CS584 Assignment 2

Vir Mittal

Code

The code for the assignment is available at `https://github.com/Vir-Mittal/CS584-Assignment-2`

Abstract

This paper analyzes the performance of an automatic classifier implementation in identifying CoM as the fall type for falls by patients diagnosed with Parkinson's disease. The paper will also compare performance across classifiers and feature set combinations to evaluate which combination gives the optimal performance.

Background

The dataset included 116 unique cases of a patient with Parkison's disease falling down. For each of these cases, the patient's description of the fall was included in a few sentences. The fall type was classified by an expert rater and could be one of the following - 'CoM', 'BoS', or 'Other'. A majority of the dataset included falls of the CoM class. Other clinical variables included were age of the patient, gender of the patient, and the location of the fall. The goal of the classifier implementation was to distinguish the CoM from the other two by generating features from the variables in the dataset and comparing performance across different classifiers.

Methods

The first step of the process was to achieve a better understanding of the dataset, especially the fall descriptions. This helped understand what the differences were between different fall types and what the classification by the expert rater was based on. It would also help in the extraction of features later as important factors would be known.

Vectorization

Once the preliminary work was completed, the next step was to vectorize each of the fall descriptions. For this, the text had to be preprocessed and a list of stemmed words was obtained for each lower-cased description using NLTK. The dataset was in the ratio 0.8 : 0.2, where 0.8 was for the training data and 0.2 was for the test data. Splitting is essential to avoid overfitting, and leaking so that performance can be measured accurately, in an unbiased way. If the data is overfitted, then the system may perform very well for the training dataset, however it would not generalize well when new data is given to it for classification.

A Count Vectorizer was then used to generate the overall vocabulary and generate the vectors for each description, based on the lists of stemmed words. An n-gram range of (1,3) was used to better tokenize the text and capture short phrases used by patients. Since the dataset was small, the maximum number of features used was limited to 200, so that the number of features is not much greater than the length of the training data. This also helps capture the most important tokens.

Classifiers

The vectors generated were then used to train automatic classifiers. For this paper, 7 classifiers were used and their performance was evaluated to choose the best one. The classifiers were Support Vector Machine, Random Forest, Naive Bayes, Gaussian Process, Ridge, and Decision Tree. Finally, a voting ensemble consisting of SVM, Random Forest, and Naive Bayes was also used.

Each of these classifiers has certain hyperparamaters that can be adjusted, giving different results. Thus, multiple tests need to be performed to get the optimal performance. A grid search method was used to find these parameters by testing different combinations based on some estimates on what the value range should be for each parameter. For example, for the SVM classifier, the C (regularization) parameter was tested for values 1, 4, 16, 64. The kernel was also tested for the values 'linear' and 'rbf'. However, tuning hyperparameters in this way also possesses the risk of overfitting. This is because the optimal parameters may be found for only the test set, but may not generalize well for other data. To overcome this, cross validation is performed on the training set. For this paper, ten fold cross validation was performed, which separates the training set into ten parts. Each repeating cycle uses 9 parts for training and 1 for testing for a particular hyperparameter. By the end of the process, the optimal hyperparameters are found and can then be evaluated on the test set.

Additional Features

In addition to the n-gram features, more features can be generated to try to improve the performance of the classifiers. Since the dataset was small, it was possible to go through the data and come up with rule-based features. These features would then need to be tested to see if they generalize well. In total, 5 additional features were tested.

The first feature was the length of the fall description. From the dataset, it seemed like patients would describe CoM falls in a more vague, longer form and describe Other falls with short sentences in the form of 'I tripped' or 'I slipped'.

The location of the fall was also vectorized and tested as a feature. For the training set, location vocabulary and vectors were generated, keeping in mind to not allow leakage of test data locations into the vectorizer. The hypothesis behind this was that patients would experience CoM falls more in their homes or in familiar environments. They would experience falls related to tripping in unfamiliar, outdoor environments.

Regular expressions were then utilized to engineer features based on commonly occurring phrases. The term, 'lost balance' or 'loss of balance' in the description seemed to be closely related to CoM falls. Thus, the description was checked for the presence of the word 'lost/loss' and 'balance', with the constraint that the two words must occur with less than 3 words in between them. The same was done for the presence of 'turned/turning' and 'quickly/fast'. The feature generated was then simply the number of occurrences of the regular expression in the description. The n-gram features would simply check if both words are present in the description, however the regex based feature additionally checks if the two words occur close together, i.e. in the same context.

Finally, word clusters were utilized to generate additional features. Word clusters group similar words together under a single concept. Thus, synonyms and spelling mistakes are also put in the same group. This helps detect similarity between sentences to a greater extent, if they contain the same concepts. For each fall description, a list of identified concepts was generated. The word clusters were then vectorized similar to how the fall locations were vectorized to generate additional features.

For each of these generated features, the feature/features numpy array (vector) was appended to the existing n-gram array (vector). The classifiers were then trained on these new vectors. The test data had to also be preprocessed and features had to be vectorized to test the performance of the classifiers.

Results

Three scores were generated for each classifier (using the optimal hyperparameters found during cross-validation). These were - Accuracy, Micro-averaged F1 score for the CoM class, and Macro-averaged F1 score. The performance was evaluated for the Micro-averaged F1 scores as the aim of the system was to detect the CoM class and this score most closely represented the performance for that metric.

Each feature was tested independently to evaluate how it impacted the classifier's performance. The results are displayed in the tables below.

Table 1 displays the performance using only n-grams as features. From the results, Naive Bayes actually performed the best with a score of 0.615.

Table 1: Scores - No additional features

Classifier	**Accuracy**	**Micro F1 score**	**Macro F1 score**
SVM	0.416	0.563	0.344
Random Forest	0.417	0.588	0.294
Naive Bayes	0.583	0.615	0.580
Gaussian Process Classifier	0.458	0.606	0.370
Ridge Classifier	0.417	0.563	0.344
Decision Tree Classifier	0.458	0.581	0.408
Ensemble Classifier	0.417	0.588	0.294

Table 2 displays the performance using the length of the description as an additional feature. The score for Naive Bayes remained the same, however the score for the SVM classifier increased.

Table 2: Scores - Length of description as a feature

Classifier	**Accuracy**	**Micro F1 score**	**Macro F1 score**
SVM	0.458	0.581	0.408
Random Forest	0.417	0.588	0.294
Naive Bayes	0.583	0.615	0.580
Gaussian Process Classifier	0.417	0.588	0.294
Ridge Classifier	0.417	0.563	0.344
Decision Tree Classifier	0.458	0.581	0.408
Ensemble Classifier	0.417	0.588	0.294

Table 3 displays the performance using the location of the fall as an additional feature. The performances across classifiers remained the same or decreased, expect for Naive Bayes which slightly increased to 0.621.

Table 3: Scores - Location of fall as a feature

Classifier	Accuracy	Micro F1 score	Macro F1 score
SVM	0.375	0.545	0.272
Random Forest	0.417	0.588	0.294
Naive Bayes	0.542	0.621	0.521
Gaussian Process Classifier	0.375	0.545	0.272
Ridge Classifier	0.458	0.581	0.408
Decision Tree Classifier	0.458	0.551	0.433
Ensemble Classifier	0.417	0.588	0.294

Table 4 displays the performance using the presence of the regular expression 'loss of balance' as a feature. The performance of the classifiers seemed to increase, especially for SVM which increased to 0.606.

Table 4: Scores - Presence of 'loss of balance' as a feature

Classifier	Accuracy	Micro F1 score	Macro F1 score
SVM	0.458	0.606	0.370
Random Forest	0.417	0.588	0.294
Naive Bayes	0.583	0.615	0.580
Gaussian Process Classifier	0.458	0.606	0.370
Ridge Classifier	0.417	0.563	0.344
Decision Tree Classifier	0.458	0.519	0.450
Ensemble Classifier	0.417	0.588	0.294

Table 5 displays the performance using the presence of the regular expression 'turned too quickly' as a feature. The performances were not impacted by this feature.

Table 5: Scores - Presence of 'turned too quickly' as a feature

Classifier	Accuracy	Micro F1 score	Macro F1 score
SVM	0.417	0.563	0.344
Random Forest	0.417	0.588	0.294
Naive Bayes	0.583	0.615	0.580
Gaussian Process Classifier	0.458	0.606	0.370
Ridge Classifier	0.417	0.563	0.344
Decision Tree Classifier	0.458	0.519	0.450
Ensemble Classifier	0.417	0.588	0.294

Table 6 displays the performance using word clusters as features. The performances increased, especially for the Ridge Classifier which increased greatly to 0.643. The performance for SVM also showed similar improvement.

Table 6: Scores - Word Clusters as a feature

Classifier	Accuracy	Micro F1 score	Macro F1 score
SVM	0.5	0.625	0.438
Random Forest	0.417	0.588	0.294
Naive Bayes	0.547	0.621	0.521
Gaussian Process Classifier	0.458	0.606	0.370
Ridge Classifier	0.583	0.643	0.571
Decision Tree Classifier	0.458	0.581	0.408
Ensemble Classifier	0.417	0.588	0.294

Finally, Table 7 displays the performance when all the features mentioned were included.

Table 7: Scores - All features included

Classifier	Accuracy	Micro F1 score	Macro F1 score
SVM	0.417	0.588	0.294
Random Forest	0.417	0.588	0.294
Naive Bayes	0.458	0.581	0.408
Gaussian Process Classifier	0.417	0.588	0.294
Ridge Classifier	0.458	0.552	0.434
Decision Tree Classifier	0.417	0.533	0.377
Ensemble Classifier	0.417	0.588	0.294

It seemed like the optimal performance would be achieved using a combination of the length of the description, presence of regex 'loss of balance', and word clusters as features with either the SVM or Ridge classifier. However, after testing combinations of classifiers and features, the optimal performance was achieved using the Ridge Classifier with word clusters as the only additional feature. The optimal hyperparameter found was using alpha (regularization parameter) = 5.0.

Discussion

It turned out that most of the additional features did not prove to be useful for the classification task. If the score remained the same, it meant that the feature was unnecessary and the system did not require it. A decrease in score indicated that the feature hindered the model's learning. It was interesting to note that for certain features, some models improved their scores, while some models did not. This shows that the feature may have been useful for specific models.

A deeper analysis of the regex based features shows that the hypothesized rules were not generalizable. Certain patients may have reported their descriptions as 'loss of balance' or 'turned too quickly'. However, others did not. This meant that the model did not find the regex based feature to be useful. This is evident in the scores as they are the same with or without the features (for the Ridge Classifier).

It is also interesting to note that both location of the fall and word clusters as features independently increased the performance of the Ridge Classifier. However, when used together, it actually decreased the performance compared to using word clusters alone.

Since the size of the dataset was small, it was also difficult to judge how useful a feature might be for a generalized case. The small test set may have contained anomalies which did not truly test the system. It was also difficult to generate more general features. The features that were generated might have been overfitted to suit the dataset. This is especially seen for the regex features as discussed above.

The word cluster feature set performed well, indicating that patients would describe their falls using the same concepts but different words, and the system was able to pick up on that.

Ablation Study

The optimal performance was achieved by the Ridge Classifier (alpha = 0.5) with words clusters as an additional feature. For this combination, an ablation study was also performed, removing one feature set at a time. The results for these are displayed in Table 8. The removal is compounding from one row to the other. For example, the first row has no features removed. The second row has length of description removed. The third row additionally removes the location of the fall (length of description is also still removed). Thus, for each row the feature for that row was removed, along with all the features in the rows above it.

Table 8: Ablation Study - Ridge Classifier

Feature Set Removed	Accuracy	Micro F1 score	Macro F1 score
None	0.458	0.552	0.434
Length of description	0.5	0.6	0.467
Location of fall	0.5	0.6	0.467
Regex 'loss of balance' check	0.583	0.643	0.571
Regex 'turned too quickly' check	0.583	0.643	0.571
Word Clusters	0.417	0.563	0.344

The scores show that the score increases when the length of the sentence and regex based 'loss of balance' check are removed, showing they were detrimental to the learning. The scores remain the same when location of the fall and regex based 'turned too quickly' are removed, showing that these features had no impact on the learning and were most likely ignored by the classifier. The scores decrease when the word clusters feature set is removed, showing that it was a useful feature.

Training Set Size Study

A performance vs. training set size graph was also generated to observe how performance changes with more/less data. This is seen in Figure 1.

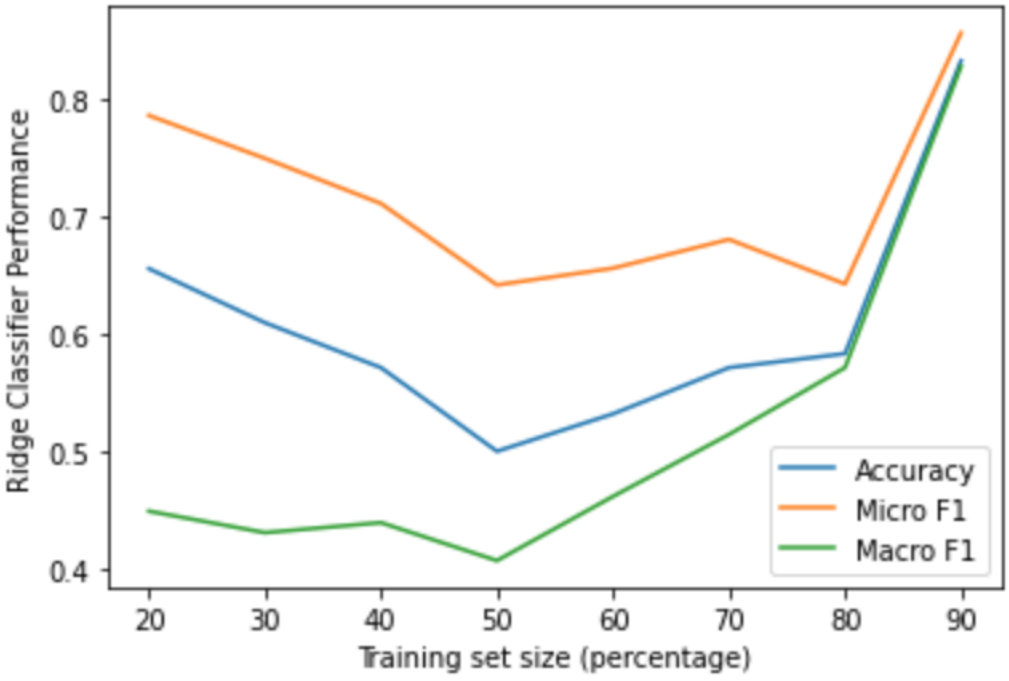

Figure 1: Performance of Ridge Classifier vs. Training Set Size

Until the training set size reaches 50 percent of the total data, the Accuracy and Micro F1 scores rapidly decrease. This is because when the training set size is small, most of the trained descriptions have a class of 'CoM', and only a few 'Other'. Thus, the classifier leans towards identifying most new test examples as 'CoM' and gets the classification correct for those examples. However, it is very bad at classifying the 'Other' class. This is seen as the Macro F1 score is much lower. Macro F1 increases with more data, as more and more diverse label examples are added to the training set, increasing performance across different labels.

After 50 percent, all of the scores are observed to increase as the classifier has more and better examples to train on, to generalize the classification. A sharp increase is seen when the training set size is 90 percent. This could be due to the resulting test set being very small, less chance for errors to occur. The trend suggests that more training data would increase the performance of the classifier. In fact, more data than the whole initial dataset can be used, and an increase will still be seen. With the current dataset, the performance does not seem to plateau yet, making it difficult to evaluate how much data might be needed to reach optimal performance.

Improvements

The classifiers might have performed better had more data been available. For the regex checking, a fuzzy matching parameter could have been included to better match results. The word clusters could have been more focused on this subject, relating to Parkinson's disease falls. More Ensembles consisting of different classifiers could have been implemented and tested, atleast including an ensemble with the Ridge Classifier. Clusters could have also been formed for location types. For example, clusters differentiating indoors and outdoors. 'Bedroom' and 'Bathroom' would be considered indoors and the label of 'indoors' might have helped in the overall classification task.

Conclusion

In conclusion, the classifier based system was not as good at identifying 'CoM' as the fall type compared to manual annotation. However, considering the dataset to train on was very small, the classifier system performed better than expected. The improvements discussed could help with some general cases that arise across cases.