Improvement of deep learning techniques for ICD-9 code classification using MIMIC-III medical notes

Stanford CS224N Custom Project, Option 3 (Graded)

Soham Gadgil
Department of Computer Science
Stanford University
sgadgil@stanford.edu

Derek Jow
School of Medicine
Stanford University
djow@stanford.edu

Anthony Li
Department of Computer Science
Stanford University
antli@stanford.edu

Abstract

ICD-9 code (International Classification of Diseases, version 9) classification has long been a burden on our healthcare system due to the time-intensive process of manual curation of disease codes. We apply current state of the art NLP models to the task of automatic ICD-9 code classification on free-text clinical notes of MIMIC-III [1]. Our goal was to exceed the baselines established by Huang et al. [2], which achieved an F1 score of 0.696 and an accuracy of 0.896 from their best model, a GRU RNN, trained and tested on notes of type "Discharge summary." With our best model, based on fine-tuning BlueBERT, we are able to achieve an accuracy of 0.8521 and F1 of 0.7074. We also evaluate the performance of these models when trained on notes of all types, which has not been done before to the best of our knowledge.

1 Key Information to include

- External collaborators / Sharing project: None
- Mentor (custom project only): Emma

2 Introduction

Diagnosing patients using ICD-9 (International Classification of Diseases, version 9) coding is important for billing procedures on patient discharge and ensuring an informative and correct patient history. ICD-9 was the standardized vocabulary of medical terminologies used for over 35 years [3]. The ICD-9 codes consist of 5 digits: the first 3 digits represent a generalized medical concept, and the last 2 digits represent specific variations of that concept. Although ICD-9 had been used as a standard in clinical care for many years, it eventually became a default system for classifying any type of medical condition, regardless of its original purpose. While ICD-9 was intended to classify diseases to allow physicians to immediately understand the state of a patient's health across systems, it soon became entangled with a plethora of other uses that made it difficult to comprehend. When the MIMIC-III dataset was collected, ICD-9 was used for diagnosing diseases, medical procedures, accidents, and billing. In the scope of this project, ICD-9 is used for discharge codes used to bill patients with the most accurate description of the treatment by their physician. As of today, ICD-9 has been replaced by ICD-10, expanding the amount of terms available from 14,025 [4] codes to $\sim$155,000 [3].

Making use of medical knowledge in electronic health records has proven to be especially difficult. Many clinical decision support models are limited by poorly structured data. Roughly 80% of time spent creating clinical models is simply processing the data, including data merging, cleaning, and extracting essential variables, severely limiting their predictive capacity. With all the extra information available, predictive models struggle to select the most important variables to make decision support useful [4].

Because traditional models struggle with unkempt data, deep learning could be the key to better clinical support because it works well with unstructured data. One avenue that would benefit from better learning approaches is ICD-9 code classification. Automatic or assisted assignment of ICD-9 codes on patient discharge would relieve a tremendous burden on our healthcare system. Currently, the process of coding is largely a manual, labor-intensive task due to the existence of thousands of possible codes each with their own fine-grained description of every possible condition. Classification is determined largely by a patient's history, treatments, and observations recorded by their physician during inpatient stay in the unstructured "clinical note" section of their Electronic Health Record. Each patient's health record also contains lab tests, vital tests, body measurements, and demographics. Many researchers incorporate the entire EHR, including all data available taken over time, to make predictions about mortality probability, 30-day unplanned re-admission probability, length of stay, and more [4]. Here, we focus on ICD-9 code prediction for the top-10 most common codes using only the free text clinical notes. One major benefit of this isolated task is that we do not need to spend excessive time harmonizing the data, including rigorous feature extraction or feature engineering.

3 Related Work

We directly compare our performance to the work of Huang et al. [2], which evaluated various deep-learning techniques on automatic ICD-9 code classification. As described by the authors, their work is intended to serve as a proof-of-concept and as a baseline for evaluating deep-learning based techniques on clinical applications. Analyzing only the clinical notes of type "discharge summary", they had optimistic results with standard deep learning NLP architectures, such as RNNs. For top-10 ICD-9 code prediction, their best model, a GRU-based RNN, achieved an F1 score 0.6957 and an accuracy of 0.8967. They also experimented with LSTMs, achieving a top-10 code F1 score of 0.6738, with CNNs achieving an F1 score of 0.6481, and with traditional logistic regression achieving an F1 score of 0.6141.

Other works have utilized all available EHR data in their task of ICD-9 code prediction. [4] performed in-hospital mortality, 30-day unplanned readmission, prolonged length of stay, and ICD-9 discharge code classification from two US academic medical centers. Their models included an LSTM RNN, an attention-based time aware neural network (TANN), and a neural network with boosted time-based decision stumps, all combined with ensembling. Using all available medical data (not just clinical notes), they made predictions for every possible ICD-9 code, ensuring each code must have an exact match to be deemed correct (For instance, code 250.4 would be different from 250.42). They were able to achieve an accuracy of 0.90, an F1-score of 0.41, and an AUC ROC of 0.87.

4 Approach

Because there are $\sim$14,025 ICD-9 codes, we simplified the problem to predict only the top-10 most common codes, an approach used by Huang et al as well. We used Google BigQuery [5] for initial data pre-processing to associate notes with their codes and remove notes that were not assigned a code in the top-10. The distribution of the top-10 codes is shown in Fig. 1. Both training and testing were performed only with notes that were assigned at least one code in the top-10, which reduced the size of our dataset from about 2 million notes to 1.2 million. The top-10 codes and their definitions are detailed in Table 1. First, we trained and tested several models on the entire set of 1.2 million notes. Then, we filtered for only the notes which were categorized as "discharge summary", which reduced our dataset size to 59,652 notes, and trained and tested additional models using the smaller set of only "discharge summary" notes.

For experiments performed on all 1.2 million notes, we used a train/validation/test split of approximately 90/5/5, but we did not end up evaluating our models over the test set due to resource constraints. For experiments performed using the smaller set of only "discharge summary" notes, we used a train/validation split of approximately 75/25 without a test set. We performed the train/validation/test splits ensuring that notes for the same patient will always be placed in the same group. Finally, we exported the data in a CSV format that can be easily read using a Pytorch dataloader, with a binary column for each of the ten codes.

Code	Definition
25000	Diabetes mellitus without mention of complication, type II or unspecified type
2724	Hyperlipidemia NEC/NOS (Other and unspecified hyperlipidemia)
4019	Hypertension NOS (Unspecified essential hypertension)
41401	Coronary atherosclerosis of native coronary artery
42731	Atrial fibrillation
4280	Congestive heart failure, unspecified
51881	Acute respiratory failure
53081	Esophageal reflux
5849	Acute kidney failure, unspecified
5990	Urinary tract infection, site not specified

Table 1: Top 10 most common ICD-9 codes and their definitions

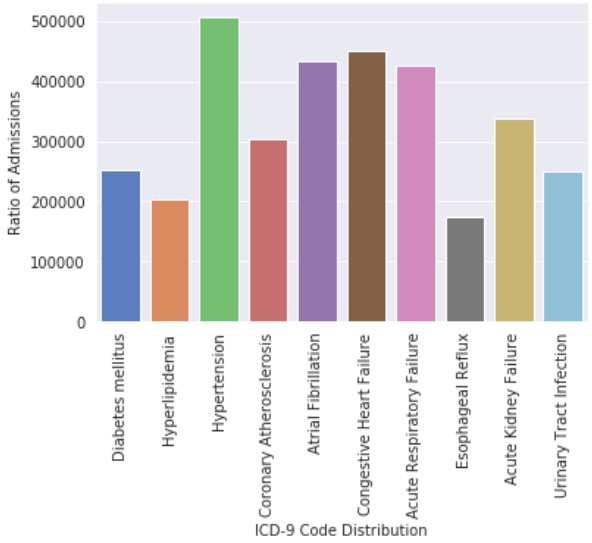

Figure 1: ICD-9 Code Distribution

Since Huang et al. already performed many experiments with RNN-based architectures, we prioritized experiments with current state-of-the-art techniques such as transformer and BERT-based models. We tested several approaches: a transformer encoder model trained from scratch, a pre-trained BERT model fine-tuned for our task, a BlueBERT model with pre-trained word embeddings derived from medical papers in PubMed [6] and MIMIC-III, and an attention-based bidirectional LSTM RNN.

Our first approach, the transformer encoder trained from scratch, was built by adapting example code for text classification from GitHub user Allen Lee [7]. This model consisted of six transformer encoder layers each with eight attention heads, operating on 256 features. Tokenization was performed using SentencePiece, which we trained ourselves on the MIMIC-III dataset. The original code was designed for single-label classification, but most notes in MIMIC-III are assigned multiple ICD-9 codes. Therefore, we made modifications to perform multi-label classification by changing the loss function to BCEWithLogits loss (multi-class cross entropy) with mean reduction, as shown in equation 1.

$$l_c(\hat{y}, y) = \{l_{1,c}, ..., l_{N,c}\}^T, l_{n,c} = -w_{n,c}\left[y_{n,c} \cdot \log\sigma(\hat{y}_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - \sigma(\hat{y}_{n,c}))\right] \quad (1)$$

Here, N is the batch size, c is the number of classes, and $w_{n,c}$ are the class weights, set to 1. We also created 10 output nodes in the final linear layer, corresponding to each of the top-10 codes. We

then passed each of the output logits through a sigmoid function and predicted any code with a score greater than a threshold. Finally, we computed the evaluation metrics, accuracy and F1 score.

The second approach was fine-tuning BERT [8], a language representation model developed by Google. We adapted the starter code for BERT from a youtube video [9]. The BERT model is pre-trained on a large corpus of unlabelled text and has developed a deep bidirectional learning representation for freeform text. It can be fine tuned to a specific task, like text classification, by simply adding a linear layer on top of the existing model. We built this model using the BertForSentenceClassification module in huggingface [10], performing similar modifications as described above to adapt it for multi-label classification. The specific model we fine-tuned was Bert-Base, Uncased, with 12 transformer blocks, 768 hidden units, and 12 attention-heads.

The third approach was fine-tuning BlueBERT, a BERT-based model pre-trained on PubMed abstracts and MIMIC-III clinical notes [11]. Like our BERT model, we added an extra linear layer on top of the existing model to transform our output into 10 classes, one for each ICD-9 code. We also implemented the BCEWithLogits loss for multi-class classification. Three small architectural variations of this were additionally explored: (1) Adding three linear layers with ReLU non-linearity instead of just one linear layer, (2) Freezing the BlueBERT weights from the first variant so that only the linear layer weights would be tuned, and (3) adding a dropout layer after the BERT layer from the second variant.

Additionally, we experimented with different techniques for dealing with notes that were longer than 512 tokens, since BERT caps at 512. Our first BlueBERT model truncated the text after 512 tokens and used that for prediction. Next, we tried splitting the text into two chunks of 512: the first 512 tokens and last 512 tokens, and combined the output logits through max pooling. Next, we tried splitting into chunks of 256. We created four 256-blocks: one at the beginning, two in the middle, and one at the end. We tried combining these chunks together using two different techniques: one by max pooling their logits, and the other by concatenating the final hidden layers together and adding an extra linear layer to reduce this to 10 output nodes.

Finally, we wanted to compare these results to our own LSTM with attention. We implemented the attention from the GitHub code by [12]. The word vectors used were obtained from the BioWordVec Model [13], which applies fastText [14] to compute 200-dimensional word embeddings using PubMed and MIMIC-III clinical notes.

5 Experiments

5.1 Data

We used the MIMIC-III (Medical Information Mart for Intensive Care III) dataset which is a large, open dataset containing deidentified medical data of over 40,000 patients staying in the Beth Israel Deaconess Medical Center between 2001 and 2012. The dataset contains demographical information, vital signs, lab tests, procedures, medications, mortality outcomes, and clinical notes. We focused solely on the free-text clinical notes written by physicians describing their patients in the ICU.

5.2 Evaluation method

We used F1 score and accuracy for each model, which were the metrics used by Huang et al. for evaluation. We computed these metrics using scikit-learn [15, 16]. For our best model (BlueBERT 512-chunked), we calculated the AUC ROC for the prediction of each code. We also tested different thresholds in the BlueBERT model and chose the threshold that maximized our F1 score, which turned out to be 0.4.

5.3 Experimental details

For our simple transformer encoder model run on the entire note corpus, we used a learning rate of 1e-4 and a dropout probability of 0.3, trained over a 90-5-5 train-validation-test split. The model took $\sim$8 hours per epoch. The same setting applied to the "discharge summaries" corpus took $\sim$ 2 hours per epoch.

For our BERT-base uncased model on the entire corpus, we trained with learning rates 3e-5 and 2e-5. The training took between 20 and 50 hours per epoch on a GPU with a batch size of 4. When we ran these models on just the notes filtered by "discharge summary", we used a 75-25 train-test split. The transformer encoder used the same learning rate of 1e-4, and the BERT model only used a learning rate of 2e-5. The BERT model took 3 hours per epoch to train.

For our BlueBERT models, we performed training on the same sets as above: the 90-5-5 split for all clinical notes, and the 75-25 split for "discharge summaries". These models used a learning rate of 2e-5. For the large text corpus, training took roughly 30 hours per epoch. On the discharge summaries corpus, training took 7 hours per epoch with a batch size of 4.

For the LSTM RNN with attention model, we used a learning rate of 1e-3. We used pre-existing word embeddings derived from BioWord2Vec. Due to to the size of the word embeddings, we limited the number of tokens loaded to the top 1 million. The max sentence size used for the LSTM was 1000 tokens. Training took 1 hour per epoch with a batch size of 128.

5.4 Results

All Clinical Notes	Training		Validation	
	Accuracy	F1	Accuracy	F1
Transformer Encoder	0.7711	0.3331	0.7780	0.3957
BERT, lr=3e-5	0.8073	0.5429	**0.8119**	**0.5128**
BERT, lr=2e-5	**0.8349**	**0.5967**	0.78	0.47

Table 2: Model performances for top-10 codes using all clinical notes

Table 2 shows the results of training our models on the full corpus of 1.2 million notes that are associated with a top-10 ICD-9 code. Unlike Huang et al., these include notes that are not of type "discharge summary." These notes can be taken at any point in time during a patient's stay, not necessarily when they are released, so models trained on these notes can be used for a wider range of tasks (beyond administrative tasks such as billing). The best model here was BERT with a learning rate of 3e-5, yielding an F1 score of 0.5128 and an accuracy of 0.8119.

Discharge Summary Clinical Notes	Training		Test	
	Accuracy	F1	Accuracy	F1
Huang et al. LSTM RNNs	0.9154	0.7445	0.8950	0.6874
Huang et al. GRU RNNs	0.9126	0.7397	**0.8967**	0.6957
Transformer Encoder	**0.9524**	**0.8393**	0.8526	0.663
BERT, lr=2e-5	0.8690	0.7204	0.8375	0.6569
BlueBERT 512-chunked, maxpool	0.8828	0.7660	0.8521	**0.7074**
BlueBERT 256-chunked, maxpool	0.8723	0.7452	0.8361	0.6769
BlueBERT 256-chunked, llayer	0.8616	0.7181	0.8382	0.6735
BlueBERT w/ addl. linear layers, lr=2e-5	0.8722	0.7446	0.8290	0.6629
BlueBERT (frozen) w/ addl. linear layers	0.8863	0.7740	0.8234	0.6534
BlueBERT (frozen) w/ addl. linear layers+dropout	0.8927	0.7873	0.8238	0.6545
LSTM with attention	0.8949	0.7275	0.8396	0.65

Table 3: Model performance for top-10 codes using "discharge summary" clinical notes

Table 3 shows performance results for models trained on clinical notes that were categorized as "discharge summary". This is the same corpus used by Huang et al., except we used a 75/25 training-validation split instead of a 50/25/25 training-validation-test split. We tested nine different models: a transformer encoder, an unmodified BERT model, six variations of BlueBERT, and an LSTM with attention. The BlueBERT 512-chunked model was split into a prefix of 512 tokens and suffix of 512 tokens, with their output states max pooled before prediction. This model beat Huang et al.'s best F1 score, producing a value of 0.7074. However, the accuracy of their GRU RNN was higher, with a value of 0.8967.

We took our best model, BlueBERT 512-chunked max-pooled, and plotted an ROC curve for each of the 10 ICD code classifiers. Figure 2 shows the ROC-Curves for all the top-10 ICD-9 Codes obtained using the BlueBERT Model. Our model works best with predicting code 41401, and performs the worst with code 59900. The average AUC ROC is 0.8769.

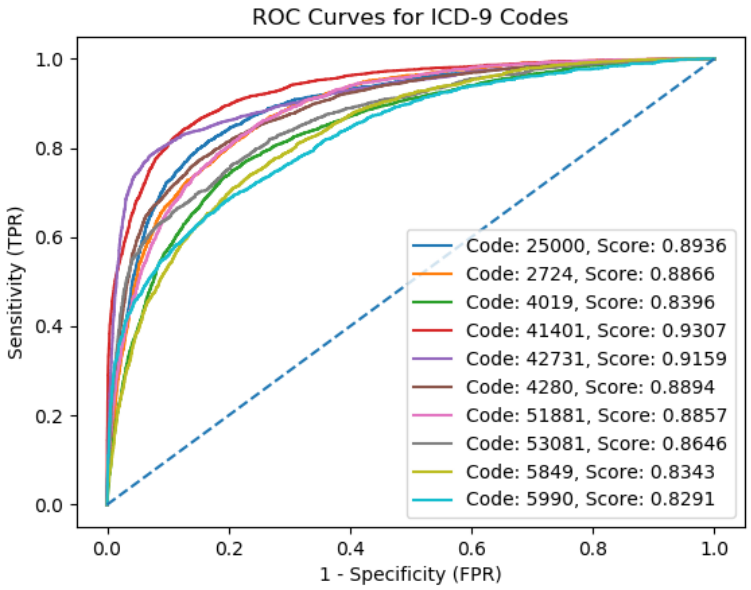

Figure 2: ROC-Curves w/ ROC AUC Scores for ICD-9 Codes

6 Analysis

Our models performed moderately well on the dataset, but we were expecting higher gains in F1 score and accuracy using BERT compared to the GRU RNN of Huang et al. Our best model, BlueBERT 512 chunked max-pooled, beat their F1 score narrowly: 0.7074 to 0.6957. However, we did not beat their accuracy, which was 0.8521 to 0.8967. We speculate that our accuracy is worse because our model is more sensitive but less precise; it is good at predicting positive for classes and detecting the code if it is real, but may predict positive too often and not always when it should. This would reflect a larger F1 score but a decrease in accuracy.

It is difficult to say conclusively that our model performs better than theirs because our F1 score was only marginally better and did not score a higher accuracy. There are different reasons why the model did not perform as well as it could. First, instead of chunking the entire text, we just combined the first and last 512 tokens, which was a decision made due to GPU constraints and time. Second, we used the BERT-base model, not the BERT-large which is larger and more compute-intensive, having 24 transformer blocks, 1024 hidden units, and 16 attention heads, thereby making it more expressive. We also opted to use the BlueBERT model that used BERT-base, similarly due to the finite resources available.

Finally, we did not perform enough data pre-processing to extract key words from the clinical notes. One idea is to filter the words by TF-IDF score and only keep words that are enriched, a technique

that was explored by Huang et al. Similarly, tokens could be passed through a knowledge base like UMLS (Unified Medical Language System) [17] that can condense medical concepts to a single token. This could be important for word embeddings because there are many ways of describing a medical condition, but the model might only be trained on one particular verbiage and not the others.

For our non-BlueBERT models, we believe that the reasons for their sub-par performance are multifold. Unlike BlueBERT, they were not trained with PubMed abstracts and biomedical literature, which would make capturing the semantic meaning of medical terms more difficult. Additionally, we did not experiment with chunking for the normal BERT model, so it was limited to only the first 512 tokens in the text. This could lead to loss of important information that could be used to make a decision.

Our LSTM with attention model also showcased sub-par performance. This could be because we only considered 1 million tokens from BioWord2Vec and restricted the max sequence length to 1000. These choices were made due to compute and time limitations. Increasing either the number of tokens loaded or the max sequence length would make the model more expressive and improve performance. Also, we used multiplicative attention in our model which does not scale well to higher dimensions. Instead, we could use additive attention which introduces a non-linearity, resulting in greater expressive power.

The performance of the transformer encoder performance was also worse than expected. This could be because clinical notes were truncated after 256 tokens, which resulted in a substantial loss of semantic meaning compared to the RNN baseline models which used the whole text. This could be fixed by filtering words by their TF-IDF scores to select the 256 best words to use in the model. Also, the training performance was much higher than the test performance, which indicates that the model was severely overfitting. This could be fixed by adding a regularization term to the loss function or by increasing the dropout probability. Furthermore, our results for the BERT models are also under-performing. It could be caused by any of the following: truncation at 512 tokens, poor word embeddings for medical vocabulary, and sub-optimal threshold for prediction.

7 Conclusion

Our results demonstrate that transformer-based neural network architectures are quite appropriate for the task of ICD-9 code classification using clinical notes, producing comparable results to the best RNN-based architectures from prior work. However, producing substantially better results than these RNN architectures may require significantly more data pre-processing and/or architectural modifications; in other words, substituting BERT for RNNs will not immediately produce dramatic performance gains in this task.

Although our best BERT-based model narrowly achieves a higher F1 score over the test data, we cannot conclusively determine that our model works better. Our BlueBERT model, split into a prefix and suffix of 512 tokens each, combined with max-pooling, achieved an accuracy of 0.8521 and F1 score of 0.7074. Our accuracy was $\sim$5% lower, and our F1 score $\sim$0.012 higher. We were hoping to achieve greater performance gains. Rajkomar et al., using additional features such as lab and vital test data, demonstrated optimistic predictive power of deep learning methods in clinical settings by achieving an accuracy of 0.90, an F1-score of 0.41, and an AUC ROC of 0.87 for all ICD-9 code classification. While fine-tuned BERT models have demonstrated to be superior to previous architectures in many other tasks, this does not appear to be the case for ICD-9 code classification with the models we tested. Producing higher quality models for this task requires more architectural ingenuity than running BERT with small modifications.

An additional contribution of our work is that we establish baseline performance metrics for ICD-9 code classification using all notes in the MIMIC-III dataset, not only "discharge summary" notes. This presents a solution to a new problem: prediction of disease at a given point in a patient's timeline. Although this concept has been tried extensively by the works of Rajkomar et al. using the entire EHR, this task has not been performed yet for prediction with just the clinical notes. We expect our scores to be used as a baseline for this task, given that adding lab and vital test data can only improve prediction.

While our results show potential in transformer-based neural network architectures, we believe that in the future we can expand our work to further increase performance. As mentioned previously in

our discussion, there is much potential in improving our data pre-processing pipeline. Transformer based models often struggle with large amounts of text, which indeed presents a problem for the lengthy discharge summaries written by physicians. We speculate that by reducing the number of words in each note, we could improve prediction. This can be performed by selecting words by their TF-IDF score or using a knowledge base like UMLS to replace verbose descriptions with terse identifiers. We also expect better performance using BERT-large and chunking the entire note into sections, rather than selecting just the prefixes and suffixes.

Overall, we have shown that transformer-based neural network architectures are promising models for use in ICD-9 code classification. Our tests demonstrated that they can perform better than prior RNN architectures. However, they will still require more data pre-processing or further adjustments to their architecture to maximize performance.

References

[1] Lu Shen Li-wei H. Lehman Mengling Feng Mohammad Ghassemi Benjamin Moody Peter Szolovits Leo Anthony Celi Roger G. Mark Alistair E.W. Johnson, Tom J. Pollard. Mimic-iii, a freely accessible critical care database. *Nature*, 2016.

[2] Jinmiao Huang, Cesar Osorio, and Luke Wicent Sy. An empirical evaluation of deep learning for icd-9 code assignment using mimic-iii clinical notes. *Computer Methods and Programs in Biomedicine*, 177:141 – 153, 2019.

[3] Jacqueline DiChiara. Icd-1 to icd-11 timeline highlights healthcares evolution. 2015.

[4] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18, 2018.

[5] Kazunori Sato. An inside look at google bigquery. *White paper, URL: https://cloud. google. com/files/BigQueryTechnicalWP. pdf*, 2012.

[6] Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174, 2016.

[7] Allen Lee. text-classification-transformer. 2020.

[8] 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.

[9] Chris McCormick. Bert document classification tutorial with code.

[10] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[11] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, pages 58–65, 2019.

[12] Prakash Pandey. Text-classification-pytorch.

[13] Zhihao Yang Hongfei Lin Zhiyong Lu Yijia Zhang, Qingyu Chen. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Nature*, s41597-019-0055-0, 2019.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[17] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.