# Knock Knock: Neural Joke Generation and Classification Final Report

**Ryan Faulkner, Richard Rex, Michael Ryan**
Department of Computer Science
Georgia Institute of Technology
{ryan58795, richardr2926, mryan47}@gatech.edu

## Abstract

The goal of our project is to generate cohesive, and humorous jokes through implementing modern developments in the field of text generation such as GPT Language Modelling and evaluating them using latest classifiers such as BERT . As an addition to our project, we have also implemented a N-gram joke generator model and a hate speech classifier which we use in order to distinguish jokes that may be offensive/insensitive to any given group of people. We also built a user-friendly web application that collects user data on how funny a given joke is in hopes of using that data to retrain a better classifier and joke generator. We evaluated the F1 of the classifiers based on an eval set of jokes and we also measured the perplexity of the generators and compared to baseline models trained on our dataset. For future steps, we want to be able to host the generator models in real time, where we can use our fine tuned GPT model and update the model in real time based on normalized user feedback. Finally as stretch goals we would like to use BERT as an adversarial classifier to make a Generative Adversarial Network joke generator. We also want to use our implemented hate speech classifiers to filter out jokes before displaying them. The code is hosted at the following link: https://github.com/Richard2926/NLP-Final

## 1 Introduction and Related Works

Imagine a world where your computer teases you for typing in your password wrong for the third time in a row. A world where your microwave jokes about you having to grab the box out of the recycling to double check the heating time on that frozen meal. This is a world where having a conversation with your AI assistant isn't a dry interaction of issuing commands and receiving acknowledgements or clarifications. Instead each conversation could be entertaining, enjoyable, and altogether more natural. With our neural joke generation model we take one step closer to this future.

There are many factors for why the study of joke generation is important. Humor adds a very human element to communication. Adding humorous quips to conversation with artificial agents will go a long way in creating socially acceptable AI. Furthermore, the study of humor has been an active area of research in linguistics for decades. Initial ideas of humor followed the incongruency theory which suggested that humor came out of unanticipated changes to an expectation (Attardo and Raskin, 1991). More recent cross-cultural studies of humor have pointed to the benign violation theory that argues humor must both violate expected norms but simultaneously not violate these expectations in a way that is too extreme or inappropriate given the context (McGraw and Warner, 2014). Building models to generate jokes can contribute further to the study of humor by showing what patterns computational models can extract from humor.

Of course, there are several past works that are relevant to this work. Past work in computational humor fits into two main areas: humor recognition and humor generation (Amin and Burghardt, 2020).

A lot of research has been done in humor recognition. (Mihalcea and Strapparava, 2005) built SVM and Naive Bayes classifiers to recognize one-line jokes. (Davidov et al., 2010) identified sarcasm by labelling some sarcastic tweets and using k-nearest neighbors to classify new inputs. (Kiddon and Brun, 2011) detected phrases compatible with the addition of "That's what she said" by using an SVM and the probability of certain words appearing in an erotic context. (Yang et al., 2015) sought to recognize humor and the humor anchor in a phrase using a method called Maximal Decrement. Recently, one-liner joke recognition has been returned to using more modern NLP methods.

ColBERT (Annamoradnejad and Zoghi, 2021), a fine-tuned BERT model trained on 200k short texts (100k jokes, 100k news headlines), boasts a 98.2% F1 Score.

Humor generation has also been an active area of research in computational humor. Originally much of computational humor generation was in the form of template completion models such as LIBJOG (Raskin and Attardo, 1994). LIBJOB filled in the blanks of the joke "How many ____ does it take to screw in a lightbulb? ____. One to ____ and ____ to ____." (Petrović and Matthews, 2013) filled in "I like my ____ like I like my ____. ____." using wordnet and Google n-grams to find word pairs that fit the expected relationships in vector space. (Sjöbergh and Araki, 2008) built a standup comedy routine generator by filling in "Speaking of ¡noun¿. ¡Joke¿" and pulling related jokes from a database. These models are capable of generating jokes conforming to a specific template, but we wanted to generate completely free-form one liner jokes. Related work exists in this area as well. (Yu et al., 2018) collected a dataset of puns and trained an LSTM to generate more. The model achieved a perplexity score of 889.07 on the pun dataset. (Ren and Yang, 2017) designed an attention based LSTM RNN network for current event joke generation. This model could be prompted with a topic and it would generate jokes based on that topic. These models scored high in creativity for joke generation, but many of the jokes did not make sense which made them score low in humor (Amin and Burghardt, 2020). In this work we use new developments in text-generation such as GPT-2 to improve the semantic meaning and humor of generated jokes.

## 2 Methods

### 2.1 Data

We are using a dataset of 200k phrases from the ColBERT joke classification paper (Annamoradnejad and Zoghi, 2021). The phrases are comprised of 100k short humorous texts scraped from reddit and 100k serious news headlines from the Huffington post. The jokes were scraped mostly from the r/jokes and r/cleanjokes subreddits. Table 1 has some summary statistic for the dataset. Table 2 shows the top 5 most common tokens across the dataset.

Some randomly sampled jokes from the dataset are: "I figured out how to talk to girls just walk up to them and press A", "What is the network admin

| Statistic | Jokes | Headlines | Overall |
|---|---|---|---|
| total samples | 100k | 100k | 200k |
| vocab size | 52350 | 61653 | 91033 |
| token count | 1199765 | 1556274 | 2756039 |
| avg wd/phr | 15.56274 | 11.99765 | 13.7802 |
| avg chr/phr | 69.99166 | 64.94949 | 67.4706 |

Table 1: Summary statistics on our dataset. (wd/phr = words/phrase)

| Class | Top 5 Tokens |
|---|---|
| Overall | like,call,trump,get,new |
| Jokes | call,like,get,say,one |
| Headlines | trump,photos,new,video,says |

Table 2: The top 5 most common tokens in the dataset (excluding stop words and punctuation)

favourite lullaby? mary had a little lan", "What did the psychic velociraptor say to his friend? dino what you're thinking".

Some randomly sampled offensive jokes from the dataset are: "Life is like a bicycle, a black will probably take it", "What do you call a gay piece of bread? a fagguette", "I like my coffee like i like my women... in the kitchen.".

Some randomly sampled headlines from the dataset are: "Interest rates near zero prevent savings accounts from growing", "Is Hillary Clinton the last democratic presidential candidate to support the death penalty?", "How to avoid disaster during this week's severe cold weather".

### 2.2 Models

The models that we decided to use are GPT-2 and BERT because they are the two leading language models presently. They are the same in that they are both based on the transformer architecture, but they are fundamentally different in that BERT has just the encoder blocks from the transformer, whilst GPT-2 has just the decoder blocks from the transformer.

GPT-2 consists of solely stacked decoder blocks from the transformer architecture. In the standard transformer architecture, the decoder is fed a word embedding concatenated with a context vector, both generated by the encoder. Furthermore, in the standard transformer architecture self-attention is applied to the entire surrounding context. We trained the GPT-2 in the standard transformer way, with a batch size of 1 with a well-defined sentence length of 150, and a $top_p$ value of 0.92 which

controls how random/creative the text generation should be. At evaluation time, we switch the model to expecting input one word at a time. We did this by temporarily saving the necessary past context vectors as object properties. Since our criteria was to test the ways that context affected the joke, GPT-2 was ideal because it works like a traditional language model in that it takes word vectors and input and produces estimates for the probability of the next word as outputs. It is auto-regressive in nature: each token in the sentence has the context of the previous words.

The next model we used was BERT, like GPT-2, which uses the transformer architecture. However, it uses the encoder part instead of the decoder part. We used BERT with the purpose of storing knowledge learned from the training data. It was an apt choice because it utilizes a multi-layer bidirectional transformer encoder consisting of several encoders stacked together, which can learn deep bi-directional representations. We feed BERT sentence embedding the whole text into hidden layers of the neural network and training it with hyperparameters as follows: 5 epochs, max sequence length of 128 and a Binary Cross entropy loss function in order to calculate loss. Further results are in the results section of the paper.

### 2.3 Baseline Models

For our baseline models, we used the RNN and a N-gram Model. The RNN is a bidirectional LSTM that has one layer and 16 embedding dimensions with 128 hidden dimensions. We output the final layer to a linear layer then use a sigmoid activation function in order to classify the joke as humorous or not. We use binary cross entropy loss as our loss function and adam as our optimizer. The state of the art classifier for joke is around 98 percent so our resulting 97 percent shown good progress towards classification.

The N Gram model is constructed by generating the count for all the words and then predicting and generating based on that. Our N Gram model also used interpolation and smoothing to account for increased accuracy where n = 15 in our case. Some of the jokes generated by the N gram model are: “Now I have to court, the Eversweet truck and breakfast is waiting to blow up.”, “He had a bad day and night he would know.”. As we can see the jokes are not very cohesive. We have also implemented smoothing for this method. Some of the jokes generated by our GPT model are “Why can’t you hear a pterodactyl go to the bath- room? because it’s extinct”, “Why did the blonde smoke weed? so she wouldn’t get high.” Our GPT model generated a perplexity of 9.5158 compared to the 3000 of the N-gram generator which clears our success criteria and our model also retains the structure of a joke. We believe the GPT performed much better because they learn relationships between words and utilize positional embeddings.

## 3 Results

### 3.1 Experimental Setup

For our RNN classifier, we used the data set of about 2000 tweets classified as hate speech or not to train with a split of .8/.1/.1 for train val and test sets. We shuffled the data and split randomly to minimize bias. We ran this same data with naive bayes and logistic regression classification techniques too, but eventually went with RNN as we received the better results from the RNN model.

For our n-gram model, we used the compilation of over 200,000 jokes from reddit/stupidstuff/waka. For joke generation, we altered the context, or ”n” considered for word generation. We tried n-gram generation by character as well, with varying ”n” values, but results were often more incoherent than with word generation.

For our BERT classifier we tried to match the procedure in the ColBERT paper (Annamoradnejad and Zoghi, 2021) and replicate the results. The train/test split was 80/20 and we used the same split from the original paper. This split was random so the training set ended up with 79,918 jokes and 80,082 news headlines. The test set ended up with 20,082 jokes and 19,918 news headlines. The bert-base-uncased model was used as the model architecture and for the pretrained weights. For our optimizer we opted to use Adam and a learning rate of $\alpha = 0.00002$.

For our GPT-2 generator we fine-tuned a gpt2-medium model on a train/test split of 0.9/0.1. The news headlines were not included in the gpt2 model training at all. This meant the training set had 90000 jokes and the evaluation set had 10000 headlines. Each joke had a $< |joke| >$ token added to the beginning and a $< |endoftext| >$ token added to the end. This was the format that the GPT-2 tokenizer was expecting these tokens in. We limited the max sequence length to 150 and implemented a repetition_penalty of 1.5 for n-grams of size 2 to

| *Joke classifier models* | Results A | P | R | F1 |
|---|---|---|---|---|
| Our RNN | 97.0 | 96.8 | 97.2 | 97.0 |
| Our BERT | **98.46** | 98.46 | **98.46** | **98.46** |
| Decision Tree† | 78.6 | 76.9 | 82.1 | 79.4 |
| SVM† | 87.2 | 86.9 | 88.0 | 87.4 |
| Multinomial NB† | 87.6 | 86.3 | 90.2 | 88.2 |
| XGBoost† | 72.0 | 75.3 | 77.7 | 81.3 |
| XLNet† | 91.6 | 87.2 | 97.3 | 92.0 |
| ColBERT† | 98.2 | **99.0** | 97.4 | 98.2 |

Table 3: Accuracy, Precision, Recall, and F1 of our different Classifiers. † ColBERT results are reported as written in the paper (Annamoradnejad and Zoghi, 2021)

prevent the model from saying the same word over and over.

### 3.2 Result Comparison

#### 3.2.1 Joke Classifier Comparison

We worked on two Joke Classifier models. An RNN model and a fine-tuned BERT model. We used the ColBERT dataset for our joke classification models so we could compare results to other works (see Table 3). Our BERT model outperformed all other classifiers with an F1 score of 98.46, even outperforming the ColBERT model. Interestingly our RNN classifier also outperformed most of the baseline models tested in the ColBERT paper. We ran BERT two times to test if the outperforming of ColBERT was replicable. The first time our BERT classifier got an F1-score of 98.56 (on a shuffled train/test set). The second time our BERT model scored 98.46 (on the exact same train/test set as ColBERT. We did not run our model more than these two times, because our BERT model took about 4 hours to train for 5 epochs using Google Colab GPUs. This training time was a prohibitive factor, but also we wanted to be conscious of recent research into the environmental impacts of large language models such as BERT and GPT (Bender et al., 2021). We did not see training our BERT model several times to be a productive use of time/power to see if our increase of just 0.26 in F1 score from ColBERT was significant. We still did perform significance testing on these two samples. Our T-test yielded a p-value of 0.101803 showing this difference was **not** significant at the $\alpha = 0.1$ level. Thus we cannot reject the null hypothesis and we must conclude that the F1 scores are not significantly different.

One reason that our BERT model might've performed better than ColBERT (although not significantly better) comes from our training strategy. Although both our BERT model and ColBERT trained

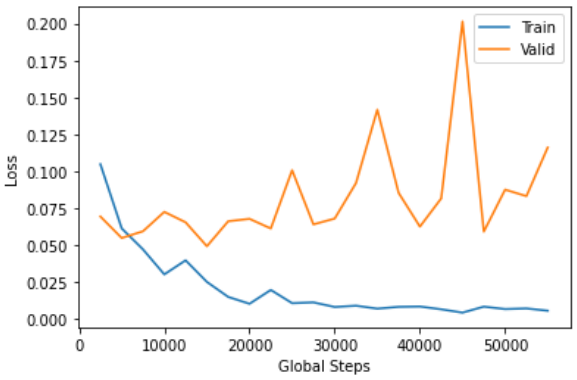


Figure 1: Training and Validation loss for BERT classification over training steps

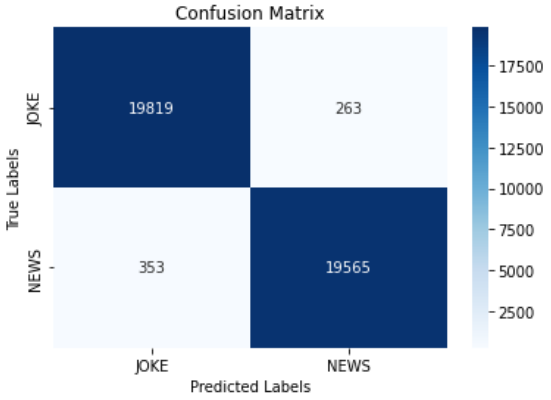


Figure 2: Confusion matrix for BERT joke/news headline classifications

for 5 epochs, our BERT model started overfitting after just 2 epochs (see Figure 1). We only saved our model when it reached a minimum validation loss. This meant that even though we trained our model for 5 epochs like ColBERT we ended up running our evaluation on the 2 epoch model. It is possible ColBERT overfit just slightly more than our model.

Although an F1 score of 98.46 is quite high our BERT model still made several classification errors (see Figure 2). Upon closer inspection these classification errors came primarily from edge cases. Sometimes news authors get creative and make their headlines into jokes. For example: "folks try a drunk driving simulator and it's a total buzz kill", "what's pink, red and gucci all over? beyonce, for some reason.", and "i see london, i see france... but i don't want to see beyonce in her underpants". Sometimes the jokes are objectively bad: "how can you be so sad when you are so beautiful?" and "if you say unique new york ( 5 times fast ) is it easy or hard?". Sometimes jokes are in the

form of headlines: ”are we as a society going to reject clickbait journalism? the answer may surprise you!” and ”very offensive man on the loose with flint and steel sparks outrage”. Overall these cases are difficult even as a person to determine if they are humor or headlines (sometimes they are both). As such we are very satisfied with the performance of our classifier.

#### 3.2.2 Generator Model Comparison

N-gram generator Results: Scored a perplexity of 3000, which was very unsatisfactory for the goals of this project. We knew we would have to turn to a different generation model if we wanted to improve. The jokes created were qualitatively okay at times, but often more incoherent stories were generated of great lengths. This was our baseline model for development on this project. Jokes showed some semblance of humor like ”What a bunch of fellow rednecks riding in the barn. That’s what He said”. Other generations were just long stories instead however: ”The man then walks over to the donkey and whispers into it’s ear, the donkey then starts laughing, so the man got free drinks for the rest of the evening did not improve his mood and eventually the construction crew, all of them ”gems-in-the-rough,” more or less, adopted her as a kind of project mascot. They chatted with her, let her sit with them while they had tea and lunch breaks, and gave her little jobs to do here and there to make her feel important.” We looked to remedy this in the development of our GPT model. Our n-gram model is very free of user input or attention so it is possible that because of this hands-off design that we are unable to get satisfactory results, and instead get long stories and a large perplexity value.

To address some of the limitations of our n-gram model we fine-tuned a GPT-2 model on 90,000 jokes. This did significantly improve the perplexity of the models scoring 9.5158. The higher cohesion of the jokes can also be seen from the randomly sampled generated jokes of each model (see Table 4). GPT-2 definitely learned certain characteristics of jokes such as setting up for a punchline with a question. There is still room for improvement with the actual humor of the joke, however. Jokes like ”How do you get a blonde to fart? pick her nose.” have all the elements of a joke without actually being funny (besides maybe appealing to childish humor of farts and picking your nose). Another issue we discovered in our GPT-2 model was direct replication of some of the jokes from the training set. For example, the joke ”How do you find will smith in a snowstorm? look for fresh prints!” and ”What’s the difference between snowmen and women? snowballs” seem really good at first glance, but then you find out these jokes are almost exactly copied from the training set. Upon even further investigation we realized this may partially be a problem with the ColBERT dataset itself. Because this dataset was scraped from reddit and not manually collected there are several repeats of jokes. For instance our model generated this joke: (**warning vulgar**) ”What’s the difference between jam and jelly? i can’t jell my c**k down a girls throat”. (censorship added) This joke shows up more than 50 times in the training set. Besides this being a potentially concerning reflection on society that this joke has been copied and reposted over 50 times to reddit, this also demonstrates a dataset shortcoming because we are not working with 100,000 unique jokes. Future work should consider applying further filtering to the ColBERT dataset to reduce replicated jokes.

Another problem we wanted to address with our generated jokes was how offensive some of the jokes being generated were. For example our model generated the following jokes (**warning offensive**) ”Have you ever tried ethiopian food? neither have they.” and ”Why do blondes have big noses? because air is free!”. This was a problem we believed we could address. In order to help mitigate this issue we trained an RNN offensiveness classifier.

#### 3.2.3 Offensiveness Classifier

RNN classifier for Offensive Jokes: Earned a accuracy of .83, an F1-score of .59, a recall of .92, and a precision of .89. This shows that our RNN classifier is an adequate model for determining hate speech among our jokes. After hand labeling some jokes of our own and testing, we received an accuracy of 97 percent for classifying offensive jokes, but we would need to study and request outside annotators to affirm this success. One way we look to implement this is by storing the data our front end currently receives to determine whether jokes are funny or offensive.

#### 3.2.4 Proposed Pipeline

With 3 key models in our project we wanted to propose a pipeline to connect these components (see Figure 3). We propose a pipeline where the user inputs a few tokens as the start of the joke.

| | |
|---|---|
| N-gram | Wee Jock's arm shoots straight up and said 'U.T.I?<br>Knock knock. Who's there? - Juan? - Juan who? - I spoke about the results.<br>When Johnny got home, all the veggies on now and then skip a beat my parents were... |
| GPT-2 | Why didn't the chicken cross a road? because it wasn.<br>My dog just showed up at the vets. i guess he's in a purrfective state of shock<br>How do you get a blonde to fart? pick her nose. |
| Base | Q: what do outlaws eat with their milk? a: crookies.<br>How many people can ride on a bird? toucan.<br>What lies upside down a hundred feet in the air? a dead centipede. |

Table 4: Randomly sampled generated joke examples from N-gram model, GPT-2 model, and original dataset

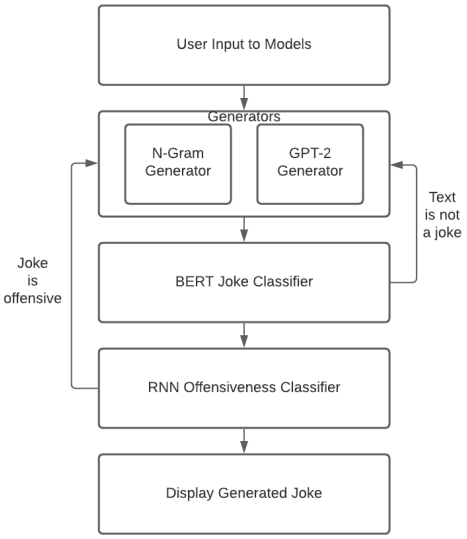


Figure 3: Proposed pipeline to combine our models for joke generation

Then based on the model they have selected, either our N-gram model or GPT-2 generates a joke. This joke is classified as either a joke or not a joke by our BERT classifier. If it is considered a joke then we can check if the joke is offensive using our RNN offensiveness classifier. If the joke is not offensive then it can be shown to the user, otherwise if the text is not a joke or if it is offensive we will request a new output from our generator.

We would like to integrate this pipeline into our webdemo (see Figure 4) so that we can ensure higher quality and less offensive jokes.

### 3.3 Work Division

Ryan - Developed the RNN model for offensive classification, and the N-gram model for joke generation. Performed analysis on his according models and did the according write-ups and presentation on them. Provided samples of n-gram generated jokes for our presentation demo.

Michael - Worked on the BERT model for joke classification, and the GPT-2 model for joke generation. Ran analysis on the multiple model improvements he developed and produced the according write-ups for his developments in the papers and presentations. Gathered and presented statistics on the original datasets used as well so that we could better understand our data. He provided the GPT-2 samples that were utilized during the demo for our presentation.

Richard - Worked on the RNN model for joke classification, and the application that integrates user feedback. Analyzed the RNN he developed and did the write-ups and presentation for his work. He developed the standalone app that was used during the presentation for our demo.

## 4 Conclusion

The implementation of attention in our models has shown, for us, to qualitatively improve the performance of humor generation. By focusing our training on the more relevant information in our data set, we can generate more cohesive shorter length jokes. This makes sense, as creating humor relies on paying attention to the most important parts or "punchlines" of the jokes, which isn't well addressed by an n-gram model. We were happy to see great improvements with attention implementations in GPT-2.

For our results, our GPT-2 perplexity was standard among the papers we looked at models for. Our BERT classifier also performed up to standards, with an F1-score and accuracy similar to the described models in the paper (Annamoradnejad and Zoghi, 2021). This is likely because we used the same data set for our BERT classifier due to its success in the field. Our GPT-2 model per-


Figure 4: The webdemo we have deployed to showcase our joke generation models and collect user feedback

formed well with the custom data set we used compared to the papers, and created jokes that could be considered humorous. We were happy with the improvements over the n-gram solution that the GPT-2 solution provided us with, and would encourage future work to also look into GPT-2 as an option for humor generation.

To continue our work, we would really like to feed back user input into our model to tune it to produce more humorous/less offensive jokes. We have not tied our offensive classification model together with the generation model yet, so we would need to look at implementing that, although we have produced both models as singular entities. We are also storing the Likert Scale responses from users using our application, however we need to also tie these back into our GPT model's weights to more finely tune the humor. Further, we would need to look up at clearing up repeat jokes, which could maybe be handled by removing duplicates from the data-set. We currently produce multiples of the same jokes or jokes that already exist in the data set.

There are also interesting areas of research that could pick up from limitations in our model. For one, our model does not take any consideration for the context that the joke is being generated for. This would be a necessary consideration for any chatbot or interactive agent attempting neural joke generation. Another interesting improvement could be tieing our BERT classifier to our GPT-2 generator in an adversarial neural network structure. Generative Adversarial Neural Networks have shown impressive performance in improving generation results so this would be an interesting improvement using some of the models we have already created. The loss function of the classifier and generator would just need to be modified in an adversarial fashion. We are excited to see where the field of computational humor goes next.

## 5 Ethics

We wanted our project to be a lighthearted source of entertainment and a proof of concept in the domain of neural joke generation. Jokes are meant to be fun and make people laugh. That being said, as humor lies on the border of what is a violation of expectations without actually stepping over the line according to the benign violation theory (McGraw and Warner, 2014), offensive jokes become a quick way to get some people to laugh at the expense of others. This was in fact a problem we saw emerge in our model. To address this we built an offensiveness classifier using an RNN. We use this model to try to catch offensive jokes before displaying them to the end user. This way people can choose whether or not they want to see an offensive joke. With further training we want to encourage our model to move away from offensive humor.

We hope that our model can be used as a launch pad for future humor generation models and possible integration into humor chatbots. With this in mind we don't want to deploy a system that is going to go around offending people. Humor is highly dependent on the context of the joke and this would be an interesting area of future research - context based joke generation. This will need to be explored further because it may be fun for your AI assistant to joke around with you when you are planning a fun vacation, but inappropriate when

you are booking travel plans to a funeral. Understanding context is a huge part of humor.

## References


Miriam Amin and Manuel Burghardt. 2020. A survey on approaches to computational humor generation. In *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 29–41, Online. International Committee on Computational Linguistics.

Issa Annamoradnejad and Gohar Zoghi. 2021. Colbert: Using bert sentence embedding for humor detection.

Salvatore Attardo and Victor Raskin. 1991. Script theory revis(it)ed: joke similarity and joke representation model. 4(3-4):293–348.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, page 107–116, USA. Association for Computational Linguistics.

Chloé Kiddon and Yuriy Brun. 2011. That's what she said: Double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 89–94, Portland, Oregon, USA. Association for Computational Linguistics.

Peter McGraw and Joel Warner. 2014. *The humor code: A global search for what makes things funny*. Simon and Schuster.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria. Association for Computational Linguistics.

Jonathan D. Raskin and Salvatore Attardo. 1994. Non-literalness and non-bona-fîde in language: An approach to formal and computational treatments of humor. *Pragmatics and Cognition*, 2(1):31–69.

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

Jonas Sjöbergh and Kenji Araki. 2008. A complete and modestly funny system for generating and performing Japanese stand-up comedy. In *Coling 2008: Companion volume: Posters*, pages 111–114, Manchester, UK. Coling 2008 Organizing Committee.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

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, Melbourne, Australia. Association for Computational Linguistics.