The NBME Kaggle Competition

Evan Vogelbaum
MIT
evanv@mit.edu

Yuval Mamana
MIT
yuvalm11@mit.edu

Luke Igel
MIT
lukeigel@mit.edu

ABSTRACT

In this paper we present our strategy for the NBME Score Clinical Patient Notes Kaggle Competition[1]. We detail how we went about data cleaning and exploration as well as the development of the three models that we formed into a final ensemble model. We conclude by discussing results of the ensemble model and lessons learned from the competition for how we can improve our model in the future.

1 INTRODUCTION

Every year, thousands of medical students participate in the National Board of Medical Examiners (NBME) Step 2 Clinical Skill Examination. In this exam, students must demonstrate proficient ability to interview patients and diagnose their ailments as well as features relevant for their treatment. Students are tested on different *cases* where each case consists of an actor who portrays various symptoms and reveals information about their lifestyle and family history. The medical student has to write a *patient note* describing their interview with the patient and all information they find relevant to the patient's ailments. The patient note is graded based on whether it notes various *features* that the actor was told to portray.

Right now, the grading process for this examination requires trained physicians to spend hours manually grading each note. This is costly in both human and financial resources, and the NBME Score Clinical Patient Notes Kaggle Competition [1] challenges participants to come up with an automated way to score the notes. As part of the competition, participants are given 1000 labeled patient notes evenly divided across 10 cases. Each labeled patient note consists of the text of the note, the features the note *should* portray (examples include "Male" or "No Chest Pain"), and for each feature either character-level bounding boxes of the places in the patient note where the feature is mentioned, or no bounding boxes if the feature is not mentioned. In addition to the 1000 labeled patient notes, participants were also given 45,000 unlabeled patient notes.

In this paper, we detail our teams strategy for the competition. Section 2 discusses our approach to data cleaning. Section 3 goes through each of the models that composed our ensemble and discusses their individual performance. Section 4 discusses our results and our takewaways from viewing the highest scoring solutions after the conclusion of the competition. Section 5 concludes the paper.

2 DATA CLEANING

Our data cleaning pipeline aims to produce a clean, consistent, and comprehensive version of the different data sets that were published by Kaggle and the NBME. The cleaning pipeline begins by manually fixing incorrect annotations and features and replacing them with the accurate text and annotation locations. These corrections rely on a data cleaning pipeline published by Y.Nakama (another competitor) and modified by our team. Next, we fixed format mistakes in the annotation data to create a standardized format that allowed for accurate evaluation of our predictions.

The last stage of the cleaning pipeline was the merging of the train data set with the features and patient notes data sets to create one data set that could be used by all models. Each record in this data set represents a distinct feature in a specific patient note.

The final data set contains the following columns:

- ID: The unique id of the record as given by Kaggle.
- Case Number: A unique id for the case of the note. A case represents an actor and the set of features that they are expected to portray.
- Patient Note Number: A unique id for a specific patient note.
- Feature Number: A unique id for the feature that is expected to be found in the note.
- Annotation: A list of strings that contains text in the original patient note which corresponds to the feature in the note. Each element is not necessarily contiguous but is human-readable. Some elements may splice together parts of a patient note.
- Feature text: The text of the feature as given by Kaggle.
- Patient note: The full text of the patient notes as given by Kaggle.
- Location: A list of tuples that represents the bounds of the characters in the patient note that correspond to the feature.

Formatting the data in this standardized format allowed us to have a consistent input and output format to all of our different models, and contributed to the efficiency of the ensemble process.

3 OUR APPROACH

3.1 Pattern-matching Model

Our baseline feature classification and localization model is a basic pattern-matching system. Given a pre-loaded corpus of features mapped to patient notes and an input patient note, the model determines which feature's set of patient notes is most similar to the input note. The model's life-cycle can be divided into two stages: initialization and prediction.

3.1.1 Initialization. Upon initialization of the pattern-matching model, the entire NBME training set is organized into a matching dictionary. The matching dictionary pairs each tuple key `(case_num, feature_num)` to a list of all annotations associated with that pair. This allows for easy retrieval of all annotations associated with a given patient feature in a particular case. For example, for the key pair `case_num 1` and `feature_num 002 ('Chest-pressure')`, the dictionary would return the list `['dad with recent heart attcak(sic)', 'chest pressure', 'intermittent episodes', 'episode',...]`. This organization method allows for easy feature classification in the Prediction stage.

3.1.2 Prediction. The pattern-matching model determines whether a given feature is present in a given patient note, and if so, the character spans of its location throughout the note. Given a `case_num (x)`, `feature_num (y)`, and `patient_note (z)`, the model will retrieve from the matching dictionary a list of all known annotations for pair `(x, y)`. These annotations are referred to as *candidates*. For each candidate `c`, the Python function `str.find()` is run on the patient note. If the candidate substring is found, the model appends the start and end indices of the found candidate in `patient_note` to the final output of spans. This simple memorization-based approach is very easy to understand and implement.

3.1.3 Results, Reflection, and Possible Improvements. This approach is very simple, but also highly brittle. We achieve subpar results in general, but strong outcomes with some particular features. There are many ways to describe most features, and we therefore found the model only performs well with features that are frequently referred to by the model's known synonyms (ex: `'male'` and `'man'`). More complex concepts in natural language such as qualification or negation are even further out of reach with this model, since `'not a male'` would still register as `'male'` under this system. Through this model, we achieve a 17% F1-micro score on a held-out test set. Despite its shortcomings, this model served as an excellent baseline during our project's development. It also allowed us to finalize our general model interface for the ensemble, where a feature, case number, and patient note are expected in order to return a list of classified character spans.

3.2 Phrasing Model

The second model we implemented is a BERT-based model that uses similarity measurements on individual phrases in the patient note to find phrases with high similarity to a given feature. We can divide the model operation to three stages: phrasing, training, and prediction.

3.2.1 Phrasing. The first, and most important, stage in our model, is the phrasing stage. In order to analyze separate phrases in a patient note, we start by dividing a note into small phrases that will later be analyzed individually.
An efficient phrasing process is crucial for the success of the model since our prediction function uses a phrase as an input. Phrases that are not similar enough to the feature they represent will not be predicted correctly.
To achieve this goal, we first divided a given patient note into sub-strings by splitting the text by commas and periods as delimiters. Then, each of these sub-strings was divided into distinct sentences using the NLTK sentence tokenizer [3]. The result of this process is a list of phrases that constitute the patient note while each phrase can be handled individually.

3.2.2 Training. Since this model does inference using feature similiarity, it does not need training in the same way our other models do. Instead we use an `all-mpnet-base-v2` model to embed our phrases and features [10]. The result of this model is a list of vector embeddings that maps each of the features to a 768-dimensional vector which can be used for tasks like clustering or semantic search.

3.2.3 Prediction. The final stage of the phrasing model is to compute the predicted features present in each patient note, and ultimately get a list of character bounds for each feature in a patient note.

The prediction is performed by encoding each phrase into a vector using the same `all-mpnet-base-v2` model used for feature embedding. We then measure the cosine similarity of the phrase to all the features in the relevant case. If no feature has a similarity score above a minimum threshold, then no feature is predicted. Otherwise, we predict the feature with the highest similarity score as being present in that phrase. The location of the phrase is then added to the predicted annotation location of the feature in the patient note.

3.2.4 Results and Possible Improvements. The performance of the phrasing model greatly depends on the quality of the phrasing stage. When the relevant text for a feature was large enough that it filled a whole phrase, the model was often able to achieve higher micro-F1 scores. For example, the feature 'Increase in appetite' had annotations that usually filled out

Feature Name	F1
Worse with deep breath	96.3%
Minimal to no change with Tums	95.6%
No depressed mood	93.2%
Family history of migraines	91.7%
1 day duration	0.0%
Onset 3 years ago	0.0%

Table 1: Selected features with high F1 score (top) and low F1 score (bottom) when evaluated by the Phrasing model.

a whole phrase, and the model predicted the feature with an F1 score of over 80%. However features like age or gender whose annotations were usually one small part of a longer sentence got lower similarity scores. Selected examples of features and their matching F1-score are displayed in Table 1.

Given the importance of the phrasing step to the success of our model, we think that most of the potential improvements will be found there. A possible way to achieve better phrasing is by using the keyword extraction method presented by Mihalcea and Tarau [6]. Using a SpaCy model, they were able to extract central keywords from a given set of sentences, that could be use by our phrasing model as input of the prediction function.

Alternatively, a more fine-grained method of phrasing could be used. For example, we could divide each note into a set of pairs or triplets of consecutive words in the text and perform the prediction on them. Unfortunately, this approach was computationally infeasible with the resources we had.

Finally, since this model was the most computationally expensive of all the models we used, we think more could be done to improve the performance and run-time of the algorithm. Such improvements include caching of our embeddings and phrased notes or doing our embedding in a parallel manner.

3.3 DeBerta Model

The third and final model in our ensemble is a large-language model fine tuned with a custom prediction head to perform token classification. The feature extractor of the model is the one created by He et al. [5]. They follow the same training strategies as in the original BERT paper [4] except that they incorporate a BPE vocabulary as described in Radford et al. [9]. To train the model, we make use of the phrase-pair inference mode for the DeBerta model. This allows us to encode the patient note as one sequence of tokens and compare each token to the sequence of tokens encoding both the feature we want to label and, in later experiments, the case that feature appears in. Each token in the patient note is given a probability of being labeled with a given feature. Any probabilities > 0.5 are converted into positive classifications at inference time.

3.3.1 Byte Pair Encoding. Because our data involves particularly rare words and abbreviations that only appear in the medical context, it is worth expounding upon the tokenization method that our DeBerta model uses and why it alleviates the potential issue that many important words will be out of vocabulary for the model.

Byte Pair Encoding, first introduced by Sennrich et al. [11], begins with a corpus of text C that the model will eventually be trained on and an initial vocabulary V. An innovation introduced in Radford et al. [9] was to let this initial vocabulary be the 256 possible numbers that fit in one byte. Since every unicode character chan be represented as a sequence of bytes, this allows all possible characters to be decomposed in terms of the base vocabulary. The building of the vocabulary then proceeds iteratively:

(1) The co-occurence of each pair of symbols $(v_1, v_2) \forall v_1, v_2 \in V$ is evaluated, where two symbols co-occur if one appears directly before another.
(2) The two most frequency co-occuring symbols $(v_1^\star, v_2^\star)$ are merged into a new symbol $w^\star$ which is added to the vocabulary. To prevent sub-optimal merges, Radford et al. [9] prevents merging across Unicode character categories for any byte sequence with the exception of spaces.
(3) The procedure repeats until the vocabulary has the desired size. In the case of DeBerta, this is $50,000 + 256 = 50,256$.

At inference time, a sequence of unicode bytes (text data) is broken down according to the learned vocabulary, matching the largest symbol possible at each step and using the base vocabulary symbols if no compound symbols match.

The use of Byte-Level BPE ensures that all unicode characters, and in particular any characters that combine to form a rare medicine-specific word, is in vocabulary for our model. The downside of this method is that no token is particularly interpretable.

3.3.2 Training and Evaluation of the DeBerta model. We found that the pre-trained DeBerta model was able to perform very well even when provided with only a subset of the 1000 labeled examples. We use grouped cross-validation on the unique id for each patient note (ensuring no patient notes overlap between our train and validation sets) to split our data into a train set of 800 note annotation pairs with 200 pairs held out for validation. After training for 3 epochs over this data, the model is able to achieve a validation micro-F1 score of 81%. While this performance was impressive, we

Feature Name	DeBerta Micro F1
Decreased Appetite	93.5%
Exercise Induced Asthma	93.6%
Tossing and Turning	94.2%
Hallucinations after taking Ambien	0%
Stress	0%
Meningococcal vaccine status unknown	0%

Table 2: The features with the top 3 (top) and bottom 3 (bottom) micro-F1 scores when evaluated by the DeBerta model.

found that there was significant imbalance in model performance across features. Table 2 shows the micro-F1 the model acheived on its top 3 and bottom 3 features.

Further investigation of the imbalance of the model corroborated our initial data exploration which showed that our training set had significant data imbalances. For example, the feature "Meningococcal Vaccine Status Unknown" has only two positive samples across the entire training set. Our first attempt to deal with this imbalance was through oversampling: randomly sampling positive instances of each feature and continuously adding them to the train set until we have an equal number of positives and negative samples. This marginally improved our model's performance on some of the features it did the worst on, but caused an overall decrease of about 1% in overall model F1.

Our other method of attempting to address data imbalance was through additional oversampling via data augmentation. Our hypothesis was that performance imbalance may have been caused by overfitting to the context of specific features. To address this, we augmented the data by randomly replacing words in the patient note by synonyms as determined by Wordnet [7]. While we originally sought to replace words that were in an annotation, so as to make our models more robust to different phrasings of key parts of a patient note, analysis of annotation lengths (see Figure 1) caused us to realize that most of the annotations we had were very tight around the critical words, and as such replacing those words may have the unintended effect of turning a positive sample into a negative one. We therefore instead opted to only modify the context around annotations, leaving the annotations themselves unchanged. For each sample in our training set, we augmented it by adding an additional 5 samples with words randomly replaced by synonyms. While this also marginally improved performance on our worst features, it reduces overall F1 by about 0.5%. Therefore for our final model we chose to use the vanilla DeBerta fine-tuned model.

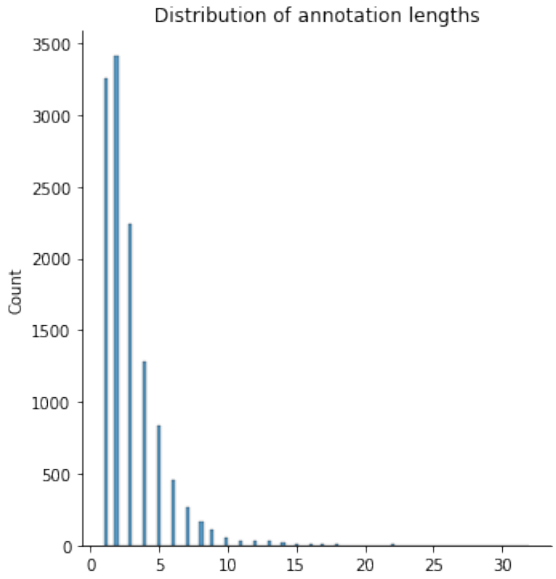

Figure 1: The distribution of annotation lengths across the train set. For each patient note annotation pair, we consider the full length of the annotation, combining together words that might be annotated in different parts of a patient note. The length of an annotation is the number of words involved in it. Most annotations are very tight bounding boxes around the relevant text.

4 RESULTS AND FUTURE WORK

4.1 Results

With three models in hand, our goal was to create and ensemble model that would combine the predictions of each model so that one model's strengths could complement others' weaknesses. While we considered advanced ensembling mechanisms including weighting of predictions, thresholding, and training models on the error of other models (e.g. gradient boosting), for simplicity and due to time constraints we opted for a relatively simple ensembling mechanism of taking the union of all models' predictions as our final predictions. This likely resulted in a model that had lower overall precision but higher recall. Given the sparsity of labels in our training data for some features, we felt that this tradeoff was acceptable. Using the DeBerta model and the pattern-matching model, our ensemble achieved a micro-F1 of 78.1% on a held-out test set. We unfortunately were not able to include the phrasing model due to time constraints.

4.2 Future Work

While our ensemble results were impressive, especially given the relative simplicity of our inference mechanism, others in the competition were able to achieve higher F1-scores–up to

89%. In this section we reflect on some of the methods that worked best and how we could incorporate them into our model in the future.

Ensembling was a common approach in the competition, however most ensembles consisted of multiple large-language models often all trained using masked word prediction over the entire data corpus. This pre-training has the distinct advantage of allowing for models to incorporate information from the entire corpus of patient notes, which was about 45 times larger than the labeled corpus. We therefore think that given additional compute and time, masked word prediction could be a pre-training method that significantly improves the performance of our DeBerta model.

Regularization was also frequently used to avoid overfitting to the limited examples present in the train dataset. Common data regularization methods including replacing words with synonyms, as we did, and dropping irrelevant sentences (those that do not contain labels) from the data. The latter approach seems particularly effective as we observed that most labels did not seem very dependent on the context that surrounded them. Competitors also made use of adversarial optimization methods such as SiFT [5] to make their classifications heads more invariant to changes in the features representing a token.

A final approach that we did not consider was the use of pseudo or meta-pseudo labels [2, 8]. Both of these methods use the predictions of models trained on the labeled data to then label the unlabeled data, treating the predicted labels as ground truths. This ended up being a powerful way to easily make use of the unlabeled patient note data without needing to generate many hand-crafted labeling functions as we had considered doing.

5 CONCLUSION

In this paper we have detailed our approach to tackling the NBME Kaggle Competition. We considered several models including pattern matching models, zero-shot phrase prediction models, and end-to-end deep learning models. We also effectively combined two of these models into an ensemble which was able to achieve competitive results on a hold-out test set. We concluded by reflecting on several methods that proved to be effective in the competition which we can use to improve our model. Going forward we hope to further explore other NLP competitions and apply the material we learned in 6.s079 to achieve even higher scores.

REFERENCES

[1] 2022. NBME Score Clinical Patient Notes Kaggle Competition. https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes. (May 2022).

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. 2020. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. *2020 International Joint Conference on Neural Networks (IJCNN)* (Jul 2020). https://doi.org/10.1109/ijcnn48605.2020.9207304

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *Proceedings of the 2019 Conference of the North* (2019). https://doi.org/10.18653/v1/n19-1423

[5] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. (2020). arXiv:cs.CL/2006.03654

[6] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Barcelona, Spain, 404–411. https://aclanthology.org/W04-3252

[7] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (nov 1995), 39–41. https://doi.org/10.1145/219717.219748

[8] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V. Le. 2021. Meta Pseudo Labels. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2021). https://doi.org/10.1109/cvpr46437.2021.01139

[9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

[10] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[11] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2016). https://doi.org/10.18653/v1/p16-1162