LYRICS-BASED MUSIC GENRE CLASSIFICATION USING STATE OF THE ART TRANSFORMER BASED MODELS

1. ABSTRACT

With the advent of music streaming services in recent years and their respective recommendation algorithms, classifying song genres is a commercially useful application of deep learning. In this paper we attempt to predict the genre of a song purely from an examination of the song's lyrical content using various pre-trained transformer based models. As lyrics exhibit a hierarchical layer structure—in which words combine to form lines, lines form segments, and segments form a complete song—we finetune several of these commercially available pre-trained transformer based models to exploit these layers and in addition learn the importance of the words, lines, and segments. We then test the fine tuned models over a 11-genre dataset that consists of over 220,000 songs in English. Through numerous experiments, we found that the RoBERTa model architecture coupled with discriminative learning techniques adapted from the Universal Language Model Fine-tuning for Text Classification paper offered the best accuracy and F1 score among all models.

2. INTRODUCTION

The ability to automate music classification is a well researched task in Music Information Retrieval [1]. Previous work on this topic has focused primarily on classifying mood, genre, and artists in which one or a combination of audio, lyrical and symbolic data is used in machine learning algorithms for these tasks [2]. While traditional approaches in text classification have utilized n-gram models and traditional algorithms such as Support Vector Machines, k-Nearest Neighbour, and Naive Bayes, recent years have seen the explosion of deep learning models such as recurrent neural networks, long short-term memory, and state of the art (SOTA) model architectures such as the famous Bidirectional Encoder Representations from Transformers (BERT). Neural methods have further been utilized for the genre classification task on audio and symbolic data. Sigtia and Dixon [3] use the hidden states of a neural network as features for songs on which a Random Forest classifier was built, reporting an accuracy of 83% among 10 genres. However, not much research has looked into the performance of these deep learning methods with respect to the genre classification task on lyrics. Here, we attempt to remedy this situation by applying SOTA language model architectures to text classification to the particular case of lyrics. Specifically, we have collected approximately 220,000 song lyrics that we will classify into 11 genres: Indie, Country, Jazz, Metal, Pop, R&B, Other, Folk, Rock, Electronic, and Hip-Hop.

3. RELATED WORK

There is some level of historical work on text mining and classification, including genre detection [4] and text analysis on lyrics [5]. In the early stages of development, music classification was mainly based on acoustic features. Lyric-based music classification, however, was not considered effective. For instance, McKay et al. [2] even reported that lyric data performed poorly in music classification. In recent years, lyric-based music genre prediction has attracted attention. Some research has combined lyrics and acoustic features to classify music genres, leading to more accurate results [6]. Lustrek [7] used function words (prepositions, pronouns, articles), specific words in genre, vocabulary richness, and sentence complexity in lyric-based song classification. He also used decision trees, Naive Bayes, discriminant analysis, regression, neural networks, nearest neighbours, and clustering. Another approach is reported by Fell and Caroline [5], which combines n-gram models and different features of a song content, such as vocabulary, style, semantics, orientation towards the world (e.g. “whether the song mainly recounts past experience or present/future ones”), and song structure. Their experiments showed the classification accuracy between 49% and 53% [5]. Canicatti [8] used a combination of song lyric data collected from the public MetroLyrics API with a database of song metadata (e.g. artist and beats per minute) to classify songs into one of five genres: rock, rap, country, jazz, and pop. These genres were specifically chosen because they are unlikely to have overlap in either lyrical content or song structure. With this, he performed a number of experiments with classifiers like decision trees, Random Forest, k-nearest-neighbor, and Naive Bayes to determine the best model. Of the accuracy measurements, Random Forest was the best of the four with about 52% accuracy while the others were between 34% and 40% [8]. Tsaptsinos [9] used a hierarchical attention network to classify music genres. The method replicates the structure of lyrics and enables learning the sections, lines or words that play an important role in music genres. He focused on using only lyrics to train a hierarchical attention network (HAN) built on top of a recurrent neural network (RNN), with an eye towards explicitly learning hierarchical features on top of the text inputs such as words, lines, and segments. He obtained their data through a signed research agreement with LyricFind, which consisted of unlabeled JSONs, and retrieved the genre classes through the iTunes API. Comparison models included the mode (the most common genre), logistic regression, long-short term memory units (as opposed to the gated recurrent units used in the HAN), and a hierarchical network (which lacked attention). The study found the HANs to be the most effective model in the complex 117-genre classification task, with a 46.42% accuracy, although it found LSTMs to suffice and perform better on the simpler 20-genre task [9].

However, there is minimal existing literature regarding song lyric genre classification using state-of-the-art (SOTA) language model architectures such as BERT, RoBERTa, ALBERT, and XLNET, which is the focus of this paper.

4. DATASET AND ANALYSIS

4.1 Data Source

We compiled our custom dataset from MetroLyrics, which is a website with a database of over 1 million songs by over 16,000 unique artists. Although it has a fairly limited number of features for each of the songs in the dataset, it does have the song name, year, artist, genre, and most importantly, the full lyrics of the songs, which are more than sufficient for our goal of classifying song genres only using the full lyric text. The songs in the dataset were categorized into the following genres: Indie, Country, Jazz, Metal, Pop, R&B, Other, Folk, Rock, Electronic, and Hip-Hop. More specifically, our dataset consists of 362,237 songs that required significant preprocessing as described in the next section.

4.2 Data Cleaning

Because MetroLyrics receives contributions from music fans who manually transcribe and enter lyrics into the site, the structure, convention, and quality can vary as there is no official governance or regulatory body to enforce standards. Thus, data cleaning was extensive. Initially, all songs that were purely instrumentals and did not contain lyrics were removed from the dataset. Almost all punctuation, non-ASCII characters and musical descriptors like “Chorus” and “Verse” were also removed from the lyrics.

In addition to these overall modifications to the dataset, lines and new line characters were preserved in the song lyrics to take advantage of applying multi-headed attention on various lines of the song and then the words of the lines. Below is a summary of data cleaning operations.

- Removed 95,680 songs that had empty lyrics, which reduced the number of eligible samples to 266,557 in total

- Lowercase all words to ensure uniformity across models
- Removed all punctuations
- Removed lyric related identifiers such as “[Chorus]” and “[Verse]”
- Removed lyrics with corrupted and non-ASCII characters

4.3 Exploratory Data Analysis

After data cleaning, we performed basic exploratory analysis to understand the makeup of the dataset. First, we notice that the dataset is grossly imbalanced in which the Rock genre dominates all other categories (Figure 1). In the results section, we describe how undersampling the majority class and undersampling the minority classes to create an equally distributed training set performed worse than the imbalanced set.

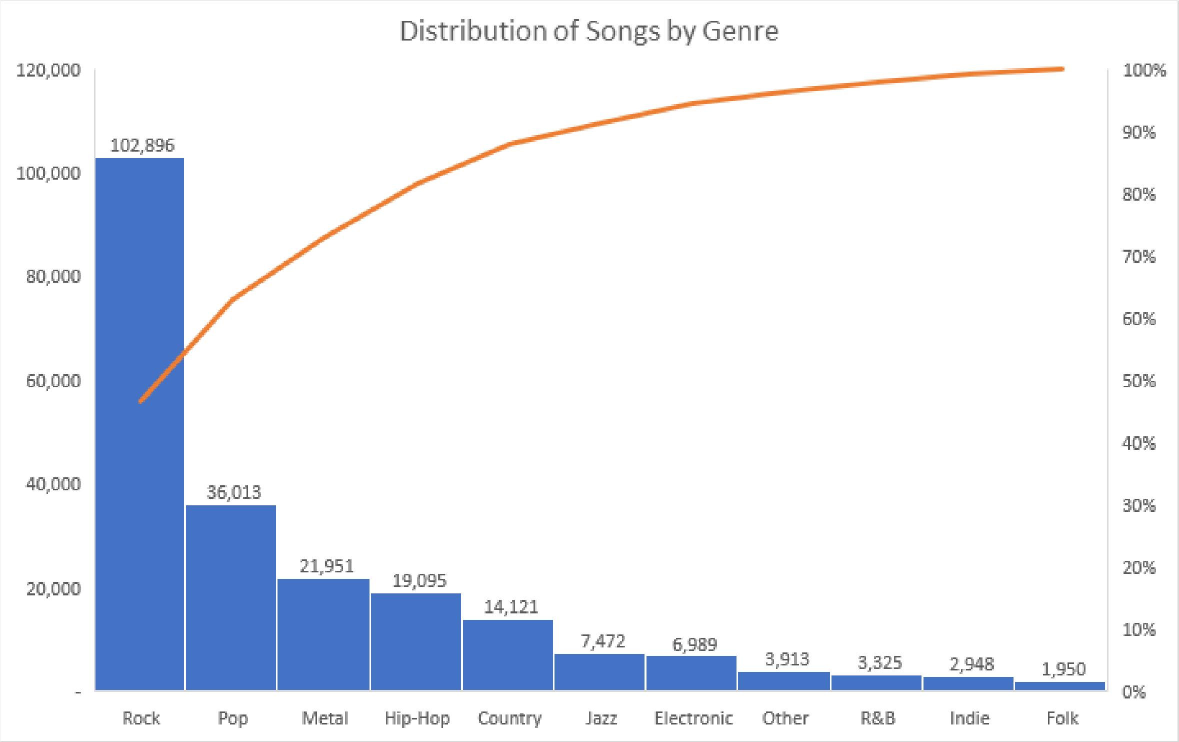


Figure 1: Distribution of songs are grossly imbalanced

Song lyrics present an unique challenge not only due to its length, but also due its repetitive nature. Unlike conventional documents, lyrics are typically composed of 1-3 verses and repeated several times in the course of a song. So while the total length of a song can be a few hundred words or more, it’s mostly due to repetition, which may not add incremental information gain while degrading model training performance. More importantly, understanding the word length of our text is vital as most of the pre-trained SOTA model architectures in our experiments have a maximum sequence length of 512 tokens. While we want to include as many “intact” lyrics as possible in our training set, doing so may not improve accuracy and add unnecessary drag to experimental velocity. The average number of words across the entire dataset was 227 and it has also been decomposed by genre in Figure 2 below. Interestingly, the average word length of Hip-Hop songs is almost twice as long as the other genres such as Pop and R&B.

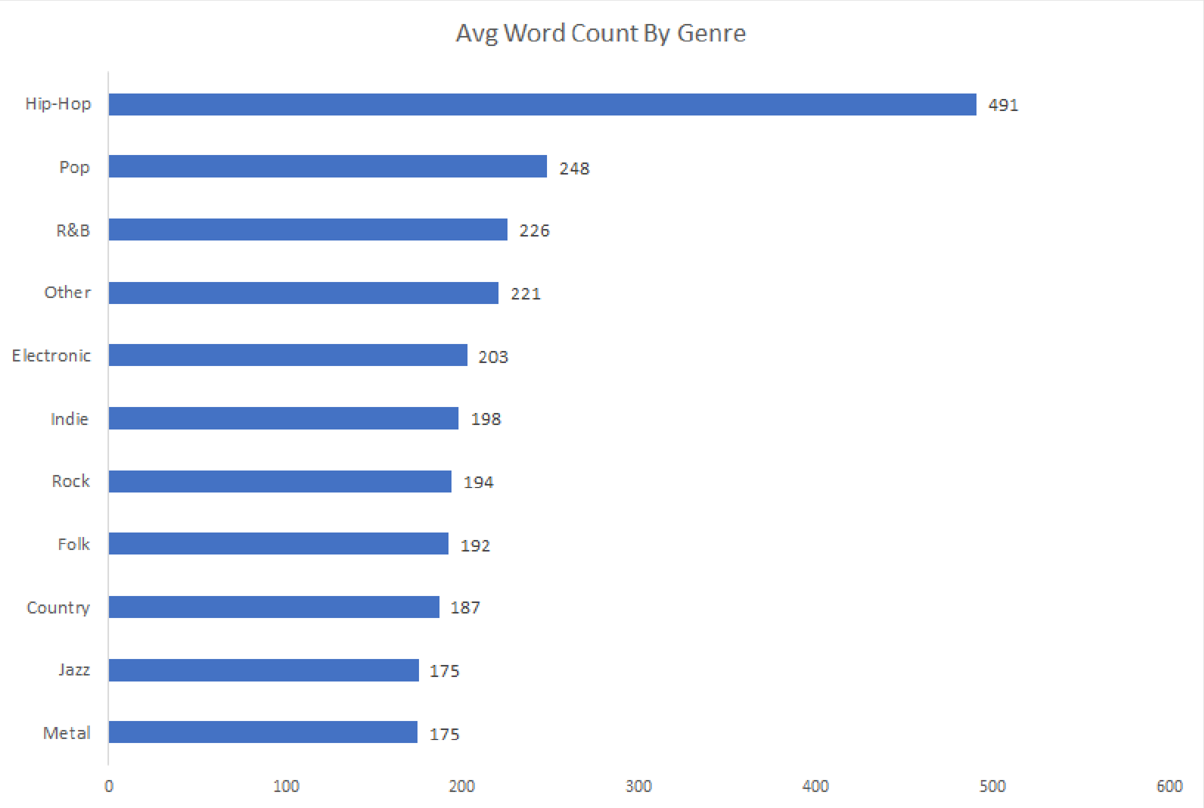


Figure 2: Distribution of word count by genre

It's also important to note that the maximum song length in the entire dataset was 6,210, which could potentially be caused by data entry or processing error. Additionally, the 99th percentile word count is 704, which means there are outliers that could degrade the model's inference accuracy. On the other hand, lyrics that contain only a few words will not have enough features for the model to extract and learn from. Doing a similar quantile analysis reveals that 99% of all songs have at least 21 words. Thus, we use these figures as lower and upper boundary conditions to further descope the dataset to 223,049 by eliminating over 7,000 songs that had lyric word count of less than 21 words or above 700. The specific distribution can be seen in Figure 3 below.

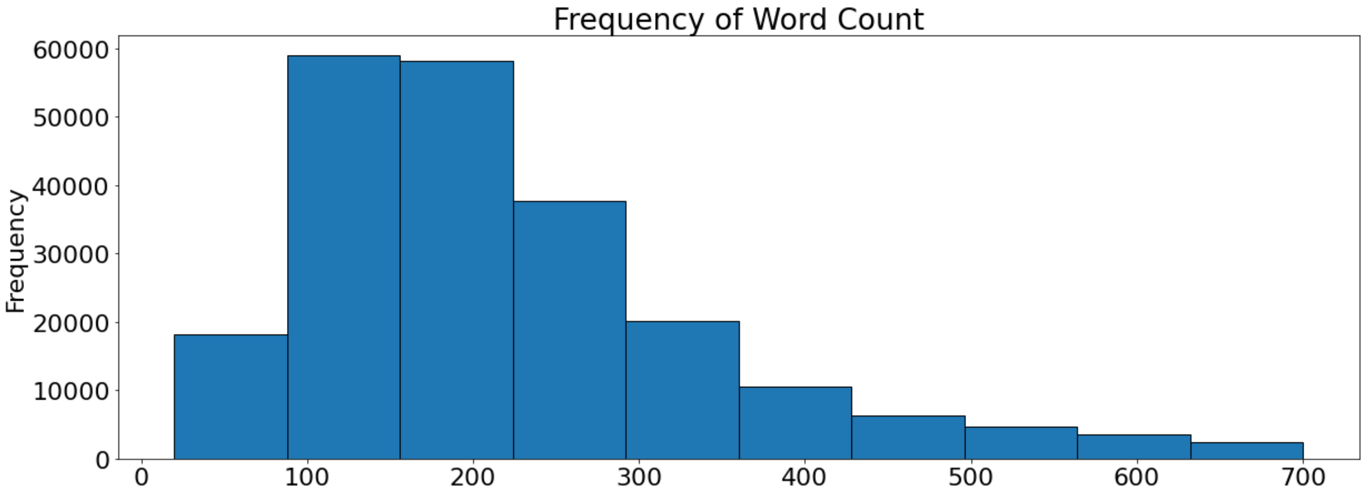


Figure 3: Frequency of word count

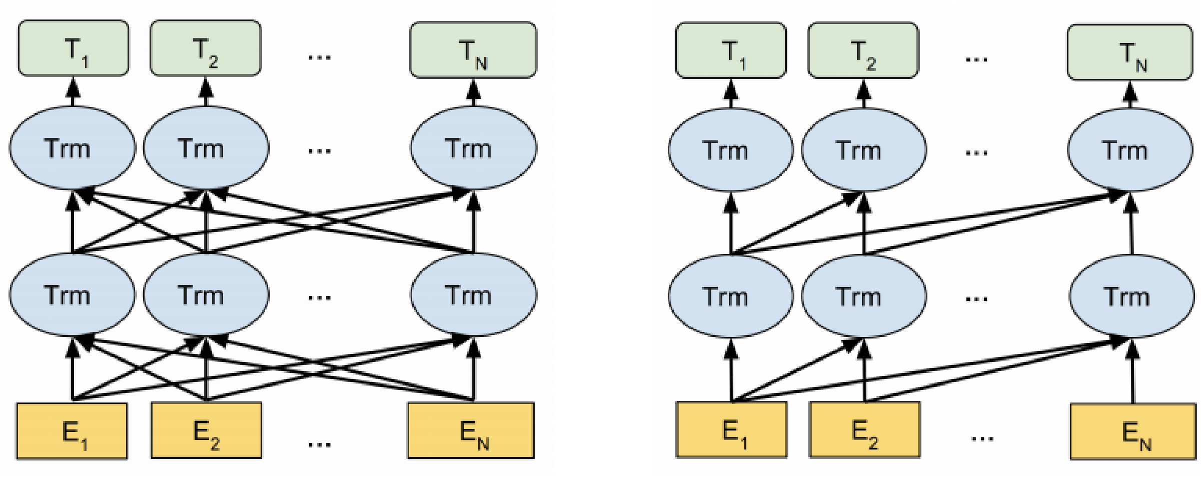
In the next section, we will briefly review the basics of the 4 SOTA pre-trained model architectures, since they have been extensively covered in other academic papers, online publications and beyond.

5. METHODS

We propose the following pre-trained SOTA language model architectures. For each fine tuned model, we evaluate models on the validation set, which produces the train loss, validation loss, weighted F1-score, accuracy, and error rate metrics. After model training, we determined the best performing model based on validation loss and applied the best model on the test set. Because the dataset is heavily unbalanced, we used the weighted F1-score as the key performance indicator. In addition, we also derived the confusion matrix to compute the accuracy per genre to gain a granular understanding model performance at the class level.

5.1 BERT (Baseline)

We implement the BERT-base model, which is an architecture developed by Devlin et al [10]. BERT has shown significant improvement over existing architectures on sequential classification tasks, obtaining state-of-the-art results on the GLUE benchmark, MultiNLI accuracy, and SQuAD v1.1 F1-score. BERT is based on the OpenAI GPT in the stacking of transformers, but uses bidirectional ones instead of only left-to-right, hence the bi-directional aspect of the name. The figures below demonstrate this distinction.



(a) BERT

(b) OpenAI GPT

Figure 4: BERT uses bidirectional transformers, as opposed to GPT which uses only left-to-right

Our implementation on BERT is based on the pretrained models provided by HuggingFace, which comes with the added linear layer after the pooled output to perform classification tasks (BertForSequenceClassification). We further fine-tune BERT by training it on the song lyrics dataset. Due to GPU memory constraints, we use BERT base uncased, which has 12 layers of transformer blocks, 12 attention heads and 110 million parameters with uncased vocabulary. Additionally, BERT and other SOTA models require specific sentence structure and formatting. We accomplish this through the BertTokenizer class by HuggingFace, which performs the following:

1. Add special tokens to the start and end of each sentence [CLS]
2. Pad & truncate all sentences to a single constant length [PAD]
3. Explicitly differentiate real tokens from padding tokens with the "attention mask" [0’s]

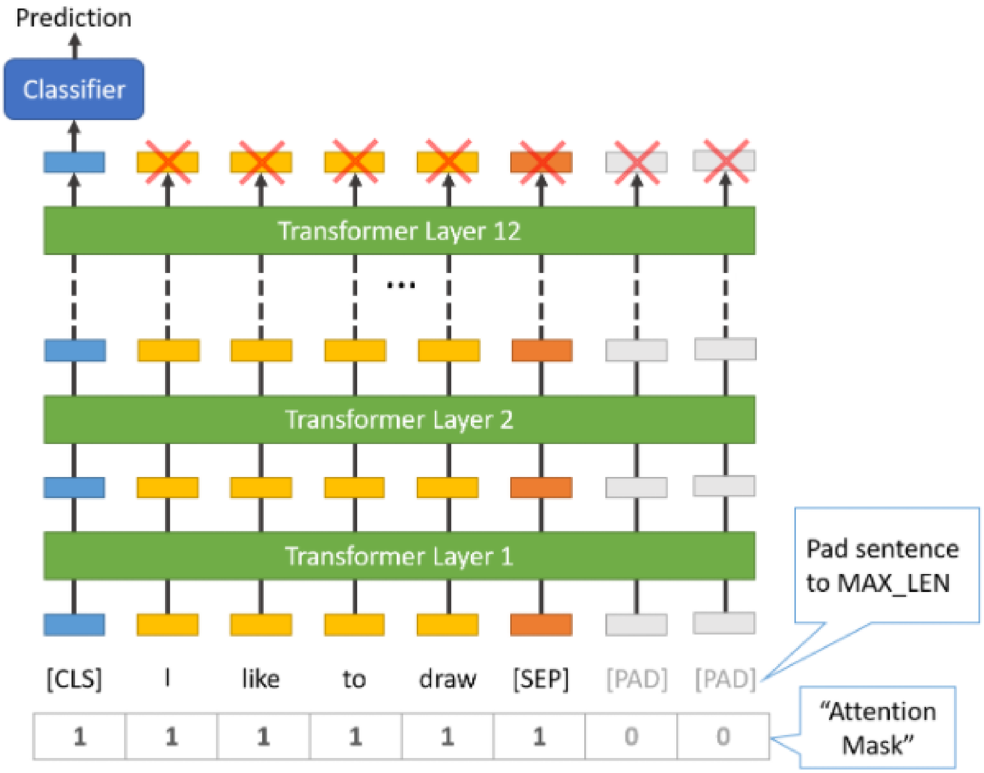


Figure 5: Illustration of BERT tokenization process with toy sentence

Because all sentences all sentences must be padded or truncated to a single, fixed length and the max length is 512 tokens, sentences longer than this will be truncated and sequences shorter than this will be padded ("attention mask").

The set of hyperparameters is as follows:

- **Train batch size**: 16
  - 32 was not used to prevent memory issues
- **Learning rate**: 2e-5 with AdamW optimizer
  - Also experimented with 3e-5, 4e-5, and 5e-5 as suggested by the Devlin et al [10]
- **Epochs**: 3
  - 4 or more epochs caused the model to overfit

5.1.1 Attempt to Correct for Class (Genre) Imbalance

Because the dataset exhibits severe imbalance, we also attempted to improve the weighted F1 score and overall accuracy with a balanced set in which each genre had the same number of samples in the train dataset. In this approach, we took the total number of samples in the train set and divided it by the number of unique genres, which produced 16,222 samples per class. For genres such as Rock, Pop, Metal, and Hip-Hop, which had more examples than the equal sample count, we applied random downsampling to achieve the target. For the other 7 classes, we applied random upsampling to achieve the 16,222 target. However, the combination of up and downsampling was unsuccessful and produced inferior results by a significant margin as shown in the results section of the report. As such, we abandoned the approach for other models.

5.2 RoBERTa

We also experiment with the RoBERTa-base model [12], which is an extension of BERT with changes to the pretraining procedure that was developed by Facebook AI. The specific modifications include more data, training with bigger batch sizes and longer sequences, larger byte-level BPE vocabulary, and the removal of the Next Sentence Prediction (NSP) objective from the training procedure. As a result of the optimization and modification mentioned above, RoBERTA outperformed both BERT and XLNet on 5 GLUE benchmark results.

We conducted a series of experiments using different combinations of hyperparameters using the model provided by HuggingFace. From the experiments, the following set of hyperparameters were determined to produce the optimal weighted F1-score on the test set:

- **Train batch size**: 16
- **Learning rate**: 3e-5 with AdamW optimizer
  - Also experimented with 1e-5, 2e-5, 4e-5
- **Epochs**: 3

In an attempt to further improve the RoBERTa experiment, we also employed techniques from the Universal Language Model Fine-tuning for Text Classification paper [13] in an attempt to further improve our RoBERTa model. Specifically, we used discriminative fine tuning with gradual layer unfreezing as well as slanted triangular learning rates to determine the learning rate to train each subsequent layer. We then train earlier layers with lower learning rates and then train later layers with higher learning rates. The rationale behind this technique is to not drastically alter the pretrained weights except for minuscule amounts in earlier layers and to be more aggressive with teaching the layers near the classification head. Furthermore, discriminative learning allows us to train the model layer by layer with different learning rates, but it also improves the F1 score while dramatically decreasing the training time.

With discriminative learning, we used the following procedure to train the model with progressive layer unfreezing:

1. Create and split the model into 14 layers (1 embedding, 12 transformer, 1 classifier)
2. Freeze all layers except for the classifier head
3. Determine optimal learning rate using fastai’s learning rate finder where the loss had the steepest descent
4. Train the model for 2 epochs
5. Save and load first cycle training
6. Unfreeze the last 2 layers (classifier and 1 transformer)
7. Determine optimal learning rate using the learning rate finder. In this and subsequent training cycles, we create separate learning rate for each layer group
8. Train the model for 2 epochs

9. Repeat steps 5 through 8 iteratively until all layers have been trained

As seen in the next section, this approach produced superior accuracy and F1-score than training the entire model via multiple epochs.

5.3 ALBERT

The third model we experiment with is ALBERT-base-v2 [14] , a "lite" version of BERT that was developed by Google and Toyota that has achieved state-of-the-art results on 3 NLU benchmarks: GLUE, RACE, and SQuAD 2.0. Because SOTA language models bag hundreds of millions or even billions of parameters, attempts to scale such models will be restricted by the memory limitations of compute hardware such as GPUs or TPUs. Thus, the motivation behind ALBERT is to implement parameter reduction techniques that could reduce the size of models while not degrading performance. Specifically, there are the 3 main modifications present in ALBERT:

1. **Factorization of the embedding matrix**: In the BERT model and its improvements such as XLNet and RoBERTa, the input layer embeddings and hidden layer embeddings have the same size. However, ALBERT isolates the size of the hidden layers from the size of vocabulary embeddings by projecting one-hot vectors into a lower dimensional embedding space and then to the hidden space, which made it easier to increase the hidden layer size without significantly increasing the parameter size of the vocabulary embeddings. This step leads to a reduction in parameters by 80% with a minor drop in performance when compared to BERT.
2. **Cross-layer parameter sharing**: Parameter sharing between different layers of the model improves efficiency and decreases redundancy. The authors proposed that since BERT, XLNet, and RoBERTa have encoder layers stacked on top of one another, this causes the model to learn similar operations on different layers. As a result, the ALBERT-large model has about 9-18x fewer parameters compared to its BERT counterpart (base and large, respectively).
3. **Inter-sentence coherence loss**: Similar to the BERT, ALBERT also used Masked Language model in training. However, Instead of using NSP loss, ALBERT used a new loss called Sentence Order Prediction (SOP). The disadvantage of this loss is that it checks for coherence as well as the topic to identify the next sentence.

For our experiments, we use the ALBERT-base-v2 provided by HuggingFace, which has been tweaked with no dropout, additional training data, and longer training time that has resulted in across the board improvement in various NLP benchmarks [14].

For the ALBERT experiments, the set of hyperparameters is as follows:

- **Train batch size**: 16
- **Learning rate**: 2e-5 with AdamW optimizer
- **Epochs**: 3

5.4 XLNet

The final model we experiment with is XLNet, which before ALBERT, achieved state-of-the-art results on multiple NLP tasks including text classification, question-answering,, natural language inference, sentiment analysis, and document ranking [15]. More specifically, it outperformed BERT on 20 tasks. XLNet is a permutation-based auto-regressive language model where all tokens are predicted but in random order. This is in contrast to BERT's masked language model where only the masked tokens are predicted. Additionally, it also stands in contrast with traditional language models where all tokens are predicted in sequential instead of random order. This approach can help the model learn bidirectional relationships and therefore better handles dependencies and relations between words. In addition, unlike the previously discussed models, XLNet is one of the few models that has no sequence length limit.

For the XLNet experiments, the set of hyperparameters is as follows:

- **Train batch size**: 16
- **Learning rate**: 2e-5 with AdamW optimizer
- **Epochs**: 3

6. RESULTS AND DISCUSSION

6.1 Training and Validation

For each model, we experiment with different combinations of hyperparameters, especially with the learning rate as it has been shown to be one of the most important hyperparameters when configuring neural networks [18]. Moreover, we held the number of training epochs (3) and batch size (16) constant across all models while manipulating and the learning rate. Specifically, validation loss, weighted F1 score, and accuracy can be illustrated in Figure 6-8 below.

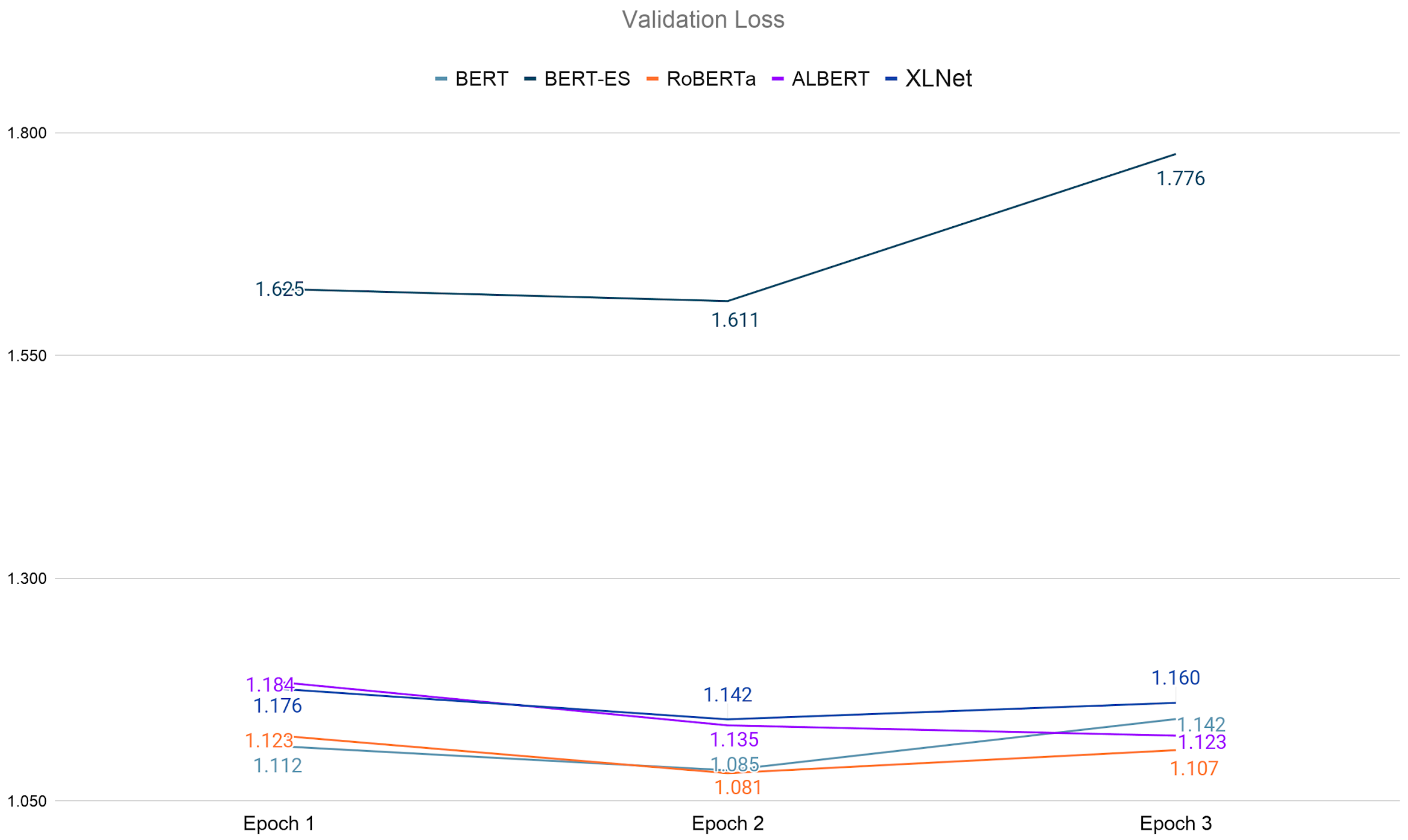


Figure 6: Loss on validation set per epoch

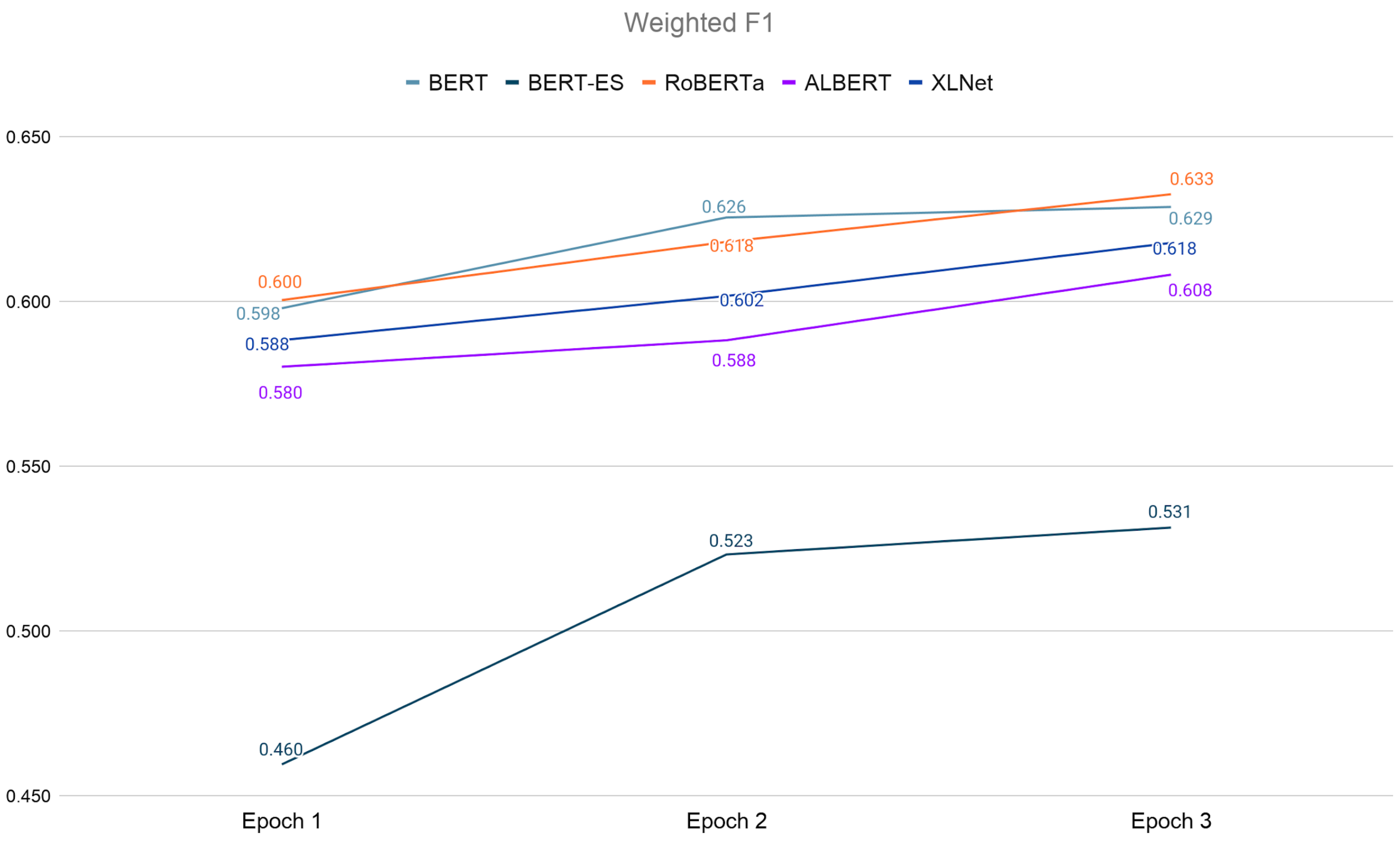


Figure 7: Weighted F1 on validation set per epoch

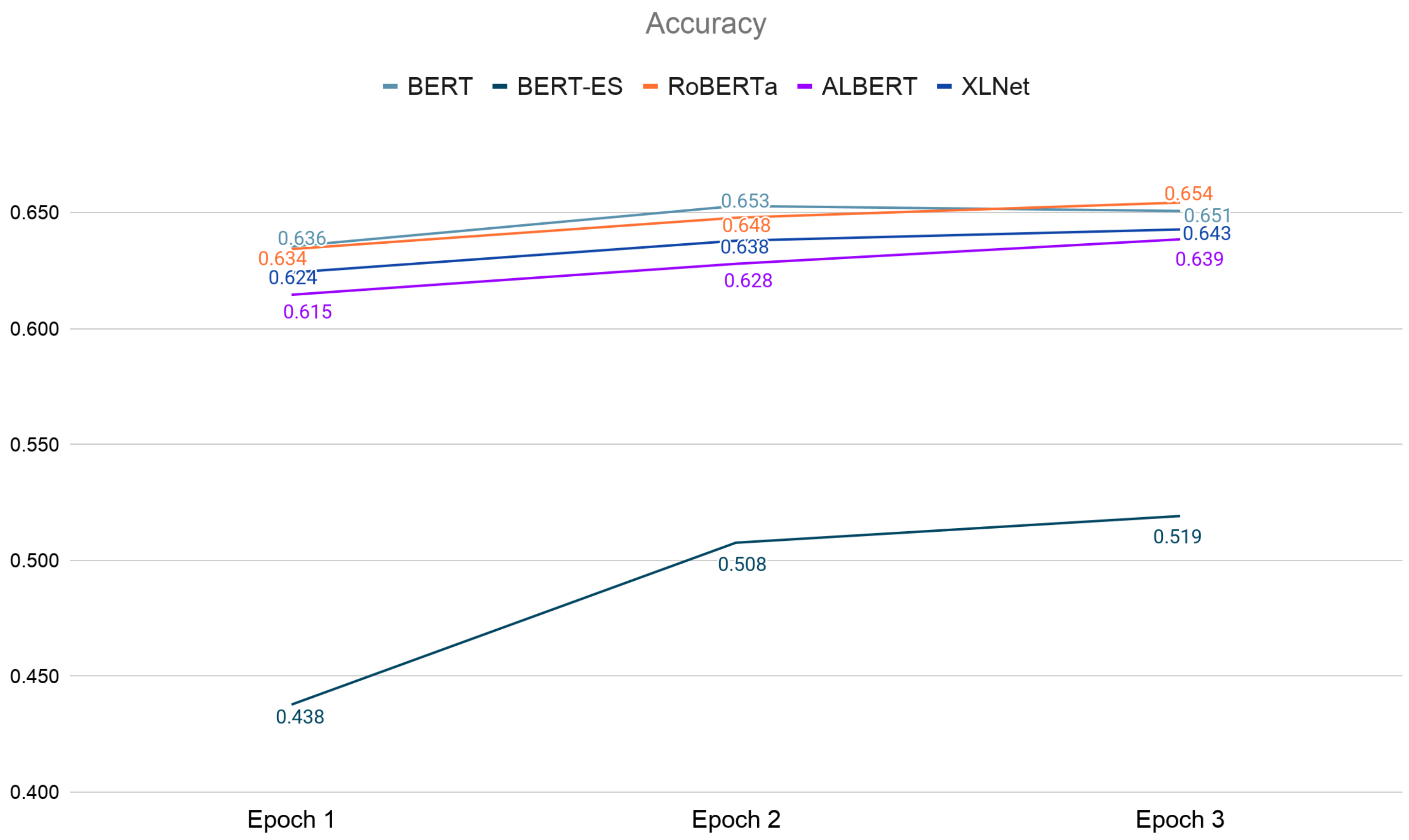


Figure 8: Accuracy on validation set per epoch

In addition to the 4 base models, we also train 2 model variants:

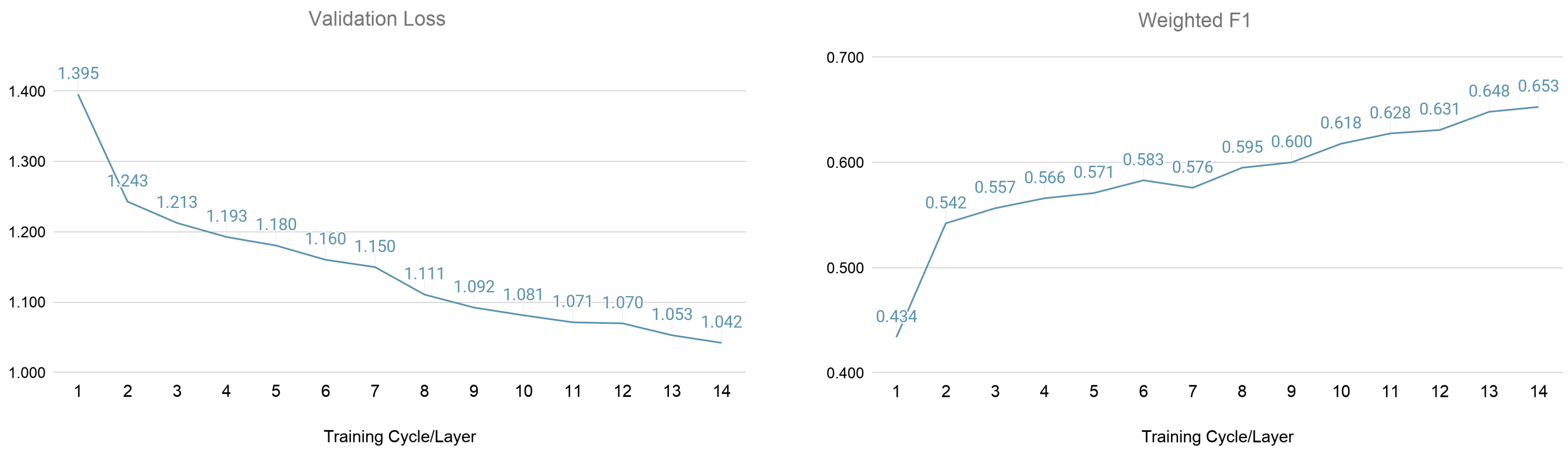
1. BERT with equal number of samples per genre in the train set (BERT-ES)
2. RoBERTa using discriminative fine tuning, gradual layer unfreezing and slanted triangular learning rates (RoBERTA-DLT); adoption of training techniques from Universal Language Model Fine-tuning for Text Classification paper [13]

Out of the 4 base models, RoBERTa produced the best F1 score while the baseline BERT model had the highest validation loss. While RoBERTa had 144 GB training data than BERT, it only led to miniscule improvements in F1. However, its validation produced significantly superior scores than BERT and 2 other models. Surprisingly, ALBERT performed the worst out of the 4 models even though it has the same number of layers, hidden dimensions, attention heads, and pre-training data size as BERT (12, 768, 12, 16 GB, respectively). One potential root cause of ALBERT's poor performance relative to BERT is its reduced parameters;

BERT-base has 108 million where ALBERT only has 12 million trainable parameters. However, it's validation loss is slightly lower than BERT, which could be due to the chance (we avoided k-fold cross-validation due to training time and GPU memory constraints).

The BERT-ES variant performed poorly across all metrics, which prompted us to discard the even sampling technique as a viable approach to address class imbalance in the hope of increasing classification accuracy. It seems this technique drastically accelerated overfitting where the training loss declined much more prominently relative to non-even sampling experiments, but it performed extremely poorly on the validation set.

Additionally, because RoBERTA-DLT involves training different layers using different learning rates (instead of a single learning rate), its training process is much different than the previous models. Figure 10 below illustrates the per training layer or cycle result (due to GPU constraints, we only trained 1 epoch per cycle). In general, we observe that RoBERTA-DFT to be the best model on the validation set. Next, we review actual predictions on the test set.



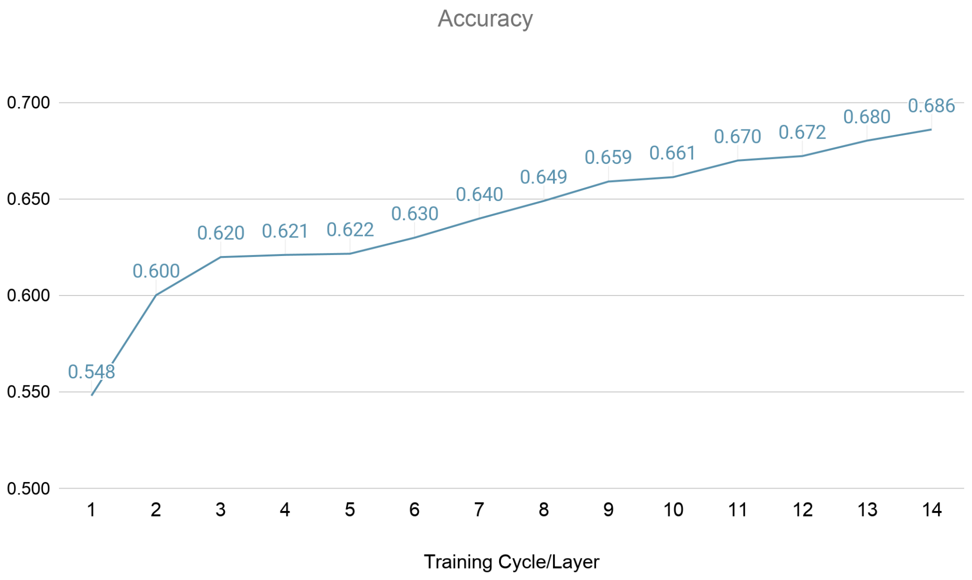


Figure 10-12: Validation loss, Weighted F1 and Accuracy per training cycle/layer

6.2 Predictions

Table 4 shows the precision, recall and weighted F1-score of the models and their variations we experimented with. Out of the 4 baseline models, RoBERTa achieved the highest weighted F1-score, which is inline with our results in the previous section Furthermore, we were able to improve the baseline RoBERTa model even more by applying discriminative fine tuning, which achieved a 4% improvement in the F1 score over baseline RoBERTa. Additionally, using an even sampling strategy in our training set with BERT produced significantly inferior results than all other models.

Model	Precision	Recall	Weighted F1-Score	Accuracy
BERT	0.631	0.653	0.629	0.653
BERT-ES*	0.581	0.525	0.539	0.525
RoBERTa	0.629	0.654	0.632	0.654
RoBERTa-DLT**	**0.638**	**0.662**	**0.649**	**0.661**
ALBERT	0.612	0.639	0.608	0.639

XLNet	0.618	0.642	0.617	0.642

Table 1: Precision, recall and F1 score across different models; bold represents best model per metric
*Even sampling
**Discriminative layer training

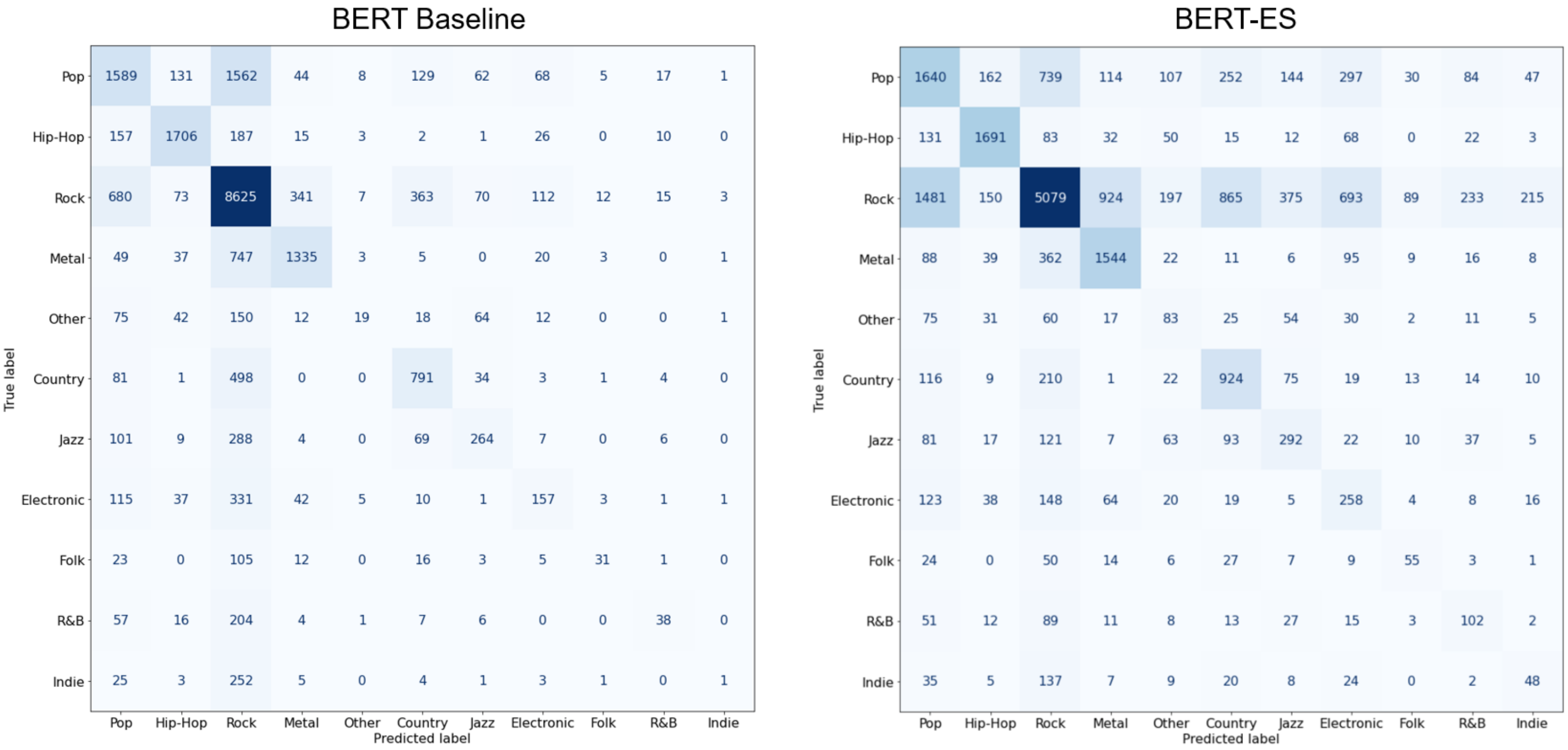
Furthermore, we also analyzed the performance of each model on a per class (genre) basis, which is summarized in table 5 below. In general, the genres that had the most example (i.e. Hip-Hop, Rock, Metal) yielded the highest accuracy, which highlights the issue with class imbalance. In a majority of the genres, RobERTa-DLT was the superior model, but in the 5 genres where samples were limited, BERT-ES was the winner. What distinguishes the performance of BERTR-ES and RoBERTa-DLT was driven primarily by the difference in training samples, especially for classes where that were on the extremes; dominant and sparse genres, respectively. For example, there is a stark contrast in the accuracy rate of Electronic, Folk, R&B, and Indie songs between the 2 models. It seems that oversampling of the minority genres dramatically increased the accuracy rate but at the expense of the majority classes such as Hip-Pop and Rock in which we had to severely downsample to ensure a balanced training set. Additionally, Indie music seems to be the genre where all of our models struggled with some returning a 0% accuracy. This could be a result of the music industry where artists that's not signed with a major record label (i.e. Sony) are considered "Indie" or independent. However, the genetic makeup of the song and its lyrics are semantically and syntactically similar to its non-Indie counterparts, but only differentiated the record label. It's also important to note that in real world applications, degrading the predictive performance of popular music genres at the cost of the less popular ones may not be prudent, since popular music genres face more traffic and user interaction.

Genre	BERT	BERT-ES*	RoBERTa	RoBERTa-DLT**	ALBERT	XLNet
Pop	43.94%	47.31%	45.44%	**49.87%**	40.87%	43.22%
Hip-Hop	80.97%	80.26%	82.11%	**83.05%**	81.68%	80.87%
Rock	83.73%	49.31%	81.86%	**82.49%**	83.82%	82.25%
Metal	60.68%	62.18%	62.23%	**63.16%**	61.27%	60.68%
Other	4.83%	**21.11%**	3.31%	4.42%	1.53%	1.02%
Country	55.98%	63.39%	61.57%	**62.90%**	49.47%	56.48%
Jazz	35.29%	37.04%	36.90%	**37.82%**	32.75%	35.03%
Electronic	22.33%	**36.70%**	23.90%	23.76%	11.24%	18.78%
Folk	15.82%	**28.06%**	15.31%	16.43%	12.25%	16.84%
R&B	11.41%	**30.63%**	12.01%	12.27%	3.90%	6.31%
Indie	0.34%	**16.27%**	0.00%	0.86%	0.00%	0.00%

Overall	65.30%	52.53%	65.40%	**66.10%**	63.90%	64.20%

Table 2: Accuracy per genre by different models; bold represents best model for each genre
*Even sampling
**Discriminative layer training

We also present the confusion matrix for each model that we experimented with.



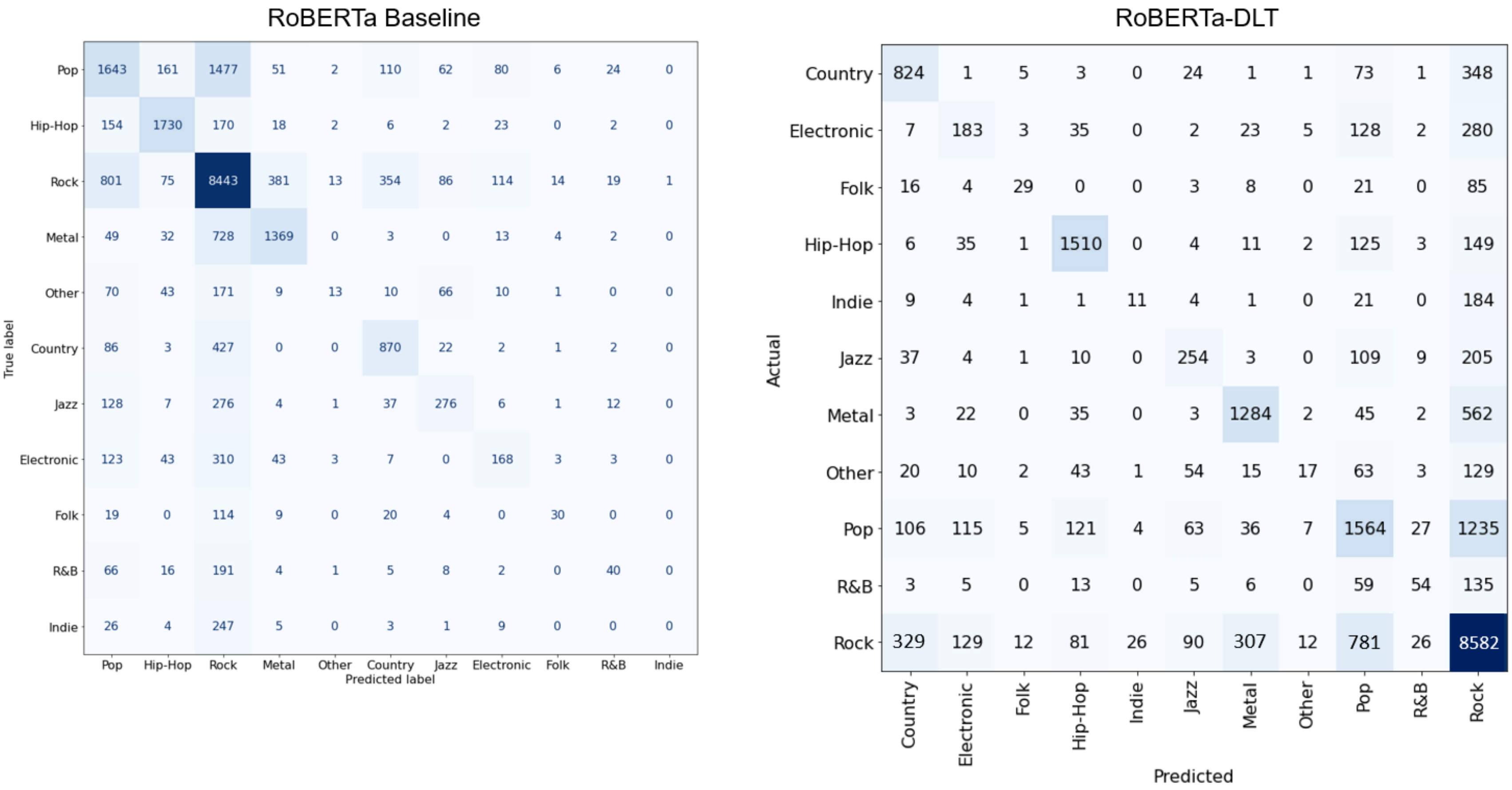
BERT Baseline

True label \ Predicted label	Pop	Hip-Hop	Rock	Metal	Other	Country	Jazz	Electronic	Folk	R&B	Indie
Pop	1589	131	1562	44	8	129	62	68	5	17	1
Hip-Hop	157	1706	187	15	3	2	1	26	0	10	0
Rock	680	73	8625	341	7	363	70	112	12	15	3
Metal	49	37	747	1335	3	5	0	20	3	0	1
Other	75	42	150	12	19	18	64	12	0	0	1
Country	81	1	498	0	0	791	34	3	1	4	0
Jazz	101	9	288	4	0	69	264	7	0	6	0
Electronic	115	37	331	42	5	10	1	157	3	1	1
Folk	23	0	105	12	0	16	3	5	31	1	0
R&B	57	16	204	4	1	7	6	0	0	38	0
Indie	25	3	252	5	0	4	1	3	1	0	1

BERT-ES

True label \ Predicted label	Pop	Hip-Hop	Rock	Metal	Other	Country	Jazz	Electronic	Folk	R&B	Indie
Pop	1640	162	739	114	107	252	144	297	30	84	47
Hip-Hop	131	1691	83	32	50	15	12	68	0	22	3
Rock	1481	150	5079	924	197	865	375	693	89	233	215
Metal	88	39	362	1544	22	11	6	95	9	16	8
Other	75	31	60	17	83	25	54	30	2	11	5
Country	116	9	210	1	22	924	75	19	13	14	10
Jazz	81	17	121	7	63	93	292	22	10	37	5
Electronic	123	38	148	64	20	19	5	258	4	8	16
Folk	24	0	50	14	6	27	7	9	55	3	1
R&B	51	12	89	11	8	13	27	15	3	102	2
Indie	35	5	137	7	9	20	8	24	0	2	48

Figure 13: Confusion matrix of BERT baseline and BERT- ES (equal sampling)



RoBERTa Baseline

True label

Predicted label

	Pop	Hip-Hop	Rock	Metal	Other	Country	Jazz	Electronic	Folk	R&B	Indie
Pop	1643	161	1477	51	2	110	62	80	6	24	0
Hip-Hop	154	1730	170	18	2	6	2	23	0	2	0
Rock	801	75	8443	381	13	354	86	114	14	19	1
Metal	49	32	728	1369	0	3	0	13	4	2	0
Other	70	43	171	9	13	10	66	10	1	0	0
Country	86	3	427	0	0	870	22	2	1	2	0
Jazz	128	7	276	4	1	37	276	6	1	12	0
Electronic	123	43	310	43	3	7	0	168	3	3	0
Folk	19	0	114	9	0	20	4	0	30	0	0
R&B	66	16	191	4	1	5	8	2	0	40	0
Indie	26	4	247	5	0	3	1	9	0	0	0

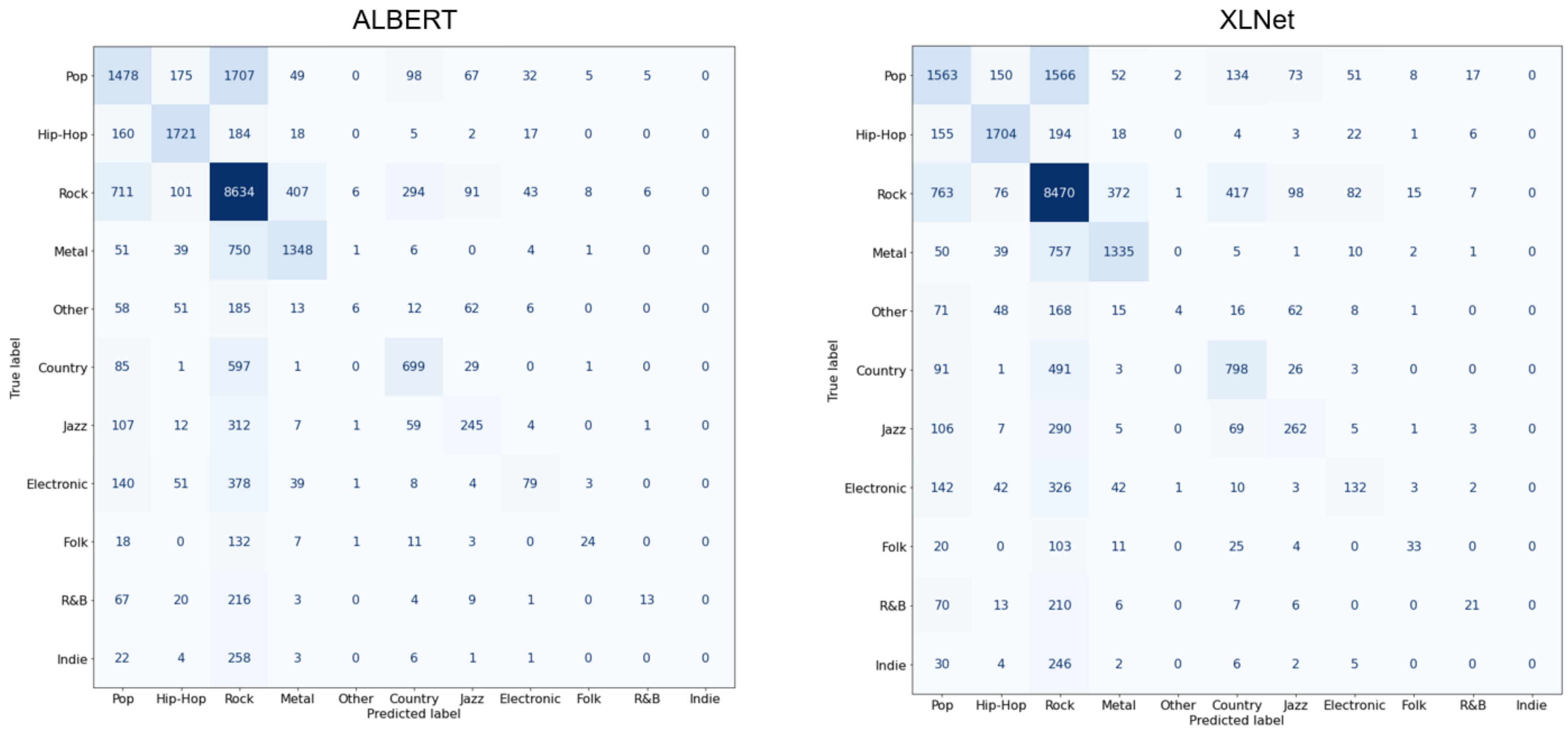
RoBERTa-DLT

Actual

Predicted

	Country	Electronic	Folk	Hip-Hop	Indie	Jazz	Metal	Other	Pop	R&B	Rock
Country	824	1	5	3	0	24	1	1	73	1	348
Electronic	7	183	3	35	0	2	23	5	128	2	280
Folk	16	4	29	0	0	3	8	0	21	0	85
Hip-Hop	6	35	1	1510	0	4	11	2	125	3	149
Indie	9	4	1	1	11	4	1	0	21	0	184
Jazz	37	4	1	10	0	254	3	0	109	9	205
Metal	3	22	0	35	0	3	1284	2	45	2	562
Other	20	10	2	43	1	54	15	17	63	3	129
Pop	106	115	5	121	4	63	36	7	1564	27	1235
R&B	3	5	0	13	0	5	6	0	59	54	135
Rock	329	129	12	81	26	90	307	12	781	26	8582

Figure 14: Confusion matrix of RoBERTa baseline and RoBERTa-DLT(discriminative layer training)



ALBERT

	Pop	Hip-Hop	Rock	Metal	Other	Country	Jazz	Electronic	Folk	R&B	Indie
Pop	1478	175	1707	49	0	98	67	32	5	5	0
Hip-Hop	160	1721	184	18	0	5	2	17	0	0	0
Rock	711	101	8634	407	6	294	91	43	8	6	0
Metal	51	39	750	1348	1	6	0	4	1	0	0
Other	58	51	185	13	6	12	62	6	0	0	0
Country	85	1	597	1	0	699	29	0	1	0	0
Jazz	107	12	312	7	1	59	245	4	0	1	0
Electronic	140	51	378	39	1	8	4	79	3	0	0
Folk	18	0	132	7	1	11	3	0	24	0	0
R&B	67	20	216	3	0	4	9	1	0	13	0
Indie	22	4	258	3	0	6	1	1	0	0	0

XLNet

	Pop	Hip-Hop	Rock	Metal	Other	Country	Jazz	Electronic	Folk	R&B	Indie
Pop	1563	150	1566	52	2	134	73	51	8	17	0
Hip-Hop	155	1704	194	18	0	4	3	22	1	6	0
Rock	763	76	8470	372	1	417	98	82	15	7	0
Metal	50	39	757	1335	0	5	1	10	2	1	0
Other	71	48	168	15	4	16	62	8	1	0	0
Country	91	1	491	3	0	798	26	3	0	0	0
Jazz	106	7	290	5	0	69	262	5	1	3	0
Electronic	142	42	326	42	1	10	3	132	3	2	0
Folk	20	0	103	11	0	25	4	0	33	0	0
R&B	70	13	210	6	0	7	6	0	0	21	0
Indie	30	4	246	2	0	6	2	5	0	0	0

Figure 15: Confusion matrix of ALBERT and XLNet

7. FUTURE WORK

In this paper, we experimented with four SOTA language models and two variants and found RoBERTa to perform the best. However, none of the scores were significantly higher over each other nor well enough to be considered to be production ready. Additionally, genres that are less popular (i.e. Indie, Folk) in reality will always be underrepresented because the number of artists producing those types of music are relatively fewer when compared with mainstream genres such as Rock and Pop. Thus, to produce a state-of-the-art classifier, it's evident that the classifier must take into account more than just the lyrical content of the song. Further research could then look into employing SOTA language models to both audio and symbolic data, and combining with the lyrics to build a stronger classifier. Moreover, because we were constrained by GPU memory (Tesla P100-PCIE-16GB), we could not experiment with larger models that possess much more parameters and attention heads, which we believe would have improved the classification performance of each model due to both the length and the repeating verse structure of song lyrics.

8. REFERENCES

[1] M. McKinney and J. Breebaart. Feature for audio and music classification. In ISMIR, pages 151–158, 2003.

[2] C. McKay, J. A. Burgoyne, J. Hockman, J. BL Smith, G. Vigliensoni, and I. Fujinaga. Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features. In ISMIR, pages 213–218, 2010.

[3] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In ICASSP, pages 6959–6963. IEEE, 2014.

[4] Mitja Lustrek. Overview of automatic genre identification 1. Technical report, Jozef Stefan Institute, Department of Intelligent Systems, Jamova 39, 1000 Ljubljana, Slovenia, 01 2007.

[5] Michael Fell, Caroline Sporleder. 2014. Lyrics-based analysis and classification of music. In COLING, volume 2014, pages 620–631.

[6] Yajie Hu and Mitsunori Ogihara. Genre classification for million song dataset using confidence-based classifiers combination. In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’12, pages 1083–1084, New York, NY, USA, 2012. ACM.

[7] Vedrana Vidulin, Mitja Lustrek, and Matjaz Gams. Training a genre classifier for automatic classification of web pages. Journal of computing and information technology, 15(4):305–311, 2007.

[8] Canicatti, A. Song genre classification via lyric text mining. International Conference on Data Mining, pp. 44–49, 2016.

[9] Tsaptsinos, A. Lyrics-based music genre classification using a hierarchical attention network. arXiv:1707.04678, 2017.

[10] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. 2018. “BERT: Pre-training of deep bidirectional transformers for language understanding”. arXiv preprint arXiv:1810.04805.

[11] Radford, A., Wu, J., Child, R., Luan, D., Amodie, D., and Sutskever, I. 2018. “Language Models are Unsupervised Multitask Learners”.

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

[13] Jeremy Howard, Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. arXiv preprint arXiv:1801.06146.

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

[15] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:1906.08237.