

Appendix E1

Training Procedures, Hyperparameter Tuning, and Other Design Decisions

The 'xgboost' Python package (18) was used for this study. A grid search of hyperparameters was used to select the number of rounds [5, 10, 20, 30, 50, 75, 100], the maximum tree depth [5, 10, 20, 30, 50, 75], and eta [0.05, 0.1, 0.2, 0.3]. Our final model had an eta of 0.1, 30 rounds, and a maximum tree depth of 20. Other parameters were set to their default values.

The random forest and SVM classifiers were implemented using the scikit-learn Python package. For the random forest, a grid search was used to select the number of estimators [20, 50, 100, 300, 500, 750, 1000] and the maximum tree depth [10, 20, 30, 40, 50, 75]. Our final model used 500 estimators with a maximum depth of 50. Other parameters were set to their default values. For the support vector machine, we used a linear kernel, L2 norm regularization, squared hinge loss, and class weights inversely proportional to class frequencies, although we tested other settings of these parameters.

For some of the above classifiers, single models are only capable of producing binary classification outputs, which is not sufficient for a multiclass, multilabel classification problem such as this one. For these models, we trained individual classifiers at the same time for each organ class, and aggregated their predictions to produce a complete set of organ predictions for each pathology report.

For the neural networks, we used pretrained GloVe vectors as word representations. We experimented with pretrained Word2Vec vectors but found GloVe to perform better in preliminary tests. Since we wanted to demonstrate the feasibility of out-of-the-box classifiers without significant modification, we opted not to use more sophisticated word representations as they require significantly more processing power both at train time and at inference time. We opted to use pretrained word embeddings rather than our own for similar reasons: ease of use comparable to TF-IDF, lack of additional computing time to prepare, and the potential performance benefits of representations trained on general purpose English text with a wide variety of grammatical constructions. Continuing to train the embeddings during classification training did not provide any additional performance benefit.

We used a grid search to identify the best performing learning rate [1e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2], number of units [100, 150, 200, 300, 500], and for the CNN, filter size [3,5,7,10]. There were minimal performance differences relating to number of units, with additional units beyond 200 providing no consistent benefit on the validation set (given the small size of this data set, small numeric variations in performance were likely to reflect particularities of the data set, the train-test splits, and the specific random seeds used to initialize the network weights). The same was true of adding additional convolutional or recurrent layers to the models. In particular, adding an additional convolutional layer resulted in a microaveraged F1 score of 95.9% at convergence, as opposed to 96.3% for the 1-layer network, and adding an additional LSTM layer did not change the performance of 96.5%.

Our final model sizes were the smallest which achieved this ‘plateau’ level of performance, to perform real-time inference on large report volumes on our clinical data servers, which do not have GPUs. Our final LSTM had 150 units and the final CNN had 200 units and a filter size of 7 tokens, both with a learning rate of $1e-3$. Models were trained until validation set loss stopped improving, with a patience of 3 epochs.

Given the class imbalance present in this data set (particularly reflected in the small number of adrenal pathology reports), we experimented with class-based loss weighting in the neural networks, in which models are penalized more harshly for misclassifying instances of rare classes, as well as oversampling from the rare classes in the data set. These approaches did not provide any performance gains in our models, even on the rare classes, so we did not use them in the final models.

Appendix E2

F1 Score

The F1 score is the harmonic mean of recall (sensitivity) and precision (positive predictive value) and is a commonly-used metric designed to incorporate both precision and recall into the evaluation of a classification model’s performance. Unlike the traditional *accuracy* metric, the F1 score does not include true negatives in its calculation, which is useful for classification tasks where the positives are relatively rare compared with the negatives. For instance, in this study, a system could have achieved high (> 95%) accuracy at classifying reports as relevant to the adrenal glands by simply predicting ‘not relevant’ for every report, as the vast majority of reports would be true negatives. However, for tasks like this, we are usually interested in whether a system can effectively identify the few positives in a large group of negatives, making F1 a more effective comparison metric than accuracy.

References

18. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016; 785–794.