BERT in Multiclass Classification of University Work Request Emails

Andrew Xiao
New York University
ax296@nyu.edu

Abstract

Language model pre-training and transfer learning has demonstrated potential for success on many natural language processing tasks. BERT (bidirectional encoder representation transformers) is a state-of-the-art language model trained on unlabeled text by conditioning bidirectionally. This pre-trained model can be fine-tuned with an additional layer for downstream tasks such as text classification. Email is a widely used medium for communication on the Internet. In this study, BERT was applied on the task of classifying work request emails from students and staff at New York University for automating form filling. Based on results from test data, BERT achieved an averaged F1 score of 90%.

1. Introduction

The purpose of this system is to classify emails from students and staff at New York University into categories useful for generating work request orders. Each email would be classified into only one category. There was also a need for automated form filling which would help make the process of providing assistance to different areas of the campus more efficiently and timely. A secondary objective is to help reduce mistakes made by manually reading emails. The main goal is to assist university employees in their work, while providing alerts for human intervention in cases where the application fails.

2. Previous Work

Recurrent neural networks, long short-term memory networks, and gated recurrent units have established themselves in the areas of language modeling and machine translation[5]. In particular, they are used in tasks involving sequential data[1]. However, the problem remains when these networks encounter long sequences leading to slow convergence in training and memory leakage[1]. Recurrent models generate a sequence of hidden states as a function of previous states and input at a specific position[5]. The burden of sequential computations is exacerbated by the length of the sequence and prevents parallelization across training examples[5]. There has been recent work improving this through the use of skip RNNs where states are updated as needed and is pushed towards less fewer state updates by a penalization term[1]. However, the high costs of sequential computation remains.

In recent years, the natural language processing community has observed the application of word embeddings in many tasks [6]. In particular, the authors of global vectors called GloVe established a model that is trained only on non-zero elements in a word-word co-occurence matrix, resulting in the improved capture of syntactic and semantic meaning from words. It has shown improvements in word similarity and named entity extraction tasks [3]. Another set of word vectors called ELMo consists of a group of learned functions of states in a deep bidirectional language model pretrained on a large corpus[4]. Recently, there has been another paradigm shift towards deep language representation models. Current techniques and language models are unidirectional such as in OpenAI GPT, where every token attends to previous ones in the self attention layers which would detract away from its usefulness in context aware tasks like question and answering[2]. In this regard, BERT can overcome this unidirectional constraint by applying a masked model pretraining objective where words are masked and the model is tasked with filling in the gaps of sentences with words based on context. Another unsupervised task is predicting the next sentence, which further enhances the ability to understand context[2]. A transformer is a model architecture that moves away from recurrence and uses attention mechanism and parallelization[5]. As a result, BERT has demonstrated its ability to advance the state-of-the-art on eleven NLP tasks[2].

3. Methods

3.1. The Data

The email data spanned seven years from 2013-2019 and was prepackaged in a mbox file derived from an employee's inbox in which permission was granted for the purposes

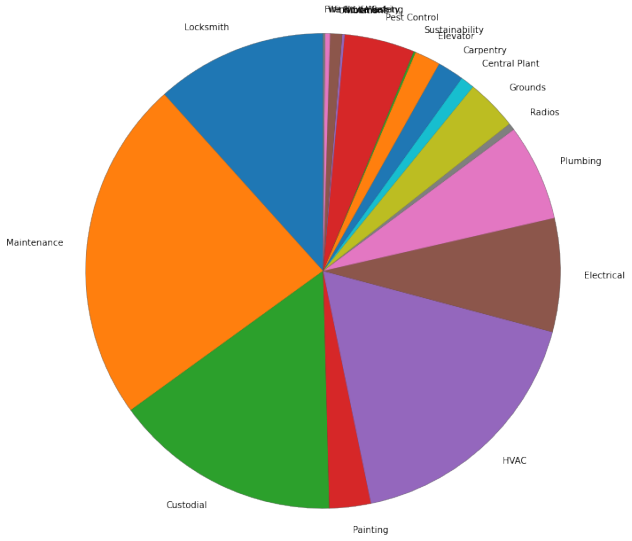

Figure 1. Distribution of Categories

of this project. The data contains no faculty or student identifiers. Preprocessing involved lower casing all text and stripping away hyperlinks, chains of characters used to delimit sections of an email, and special characters. In emails, many common salutations, greetings, and expressions of gratitude were eliminated using regular expression patterns. Some features peculiar to emails such as phrases like "forwarded message" and "on monday/mon, jan 28, 2014 at 12:00 am someone wrote:" were also stripped from the emails. Since the emails originated from a Gmail inbox, all messages belonging to the same thread possess a unique thread-id. After pre-processing, the thread-id, message-id, subject, to, from, date, and body values were extracted from the metadata and converted to a JSON file for faster operations. This file was later modified so that every thread-id was mapped to all the messages belonging to that thread.

In order to associate a true label with each thread, a database containing generated work request IDs along with their problem codes was queried. For every email thread that had any mention of a work request ID, it would be matched to the corresponding ID and problem code in the database. Upon further investigation into the distribution of categories in the threads based on the problem codes, it was observed that Pest Control (4.8%), Plumbing (6.5%), Electrical (7.7%), HVAC or Heating, Ventilation, and Air Conditioning (17.6%), Custodial (15.4%), Maintenance (23.3%), and Locksmith (11.6%) were predominantly represented as shown in Figure 1. These became the categories of interest in this project.

3.2. Model Architecture

The model architecture consists of an encoder which maps an input to a sequence of continuous representations, followed by a decoder as depicted in Figure 2[5]. The encoder is comprised of 6 layers, with each layer having 2 sub-layers, one being a multi-head self-attention component and the other a feed-forward network. The outputs from these components are normalized and are of dimension 512[5]. Similarly, the decoder is also comprised of 6 layers with an additional third sub-layer responsible for multi-head attention on the output from the encoder. Like the encoder, residual connections are applied around the sub-layers along with normalization. Furthermore, each layer in the encoder and decoder contains a fully connected feed-forward network consisting of a couple linear transformations with a ReLu activation[5].

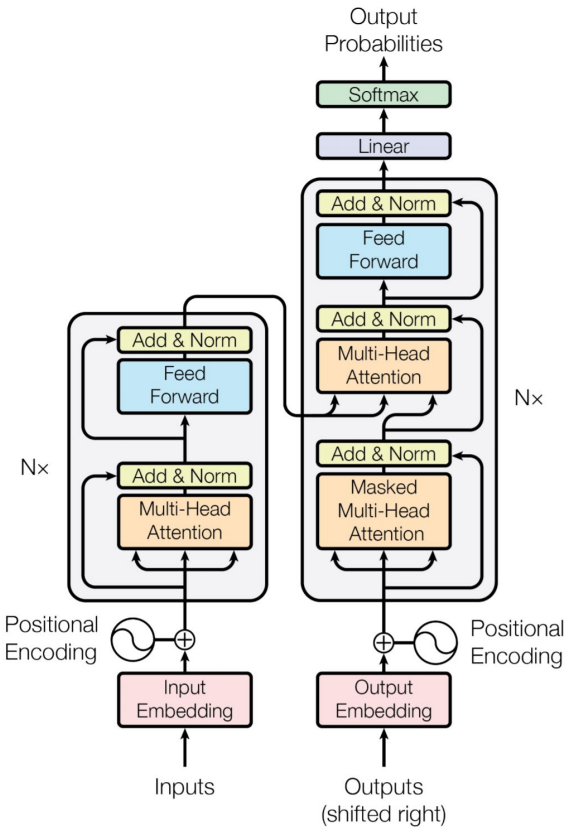

Figure 2. BERT architecture (Vaswani et. al)

3.3. Process Workflow

The workflow consists of three components: input preparation, data splitting, and data prediction (Figure 3). The Python library called transformers was used whereby the pretrained BERT model could be fine-tuned with an ad-

ditional output layer for classification. The bert-uncased model (12-layer, 768-hidden, 12-heads, 110M parameters) trained on lower-case text from the Toronto Book Corpus and Wikipedia was picked for the experiments. The preparation stage is required in order to prepare the input sequences in the format of training examples that the BERT model was pretrained on. The sentences were tokenized using WordPiece, special tokens [CLS] and [SEP] were appended to the ends of each sentence, tokens mapped to input IDs, padding was arbitrarily added up to a length of 450 due to BERT's maximum limit constraint of 512, and created attention masks for the tokens. The attention masks designate the tokens that need to be attended to versus padding tokens. The training set is comprised of 90% of the data and the remaining for the validation set. The validation set will be used for fine-tuning the model and a separate randomly picked set of 80 examples will be used for the model prediction testing. The benefits of self-attention include more computations that can be parallelized and shorter path lengths between any combinations of positions in the input sequence[5]. A point to emphasize is that a self-attention layer results in a constant number of sequential operations, whereas recurrent layers require linear time number of sequential operations[5].

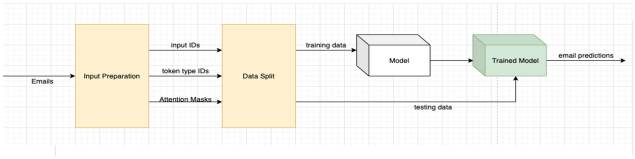

Figure 3. Workflow of Classification Mini-System

4. Experiments

4.1. Results

The model training was conducted with a batch size of 16, learning rate set to 2×10^{-5}, Adam optimizer ($\epsilon = 1 \times 10^{-8}$), and trained for 4 epochs. The total training time took 7 minutes and 23 seconds using a single GPU (Tesla P100-PCIE-16GB) from Google Colaboratory. As observed from losses in Figure 4, there does not appear to be overfitting as the validation curve decreases and plateaus at epoch 2.

4.2. Evaluation

In the testing set, the 80 random examples were split up into 5 batches, each containing 16 examples. Due to the unbalanced nature of the classes, a naive metric using accuracy would lead to an inflated view of the classifier's predictions on the majority class. Three metrics were used to evaluate the performance of the model: microaveraged F1 score, Matthew's Correlation Coefficient (MCC) scores, and Cohen's κ scores. The F1 score is defined as the harmonic mean of recall and precision as shown in Equation 1:

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{1}$$

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2}$$

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3}$$

One of the drawbacks of the F1 score is the lack of consideration for true negatives, whereas MCC takes this into account as demonstrated in Equation 2. In order to get a high MCC score, the classifier has to make correct predictions on the majority of negative and positive cases, without regard to skewness of examples among the classes. The Cohen's κ is a statistic where p_e is the probability of chance agreement and p_o is the relative observed agreement between raters. It is used to measure two raters' agreement on N items classified into mutually exclusive categories and takes into account agreement occurring by chance (Equation 3). The averaged F1 score across the batches was 90% (Figure 5), 88.3% for the mean MCC score (Figure 6), and 87.7% for mean Cohen's κ score (Figure 7). This demonstrates that the model is performing quite well against unseen email threads and has a similar average across all three metrics.

5. Conclusion and Future Work

The system was built into a Flask application with an endpoint that receives an input email and returns the predicted class and the corresponding probability. This probability could be used to determine the degree of confidence of the prediction and whether a human check is necessary. The application was built into a Docker image for integration as a microservice. Some directions for future work include using the larger BERT pre-trained model (24-layer, 1024-hidden, 16-heads, 340M parameters). In addition, there are at least four other categories with a small number of examples that could be included in the training of the model to see if those categories could also be classified. It is certainly sensible for an email to fall into one or more categories and this could be another area for work. For instance, an issue with the air conditioning system could be classified as an electrical and HVAC issue. Lastly, there are going to be inputs with incorrect classifications. In these cases, it would be prudent for human intervention to determine the proper class and save these corner cases in a separate file to include in the next iteration of training an improved model.

Figure 4. Training and Validation Losses

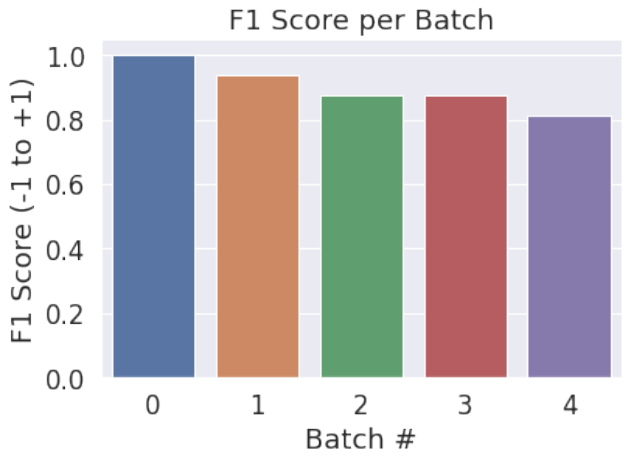

Figure 5. F1 Scores

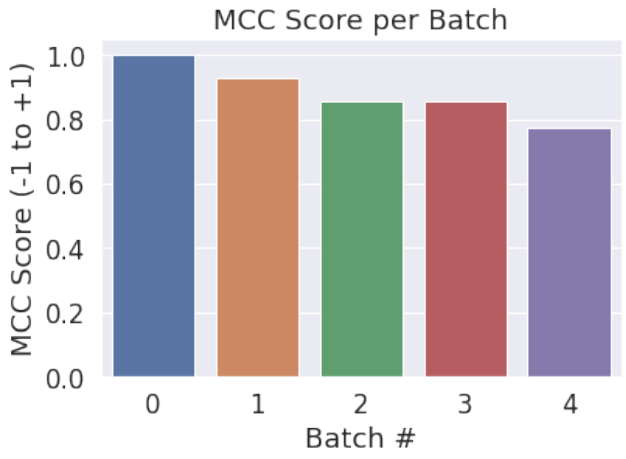

Figure 6. MCC Scores

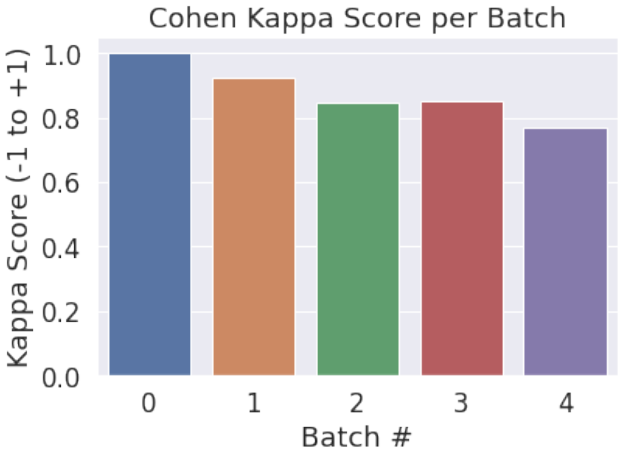

Figure 7. Kappa Scores

References

[1] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017. 1

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[3] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 1

[4] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. 1

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 3

[6] J. Zhao, Y. Zhou, Z. Li, W. Wang, and K.-W. Chang. Learning gender-neutral word embeddings. *arXiv preprint arXiv:1809.01496*, 2018. 1