Comparing Fine-Tuning and Feature-Extraction Approaches for Genre Annotation on the Brown Corpus

Steven Li
University of Illinois Urbana-Champaign
LING 413
sl212@illinois.edu

Abstract

This paper studies genre classification on the Brown Corpus through the use of multiple deep learning architectures. We compare a fine-tuned BERT model against feature-extraction approaches where BERT embeddings are fed into simpler classifiers (a NN, a CNN, and a LSTM) and a majority-class baseline. The fine-tuned BERT model outperforms the feature-extraction methods, achieving a weighted-average F1-score of 0.60, showing the strengths of pretraining as well as its more complex architecture. The model's performance also exposes linguistic insights into the characteristics of the 1961 corpus' genres.

1 Introduction

Linguistic annotation converts raw data into structured information, which helps expose implicit linguistic features. In this paper, we approach genre classification as a form of automated linguistic annotation.

The focus of this study is the Brown University Standard Corpus of Present-Day American English which a collection of texts from 1961 (Kučera and Francis, 1967). The core idea is a comparative analysis of applying different models to assign genre labels to the sentences from this corpus.

To evaluate these strategies, we benchmark four architectures against a majority-class baseline. In particular, we focus on the differences between fine-tuning and feature-extraction approaches. Beyond the performance metrics, these results also show us characteristics of the Brown Corpus genres.

2 Background

2.1 The Brown Corpus

The Brown Corpus contains about one million American English words and was published in 1961 (Kučera and Francis, 1967). It has 500 text samples, each with approximately 2000 words. Most importantly, it has balanced data across 15 genres, making it an ideal dataset for training and evaluating genre classification models.

2.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a powerful pre-trained language model (Devlin et al., 2019). We explore two main ideas for applying BERT to classification:

- **Fine-Tuning**: In this paradigm, a classification layer is appended to the pre-trained BERT model. Then, the entire network's weights, including the original BERT parameters, are updated during training. This allows the model to adapt its entire language representations.
- **Feature Extraction**: In this paradigm, the pre-trained BERT model's weights are not changed and it is used as a frozen feature extractor to convert text into semantic embeddings. These embeddings are then fed into separate neural networks which are trained from scratch.

3 Methodology

This project's implementation is divided into two major sections: data preprocessing and the implementation of five classification systems.

3.1 Preprocessing

3.1.1 Corpus Loading and Partitioning

The Brown Corpus, containing 57,340 sentences, was loaded using the NLTK library (Bird et al., 2009). To ensure that the model is able to generalize, the dataset was partitioned into training (80%), validation (10%), and test (10%) sets using a fixed random seed of 42.

3.1.2 Tokenization

Afterwards, the data is tokenized, so the raw text is converted into a format suitable for BERT. We used the bert-base-uncased AutoTokenizer from the Transformers Hugging Face library (Wolf et al., 2020).

This tokenizer uses a WordPiece algorithm, which breaks words into subword units based on a pre-trained vocabulary. For example, a word like "stylistically" might become ‘["style", "##istic", "##ally"]‘. This approach allows the model to handle a large vocabulary and process out-of-vocabulary words by representing them as sequences of known subwords. Each sentence is then formatted with special tokens: [CLS] for the start and [SEP] for the end.

To handle different sentence lengths across the dataset, all sequences were standardized to a fixed length. We set a max length of 512 tokens for the fine-tuning experiment and 128 for the feature-extraction experiments (for faster results). Sentences longer than this limit are truncated, while shorter ones are padded with [PAD] tokens. The tokenizer generates three inputs for the models:

1. input_ids: A sequence of integer IDs corresponding to the tokens in the vocabulary.

2. attention_mask: A binary mask consisting of 1s and 0s indicating which tokens are real (1) and which are padding (0). This ensures that BERT’s self-attention mechanism does not utilize the padding tokens.

3. token_type_ids: A sequence of intengers indicating which sentence a token belongs to.

Finally, the string genre labels were class-encoded into integer indices.

3.2 Modeling Architectures

3.2.1 Fine-Tuned BERT

This model uses the standard AutoModelForSequenceClassification architecture from Hugging Face. It consists of the bert-base-uncased model (a 12-layer Transformer encoder) followed by a dropout layer (p=0.5) and a single linear classification layer that maps the 768-dimensional output of the BERT model to a 15-dimensional vector (one for each genre). During training, gradients are backpropagated through the entire network, adjusting all parameters of BERT. The model was trained for 3 epochs with a learning rate of 2×10^{-5} using the AdamW optimizer (Figure 1).

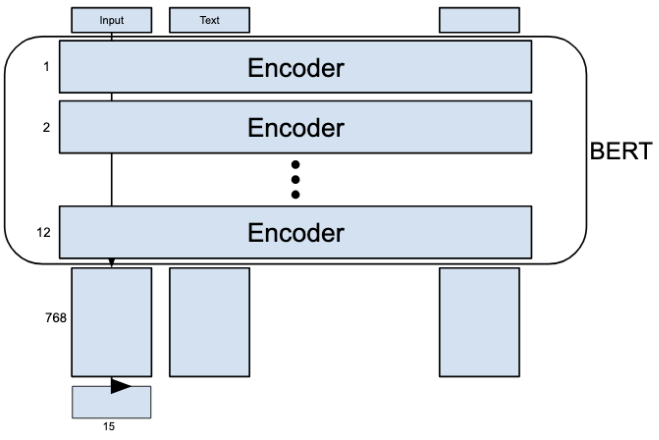

Figure 1: BERT Model Architecture.

3.2.2 Feature-Extraction Models

Next, we trained three systems using the pre-trained bert-base-uncased model as a frozen feature extractor. For a given input sentence, it produces a sequence of 768-dimensional embedding vectors. These static embeddings are then fed into the following neural net classifiers.

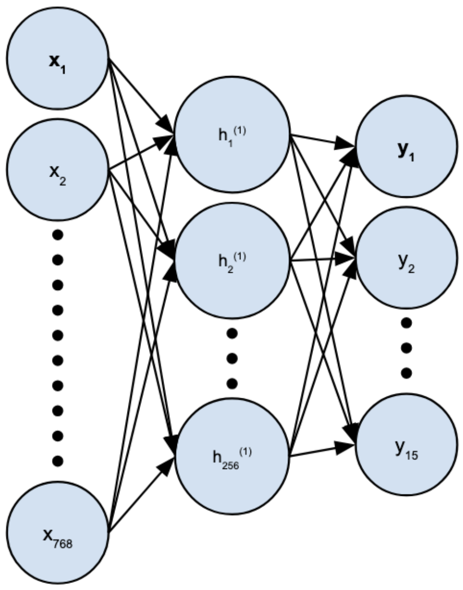

Figure 2: NN Model Architecture.

Simple Neural Network (NN) This model serves as a baseline for the feature-extraction approach.

- **Architecture**: The sequence of token embeddings from BERT is first averaged across the words, resulting in a single 768-dimensional vector as input. This vector is then passed through a hidden layer with 256 neurons and a ReLU activation function, followed by a dropout layer (p=0.5) and a final output layer with 15 neurons for classification (Figure 2).

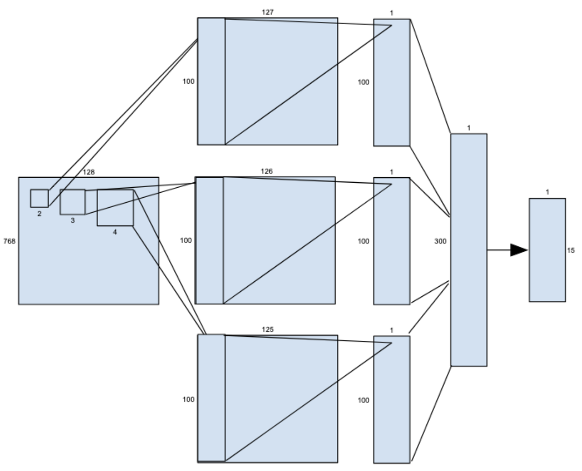

Figure 3: CNN Model Architecture.

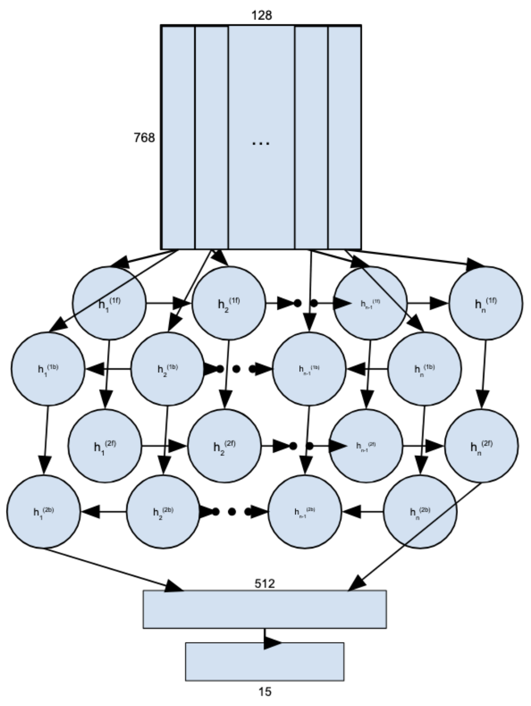

Figure 4: LSTM Model Architecture.

Convolutional Neural Network (CNN) This model is designed to identify local patterns (like n-grams) within the sequence of BERT embeddings.

- **Architecture**: The sequence of embeddings is treated as a 1D image where the channels (normally red, green, and blue) are the 768 embedding dimensions. It employs three parallel 1D convolutional layers with filter sizes of 2, 3, and 4, and 100 filters each. These filters slide over the sequence to detect patterns across adjacent embeddings. A ReLU activation is applied, followed by a maxpool layer for each filter bank, which captures the most important feature detected. The outputs of the pooling layers are concatenated, passed through a dropout layer (p=0.5), and finally to a fully connected layer for classification (Figure 3).

Bidirectional LSTM (BiLSTM) This model is designed to capture sequential dependencies and long-range context across the entire sentence.

- **Architecture**: The sequence of BERT token embeddings is fed into a 2-layer Bidirectional LSTM with a hidden dimension of 256. Being Bidirectional allows the LSTM to processes the sequence from left-to-right and right-to-left simultaneously. The final hidden states from both the forward and backward passes are concatenated to form an improved representation of the sequence. This representation is regularized with dropout (p=0.5) and then passed to a final linear layer for classification (Figure 4).

3.2.3 Majority-Class Baseline

This is used as a baseline to compare the other models to. It will always predicts the most frequent class in the training data.

4 Results

The evaluation clearly shows the strengths of the fine-tuning for this classification task.

As shown in Table 1, the Fine-Tuned BERT model achieved the highest performance with a weighted-average F1-score of 0.60. This significantly outperforms the feature-extraction models, which scored between 0.44 and 0.47. Among the feature-based models, the CNN and Bidirectional LSTM showed slight advantage over the simple NN, showing that their specialized architectures for capturing local and sequential patterns offer some benefits. However, all neural network models performed significantly better than the majority-class baseline (F1=0.04).

Model	Precision	Recall	F1-Score
Fine-Tuned BERT	**0.60**	**0.60**	**0.60**
TextCNN + Emb.	0.51	0.48	0.47
BiLSTM + Emb.	0.49	0.47	0.47
Simple NN + Emb.	0.47	0.45	0.44
Majority Baseline	0.02	0.14	0.04

Table 1: Overall weighted-average performance of all systems. "Emb." means that static BERT embeddings were used.

Genre Name	Precision	Recall	F1-Score
adventure	0.49	0.58	0.53
belles_lettres	0.57	0.63	0.60
editorial	0.41	0.38	0.39
fiction	0.54	0.46	0.49
government	0.72	0.75	0.73
hobbies	0.76	0.71	0.73
humor	0.34	0.23	0.28
learned	0.79	0.76	0.78
lore	0.55	0.59	0.57
mystery	0.57	0.54	0.55
news	0.67	0.66	0.67
religion	0.68	0.59	0.63
reviews	0.62	0.52	0.57
romance	0.44	0.52	0.48
science_fiction	0.69	0.41	0.52

Table 2: Per-genre performance of the Fine-Tuned BERT model.

The per-genre performance of the BERT model (Table 2) shows that it excels at labeling genres with more unique vocabularies like "learned" (F1=0.78) and "government" (F1=0.73), while it struggled with genres with more variation like "humor" (F1=0.28) and "editorial" (F1=0.39).

5 Discussion

5.1 The Strengths of Fine-Tuning

One of the key findings is the significant gap in performance between fine-tuning and feature extraction models. While the frozen BERT embeddings provide a strong baseline by allowing the neural networks to gain semantic understanding of the sentences, the embeddings are not as specialized when compared to fine-tuning. Fine-tuning allows the model to adjust BERT's attention mechanisms and internal networks to become sensitive to the subtle markers that differentiate genres, while the feature-extraction models are limited to learning patterns from the embeddings with a shallow classifier.

5.2 Linguistic Analysis of Model Results

The model's successes and failures show that some genres like "learned" and "government" texts are likely to have unique and consistent vocabularies as well as frequent syntactic patterns, which may form detectable signals for the model.

On the other hand, the model's struggle with 'humor' is expected. Humor often relies on pragmatics and world knowledge, which is not easily captured by distributional patterns. In addition, the model's struggle among fictional sub-genres suggests that sentence-level features may not be enough to differentiate them very well.

6 Conclusion

Our results show that end-to-end fine-tuning of a BERT model is superior to feature-extraction approaches, where embeddings are used along with a NN, CNN, or Bidirectional LSTM classifier. The fine-tuned model achieved a weighted F1-score of 0.60, demonstrating its ability to update its internal representations to capture subtle feature differences within the genres. The analysis of its performance also provides linguistic insights into the Brown Corpus genres and highlights the difficulty of classifying pragmatically complex categories like humor as well as semantically complex categories like the fictional subgenres.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

Jacob Devlin, Ming-Wei Chang, Lee Kenton, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Henry Kučera and W. Nelson Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing.

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.