Getting a Sense of Humor: Joke Punchline Generation with User-Specified Levels of Humor

Alec Korotney
akorot@umich.edu

Kevin Wang
musicer@umich.edu

Abstract

Machine joke generation is a challenging task that requires a model to produce output with not only similar meaning and form, but also a similar "sense of humor" as a human-generated joke. Even more difficult is generating jokes with a specified level of humor; some jokes are funnier than others, and generating jokes with different levels of humor requires a model to more thoroughly understand what a "sense of humor" really is. We devise two approaches for generating jokes with variable levels of humor: a multiple-model method that simply trains separate models for a discrete number of desired humor levels, and a single model that takes in a continuous-valued humor level as an additional input at inference time. We achieve BLEU scores above 0.1 for both approaches, and as far as we know, our work represents the first attempt to generate jokes with specific desired humor levels. Our code is available at https://github.com/ABKor752/Joke-Generator.

1 Introduction

Generating realistic jokes is a difficult task for a machine, requiring a model to understand not only human-perceived humor but also what assemblage of words constitutes a "proper" joke. As an NLP task, training a model to generate a coherent and funny joke, given some sort of prompt, requires a focus on word sequencing, grammatical structure, and in the hardest cases, humorous neologism such as "de-calf-enated" (decaffeinated) or "flew-id" (fluid) as part of a pun.

Previous work on incorporating NLP with jokes can be divided into two categories: joke generation and joke classification. The former faces the issues noted above, while the latter additionally faces the same challenges as traditional sequence-to-label tasks. Combining these tasks yields a text generation problem with class label as an additional input.

Our work explores whether a machine can not only generate meaningful **punchlines** from joke **bodies**, but also generate a punchline given a **specified level of humor**. In other words, given a joke body and some level of humor (e.g. funny, unfunny, mediocre), our model should generate a punchline that has the appropriate level of humor. To complete this task, we utilize transformers, a powerful model architecture with the ability to properly navigate the complex grammatical and semantic structure of a joke body and yield a meaningful punchline.

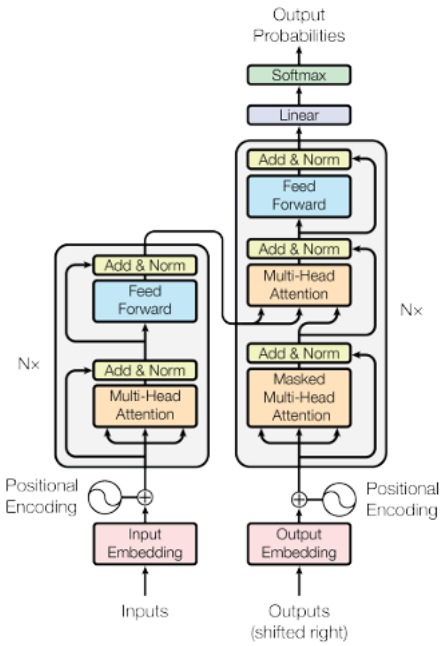

Figure 1: The transformer architecture from (Vaswani et al., 2017). We utilize transformer models in conjunction with data preprocessing to generate joke punchlines with a desired humor level.

In addition, this model must take humor level as an input. Depending on the approach taken, we will first filter the input to a single model in a multiple-model approach or directly add a scalar humor level value as an input. The methodology is elaborated on in Problem Approaches.

2 Related work

Weller and Seppi (2019) was the first paper to apply transformers to humor detection. Hasan et al. (2021) built upon this to explore transformer-based humor detection in multimodal settings, and Ziser et al. (2020) explores transformer-based humor detection specifically in product question answering systems. Prior to this work, non-transformer methods were used, including N-grams in Taylor and Mazlack (2004) and tree-based methods in Purandare and Litman (2006). Transformer methods outperform non-transformer methods consistently across multiple established datasets.

There is relatively little work in generating joke punchlines because it is difficult for machine learning models to conceptualize humor and generate humorous text. One of the earliest papers to attempt this is Petrović and Matthews (2013), which uses unsupervised methods to generate jokes fitting a specific structure. The methods in this paper are not strong enough to handle joke generation that isn't attached to a specific structure - hence our reliance of transformers. A number of papers build upon Weller and Seppi (2019) and use transformers to generate jokes. Weller et al. (2020) attempts to "translate" normal text into humorous text, and Zhang et al. (2020) attempts to produce punchlines given a joke body (referred to in the work as a joke context). We differ from this latter work by also adding a humor parameter, allowing the user to decide how funny they want the outputted joke to be. As far as we know, we are the first group to attempt to use a pretrained humor detection model to augment existing joke data and train a transformer to generate jokes with various levels of humor.

3 Data

We use the preprocessed Reddit joke dataset from (Weller and Seppi, 2019). Each joke is divided into Reddit post, Reddit body, and upvote count. A portion of our project requires dividing the dataset into funny and unfunny jokes, which the original paper does by labeling all posts with more than 200 upvotes as "funny" (label 1) and all other posts as "unfunny" (label 0).

Because the majority of a joke may be contained in the Reddit post body, we split the data in the paper into joke bodies and punchlines in the following way: the final sentence (or grammatically correct segment of the Reddit body) is the punchline, while the rest, title and body included, make up the joke body. Our justification for this is that in general, the punchline is the final part of a joke and so is most often contained in the final sentence.

We further preprocess the Reddit data as follows:

- Any post whose content has been deleted by the user or removed from Reddit will have "[removed]" or "[deleted]" as a Reddit body. These posts are removed from the dataset.
- Oftentimes, Redditors will add addendums to their post to thank people for giving them awards, or to acknowledge comments. These typically begin with some form of "Edit:". Any posts containing such a string is truncated to not include "Edit:" or anything after it.
- As a public social media platform, typos are to be expected. Since it is hard to resolve all typos, the ones we focus on primarily are accidental concatenation of two words (e.g. "Thefather", "heSaid"). For simplicity, we focused on splitting words which have an extraneous capital letter in the middle. Theoretically, these are more common since these word concatenations can also be caused by inaccurate newline processing from (Weller and Seppi, 2019). An extension of this project could be to further improve how typos and errors are dealt with in the raw data.
- Extraneous punctuation was removed from the data, including quotes and HTML coded characters (e.g. ,).

The resulting processed data is in the format shown in Table 1.

Label	Body	Punchline
1	Why did Stalin only write in lowercase?	He was afraid of capitalism!
0	The of Berkshire Hathaway should open diner.	And call it Warren's Buffet.

Table 1: Examples from Reddit data after preprocessing.

4 Problem Approaches

We approach this problem in two ways, using either a discrete or a continuous humor level input.

4.1 Discrete Humor Level

We begin by training a model which can generate jokes that are either *funny* or *unfunny*. To do this, we design a base architecture sequence-to-sequence transformer model that maps joke bodies

to punchlines, and we train this architecture on all jokes with label 1, followed by all jokes with label 0, resulting in two model instances (one for funny joke generation, another for unfunny joke generation).

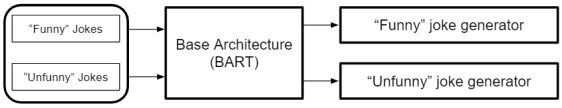

Figure 2: Model setup for the discrete-valued humor level approach.

The base architecture was implemented with the following attributes:

- HuggingFace `facebook/bart-base` model structure
- `BartTokenizer` tokenizer
- `BartForConditionalGeneration` pre-trained model
- `DataCollatorForSeq2Seq` data collator
- `Seq2SeqTrainer` trainer (with `Seq2SeqTrainerArguments`)

We tokenize the data with truncation and maximum-length padding. We trained with two different data splits: one where there was no split at all and the same data was in both the training and validation sets, as well as one where training and validation sets were created using a 92%:8% split. The differences in results for these preprocessing methods are discussed in Discrete Approach.

Each model was trained and evaluated on each set of data (funny and unfunny) separately. At inference time, jokes with a desired humor level of 0 are run through the unfunny model, and jokes with a desired humor level of 1 are run through the funny model.

4.2 Continuous Humor Level

Unlike the discrete model, we use only a single model instance in the continuous humor level case. If there are infinitely many possible humor level values, as is the case when we draw humor level values from the range $[0, 1]$, it becomes impossible to train a separate instance for each desired humor level as in the discrete case. As such, we instead set up the problem so that the humor level becomes another input into the model alongside the joke body. Inputting the same joke body with different humor levels should produce different punchlines with different levels of humor.

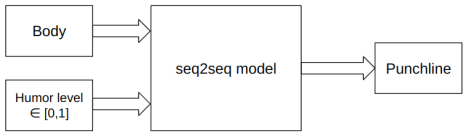

Figure 3: Model setup for the continuous-valued humor level approach.

We use an identical base architecture for the continous case as in the discrete case. The main difference between the two is the preprocessing step: in the continuous case, the output of the preprocessing step from the discrete case is further augmented to generate more data with additional information. For each pair of body and punchline, we follow the following procedure:

1. The punchline is duplicated N times, where N is a hyperparameter. For our experiments, we set $N = 5$.
2. Each duplicated punchline is paraphrased by a pretrained parapharsing model. We use IBM's Quality Controlled Paraphrase Generation model from (Bandel et al., 2022), which we use to vary the lexical, semantic, and syntactic structure for each paraphrase.
3. The body is concatenated with each punchline to produce N jokes with the same body but different punchlines.
4. Each joke is assigned a humor level based on the output of a pretrained humor detection model. Due to difficulties reproducing results from the original GitHub repository, we reimplemented the model from Weller and Seppi (2019) using the HuggingFace API. This is included as part of our GitHub repository under the `humor_detector` directory.

The tokenizing function for the continuous case also differs from the discrete case. Instead of simply returning the tokenized joke body and punchline, we insert the humor level before every word in the joke body before tokenization, as shown in Figure 5. Since the same joke body with a different humor level should produce a different punchline, we need to somehow incorporate the humor level

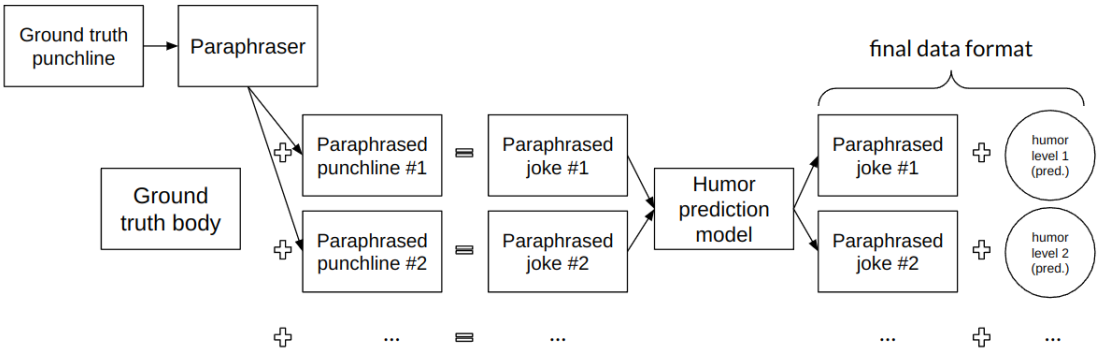

Figure 4: Data preprocessing and augmentation for continuous humor level case. Each joke body is duplicated and paired with a paraphrase of the original punchline, and an overall humor level is assigned to the resultant joke.

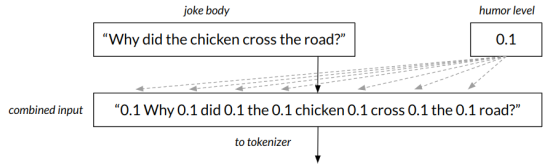

Figure 5: Tokenization step for continuous humor level case. The humor level is inserted before every word in the joke body, and the result is tokenized.

into the model input. We insert the humor level before each word to avoid problems with long-range dependencies; this is an easy way to maintain a seq2seq model architecture.

This approach is borrowed from (Bandel et al., 2022), which inserts labeled several floating point values before each token; since we only have one floating point value that always refers to humor level, we do not need to label it in the tokenization for fine-tuning.

Notably, because setting $N = 5$ effectively quintuples both the training and test datasets, training time increases considerably at this stage. We required multiple GPUs and several hours to finish finetuning this model, meaning we had less opportunity to conduct a full hyperparameter search.

5 Evaluation

We use BLEU score to evaluate the efficacy of our models. In particular, we use the BLEU metric provided by (Papineni et al., 2002).

5.1 BLEU Score definition

The BLEU score is a measure of the similarity between two strings of text, typically used for giving a quantitative rating of the quality of a machine translation task. Given a predicted translation and a set of *references* that the translation could match to, the BLEU score is a measure of the similarity between the prediction and the references. (For this project, since each joke has only one correct punchline, there is exactly one reference for each joke in the testing set.)

When analyzing n-grams up to some N in the prediction and references, assigning a precision score p_n and a weight w_n to each n, The BLEU score is computed as

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where the positive weights w_n add up to 1, and BP is the *brevity penalty* defined as follows, where c is the translation length and r is the reference length:

$$\text{BP} = \begin{cases} 1, & c > r \\ e^{(1-r)/c}, & c \leq r \end{cases}$$

The BLEU score is computed as a real number value between 0 and 1. The closer it is to 1, the closer the prediction is to matching the reference. A score of 1.0 means a perfect translation, which is hard for even a human to achieve for any generic translation task.

5.2 Appropriacy of BLEU score

BLEU score was the simplest metric appropriate for our task, and it worked out of the box. However, BLEU score does not capture many of the nuances of humor. For example, since BLEU is based on exact matches in n-grams and not on meaning, generated neologisms like "impasta" (a play on "imposter") will always be penalized if they do not match a reference, even if the neologism is appropriately humorous. Additionally, BLEU cannot capture the function of words in a sentence;

missing an article or an adverb or other non-critical part of speech carries the same penalty as missing the subject or main verb. Consequently, many generated jokes have poor grammatical structure and do not make sense. A metric that better captures grammatical structure could have produced stronger results.

Other NLP papers surrounding the subject of humor have used different metrics to evaluate a model's understanding of humor. The most foolproof but resource-expensive method is human evaluation, e.g. A/B testing as done in (Weller et al., 2020). Although effective, human evaluation would have taken more time and resource investment than what would have been reasonable for our group, so we decided against it. An older paper, (Petrović and Matthews, 2013), uses an approximation of log likelihood called rank of likelihood (ROFL). This metric works because this paper works in a smaller, more restrained space where it is possible to rank the likelihood of every single possible joke; with more powerful models like ours and a less confined problem space, an approach like this that relies on enumerating every possible joke is simply not possible. As such, BLEU was the best available metric for us. The ideal scenario would be human evaluation a la (Weller et al., 2020), but this was unfortunately not doable for our team in this timeframe.

6 Results

We analyze the results of our two approaches separately.

6.1 Discrete Approach

Evaluation began primarily on the unfunny model which, by assumption, would be the harder one to train due to the slightly smaller amount of data and the unexpected punchlines (due to lack of humor). Once satisfied with the results from the unfunny dataset, we tested both the funny and unfunny models in tandem.

Using the entire training dataset as the validation set as well, we produced BLEU scores for corresponding hyperparameter configurations, a sample of which is shown below for *unfunny* jokes.

(epochs, learning rate, weight decay)	BLEU
(5, $2e^{-5}$, 0.01)	« 0.06
(5, $6e^{-5}$, 0.01)	0.0753783
(15, $6e^{-5}$, 0.01)	0.0963968
(15, $6e^{-5}$, 0.02)	0.1089136

Table 2: Sample BLEU scores for unfunny joke model

Running on a virtual 64GB GPU with 4 CPUs takes roughly 8 hours to run for 15 epochs.

Using the 92%:8% train/validation split with the best set of hyperparameters found above, the BLEU score ends up being 0.0879357, a worse performance. Keeping the hyperparameters and using the training set as validation, the BLEU score for *funny* jokes is 0.1698145. Despite not having a baseline for our evaluation due to the original nature of the project, we can safely say that the jokes are not simply gibberish and do indeed follow the desired punchline to an extent.

Some example joke outputs are shown below:

Body	Punchline (actual)	Punchline (generated)
What did Sparticus do to the cannibal who ate his nagging wife?	Nothing, he's gladiator	Nothing, he's Gladiatorutt
Imagine if Americans switched from pounds to kilograms overnight...	There would be mass confusion	There would be mass confusion
Do you know how to avoid clickbait?	Obviously not	It avoid on

Table 3: Example jokes from funny model (with best hyperparameters)

Immediately, we notice a similar structure between the actual and generated jokes. Some jokes correctly predict the punchline, and others get close. However, the shorter punchlines seem to be harder to predict and even lead to grammatical inaccuracy. Though the wide range of punchline accuracy is worth noting, the similar structure is still quite impressive. The same situation arises with the unfunny joke model.

Body	Punchline (actual)	Punchline (generated)
Please, please don't start growing marijuana on your cattle farm	The steaks are too high	The steaks are high high
Ivermectin makes your blood lethal to mosquitoes	Suck it, mosquitoes!	Butits it, it!

Table 4: Example jokes from unfunny model (with best hyperparameters)

6.2 Continuous Approach

We trained the model with continuous-valued humor level on three different hyperparameter settings, with the resultant BLEU scores reported in Table 5.

(epochs, learning rate, weight decay, batch size)	BLEU
(3, $3e^{-5}$, 0.02, 2)	0.1264081
(3, $6e^{-5}$, 0.02, 1)	0.1340474
(3, $6e^{-5}$, 0.02, 2)	0.1321528

Table 5: BLEU scores with given hyperparameter settings for model with continuous-valued humor level input.

Owing to the larger dataset resulting in longer train time, we doubled the number of GPUs for this model. This results in a similar train time of 8 hours. We again use a 92%:8% train / validation split, this time for all hyperparameter settings.

We display example generated jokes in Table 6. The generated jokes are roughly similar in meaning to the ground truth punchlines, but exhibit errors in grammatical structure (likely due to shortcomings of the BLEU metric discussed in Appropriacy of BLEU score). These punchlines are generated with the humor level input assigned to the original joke body / punchline by our pretrained humor detection model; in other words, we expect these punchlines to be similar to those from the original data. However, doing this essentially assigns the model to reproduce the original ground truth punchline, rather than attempt to use the desired humor level to tell original jokes with differing humor levels. To analyze the flexibility of our model and its ability to generate novel punchlines with different humor levels, we generate punchlines with new levels of humor in Table 7 for the same two joke setups in Table 6.

In the first two rows of Table 7, we change the humor level input so that the originally unfunny joke is now generated with humor level 0.99 (very high), and the originally funny joke is now generated with humor level 0.01 (very low). The punchline for the former is drastically different from the original generated punchline, implying that the difference in humor level does influence the output. Subjectively speaking, however, we do not believe this newly generated punchline is funnier than the original. It is more grammatically coherent, but the punchline does not connect well with the joke body to produce something funny. On the other hand, the second joke produces a very similar punchline even with the "opposite" humor level. Said punchline just seems to be a paraphrase of the originally generated punchline, with a negligible difference in meaning.

The next two rows in Table 7 examine generated punchlines where the desired humor level is 0.5, not quite unfunny but not quite funny either. We observe similar issues, where the generated punchline for the first joke with humor level 0.5 (row 3) has completely different structure and meaning from the original generated punchline, and the generated punchline for the second joke with humor level 0.5 (row 4) is a paraphrase of the original generated punchline with very little change in meaning. Since both sets of joke generations follow a similar pattern, this may indicate that the humor level input serves little other purpose than as random noise to influence which paraphrase of a given punchline to produce.

6.3 Discussion

In terms of BLEU score, we found that the best model trained on a continuous-valued humor level outperforms the best model trained on a discrete-valued humor level (for the base unfunny case). This may be explained by data augmentation; since paraphrasing produces similar punchlines that the model must learn to generate during training, the model may be more robust against adversarial synonyms (Zhao et al., 2017) and see better performance. Notably, the best-scoring discrete approach model trains for roughly the same amount of time as the best-scoring continuous approach model, as it trains five times as long (15 epochs versus 3 epochs) but on one-fifth as much data.

We suspect that the fastest and easiest way to

Body	Punchline (actual)	Punchline (generated)	Humor level
I heard my son say his first words to me today...	Where have you been the last 20 years?	I are you been? last 25 years?	0.00235891
This German Shepherd comes to defecate on my lawn every day.	His dog came over last night	He dog came over last night	0.99884134

Table 6: Example jokes from continuous-valued humor level model, using original humor values calculated by the pretrained humor detection model.

Body	Punchline (generated)	Desired humor level
I heard my son say his first words to me today...	I want you to tell me what happened to your father, and you can tell me	0.99
This German Shepherd comes to defecate on my lawn every day.	He brought a dog in the house yesterday.	0.01
I heard my son say his first words to me today...	505050, he said.	0.50
This German Shepherd comes to defecate on my lawn every day.	He brought a dog in the house last night.	0.50

Table 7: Example jokes from continuous-valued humor level model, using original humor values calculated by the pretrained humor detection model.

improve performance on this problem is to collect more data. As noted above, a 92%:8% split of training and validation sets had a worse BLEU score at test time than using the entire training set for both training and validation. The addition of a separate validation set should improve the performance, so we theorize the drop in performance to be a result of not having the additional data from the 8% of data allocated to validation. If the number of data examples explains such a wide performance gap between the two settings, then collecting even more data should theoretically make performance even stronger. Given stronger hardware, another avenue to explore is using a model architecture with more parameters or larger batch sizes. Models with more parameters generally tend to generalize better given additional training time, while larger batch sizes will result in smoother descent on the loss function.

Notably, in the additional preprocessing steps taken in the continuous-valued humor level approach, most jokes were assigned very high or very low humor levels (above 0.90 or below 0.10). This indicates very high logit values for the pretrained humor detection model, which may indicate too much time spent training. As such, the continuous-valued humor level model struggles to produce mediocre jokes with humor levels close to 0.5. To improve the model's results on producing jokes with a more complete range of humor levels, it may be useful to generate data by hand and manually label jokes' humor levels. In particular, almost every copied joke body had punchlines that were all high in humor level or all low in humor level. It may be more effective to generate different punchlines with both high and low humor levels for the same punchline, so as to ensure that the humor level input influences the output sequence.

7 Conclusion

We present novel models that can generate punchlines to joke bodies given a desired level of humor. Both models score above 0.1 in BLEU score, indicating significantly better performance than random text generation. Although our continuous-valued humor level model does not truly capture intermediate levels of humor, we demonstrate an approach that does work that we believe could succeed if enhanced by more carefully collected data.

From what we know, this is the first time someone has tried to generate jokes with specific humor levels.

There are several avenues for future research and improvement, including but not limited to the following:

- Many Reddit posts contain spelling errors other than the "word concatenation" analyzed during preprocessing. However, some words are purposefully misspelled for the purposes of the joke. Future research can look into detecting accidental versus intentional typos in jokes, which could further improve preprocessing of joke generation.
- The discrete joke case can be further evaluated on the entire joke training dataset, not just on funny and unfunny jokes individually, so that a maximum BLEU score is potentially observed for joke generation itself, ignoring joke humor level.
- A majority of the jokes used in the continuous case have extreme humor levels (above 0.90 or below 0.10). By improving the usage of the joke humor level model or even the model itself, better logits can be used to train the continuous model.
- Several generated jokes do not have grammatical accuracy. This can be included as a metric in evaluation and have a larger focus in training to ensure the jokes themselves make semantic sense.

Others looking to improve upon our work can access our code at https://github.com/ABKor752/Joke-Generator. We hope that this project provides a valuable starting point for future research in using humor levels in joke generation, further improving a machine's ability to understand humor and the different degrees of humor that a joke can have.

References

Elron Bandel, Ranit Aharonov, Michal Shmueli-Scheuer, Ilya Shnayderman, Noam Slonim, and Liat Ein-Dor. 2022. Quality controlled paraphrase generation.

Md Kamrul Hasan, Sangwu Lee, Wasifur Rahman, Amir Zadeh, Rada Mihalcea, Louis-Philippe Morency, and Ehsan Hoque. 2021. Humor knowledge enriched transformer for understanding multimodal humor. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12972–12980.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. 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.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for F*R*I*E*N*D*S*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215, Sydney, Australia. Association for Computational Linguistics.

J.M. Taylor and L.J. Mazlack. 2004. Humorous wordplay recognition. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3306–3311 vol.4.

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

Orion Weller, Nancy Fulda, and Kevin Seppi. 2020. Can humor prediction datasets be used for humor generation? humorous headline generation via style transfer. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 186–191, Online. Association for Computational Linguistics.

Orion Weller and Kevin D. Seppi. 2019. Humor detection: A transformer gets the last laugh. *CoRR*, abs/1909.00252.

Hang Zhang, Dayiheng Liu, Jiancheng Lv, and Cheng Luo. 2020. Let's be humorous: Knowledge enhanced humor generation. *CoRR*, abs/2004.13317.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *CoRR*, abs/1710.11342.

Yftah Ziser, Elad Kravi, and David Carmel. 2020. Humor detection in product question answering systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 519–528, New York, NY, USA. Association for Computing Machinery.