# Assignment 2: Classifying Different Falls

**Gary Vestal, PhD Student**[1]
[1]**Emory University, Atlanta, Georgia, United States**

## Methods

### Feature Set

The data-set consisted of 116 descriptions of different falls. Of these, 81 were falls tagged with ”CoM” (center-of-mass). The data-set also data data for other types of falls, but I was only concerned with distinguishing these CoM falls from non-CoM falls, so I tagged the other 35 falls with ”Other.” There was not a very large discrepancy in the number of CoM falls vs Other falls, so I used f1-macro-score as my primary evaluation metric.

My feature set consisted of:

- Word tokens of the fall description. Tokens were stemmed, had stop-words removed, and consisted of n-grams up to tri-grams. Tokens were vectorized using a bag-of-words model.
- Tags of the words from the fall description using the 50mpaths TweetNLP cluster database[1]
- Tags of the words from the fall description using metamap[2]
- Tags of the words from the fall location using metamap[2]
- The (normalized) duration of the fall

For all tags found by metamap, all tags with a confidence score less than 1.0 were discarded. I kept 10000 features from n-grams and 1000 features from each type of tags, although it’s unlikely that this many were found for such a small data-set.

### Classification

I split the data-set into 80% optimizing and 20% evaluation. The optimizing data-set was then further split between training and testing using 5-fold cross-validation. While 10-fold cross-validation is standard, I chose to only use 5 folds, since the data-set was so small. That way, each fold would have a reasonable number of data points in its test set.

I used a Naive Bayes Classifier as a baseline, then tested five other classifiers: an SVM classifier, a Random Forest classifier, a K-Nearest-Neighbors classifier, a Neural Network with one hidden layer, and an ensemble classifier that combined the other four classifiers into one using a soft voting scheme. I optimized each one for f1-macro-score over its hyperparameter space using my 80% optimization set.

After optimization, I evaluated each classifier on my 20% evaluation set, measuring each one’s f1-macro-score, f1-micro-score, and a 95% confidence interval for its f1-macro-score using bootstrapping. My bootstrapping used 1000 iterations, each time re-sampling the test set with replacement. For each classifier, these optimal hyper-parameters, f1 scores, and 95% confidence intervals are shown in Figure 1.

### Further Evaluation of My Pipeline

Using the best classifier (which turned out to be my KNN classifier), I performed an ablation study, reevaluating my pipeline on sub-sets of my feature-set. I used 5 subsets, where each one consisted of the full feature-set with one of the five groups of features removed. Results are shown in Figure 2.

In addition, to see how the performance of my classifier scaled with the size of its dataset, I re-ran its classification with subsets of my training set of various sizes. Each time, I randomly created a training set subset by randomly sampling from my original training set. The evaluation set was the same for each training set subset. Each subset was contained enough data points to have a testing set of at least size 10.

## Results

### Classification

**Figure 1:** Metrics of Optimized Classifiers

| Classifier | Confidence Interval | F1 Macro | F1 Micro |
|---|---|---|---|
| GaussianNB() | [0.35135135135135137, 0.7473684210526316] | 0.521368 | 0.708333 |
| SVC(C=0.25, kernel='linear', probability=True) | [0.33333333333333337, 0.7] | 0.494737 | 0.666667 |
| RandomForestClassifier(n_estimators=30) | [0.3333333333333333, 0.45454545454545453] | 0.400000 | 0.666667 |
| KNeighborsClassifier(n_neighbors=1) | [0.36461844265837007, 0.7501893939393939] | 0.571429 | 0.583333 |
| MLPClassifier(hidden_layer_sizes=(71,)) | [0.35135135135135137, 0.718963068181818] | 0.521368 | 0.708333 |
| VotingClassifier(estimators=[('SVM',\n ... | [0.35135135135135137, 0.7491289198606272] | 0.521368 | 0.708333 |

Looking at the F1 scores, the classifier with the best performance was the K-Nearest-Neighbors classifier. However, looking at the 95% confidence intervals, no classifier was significantly better than any other. They were not even able to beat the Naive Bayes baseline. The fact that the top and bottom scores for the KNN classifier were higher than those of the Naive Bayes classifier suggests that it may be possible to make a significantly better classifier, but I certainly have not found it here.

Nonetheless, because the KNN classifier had the best performance, it was used for further evaluation.

### Ablation

**Figure 2:** Metrics of Ablation Study. Compare to 0.57 F1-Macro / 0.58 F1-Micro without any features removed

|  | F1 Macro | F1 Micro | Features Removed |
|---|---|---|---|
| **0** | 0.607985 | 0.625000 | Fall Duration |
| **1** | 0.571429 | 0.583333 | Metamap Location Tags |
| **2** | 0.571429 | 0.583333 | Metamap Description Tags |
| **3** | 0.288889 | 0.333333 | TweetNLP Description Tags |
| **4** | 0.469287 | 0.625000 | N-Grams |

Perhaps the one unsurprising result is that removing the n-grams hurts performance. Everything else, though, is both interesting and telling.

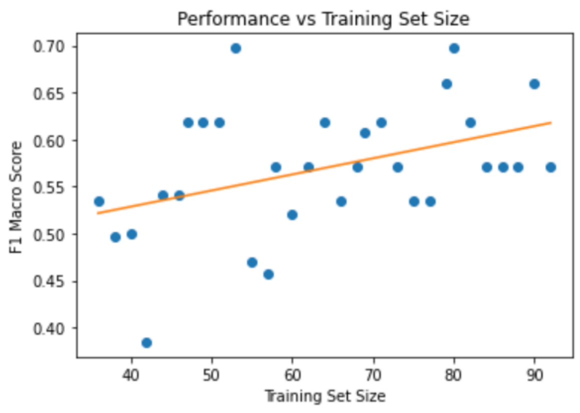
Fall duration actually proved to be a hindrance to the classifier. Not only is one's fall type irrelevant to it's duration, but there is actually a spurious inverse correlation between fall type and fall duration. The metamap tags were utterly useless to the classifier. This may be due to how many different tags metamap has compared to the size of our dataset. Metamap provided too many tags for our classifier to find any patterns in them.

This may also explain how we see such a tremendous drop in performance when we remove the TweetNLP tags. The TweetNLP tags are far more general, so there were more repeated tags in the data, giving our classifier the chance to find useful correlations. This same logic also applies to the TweetNLP tags vs n-grams: the space of n-grams used to descibe a fall was much bigger than the number of falls recorded. Thus, for our bag-of-words representation of our n-grams, there weren't enough repeated words to find many correlations, so the TweetNLP tags of these words were actually more useful to the classifier than the words themselves.

All of this suggests that we don't have enough data to effectively use sparse representations, like bag-of-words token vectors. This could be addressed in future models by using dense vector representations, or simply by using a bigger dataset.

### Performance Vs Training Set Size

**Figure 3**



Results of how my pipeline scales with training set size are shown in Figure 3. The figure shows a vague positive trend with increasing training set size. However, it is very noisy. Also, the training set is already very small, so results from subsets of that training set are even less reliable. Overall, the figure suggests that adding more data to the dataset will help, but is too reliable to be conclusive.

### Discussion

Overall, this study suggests that building a classifier capable of distinguishing CoM falls from other types of falls may be possible, but there will be significant obstacles to overcome before such a classifier is good enough for deployment.

None of my classifiers were able to beat the Naive Bayes baseline. Moreover, each of them had rather mediocre performance, with F1-macro scores around 0.57 at best. An ablation study reveals that several of the metrics used were useless (or worse, counterproductive). In addition, more dense features, like TweetNLP tags, proved to be much more useful than more sparse features, like n-grams and metamap tags. This suggests that a much better classifier could be made by removing these unhelpful features, using dense vector representations, and, as usual, using a larger dataset.

### References

1. Owoputi O, O'Connor B, Dyer C, Gimpel K, Schneider N, Smith NA. Improved part-of-speech tagging for online conversational text with word clusters. InProceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies 2013 Jun (pp. 380-390).
2. Aronson AR, Lang FM. An overview of MetaMap: historical perspective and recent advances. Journal of the American Medical Informatics Association. 2010 May 1;17(3):229-36.