# Still Funny? Paraphrasing Jokes With Fine-Tuned T5 and BART

**Meixin Yuan**
mayyuan@umich.edu

**Yixin Zheng**
yixinzh@umich.edu

## Abstract

Paraphrase is a popular NLP task. In this program, we explored training language models for paraphrasing jokes to see if the machine learned model could capture the humor in human generated jokes. There are a few state-of-art language models that are appropriate for tuning paraphrasing models. In this program, we explored the T5 (Text-to-Text Transfer Transformer) and BART (Bidirectional and Auto-Regressive Transformer) language models with Transformers and SimpleTransformers packages. We evaluated the generated sentences with USE (Universal Sentence Encoder) scores and from human perspectives. We found that both T5 and BART are effective in paraphrasing tasks with BART slightly outperforming T5. Transformers with PyTorch Lightning and SimpleTransformers both provide good processing efficiency. While Transformers and PyTorch Lightning provide more flexibility and control on parameters, SimpleTransformers is much more easy to use. However, it is hard to capture the humor in jokes for language models because humor relies much on the lexical and cultural contexts.

## 1 Introduction

Paraphrasing is a common task in language use. However, people often find paraphrasing to be hard because the original text constraints people's thoughts in language use. Paraphrasing jokes can add more difficulties because paraphrasing may not capture the humor in the joke. What if there is an algorithm that generates paraphrase automatically? This project utilizes the paraphrase data set Google PAWS to train an algorithm for paraphrasing sentences with given jokes. We aim to fine-tune two models and compare their performances in paraphrasing tasks and compare different fine tune approaches. We fine tuned a T5 (Text-to-Text Transfer Transformer) with HuggingFace Transformers with PyTorch Lightning framework, and BART with SimpleTransformers. We use the Universal Sentence Encoder (USE) to examine the sentence structures and meanings and the USE score will demonstrate whether the paraphrasing has been implemented effectively. With the trained algorithms, we select a few joke examples and paraphrase them with the trained algorithm and conduct human evaluations. We compared the performances of both fine-tuning methods and language models. Finally, we will discuss if there is potential to improve the algorithms.

## 2 Data: PAWS: Paraphrase Adversaries from Word Scrambling

We use Google PAWS (Wiki) data for this task. PAWS is a dataset with 108,463 well-formed paraphrase and non-paraphrase pairs with highly overlapping lexical meanings. It includes the human-rated label that indicates each pair's fluency and paraphrases judgments. It has proven to be a very effective dataset for training paraphrase and other language models that involves pairwise comparisons. The data structure is demonstrated below:

| Field | value |
|---|---|
| id | 1 |
| Sentence1 | In Paris, in October 1560, he secretly met the English ambassador, Nicolas Throckmorton, asking him for a passport to return to England through Scotland. |
| Sentence2 | In October 1560, he secretly met with the English ambassador, Nicolas Throckmorton, in Paris, and asked him for a passport to return to Scotland through England. |
| label | 0 |

### 2.1 Data exploration

We conduct some data exploration for the selected dataset. Figure 1 and 2 shows the distribution of the sentence length for sentence 1 and sentence 2. Then, we create two wordcloud for each of the sentence field to see the frequent words included in this dataset.

*Figure 1. Sentence length of phrase 1*

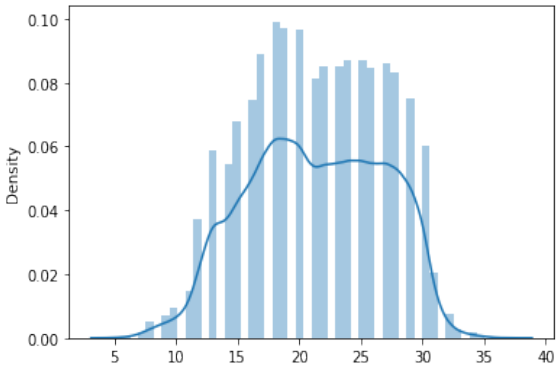



*Figure 2. Sentence length of phrase 2*

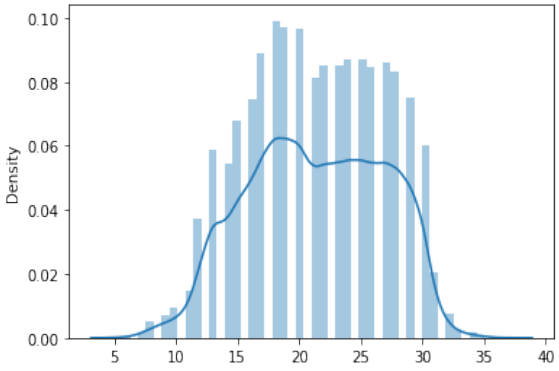
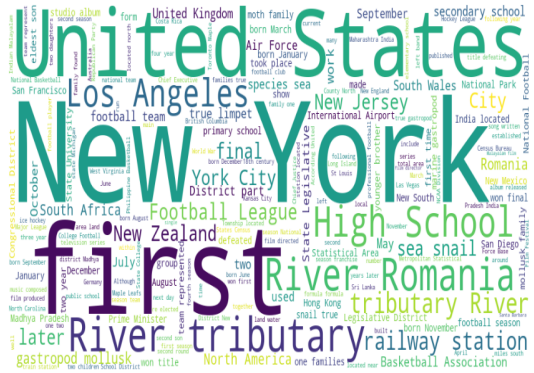



*Figure 3.Wordcloud of phrase 1*


*Figure 4.Wordcloud of phrase 2*




Both sets of the plots demonstrate that sentence 1 and sentence 2 are quite similar in terms of the length of each record and the frequent words for each data. The average length of the record is 21 words with a maximum of 37.

## 3 Related Work

We identified several previous works of paraphrasing tasks in NLP. Fu et. al. (2020) proposed a paraphrase generation method with latent bag of words. Witteven Andrews (2019) explored the use of the large language model GPT-2 to conduct sentence paraphrasing.T5 is a unified framework that covers text-based language problems in text-to-text format (Raffel et. al., 2020). It is trained on an open-source training dataset called Colossal Clean Crawled Corpus (C4). T5 has achieved state-of-art results for many NLP benchmarks and demonstrated great flexibility. Basically, the T5 structure reframes all NLP tasks into a unified text-to-text format, which means it takes both input and output as text strings. Unlike its predecessors GPT or BERT which only takes one set of token ids in the same training process, T5 allows us to train input source data and output target data at the same time. As the figure below demonstrates, T5 can deal with all sorts of NLP tasks.

BART is Bidirectional and Auto-Regressive Transformers. It is a denoising autoencoder for pretraining sequence-to-sequence models. With the standard seq2seq architecture, it not only has the bidirectional encoder like BERT, but also has a left-to-right decoder like GPT.

USE (Cer et. al., 2018) is a model developed to encode sentences into embedding vectors that specifically target transfer learning to NLP tasks. It is a widely accepted method in analyzing sentence similarity. Thus, we also adopt USE to evaluate our models.

## 4 Method

As indicated earlier, we tuned two models for this project. One with T5 and another with BART. For both of the models, our process involves the following steps: 1) process the data to feed into the specific data structure needs for each of the model tuning process, 2)set up model and arguments and build/choose appropriate training functions, 3) tune hyperparameters, 4) train the model, and 5) test and evaluate the model. We will introduce the process for each of the model-tuning processes below. Besides, we created a baseline model for comparison. The baseline model is detailed in the evaluation and results section.

### 4.1 Fine-tune T5 with Transformers and Pytorch Lightning

We used the HuggingFace Transformers to implement T5 with PyTorch Lightning framework. Our code structure refers to PyTorch Lightning documentation[1] and the work from Suraj[2]. We ran our code in Google Colab Pro with a GPU of around 27 GB.

We adopted and adapted to the structure and some functions created by Suraj as our starting point to build our PyTorch Lightning structure for our tuning process. Two pre-trained T5 models are used in this process. We used T5-small for tuning hyperparameters and T5-base for the training process.

The input data is the dataframe of PAWS data and we conducted necessary data processing as follows. First, we have built a paraphrase dataset and created a class for generating data for paraphrasing. Basically, the generated data set is a dictionary of four matrices: source id, source attention mask, target id, and target attention mask. Here, the source refers to sentence1 in the data, while the target refers to sentence 2 in the data. Each record is tokenized with the selected tokenizer from the pretrained T5 model, with each of them being padded to the specified maximum length.

Building the T5 fine tuner class is the most complex part of this tuning process. We inherited the LightningMoule class. Lightning has built in many micro-optimizations automatically and provides some additional parameters allowing for faster process and multi-task processing. For example, when creating data loaders, we set the number of works as 4. It means more than the main process is loading the data. Another trick to improve the process is to clear the optimizer at each step. Besides, we improved the optimizer performance to set no decay for bias and Layer.Normal.weight for AdamW optimizer.

### 4.2 Fine-tune BART with Simpletransformers

We used the library Simpletransformers[3], which is developed, maintained, and updated by Thilina Rajapakse as well as other contributors. This library is built on the work of Hugging Face team and their Transformers library.

We adopted and adapted to the structure and some functions created by Thilina Rajapakse as our starting point to build our structure for our tuning process. The pre-trained model is a BART model. We used "facebook/bart-large" for tuning hyperparameters and for the training process.

After the data pre-processing, we assign the hyperparameters of seq2seq as a dictionary. To construct the pretrained model, we pass the encoder decoder type of "bart", that name of "facebook/bart-large", as well as the dictionary arguments to the seq2seq model in the simpletransformers library. Then we fit the model on our training and development data set.

The loss of the BART model converges at epoch 2.To fine tune the model, we tried different values of learning rate and epsilon. It turns out the learning rate of 5e-5 and epsilon of 1e-8 returns least loss. So we use the epoch of 2, learning rate of 5e-5, as well as epsilon of 1e-8 to train the model.

[1] https://pytorch-lightning.readthedocs.io/en/latest/?$_ga$ = 2.222514451.980334004.1619567125 − 1748674786.1619567125

[2] https://github.com/patil-suraj/exploring-T5/blob/master/t5$_{fine_tuning}$.ipynb

[3] https://github.com/ThilinaRajapakse/simpletransformers

| Learning rate, epsilon | epoch | Train perplexity | Validation perplexity |
|---|---|---|---|
| (5e-08, 1e-08) | 1 | 4.023 | 4.469 |
| (5e-08, 1e-08) | 2 | 4.015 | 4.468 |
| (5e-08, 1e-06) | 1 | 4.045 | 4.469 |
| (5e-08, 1e-06) | 2 | 3.843 | 4.469 |
| (5e-06, 1e-08) | 1 | 3.878 | 4.244 |
| (5e-06, 1e-08) | 2 | 3.918 | 4.164 |
| (5e-06, 1e-06) | 1 | 4.072 | 4.299 |
| (5e-06, 1e-06) | 2 | 3.902 | 4.230 |
| (0.0005, 1e-08) | 1 | 3.112 | 2.506 |
| (0.0005, 1e-08) | 2 | 2.565 | 2.404 |
| (0.0005, 1e-06) | 1 | 3.158 | 2.551 |
| (0.0005, 1e-06) | 2 | 2.650 | 2.439 |

Then we conducted the parameter tuning for learning rate and epsilon value for the optimizer. The parameter tuning results is demostrated below. We choose to report the perplexity score for each of the eapoch for each set of paramters. We choose the learning rate of 5e- 5 and epsilon of 1e-8 for the final training process. We trained the model with the full train and development dataset with T5-base pre-trained encoder. The training process took around 1hr 50min in Google Colab Pro for 2 epochs.

## 5 Evaluation and Results

### 5.1 Baseline model

We constructed a baseline model by re-sample the words in the input sentence. Setting the same length of sentence, we randomly choose the word from the input sentence construct the output sentence.

We did both machine evaluation and human evaluation for three models of baseline model, T5 model as well as the BART model.

### 5.2 Machine Evaluation

In the Google data, there are input sentences and targeted sentences. After training the model on the training and development data, in the test set, we use the input sentences to generate the predicted sentences, and compare the predicted sentences by our model with the targeted sentences in the Google test data set. The comparison is done by USE score.

As for USE score, after embedding from the universal sentence encoder, the USE calculation, we will get the sentence similarity scores for each record pairs.

The USE score is 0.703 in the baseline model, while it is 0.963 in the BART model and 0.912 in the T5 model.

| Model | USE Score |
|---|---|
| T5 | 0.912 |
| BART | 0.963 |
| Baseline | 0.703 |

### 5.3 Human Evaluation

The evaluation is applied to the paraphrased jokes after the best model is used for it. Besides evaluated by USE, this time we also evaluate the paraphrased sentence from the perspective of fluency,coherency and fun.

We select three jokes from the scoutlife website[4] and pass it to our models. Here we present one example of our results.

| Model | Sentence |
|---|---|
| Original | A kid finds a magical lamp.He rubs the lamp, and a genie appears and says, “What is your first wish?” The kid says, “I wish I were rich!” The genie replies, “It is done! What is your second wish, Rich?”. |
| T5 | A kid finds a magical lamp, and rubs the lamp around him and appears. A genie says to her what first wish is? The kid says, “I wish I were rich!” The genie replies: It is done! What is your second wish for Rich?” |
| BART | A kid finds a magical lamp and rubs it. A genie appears and says, “ What is your first wish? ” The kid says, “I wish I were rich!” The genie replies, “ It is done! What is your second wish Rich? |
| Baseline | appears says, rubs wish?” magical a genie lamp.He magical is and and “What finds first genie and lamp, first genie. What says, The replies, I I The What says, is “It kid genie were What kid “It What “It The. |

## 6 Discussion

According to USE score, the BART performed best, while T5 took the second best.The fine tuning of both of the models took around the same time. Both of them outperform the Baseline model.

[4]https://jokes.scoutlife.org/jokes/long-jokes/

From the human evaluation, the paraphrase sentences generated by both the T5 and BART are mostly fluent and coherent sentences. However, the baseline mode is insufficient in this respect. In general, both the T5 and BART generated sentences can capture most of the meaning of the original joke. However, there are much variances in how funny the paraphrase is. In the demonstrated example, the BART generation captures more of the humor from the original joke than the T5 generation. However, some unintended generation could also be fun. For example, the following example shows that T5 generates a joke with opposite meaning of the original one, but also seems to be fun.

| Model | Sentence |
|---|---|
| Original | A man walks into a library, approaches the librarian and says, "I'll have a cheeseburger and fries, please."' The librarian says, "Sir, you know you're in a library, right?"' '"Sorry," he whispers. "I'll have a cheeseburger and fries, please." |
| T5 | A man walks into a library and approaches the librarian and says, "I want to have cheeseburger and fries," please.The librarian says: "Sir, you know that your library is there?"He screamed, "Sorry," I'll have a cheeseburger and fries please. |

## 7 Conclusion

We used the Google PAWS data to fine-tune two state-of-art language frameworks for this project. We have constructed the re-sampling baseline model, and fine tuned T5 and BART model. We used the tuned hyperparamters and trained the model with the full-size training/development data of PAWS. To evaluate the result of the generation, we calculate the average USE scores based on model generated results and test data. We finally discussed the results of paraphrasing from human perspectives.

Both T5 and BART model performed better than the baseline model, which indicates the fine-tuning is meaningful. As for the paraphrasing use, BART keeps most precise meanings of the original sentence, so it is considered to be more appropriate for paraphrase tasks. However, we also identify that the nuances between BART and T5 model might be result of the parameters used and different piplines of our training process.

Both the PyTorch Lightning framework and the framework provided by Simple Transformers are proved to be quite efficient. However, Transformers with Lightning allows more flexibility and more control during the process. However, SimpleTransformers is very simple and straightforward although there are more invisible processing and make it harder to debug.

Our attempts demonstrate that paraphrase joke is a quite complex task as humor relies much on the lexical and cultural contexts.

## 8 Other Things We Tried

We have considered algorithm of Latent Bag of Words for Paraphrasing, but we failed for the library version conflict. The module "tensorflow.contrib" doesn't exist in the tensorflow 2.x version. We tried to install the version 1.14, but it is not listed in the PyPI website. We also tried "tf_upgrade_v2" function which is for automatically upgrading code to tensorflow 2; but it doesn't work for the configuratioin.

For tuning the T5 model, we tried to apply a conventional training pipline without Llightning, however, it was not as efficient and there are some problems in validation process.

In the BART tuning parameters, we tried to store the perplexity and loss for each value of parameters; however, the cuda will be out of memory if we tried more than 2 loops. So we tried to turning only one or two loops each time and didn't store it in the data frame.

## 9 What You Would Have Done Differently or Next

We may try Latent Bag of Words for paraphrasing. For the version conflict, we have tried to install tensorflow 1.14 in local annoconda; it seems a package can be found, but conflict with python version. we should build a virtual environment of lower python version to see whether it works.

We found that our generations have much overlap with the original sentences.However, we were unable to efficiently deal with the problem. If we continue on this work, we may try to revise the generation process to limit the overlap.

Finally, as there are not good dataset for paraphrase jokes in specific, we used PAWS in our case. However, it might not represent the semantic styles

of jokes. Therefore, we would like to try further fine tuning the model with jokes if better data is available.

## 10 Group Effort

The project topic and goals are co-developed by both of the authors. Meixin Yuan conducted the data exploration, fine tuning of the T5 in transformers and Pytorch Lightning, and contributed on the human evaluation. Yixin Zheng created the baseline model, conducted the fine tuning of BART and machine evaluation. The report and medium post is written by both of the authors.

## 11 Acknowledgement

We appreciate the comments and suggestions from Prof. David Jurgens Chulei Chen,Qi Sun,Lea Wei, Hisamitsu Maeda, Aparna Ananthasubramaniam,Zhiyuan Zhan.

## References

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., Kurzweil, R. (2018). Universal Sentence Encoder. ArXiv:1803.11175 [Cs]. http://arxiv.org/abs/1803.11175

Dolan William. B., Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. Proceedings of COLING 2004, Geneva, Switzerland.

Fu, Y., Feng, Y., Cunningham, J. P. (2020). Paraphrase Generation with Latent Bag of Words. ArXiv:2001.01941 [Cs]. http://arxiv.org/abs/2001.01941

Quirk, Chris, Chris Brockett, and William B. Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation, In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 25-26 July 2004, Barcelona Spain, pp. 142-149.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. ArXiv:1910.10683 [Cs, Stat]. http://arxiv.org/abs/1910.10683

Rajapakse, T. (2020, August 5). BART for Paraphrasing with Simple Transformers. Medium. https://towardsdatascience.com/bart-for-paraphrasing-with-simple-transformers-7c9ea3dfdd8c

Witteveen, S., Andrews, M. (2019). Paraphrasing with Large Language Models. Proceedings of the 3rd Workshop on Neural Generation and Translation, 215–220. https://doi.org/10.18653/v1/D19-5623

Zhang, Y., Baldridge, J., He, L. (2019). PAWS: Paraphrase Adversaries from Word Scrambling. ArXiv:1904.01130 [Cs]. http://arxiv.org/abs/1904.01130