. In order for the input data to be fair with all models, we gave the S-BERT sentence embeddings for the Setup and Response as inputs which were generated using a pre-trained BERT network.

For hyperparameter tuning we varied the max-iter parameter (5000, 10000, 15000) of logistic regression, which mostly gave equally good F1 scores on the validation set. Moreover, for XGBoost we tried different combinations of learning rate (0.1, 0.3, 0.5, 0.7, 1.0), max depth (3, 5, 10) and gamma (0.1, 0.2, 0.5) parameter values after which we found the optimum values to be learning rate = 0.5, max depth = 5 and gamma = 0.2. The greater the max depth and gamma value, the greater the training scores but they seemed to be overfitting on the dataset as the validation scores were comparatively low.

5.2 Deep Learning

We implemented 5 deep learning architectures, where each proposed architecture could be used to study different inductive biases which help us in distinguishing between jokes and non-jokes. We use the term Setup/Question/Builtup and Response/Answer/Punchline interchangeably.

Note: Although we have mentioned LSTM as the recurrent layer, we also experiment by using Vanilla RNN, GRU and BiLSTM for each of the configurations shown. Also, hyperparameter tuning is conducted for all models but we make a detailed explanation for a couple of architectures.

5.2.1 Architecture 1. The first architecture (Figure 1) we tried is one that can be considered as a baseline for Deep Learning models. It takes the combined Setup and Response sentence as input and passes it through the LSTM network to get the sentence (Q+A) embedding. This is passed into fully connected layers to generate the final prediction. The inductive bias behind this model is that the (Q+A) embedding obtained encapsulates the meaning of the sentence, including whether it is a joke or not. The final hyperparameters were: 1 recurrent layer, hidden state size = 128, learning rate = 0.001, Adam optimizer , training epochs = 5, and 300 dimension GloVe embeddings for input words

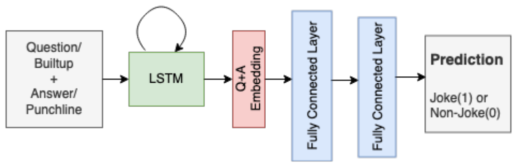


Figure 1: First Architecture

5.2.2 Architecture 2. The second architecture (Figure 2) uses Setup and Response inputs in parallel and passes them into LSTM networks to produce embeddings for each of them. We next concatenate these embeddings and pass them into fully connected layers to produce the final predictions. In this architecture we are assuming that there is no interaction between words in the setup and response and thus we are only accounting for interaction of words within each part separately. We believe that if there actually exists this independent context relation between the words in Setup and Response, then this model might perform better.

For hyper-parameter tuning we tried to vary the learning rate (0.001, 0.0005, 0.0001) for Adam optimizer, hidden state size (64, 128) for recurrent layer and the number of training epochs (3,5,10). The LSTM network used for Setup and Response was kept exactly same, each having one recurrent layer. Also the fully connected layers had sizes ($2 \times hidden\_size$) and ($hidden\_size$) respectively. We observed that using a learning rate of 0.0001 or 0.0005 caused the model training to become very slow, requiring at least 10 epochs, while a learning rate of 0.001 resulted in the same performance in only 3 epochs. While training for 5 epochs was also giving similar results, the validation performance deteriorated due to overfitting. Increase in hidden-state size also lead to an improvement in performance scores. Hence, our final model hyper-parameters were learning rate = 0.001, hidden state size = 128, training epochs = 3, and 300 dimension GloVe embeddings for input words.

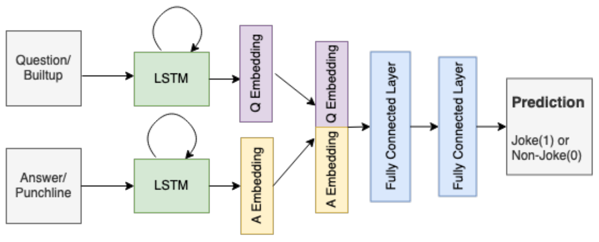


Figure 2: Second Architecture

5.2.3 Architecture 3. The third architecture (Figure 3) first takes as input the Setup and passes it into an LSTM layer to produce the Setup (Q) embeddings. These embeddings serve as the context to the next LSTM layer which takes Response (A) as its input. The output of the second LSTM is thus an Q+A embedding which is passed into fully connected layers to produce the final predictions. With this architecture we are assuming that the Setup embeddings create a context which can be utilized while looking at the Response, thus creating more coherent embeddings for classification. Our final model hyper-parameters were 1 recurrent layer, learning rate = 0.001, hidden state size = 256, training epochs = 3, and 300 dimension GloVe embeddings for input words.

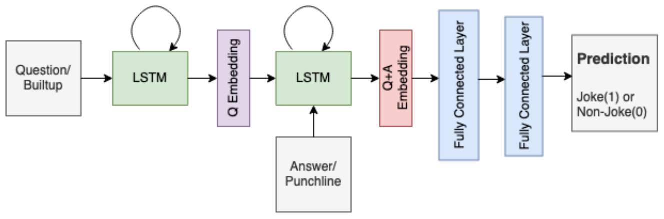


Figure 3: Third Architecture

5.2.4 Architecture 4. The fourth architecture (Figure 4) uses a sentence embedding for the Setup and Response (S-BERT) as input. This is passed into a pre-trained BERT network, followed by several fully connected layers to finally produce a 0/1 output. According to our intuition, since this model produces state of the art BERT contextualized embeddings, it should be better able to capture the hidden notion of a sample being a joke or a non-joke. The final hyperparameters for this model were: model = BERT-base, BERT network learning rate = 0.00001, FC layer learning rate = 0.001, max sentence length = 64, epochs = 1.

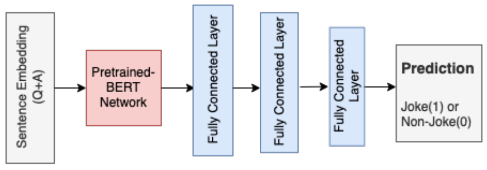
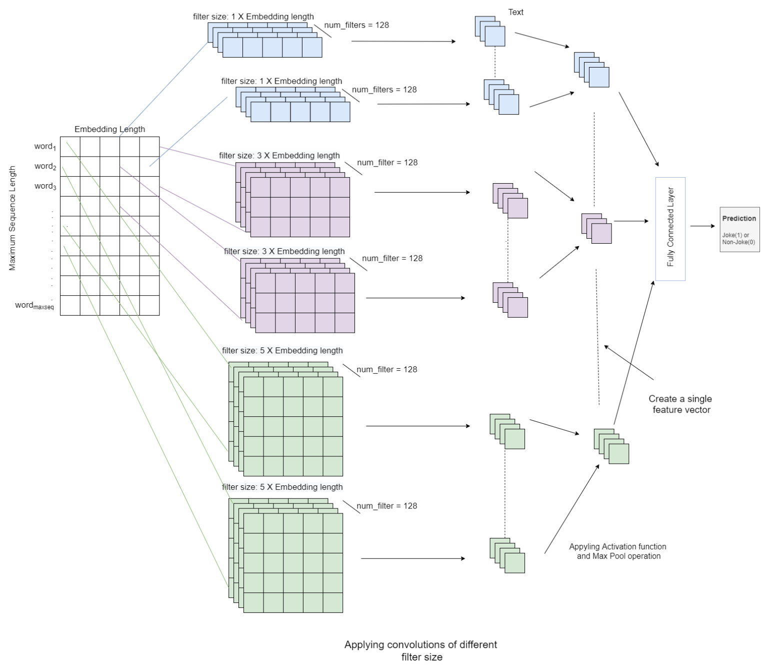


Figure 4: Fourth Architecture

5.2.5 Architecture 5. This architecture is based on using CNNs for text classification (Figure 5). It takes GLove embeddings of Setup and Response as an input and applies convolutions of filter sizes (f) 1, 3 and 5 to the embedding matrix of the size MAX_SEQ_LENGTH (max_{seq}) × EMBEDDING LENGTH. The width of the filter would be equal to that of the embedding length. We used 128 filters ($num\_filters$) for each size of the filter. After applying convolutions with $stride$(=1 in our case) and $padding$(=0 in our case) we would get the output whose dimension would be $\left(\lfloor \frac{max_{seq}-f+2\times padding}{stride} \rfloor + 1)\right) \times 1 \times num\_filters$, We apply ReLU Activation function and then do 1D Max Pooling to get a output of size $1\times1\times num\_filters$, for each size of the filter. The output that we get from each size of the vectors are the stacked up together to get total dimension as $\left(\lfloor \frac{max_{seq}-f+2\times padding}{stride} \rfloor + 1)\right) \times n_f \times num\_filters$, where n_f is the number of filter sizes we have used and the stacked output is flattened and then passed to the fully connected layer and



filter size: 1 X Embedding length
num_filters = 128
filter size: 3 X Embedding length
num_filter = 128
filter size: 5 X Embedding length
Embedding Length
Maximum Sequence Length
word1
word2
word3
wordmaxseq
Text
Fully Connected Layer
Prediction
Joke(1) or Non-Joke(0)
Create a single feature vector
Applying Activation function and Max Pool operation
Applying convolutions of different filter size

Figure 5: Architecture 5 - CNN with multi-size filters

final softmax layer is applied to get the final prediction. The inductive bias behind this model is that the sequence of words contains some semantic meaning which makes it a joke or non-joke and can be picked up by this architecture.

5.3 Advanced/Interpretable Deep Learning

5.3.1 LSTM + Attention. This architecture is an advancement to the Architecture 1 discussed in Section 5.2.1. It makes use of additive self-attention to weigh the past hidden states produced by the individual tokens based on their importance in determining whether the sentence is a joke or not. This is our first step in interpretability of Deep Networks wherein we can get insights into which words are important according to our model in distinguishing between a sentence being a joke or not, and also to verify whether our notion of the same aligns with it.

For hyperparameter tuning, we tried to tune the hidden state size (64, 128, 256), learning rate (0.001, 0.0005, 0.0001), number of layers FC layers in the attention module (1,2,3), and the number of training epochs (3,5,7,10) to optimize on the performance on the validation set. We observed that we required considerably more number of epochs to train a model with attention, and decreasing the learning rate resulted in a more stable learning curve. The final hyper-parameters that we chose were hidden state size = 128, learning rate = 0.0005, 2 FC layers in attention module with dropout probability of 0.3, 7 training epochs and 300 dimension GloVe embeddings for input words.

5.3.2 Roberta, Albert, XLNet. We used pretrained models namely, Roberta, Albert and XLNet, [11] [9] [16] which were fine tuned using the Simple Transformer library build on top of the Hugging Face repository [2]. The models learn the contextual relations between the words in a text, basically a contextualized embedding of the sentence. We used roberta-base model type for Roberta which has 125M parameters, albert-base-v1 for Albert (11M parameters) and xlnet-base-cased model type (110M parameters) for XLNet. The default parameters were used in case of Albert and XLNet while for Roberta, a modified learning rate of 1e-5 was used.

5.3.3 LIME. This method is an explanatory technique that enables the model results to be interpretable as it supports the decision made by the model, by determining the group of words/features that have influenced the result. LIME [13] tries to combine interpretability with locally fidelity, meaning that it shows how the model will behave under the given vicinity of the sample under investigation.

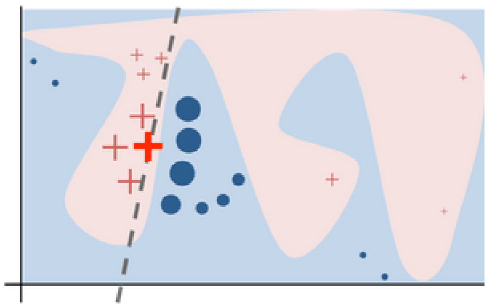


Figure 6: Local linear approximation in LIME (Figure taken from [13])

The way LIME works is by taking a sample as an input and generates various perturbations of the sample to describe the vicinity of the data. For example, consider Figure 6, where the blue and the red regions correspond to the two classes being considered, and is clearly non-linear. The bold red plus is the instance being explained, for our case, this might be a joke or a non-joke sample. We then sample instances around this instance using various perturbations, and weigh them according to their proximity to the sample being explained. As can be seen from the figure, some perturbation can result in the sample being in the red region and some can result in the sample being in the blue region. A linear model is then learnt (dashed grey line) on top of these samples in an attempt to approximate the model well in the vicinity of the sample being explained. The probabilities of perturbed samples from this model fit can then be used to get importance of the words/features in the sample being explained, and also the likely direction the word/feature would drive the final prediction.

5.3.4 First-Derivative Saliency. This method [10] is based on finding how much an input contributes to the final decision of a Deep Network by calculating the first derivatives. More formally, for a classification model, an input E is associated with a gold-standard class label c. For the models that we employ in our analysis, the input E refers to the word embeddings of the words constituting a sentence. Given this input word embeddings E along with the associated class label c, the trained model associates the pair (E, c) with a score $S_c(E)$. The goal in interpretation, is to then decide which unit in E makes the most significant contribution to $S_c(E)$, and thus the decision of the of the class label c.

In Deep Networks, the class score $S_c(E)$ is highly non-linear function but can be approximated by taking the derivative of the loss function with respect to the word embedding of the words in the sentence, that is,

$$w(e) = \frac{\partial(S_c)}{\partial e}$$

where $w(e)$ is the derivative of the S_c with respect to the embedding e. The higher the absolute value of this derivative, the more a word is responsible for the final decision of the class label c.

6 ANALYSIS

Table 3 presents the performance of all models on Data Split 1 and 2. We use macro F1 score as the evaluation metric for validation data in both the splits and test data in split 1, since these datasets contain an imbalanced proportion of jokes and non-jokes. Further, since test data in split 2 contains only jokes, we measure the accuracy of predictions.

6.0.1 Data Split 1. From the performance metrics reported in Table 3, we see that clearly the Deep Learning based models perform better in comparison to the baseline models. Although, if we try to make the same comparison between Deep Learning and Advanced Deep Learning models we see that there is no significant improvement in the performance metrics. This indicates that, if one is interested in performance, it is fairly easy to distinguish between a joke and a non-joke just by using a simple Deep Learning based and there is no need to employ an Advanced Deep Learning model.

Model Name	Data Split 1		Data Split 2	
	Val F1	Test F1	Val F1	Test Acc
Non-Deep Learning				
Logistic Regression	0.122	0.13	0.0327	0.0163
Naive Bayes	0.36	0.363	0.2743	0.9657
XGBoost	0.8325	0.828	0.8047	0.7328
Deep Learning				
Architecture 1 - RNN	0.925	0.924	0.986	0.020
Architecture 1 - LSTM	0.954	0.953	0.991	0.025
Architecture 1 - GRU	0.955	0.955	0.993	0.018
Architecture 1 - BiLSTM	0.958	0.959	0.992	0.046
Architecture 2 - RNN	0.954	0.950	0.988	0.016
Architecture 2 - LSTM	0.968	0.964	0.992	0.029
Architecture 2 - GRU	0.963	0.961	0.992	0.056
Architecture 2 - BiLSTM	0.968	0.966	0.994	0.028
Architecture 3 - RNN	0.917	0.908	0.97	0.022
Architecture 3 - LSTM	0.967	0.965	0.994	0.020
Architecture 3 - GRU	0.965	0.962	0.992	0.025
Architecture 3 - BiLSTM	0.969	**0.968**	0.994	0.017
Architecture 4 (BERT+FC)	0.963	0.960	0.961	**0.927**
Architecture 5 (CNN)	0.95	0.952	0.989	0.06
Advanced Deep Learning				
Attention - RNN	0.929	0.912	0.989	0.014
Attention - LSTM	0.955	0.950	0.992	0.026
Attention - GRU	0.942	0.941	0.993	0.032
Attention - BiLSTM	0.960	0.952	0.992	0.088
RoBERTa	0.961	0.960	0.999	0.012
ALBERT	0.957	0.956	0.95	0.881
XLNet	0.967	0.967	0.921	0.814

Table 3: Model Performance across Different Methods

We also report the training and validation F1 curves for various models. Figure 7 shows the learning curves for Architecture 2 and Figure 8 shows the same for our attention-based architecture. These curves indicate that the performance of LSTM, GRU and BiLSTM are quite similar while Vanilla RNN lags behind.

6.0.2 Data Split 2. The notion behind having a Data Split 2 was to check the generalizability of these models in correctly identifying jokes by having a separate test set of jokes coming from a source that the model has never seen. As can be seen from the performance metrics only the XGBoost model was able to generalize well among the non-Deep Learning models. Although Naive Bayes also gave a high performance on the Test set, but we suspect that this was due to it incorrectly predicting everything as a joke, as can be seen by its poor performance on the validation set.

Except for the Deep Learning model that was based on using the BERT sentence embeddings, all the other architectures failed to generalize on the test set, athough the performance on the validation set was very high. This indicated that the models might have learnt some patterns that do not necessarily make a sentence joke, but it able to distinguish between the samples it was trained on.

The attention based Advanced Deep Learning also failed to generalize on the test set. Although other BERT based advanced models like ALBERT and XLNet generalized quite well, except for RoBERTa which might be due the fact that it was trained on a different text dataset (coming from Facebook) compared to the former mentioned models (coming from Google).

6.0.3 Key Observations. The above performance analysis bring out the following key observations:

(1) If the task is to just distinguish between jokes and non-jokes without bothering much about generalizability, then simple Deep Learning models are sufficient.
(2) If we want a model that generalizes well, that is it picks up the essence of a sentence being a joke, one cannot rely on simple and even some advanced Deep Learning models. The performance metrics clearly indicate that these models are picking up some patterns that do not necessarily capture the essence of a joke, hence pointing to the need for further investigation in understanding what these models have learnt.
(3) BERT based model are better able to generalize on unseen joke dataset, although the figures are not clearly indicative of the whether the model learnt something relevant to identifying jokes. It is likely that it might have learnt some other pattern that generalized well.

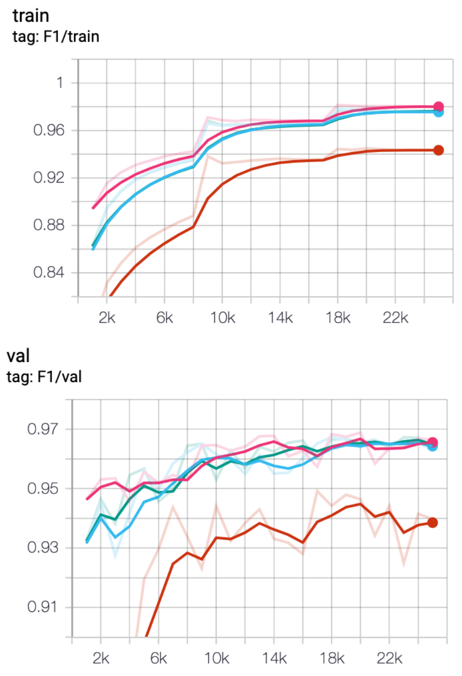


Figure 7: Learning Curves for Deep learning models (Architecture 2) where Red = RNN, Light-Blue = GRU, Green = LSTM and Pink = BiLSTM

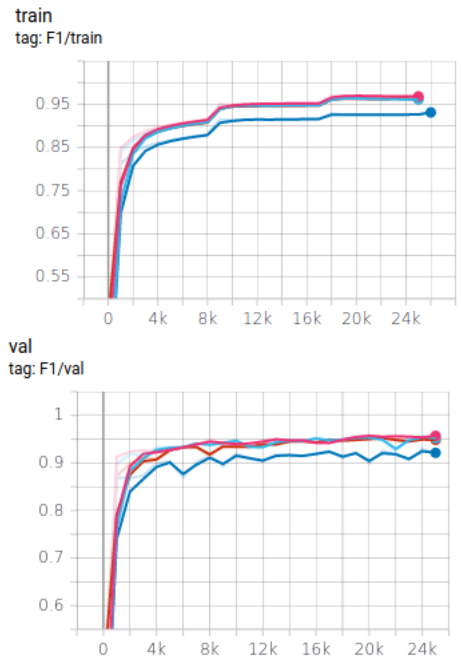


Figure 8: Learning Curves for Advanced deep learning model (Attention-based architecture) where Dark-Blue = RNN, Light-Blue = GRU, Red = LSTM and Pink = BiLSTM

7 ABLATION STUDIES

7.1 Description

For the purpose of testing our different joke rationales and understanding what our deep learning models had actually learnt, we created different variants of our test data set (split 1) containing both jokes and non-jokes. We next describe each of these splits, and give examples in the format [Original Sentence] -> [Altered Sentence]

(1) **Reverse Setup and Response** - Here, we reversed the order of setup and response. So, the sentence started with a response and was followed by a setup. The intuition behind this was to understand whether our model is able to identify that a joke becomes funny because of the setup providing a context or background information to the punchline which follows afterwards. Consider an example sentence:

What word becomes shorter when you add two letters to it? Short. -> Short. What word becomes shorter when you add two letters to it?

(2) **Jumbling the words** - The order of words gives a semantic representation to the sentence. We wanted to experiment whether our model takes into account the sequence of words when deciding whether a sentence is funny. According to our intuition, shuffling of words should change the attention of

model to different words and thus result in changing results from joke to non-joke. Consider an example sentence:

What did the octopus make for desert? ...Octopie -> the make desert? did ...Octopie What for octopus

(3) **Supplying a different context** - Another important factor for a funny Question-Answer pair is the relevance of context between the Setup and Response. That is, if the Question and Answer belong to different topics or are unrelated, the sentence will not be funny. To test whether our deep learning model also takes this into consideration, we shuffled the responses in our test data set so that each setup is matched with a random response. Consider an example sentence:

What did the octopus make for desert? ...Octopie -> What did the octopus make for desert? Claire Forlani, was not expecting the accent or cussing. She is also one of the hottest women I have ever seen.

(4) **Removing most-attended words** - An important part of a joke are the subjects of discussion, i.e. the characters, objects, etc. along with the verbal phrases, which define the activity the subjects are involved in. We believe that these keywords play an essential role in the context of the joke and that these words are given the highest attention by our deep learning model (Section 5.3.1). If we omit these keywords, then the sentence should lose its humor. Thus, by identifying and removing the highest attention word in every sentence, we created a new data split and studied how the attention scores and classification results changed. Consider an example sentence where the word 'ducks' got the highest attention:

Why do ducks have webbed feet? To stamp out forest fires! -> Why do have webbed feet? To stamp out forest fires!

(5) **Effect of punctuation characters** - While we removed weird characters which we believed should not be used by our model in making distinction between jokes and non-jokes, valid punctuation characters were still present in our test data set. Since their presence was skewed (as shown by Table 4), we created a data set which did not have these characters, to find out whether our model predictions vary for this case. Consider an example sentence:

Did you hear oxygen and magnesium started dating? I know, right! Like, OMg! -> Did you hear oxygen and magnesium started dating I know right Like OMg

7.2 Ablation Study Results

Table 5 shows the Recall score (of jokes) for architecture 1 - LSTM model on various ablation datasets described in the previous section.

Punctuation	Proportion in Non-Jokes
Exclamation (!)	0.525
Comma (,)	0.878
Single Quote (')	0.795
Double Quote (")	0.951
Question mark (?)	0.787
Full stop (.)	0.809

Table 4: Relative Punctuation distribution

We calculate recall score for this case because we are interested in examining how many jokes were mis-classified as non-jokes due to the particular ablation. We thus expected the recall to drop drastically for all cases. However, we observe that the drop in cases 3, 4, 5 and 6 is not substantial. This probably hints towards the fact that first, the deep learning model is unable to recognise the relevance of contexts between setup and response. Second, in the absence of most-attention word, it figures out the other important words and determines the sentence was a joke. Finally, it does not pay total attention to punctuations as a major distinguishing feature. Our additional ablation experiments where we only have setup as input (6) or response as input (7) indicate towards the fact that our model focuses more on the setup and is able to perform pretty well even without the response.

Exp	Description	Test Recall
1	Reverse Setup-Response	0.562
2	Jumbling words	0.518
3	Inappropriate Context	0.825
4	Most-attended word removed	0.909
5	Without Punctuation	0.794
6	Only Setup	0.731
7	Only Response	0.455
-	Original Test set	0.954

Table 5: Performance of model on Ablation datasets

8 INTERPRETATION

The observations from the model performance results and the ablation studies, both point towards the fact that the model has learnt some peculiarities in the dataset, which not-necessarily relate to a sample being a joke or non-joke, but is sufficient enough to be exploited for classification purpose. Following we employ our attention based model and other statistical techniques mentioned in Section 5.3, to come up with an explanation for these peculiarities.

8.1 Word Level Statistics

Although we tried to make the vocabulary same for jokes and non-jokes, it was difficult to also account for keeping the relative occurrence of each word same in both the categories. Table 6 shows the top 5 most frequent words in jokes and non-jokes with their respective relative proportions. As can be seen, certain words like *lightbulb* are more likely to occur in a joke sample compared to a non-joke sample. In the following sections we study sentences

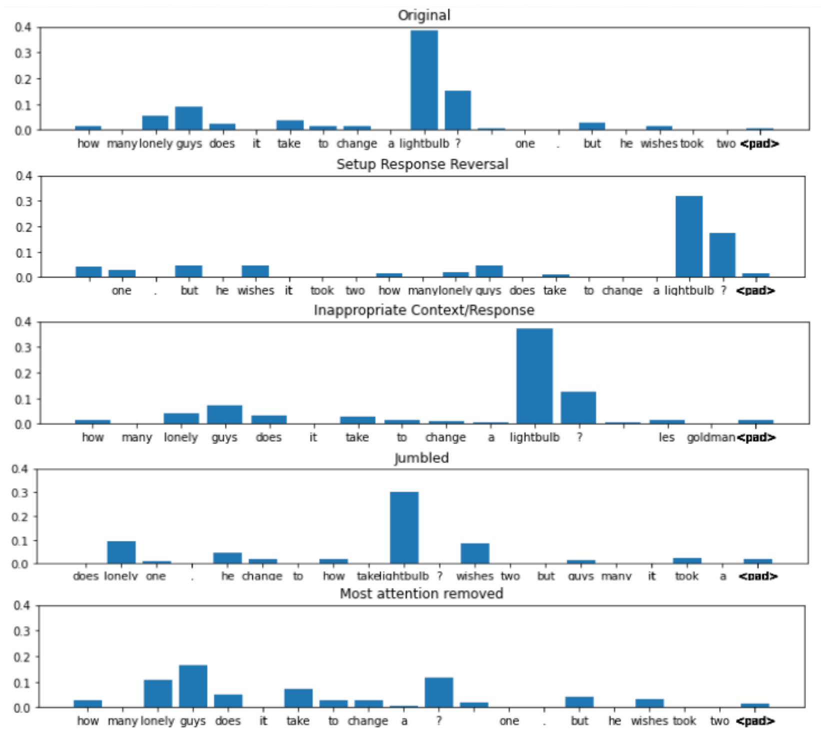


Figure 9: Attention on the words making up a joke for different ablation studies

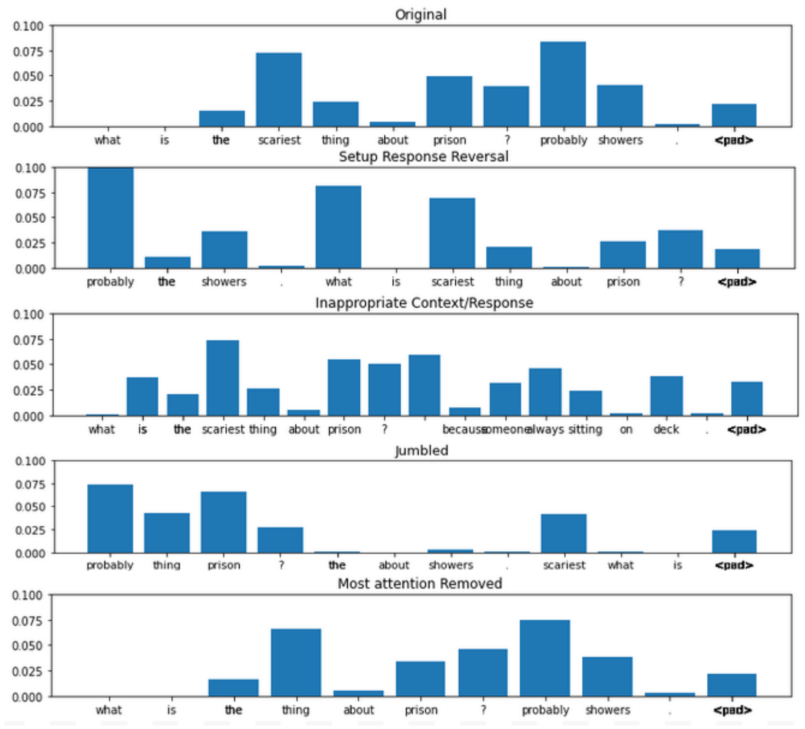


Figure 10: Attention on the words making up a non-joke for different ablation studies

containing these words to understand whether our model learnt and used this difference in word distribution as a distinguishing feature.

Jokes		Non-Jokes	
Word	Proportion	Word	Proportion
lightbulb	0.9777	moment	0.9969
bulb	0.9642	weird	0.9943
dyslexic	0.9545	comment	0.9935
jew	0.95	scariest	0.9911
busty	0.9444	dumbest	0.9908

Table 6: Relative Word-Frequency Distribution

Using the words common to jokes and non-jokes, we also calculated the Pearson correlation between the likelihood of a word being in joke sample with the median attention score this word gets across all the joke samples it occurs in. This had a value of 0.112 with a p-value of 7e-24. Carrying out a similar experiment for non-jokes, the correlation was -0.104 with a p-value of 9e-21. These figures indicate that relatively frequent words in jokes are given higher attention scores when making classification decision, while this does not happen with frequent words in non-jokes.

8.2 Analyzing Attention Scores

Given the relatively high occurrence of the word *lightbulb* in the jokes sample, and similarly the high occurrence of the word *scariest* in the non-joke sample (Table 6), we next look at one example of joke and non-joke containing each of these words and study the effect of how the attention these words get varies based on the ablation studies being conducted.

Figure 9 shows how attention varies for different words for the joke *"How many lonely guys does it take to change a lightbulb? One, but he wishes took two"*. Looking at the plots, we make a few important observations:

(1) The word *lightbulb* gets a very high attention compared to the other words in the original sample, indicating that this word was majorly responsible for the sample being categorized as a joke.
(2) Despite of manipulating the sample based on the ablation study being done, that is reversing the setup and response, providing inappropriate response to the context or even jumbling the words in the sample, the sample always got categorized as a joke. Which is unusual as one would expect a joke to get categorized as a non-joke in each of the above cases.
(3) In each of the above mentioned ablation studies, the word *lightbulb* gets the highest attention. This suggests that the model might be looking for some keywords in the sample to make the classification and not the essence of a sample being a joke. This also provides some explanation as to why the test recall score was high despite of the ablation study being conducted in Table 5.
(4) On removing *lightbulb* from the sample, we notice that the sample still gets categorized as a joke, but the attention the word *lightbulb* got now gets distributed to other words in the sample, indicating that there might be an hierarchy in the order of words that the model looks for while making a classification.

A similar analysis was conducted on a non-joke sample: *"What is the scariest thing about prison? Probably showers"*. The results that we got here were similar to the one we got for the joke sample described above (Figure 10), but with the sample being always categorized as a non-joke despite of the ablation study being employed. In addition, the word *scariest* was not the most attended word in the original sample, unlike *lightbulb*, but the patterns it demonstrated were similar to *lightbulb* as in the previous case.

8.3 LIME

As used by Ribeiro et al [13], we use the LIME technique to interpret the results that are generated by our trained model, to determine how it came to the conclusion that it has made. In order to maintain uniformity among the different interpretability techniques, we used the same samples for joke and non-joke which contain the words *lightbulb* and *scariest* respectively.

Both figures (Figure 11 and Figure 12) contains two different visualizations for each sample where the original sample has been highlighted with different colours of various intensities where the colour represents the class (Orange for joke and Blue for non-joke) and the intensity represents the effect the highlighted word has on the model's decision. The bar plot represents the same knowledge (Red for Non-joke and Green for Joke) but along with the weights that are assigned to each feature (i.e. each word) by the model, so that the effect can be compared numerically.

Supporting the claims made before, we can see that the words *lightbulb* and *did* in the joke samples (shown in Figure 11) have been given the greatest weight by the machine learning model. Both visualizations (text highlight and bar chart) suggests that the words *lightbulb* and *did* are majorly responsible for the classification of the entire input sentences to a joke. We can see how the weights assigned by our model to different words of the sentence goes hand-in-hand with our ablation studies on how the focus is on the presence of certain keywords that have relatively high occurrence in the joke sample, and not on the essence of the sample being a joke. Likewise, the same pattern can be observed in the non-joke sample (shown in Figure 12) where the words *scariest* and *you* are considered to be responsible for the decision made by the model, and also *scariest* is the top 4^{th} most commonly occurred word in all non-jokes.

8.4 First-Derivative Saliency

Similar to the idea used by Li et al. [10], we employed the method of First-Derivative Saliency to investigate into patterns that the model is trying to learn. Carrying our analysis from before we investigated the importance of words in a joke that contains the word *lightbulb*. As can be seen from Figure 13, most of the words that occur in the setup of the joke are important in categorizing this sample as a joke,

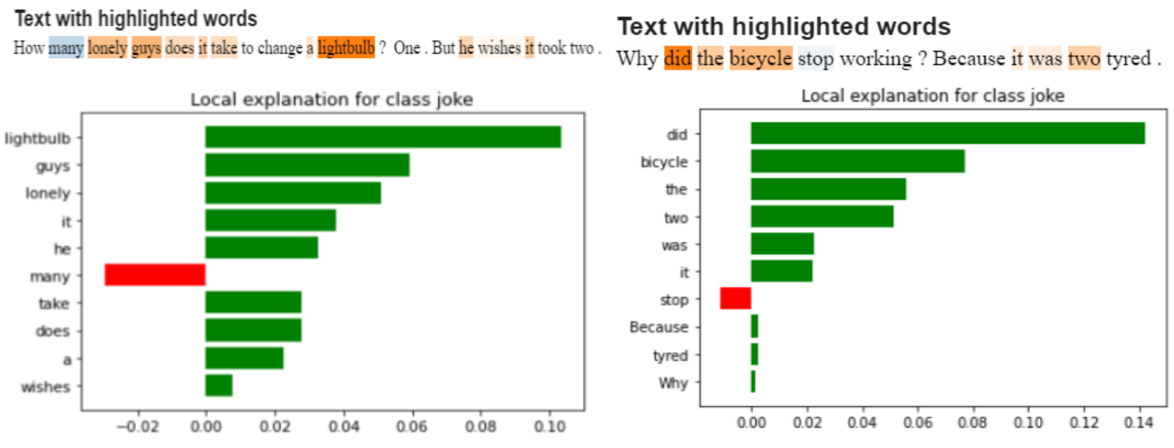


Figure 11: LIME technique on Joke samples

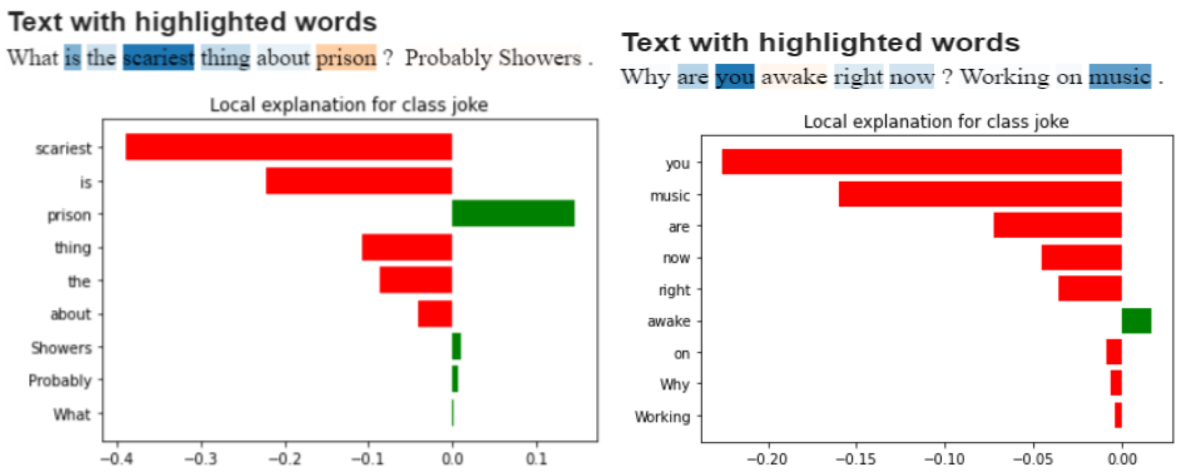


Figure 12: LIME technique on Non-Joke samples

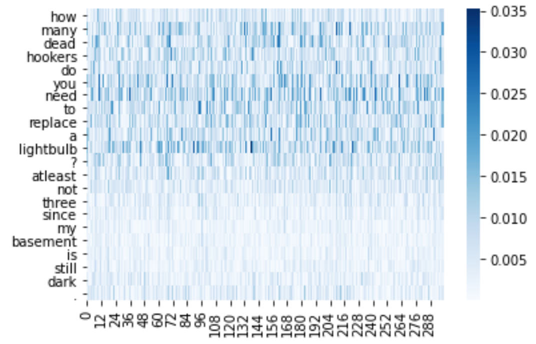


Figure 13: Importance of words in a joke as calculated based on first-order derivative. A darker color indicates higher importance of the word in final classification

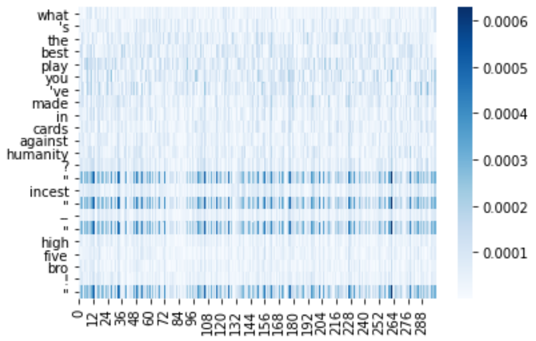


Figure 14: Punctuations like quotation marks dominating the sample classification as a non-joke

and it is not just the word *lightbulb* that dominates the classification, as was observed in the previous two analysis. Although there are other samples as shown in Figure 14 which indicate that the model is placing more emphasis on identifying quotation marks in the sample to make the classification as a non-joke, and not looking into the actual structure of a non-joke.

In addition to the take-away from Figure 13, it was also noted that for most of the non-jokes, the classification is dominated by the words in the setup. That is, for the first-order derivative saliency heatmaps for jokes, we observe a darker patch on the setup side and a lighter side on the response side. This observation is in consensus to the results of the Ablation study in Table 5 where it was seen that the setup itself is sufficient to correctly classify a joke sample.

9 DISCUSSION AND CONCLUSION

Based on our model performance analysis and interpretation studies, we saw that our Deep Learning models were more focused on finding certain peculiarities in data that not necessarily lead to an appropriate understanding of the problem in hand. We uncovered one of many such peculiarities, and for now we can sufficiently say that one of the answers to the question *"What is Humor for Deep Learning Model"* is *"Finding words with relatively more occurrence in humorous text and base your understanding of a humor on that"*. We also acknowledge that it is generally difficult to say what a model is learning, and that a Neural Network is more of like a black box to to be used with caution. There are many hidden features that a model can learn from the data, and controlling everything can be tricky. That been said, there are also techniques and model architectures that we discussed in this paper, that can help us understand this black box and maybe turn into in a grey box where decisions are more interpretable.

There are a few aspects of this project that can be investigated on in the future. First, our performance analysis revealed that the architectures using the BERT embeddings were able to generalize better on the test dataset from Data Split 2, and it would be interesting to see what patterns these networks are picking up that are helping in generalization, and whether these patterns correlate to our understanding of whether something is humorous. Second, we relied on curating our own dataset from different sources, which were not specifically crafted for the purpose of joke identification. Although a hand-crafted dataset for joke identification does not exist as of yet, it would be interesting to see how our models perform on the same. Lastly, to add on to the interpretability of Deep Learning Models, we could employ a variant of Dynamic Memory Networks [15] which applies attention and memory mechanisms by marking certain parts of the Response that are critical for a text to be a joke to exhibit certain reasoning capabilities between the Setup and Response which would make a text humorous.

REFERENCES

[1] [n.d.]. *Fmcumhaill/Reddit-QA-Corpus.* https://github.com/FMCumhaill/Reddit-QA-Corpus

[2] [n.d.]. *Hugging Face Github Repository.* https://github.com/huggingface/transformers

[3] [n.d.]. *The Stanford Question Answering Dataset.* https://rajpurkar.github.io/SQuAD-explorer

[4] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. " What is relevant in a text document?": An interpretable machine learning approach. *PloS one* 12, 8 (2017).

[5] Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers).* 113–117.

[6] Luke De Oliveira and Alfredo L Rodrigo. 2015. Humor detection in Yelp reviews. *Retrieved on December* 15 (2015), 2019.

[7] Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics* 43, 4 (2017), 761–780.

[8] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[9] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[10] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066* (2015).

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[12] Abhinav Moudgil. [n.d.]. *Short Jokes.* https://www.kaggle.com/abhinavmoudgil95/short-jokes/#shortjokes.csv

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 1135–1144.

[14] Jiri Roznovjak. [n.d.]. *Question-Answer Jokes.* https://www.kaggle.com/jiriroz/qa-jokes

[15] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning.* 2397–2406.

[16] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems.* 5754–5764.