# Music Recommendation for Podcast Scripts: Detecting Emotion from Text

Stanford CS224N Custom Project

**Anne Lee**
Department of Computer Science
Stanford University
alee7368@stanford.edu

**Shreya Ravi**
Department of Computer Science
Stanford University
sravi2@stanford.edu

**Alex Tsun**
Department of Computer Science
Stanford University
alextsun@stanford.edu

## Abstract

In this project, our task is to generate music recommendations based on podcast scripts. Given a podcast script, our model will detect the emotion from the text to recommend relevant music. To achieve this, our model has been trained on a Twitter dataset with labelled emotions. Then, the model will be trained on podcast scripts through domain adaptation. Initial approaches include building a CNN, LSTM, RNN, BERT, and ALBERT model for emotion detection on the Twitter dataset. In the future, we will perform domain adaptation for podcast scripts and fine-tune our BERT and ALBERT models. Finally, we will have labelled music with emotions, and recommend a random subset of the music with the same predicted emotion of the podcast.

## 1 Key Information to include

- External collaborators: Mercan Topkara (Luminary Media)
- Mentor: Dilara Soylu
- Sharing project: N/A

## 2 Introduction

Recently, detecting emotion from text has become an increasingly popular research field due to the vast potential application in media, marketing, and countless other fields. For example, emotion detection can be used by marketing teams to analyze sentiment on newly released products or by publishing agencies to categorize stories or news articles by emotion.

Sentiment analysis is already a widely researched field in natural language processing, though much of human emotion cannot be entirely encapsulated on a scale ranging from positive to negative sentiment. For example, though disgust and anger both represent negative sentiments, their connotation differs significantly and can be useful in many applications.

Extracting emotions, not just positive and negative sentiment, is an extremely challenging text due to the nuances of natural language. Emotions are highly complex and cannot be represented on a one-dimensional level, so classifying text into one category is extremely difficult. Additionally, people respond differently to events and express emotions in varying ways – adding to the complexity of the challenge. There is also a lack of large, labelled emotion datasets.

In this project, we aim to detect emotions of podcast transcripts with the ultimate goal of generating music recommendations. First, we trained a model on Twitter data with labeled emotions. Then, we apply transfer learning to adapt this model to detect emotion of podcast transcripts. Finally, we recommend a random subset of music categorized by extracted emotion.

Throughout the project, we have spoken with many journalism professors, podcast producers, and other experts in the field. Finding music for podcasts is currently a time-consuming and laborious process, which we aim to solve through natural-language processing. Ultimately, we wanted to launch a platform to help content creators find music based on their podcast scripts.

## 3 Related Work

Many researchers have been working on extracting emotion from text. Below we describe both supervised approaches for detecting emotion, as well as unsupervised approaches.

### 3.1 Supervised Approaches for Detecting Emotions in Text

Because there are few emotion-labeled datasets, many supervised approaches for emotion detection use data collected from blogs or social media using hashtags and emojis as the emotional category. For example, Wang et. al (2012)[1] curated a dataset of 2.5 million tweets mapped to seven basic emotions, trying various combinations of features, including the usage of different lexicons, different n-grams, and part-of-speech tagging (POS). Ultimately, they found that ”n-gram(n=1,2), LIWC lexicon, MPQA lexicon, WordNet-Affect, and POS” was the most optimal combination of features. However, the final classifier’s F-measures reached as high as 0.72 for joy, and as low as 0.13 for surprise. In another study, Li and Xu (2014)[2] extracted features that are relevant/meaningful to emotions. The team extracted emotion ”causes” based on predefined linguistic patterns as features, and used Support Vector Regression (SVR) to create the classifier, ultimately achieving better precision but lower recall. Li et al. (2015)[3] focused on sentence-level classification instead of document-level by creating a Dependence Factor Graph (DFG). The team was ultimately able to achieve accuracy levels of 63.4% with F1 of 0.37, a significant improvement in comparison to previous methods.

### 3.2 Unsupervised Approaches for Detecting Emotions in Text

Agrawal and An (2012)[4] extracted NAVA words (nouns, adjectives, verbs, and adverbs) and dependencies between these words as contextual information. Then, emotion vectors for words were computed based on the semantic relatedness through Point-wise Mutual Information (PMI). Then, a vector was computed for each word using its PMI and the contextual information from dependencies. Finally, by aggregating the vectors of each word in a sentence, a vector was generated for each sentence. Overall, this approach had better results in comparison to other unsupervised approaches, and had nearly comparable results to some supervised approaches.

### 3.3 Limitations and Challenges

Due to the lack of labelled emotion datasets, many researchers (Wang, Li, etc.) have relied on tweets with hashtags and emojis as labels. This may not be the most optimal dataset, as hashtags and emojis may not consistently correlate with the text within the tweet. Additionally, people respond differently to events and express emotions in varying ways – adding to the complexity of the challenge. Finally, in the supervised approaches described above, there are high levels of misclassification in underrepresented classes. To mitigate this, researchers can modify the learning algorithm to adapt to imbalanced data, increase the cost of majority classes during training, or increase the amount of data in underrepresented classes.

Additionally, there is often a high degree of misclassification among related classes, such as ”anger” and ”sadness” or ”joy” and ”love.” Because emotions are highly complex and cannot be represented on a one-dimensional level, future approaches can define emotions based on higher level dimensions and categorize text into multiple classes.

## 4 Approach

### 4.1 Overview of Our Approach

Our main goal is to classify emotion (5 classes) on a small podcast dataset (∼1000 examples). To do this, we wanted to apply transfer learning/domain adaptation by first training on a larger Twitter dataset (∼29000 examples). We had 4 main stages of experiments in achieving our final result.

1. Train our models on just the Twitter data.
2. Train our models on preprocessed Twitter data. This meant cleaning the tweets (e.g., removing punctuation, tags, etc.).
3. Apply transfer learning to our podcast dataset with pretrained models from the previous stage.
4. Directly train our models on the podcast data, as a comparison.

We describe each stage further in our experimental details section, but the neural model architectures we used remain the same across the experiments, and will be described here.

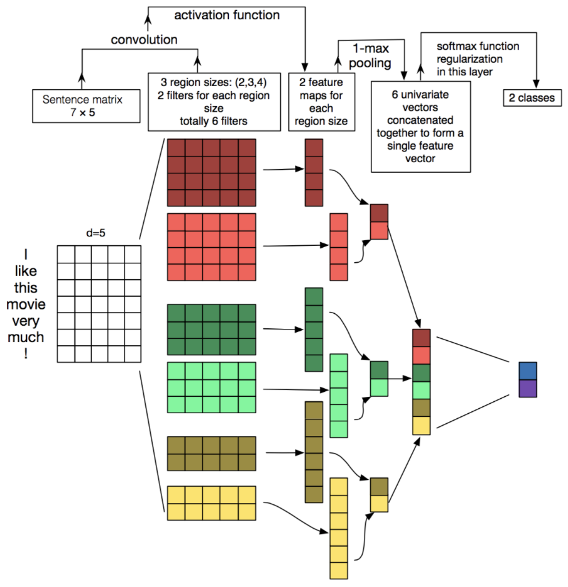
### 4.2 Baseline

Our baseline approach for classification of text was using RNN's, LSTM's, and CNN's. We used a standard tokenizer for each of the approaches and pretrained GloVe word embeddings to get a sequence/matrix of vectors per word. Below we describe the models in more detail. We used pre-existing code from here (Github https://github.com/prakashpandey9/Text-Classification-Pytorch) as inspiration for these baseline classification models, and from here (https://mccormickml.com/2019/07/22/BERT-fine-tuning/) for BERT models.

#### 4.2.1 Convolutional Architecture

For our CNN, we used several convolutional layers, followed by max-pooling and concatenation to convert our sentence matrix into a single vector. We followed up with a fully-connected layer and softmax to get predictions. The below architecture image is from [5]. Details such as filter sizes are described in experimental details.

Figure 1: CNN Architecture for Text


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

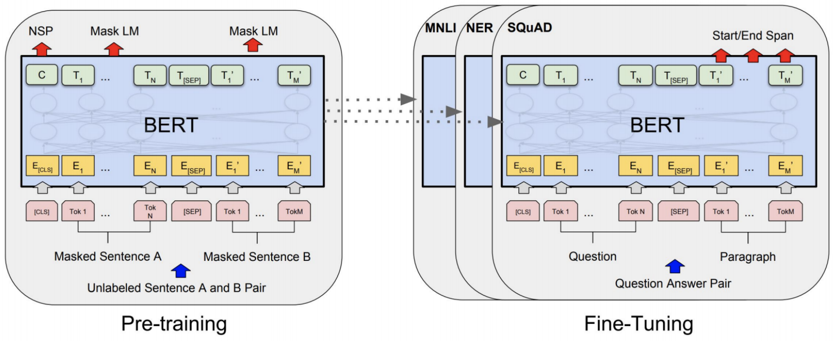
#### 4.2.2 Recurrent Architectures

We used a standard LSTM/RNN architecture to convert our sequence of word vectors into an embedding for the entire sentence. We followed up with a fully-connected layer and softmax to get predictions. We added a dropout layer before the fully-connected layer to improve learning without overfitting. We also created our own Bidirectional LSTM to get a better representation of the text before feeding it through our classifier.

### 4.3 Main Approach

Our main approach was to use pretrained BERT [6] and its variants such as ALBERT [7] to retrieve bidirectional sentence embeddings, and feed them through classifiers consisted of fully connected layers and nonlinearities. Pretrained BERT has been shown to get SOTA results on many NLP tasks, and mainly due to the powerful sentence representations it learns using transformers and multi-headed attention. We used the Wordpiece tokenizer rather than the standard ones for baseline (lower case + split on whitespace), to adhere to its predetermined vocabulary. We froze all the BERT layers as the iteration time would be too expensive and we did not have the computational resources to fine-tune that many parameters. The below architecture image is from Jacob Devlin's guest lecture slides.

Figure 2: Fine-Tuning with BERT



## 5 Experiments

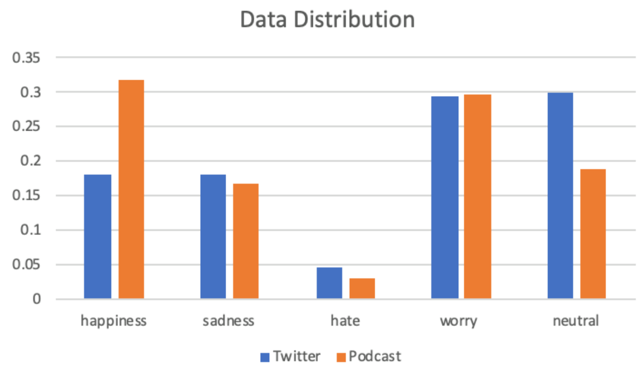
### 5.1 Data

Initially, we scraped Wattpad (an online storytelling platform) for text to train on. However, after gathering the data, we realized the stories were not of high enough quality and there would be significant pre-processing that would be required. Instead, we have been using a Twitter dataset called the Sentiment Analysis: Emotion in Text dataset by CrowdFlower, which contains 40k samples of Tweets with 13 emotion labels. We narrowed down the dataset to contain 5 main emotions (Happiness, Sadness, Neutral, Worry, Hate) with 29k samples. For transfer learning, we have been manually labeling podcast transcripts that Luminary Media shared with us. We wrote a data-labeler interface and labelled 1.5k paragraph excerpts.

### 5.2 Evaluation method

We used the weighted F1 metric which is a standard for multi-class classification. That means we computed per-class F1 scores, which is defined as the harmonic mean between precision and recall $\frac{2PR}{P+R}$, where $P$ is precision and $R$ is recall. Then, we take a weighted average over the classes, weighted by the proportion of true labels in that class. This is a better metric than standard accuracy, since if there is class imbalance (99% one class for example), then we could get 99% accuracy just by guessing that class.

Figure 3: Podcast Data Distribution



### 5.3 Experimental details

For the BERT and ALBERT models, we used the pretrained base models with an extra fully connected layer from dimension 768 (embedding of [CLS] token) to dimension 5 (number of labels) followed by a softmax layer, while freezing BERT parameters and only training the last layer. We called this model BERT_Classify. We then made this model deeper by adding a fully connected layer between the BERT model and the classification fully connected layer, and adding a nonlinearity (a ReLU layer) to allow it to learn more complex relationships. We also added dropout layers before each fully connected layer to help the model train without overfitting. We called this model Bert_DeepClassify.

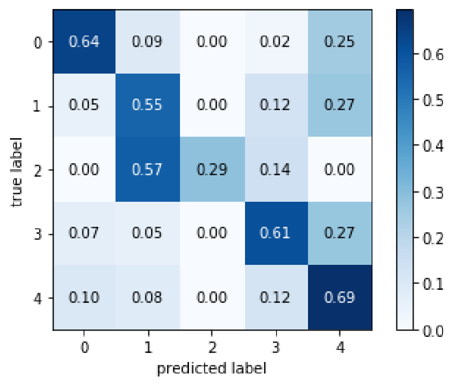
For the CNN, we had 100 filters of size 3, 4, and 5 by text length respectively, and then we RELU'd and maxpooled the output. Then, we had a dropout of 0.5 and softmax layer from 300 (3*100 filters) dimensions to 5 dimensions. For the RNN, LSTM, and Bi-LSTM, we used a hidden size of 256, and GloVe embeddings of dimension 300.

### 5.4 Results

Our results are detailed in Table 1 of the Appendix. The F1 and Accuracy Scores were higher for BERT (F1: 51.53 and Accuracy: 52.11) and ALBERT (F1: 47.81 and Accuracy: 49.36) in comparison with our baseline models (CNN, RNN, LSTM, Bi-LSTM). BERT also performed better than ALBERT. This is expected since BERT is more powerful than ALBERT, which is in turn more powerful than all of our baselines. In the future, we will focus on optimizing our BERT model instead of running both BERT and ALBERT.

Figure 4: Confusion Matrices of the 2 Final Models

(a) Directly on Podcast Data



(b) Domain Adaptation from Twitter to Podcast Data

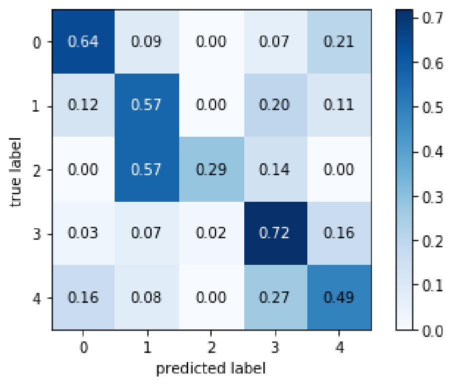


Table 1: Results for Twitter Models on Twitter Data

| Model | Original Data | | Pre-processed Data | |
|---|---|---|---|---|
| | F1 Score | Accuracy | F1 Score | Accuracy |
| CNN | 43.73 | 45.00 | 45.73 | 42.63 |
| RNN | 37.69 | 44.07 | 42.6 | 44.94 |
| LSTM | 37.68 | 40.72 | - | - |
| Bi-LSTM | 43.00 | 45.30 | 36.92 | 40.63 |
| BERT_Classify | 50.57 | 51.10 | 50.49 | 50.56 |
| BERT_DeepClassify | 51.53 | 52.11 | 50.93 | 51.29 |
| ALBERT | 47.81 | 49.36 | - | - |

Table 2: Results for Models Trained Directly on Podcast Data

| Model | F1 Score | Accuracy |
|---|---|---|
| CNN | 37.76 | 38.92 |
| Bi-LSTM | 56.97 | 44.08 |
| BERT_DeepClassify | 63.08 | 61.03 |

Table 3: Results for BERT model with Transfer Learning from Twitter to Podcast

| Model | F1 Score | Accuracy |
|---|---|---|
| BERT_DeepClassify | 62.31 | 61.36 |

## 6 Analysis

For the first approach of training directly on the podcast data, we found that BERT_DeepClassify was our best performing model, as seen from the results in Table 2. One shortcoming of this model was the lack of podcast transcripts that were labeled with emotion, since we only had $\sim$1.5k samples.

We believed we could address this problem with our second approach of training on the larger Twitter dataset, and retraining only the last layer of our model to fine-tune the weights on our podcast dataset. We thought that this would yield better results due to having fewer weights to train, and thus fewer data required to obtain good results.

In order to determine which model we would use to perform the transfer learning step, we experimented with various model architectures listed in Table 1, and we found that Bert_DeepClassify was the best model for this portion of the task. Thus, we decided to use this architecture to implement the transfer learning step, and we obtained the results in Table 3.

### 6.1 Data Limitations

We observed some limitations with our data that we were not able to overcome using the transfer learning approach. The first is inherent differences in samples in the two datasets. Each podcast excerpt was longer, with the median podcast excerpt length being 116 words and the median tweet length being 12 words. Additionally, we noticed there were a lot more out-of-vocabulary (OOV) words in the twitter dataset as a result of the mediums of Twitter and podcasting being different. We attempted to adjust for this difference by removing common OOV words by removing all the Twitter mentions ("@user") and the hashtags ("#tag") from the dataset and training on the remainder of the words for the initial training before the fine-tuning on the podcast data. This pre-processing only marginally helped, however.

By further observing the data directly, we can see that the quality of twitter dataset resulted in some of the incorrectly classified data. For example, the following tweet was classified as `sadness`:

Just got home i love stake and shake milkshakes

This tweet should have been labeled as `happiness` and likely contributed to the model's overall misclassification of `sadness` tweets as `happiness` tweets. This is only one example, but this mislabelling of tweets existed across all emotion categories.

## 6.2 Further Reasons for Misclassification

We observed very similar performances between the two BERT_DeepClassify models, one of which trained directly on the podcast data and the other which used the transfer learning approach. By studying the confusion matrices of the models (Figure 5, (a) and (b) respectively), we make a few key observations. The first is that in both models (and slightly more so on the model trained directly on the podcast data), the emotions `sadness` (Emotion 1) and `worry` (Emotion 3) are often confused. More specifically, `sadness` (Emotion 1) is often confused with `worry` (Emotion 3) rather than the other way around, and this is likely due to the fact that there is almost twice as many examples in both the twitter and podcast datasets for the `worry` emotion compared to the `sadness` emotion.

It also seems like the model attaches emotion strongly to certain words or phrases and is thus unable to put it in the larger context of the sentence and classify it as such. For example, the model incorrectly predicted that the following `neutral` tweet was a `sadness` tweet.

> Selling my Bonnaroo ticket. Can't get any time away from work. Anyone in the DC area interested? I can meet up!

It is likely that the model fixated on the phrase "Can't get any time away from work", which typically entails a lot of negative, sad emotions but the tweet is actually neutral because it is largely informative and this tweet is a means of selling, which is not associated with an emotion in particular.

## 6.3 Ablation Experiments

By testing various model architectures and comparing them based on the accuracy and F1 scores on the twitter dataset, we were able to determine which model performed best. Table 1 shows the results of these experiments with each model architecture. We found the best architecture for the Twitter data to be BERT_DeepClassify model, and doing these experiments told us where this model failed to produce accurate classification predictions and how to progress forward in fine-tuning this model on the podcast data.

We also performed ablation experiments to determine whether pre-processing the twitter data would help, and we found that on our simpler, baseline architectures it helped a lot but on our transformer architecture it did not increase performance very much. However, we decided to use the model trained on pre-processed inputs anyways because it better matched the domain we were trying to adapt to.

Overall, these ablation experiments told us that pre-processing the twitter data to match the podcast transcript domain slightly better and using the BERT_DeepClassify model would likely give us the best results after we implemented our transfer learning step.

## 6.4 Errors from Podcast Data

We hand-labelled the podcast data, so the quality of the labels is very good, although the data distribution (the number of samples of each emotion in the dataset) is a bit more skewed than for the twitter dataset because podcasts are less likely to have `hate` or `sadness` as predominant emotions as opposed to `happiness` due to the differences inherent in the media. As a result of this skewed dataset, both models predicted the following excerpt to belong to the category `happiness` when in reality it was `neutral`.

> So I decided to pretend I was from immigration. And this is how the call went. Hi Mrs. This is call in from the Australian Immigration Department. I have your followed right here in front of me ... She's obviously desperately trying to be very polite because potentially this person on the phone holds a lot of power over how her life kind of goes and blame Armstrong. He's your fiance Xeno and what is blinds Occupation? Thank you.

Additionally, it was likely harder to detect emotion on longer pieces of text. This was probably made even harder by training on twitter data, which is shorter, and thus the model likely was optimized for the emotion-related words to be very concentrated and there to only be one or two in the entire text. However, this is not always the case for podcasts, and as we saw when we labeled our data, emotion-related words appear across the long text to convey a more general mood. This is typically more subtle than emotion from a tweet.

## 7 Conclusion

Through our many experiments, we discovered that RNNs and vanilla LSTMs cannot match the performance of bi-LSTMs which encode important information about both directions in the text. CNN's are also surprisingly effective at handling text data, and perform as well as bi-LSTMs though they train even faster due to the parallelization available in the architecture.

The most modern systems such as BERT, not only take into account bidirectionality, but also use stacked transformer and attention mechanisms which are massively parallelizable as well. In some sense, it takes the best of both recurrent and convolutional architectures, and adds more, and is able to perform at an even higher level. However, though BERT is powerful and helps generate rich sentence embeddings, it has so many parameters that the forward and especially backward time is extremely slow, and hence we use fine-tuning and freeze the BERT layers. This saves a lot of time and space, in not storing and computing gradients for these layers.

We are happy we were able to get F1 scores in the 50's for the Twitter data and 60's for the podcast data, when random guessing across 5 classes would give around 20. The most likely explanation for the very similar results in terms of F1 scores, accuracies, and even confusion matrices between the 2 approaches is that the pre-training on the twitter data didn't add much additional useful data. This is due to the quality of the data and inherent differences between the types of data being trained on (e.g. the Twitter data has a lot of slang and OOV words and is shorter). As a result, the fine-tuning likely did the bulk of the work in detecting the emotion from the podcast excerpts and both models had similar performances. Overall, the BERT_DeepClassify model in both transfer learning and direct on podcast eventually converged to similar performance, and the fine-tuning initialization didn't really help.

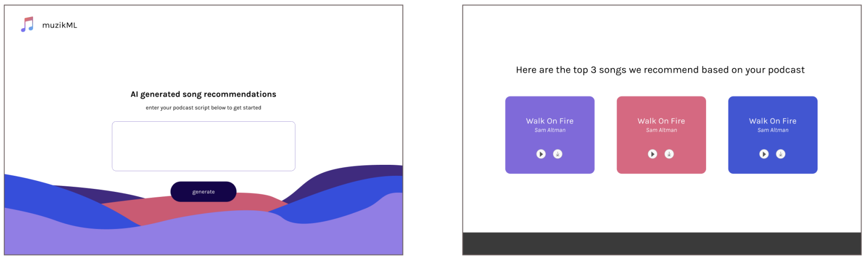
## 8 Impact

Throughout our project, we have been reaching out to and connecting with several contacts from media companies to design our project around their needs. Professor Jay Hamilton (Chair of the Department of Communication) and Professor Serdar Tumgoren (Professor in Professional Journalism) have been valuable mentors. We have also been speaking with contacts from Vox, The New York Times, and the Stanford Daily.

## 9 Future Work

The Stanford Daily has agreed to test our model/product. In the future, we hope to develop a front-end interface to help the Stanford Daily, Vox, and other podcasts generate music recommendations. We began development here: https://thawing-retreat-13585.herokuapp.com/. Screenshots of the platform is shown below.

Figure 5: Music Recommendation System Front-End



AI generated song recommendations

enter your podcast script below to get started

Here are the top 3 songs we recommend based on your podcast

## References

[1] Krishnaprasad Thirunarayan Wenbo Wang, Lu Chen and Amit P Sheth. Harnessing twitter" big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust (PASSAT)*, 2012.

[2] Weiyuan Li and Hua Xu. Text-based emotion classification using emotion cause extraction. In *Expert Systems with Applications*, 2012.

[3] Rong Wang Shoushan Li, Lei Huang and Guodong Zhou. Sentence-level emotion classification with label and context dependence. In *International Joint Conference on Natural Language Processing*, 2015.

[4] Ameeta Agrawal and Aijun An. Unsupervised emotion detection from text using semantic and syntactic relations. In *Web Intelligence and Intelligent Agent Technology-Volume 01*, 2012.

[5] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.