NLP Maps Final Meeting and Sendoff

Andrew Simon, Meenal Rawlani, Shijie Zhang

Introduction

- Update on Model Optimization
- Deep dive of model performance over dataset
- Selection Embedding Algorithm
- Closing Remarks and Final Conclusions

Update on Model Optimization

- Good News: Successfully restored tensorflow models into the ensemble
- Bad News: Performs equally as well as random forest, offering no advantage over RF
- Implemented OverSampling, however just overfit the data
- Used Optuna for optimizing RF
- Strongest model is Random Forest with 1100 decision estimators for ELMo, however optimization varies amongst models

Update on bag-of-words, tf-idf and bert

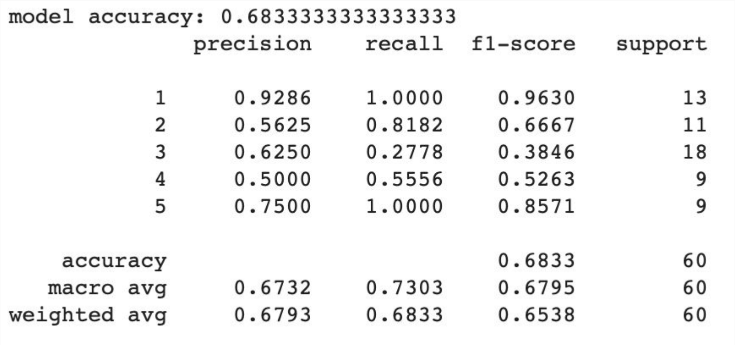
- No significant improvement observed in model performance
- Random forest classifier shows promising results
- Attempted data duplication to increase training set size, resulted in overfitting, poor performance on the test data

GloVe and fastText

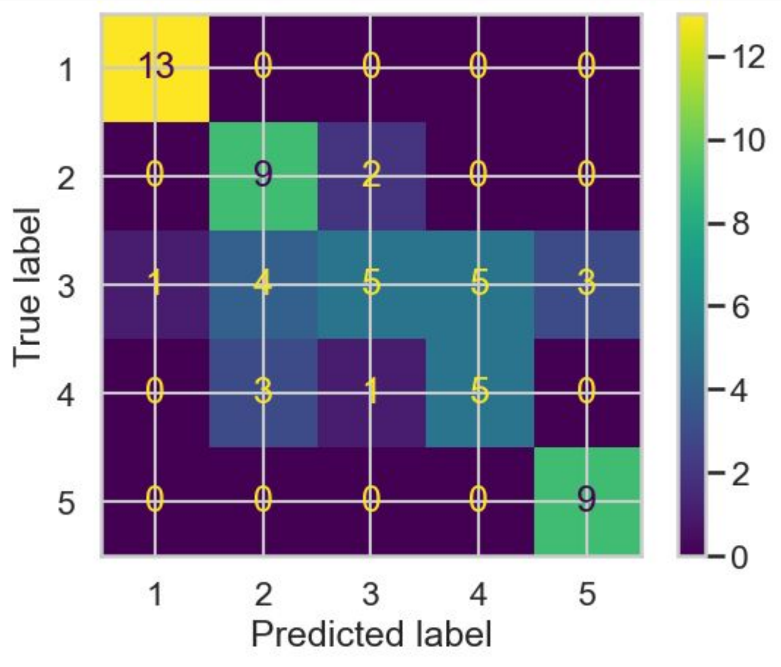
- Tried simpler linear models and more complicated models such as XGBoost, but Random Forest still performs best among them
- Implemented PCA, no significant improvement
- Using Optuna for hyperparameter hunting, both RF models of GloVe and fastText perform a bit better. Model score increases from ~0.6 to ~0.7. No further improvements for hyperparameter optimization

GloVe performance

- Level 2 performs much better than before
- Slightly better performance for Level 3&4

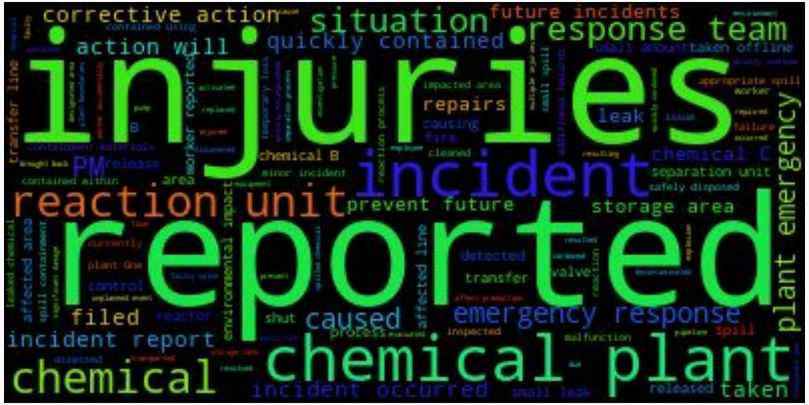


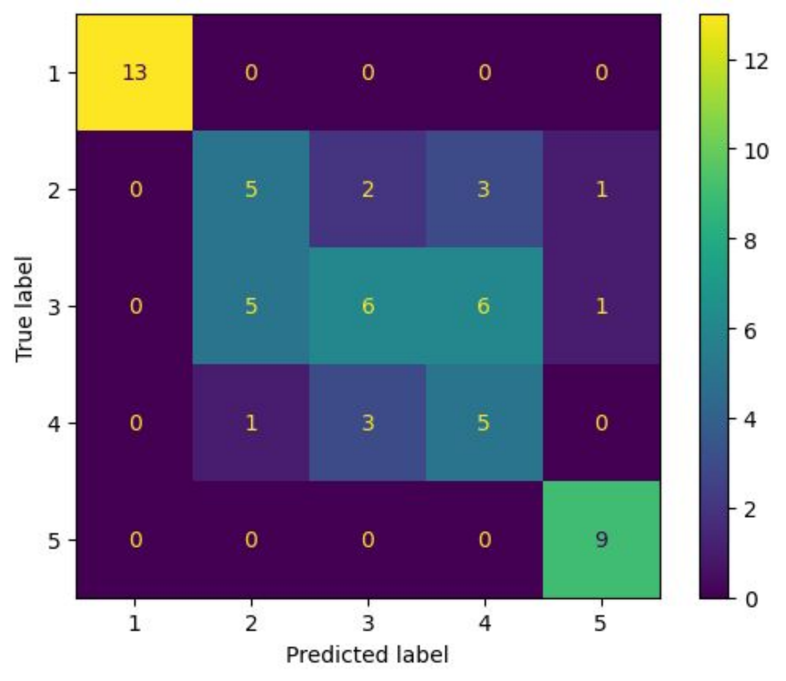
```
model accuracy: 0.6833333333333333
              precision    recall  f1-score   support

           1     0.9286    1.0000    0.9630        13
           2     0.5625    0.8182    0.6667        11
           3     0.6250    0.2778    0.3846        18
           4     0.5000    0.5556    0.5263         9
           5     0.7500    1.0000    0.8571         9

    accuracy                         0.6833        60
   macro avg     0.6732    0.7303    0.6795        60
weighted avg     0.6793    0.6833    0.6538        60
```



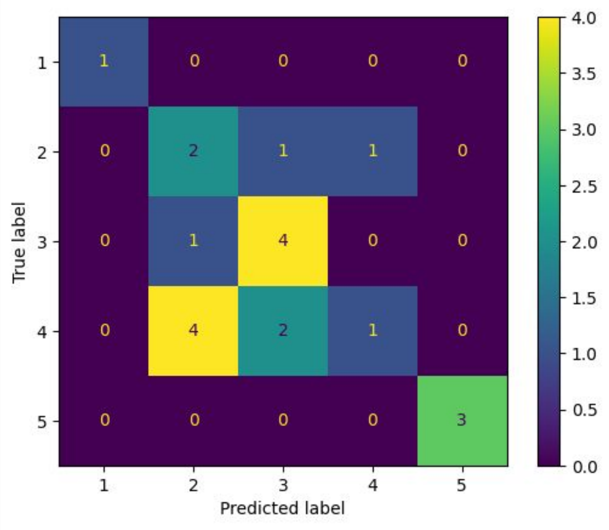
fastText performance

- fastText performance improves a bit
- Still struggles with Levels 2-4

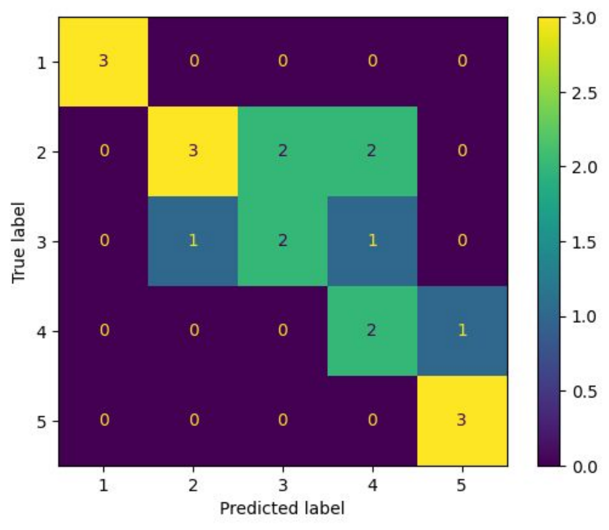




Deep Dive of performance on datasets

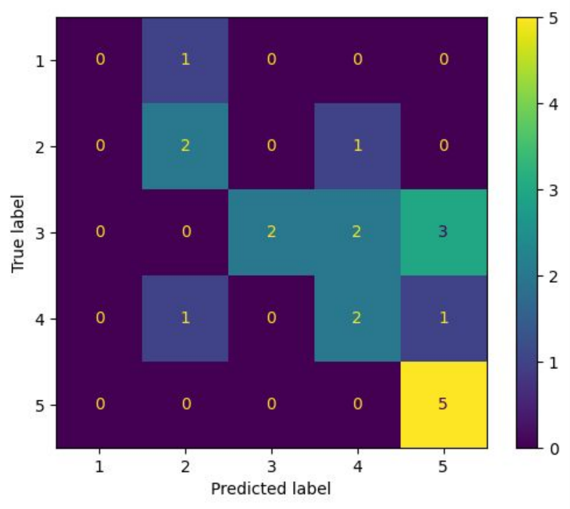


ELMo Embeddings on Same Person Reports, RF score 0.55 TF Score 0.55

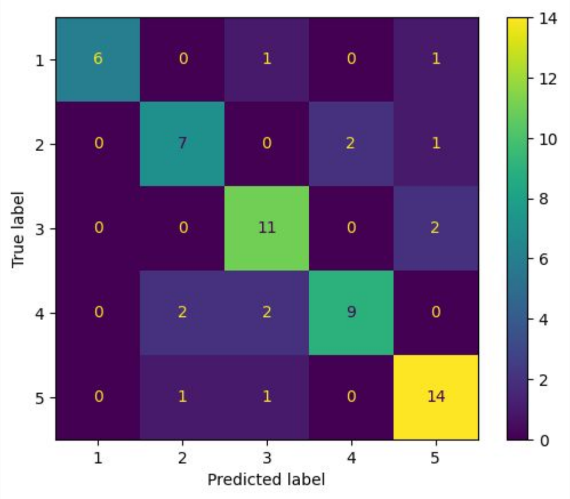


ELMo Embeddings on Multiple People Detailed Reports, RF score 0.65TF Score 0.70

Deep Dive of performance on datasets

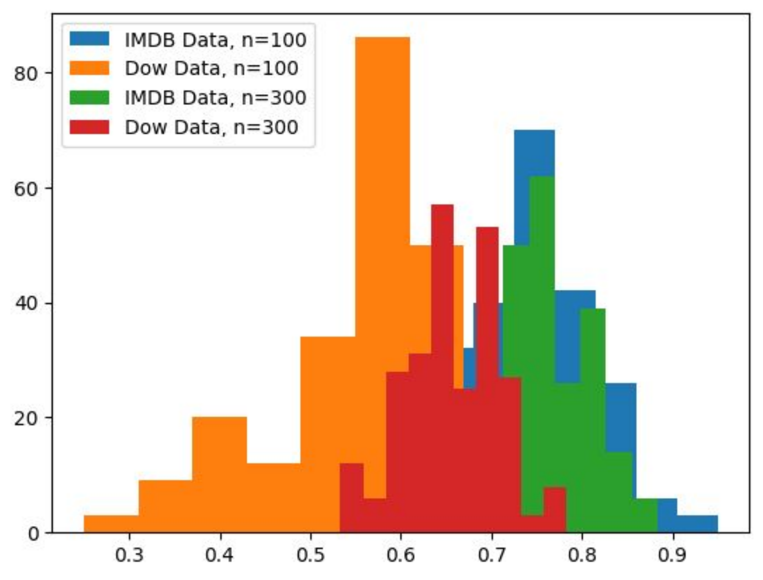


ELMo Embeddings on Multiple People Low Detail Reports, RF score 0.55 TF Score 0.65



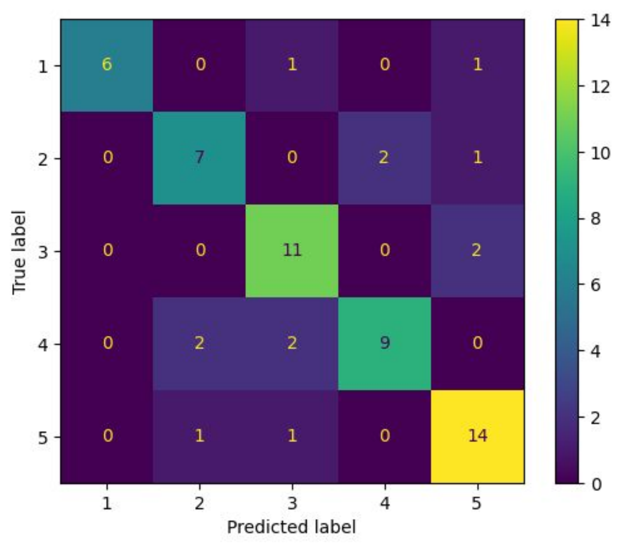
ELMo Embeddings on Concatenated data Reports, RF score 0.783 TF Score 0.65

Deep Dive of performance on datasets



- Histogram of RF scores for ELMo embedding sampled across 250 random states
- There is a high degree of variance of how machine learning models are able to perform on the data, in this case the best method to increase model performance is to increase training data
- By themselves there is little difference in model performance across sheets
- Optuna was only able to increase model performance by several points
- Tensorflow can perform marginally better on small (~100) datasets

Selection Embedding Algorithm



True label \ Predicted label	1	2	3	4	5
1	6	0	1	0	1
2	0	7	0	2	1
3	0	0	11	0	2
4	0	2	2	9	0
5	0	1	1	0	14

- Confusion Matrix Generated by substitution of intermediate level incident labels from ELMo onto Word2vec labels
- The confusion matrices are identical, meaning there is no contribution of Word2vec onto the ensemble
- Reducing the training data of ELMo reduces accuracy, a tradeoff between accuracy and computational expense

Closing Remarks and Recommendations

- Github repo will be finalized on Wednesday June 7
- Largest priority for increasing model performance is increasing training data, until then an accuracy ceiling has been reached
- Contextual embedding models are best suited for the task, with a lower level machine learning algorithm such as sci-kit learn
- Last steps are to finalize user interaction and reduce contextual embedding training times