**5**

In this chapter the results of the whole text mining process are presented and evaluated. The chapter starts with the test results of different tokenizer configurations. Then, based on the best tokenizer configuration, evaluations of the tokens in the chess annotation data are presented. Before applying the classifiers on the data sets, the feature models are evaluated separately by checking the most informative attributes and by applying basic queries on the word embeddings. Afterwards, the achieved accuracies for all combinations of problem, data set, model and classifier are compared and analyzed. Finally, the misclassifications are viewed using examples, confusion matrices and different cost weights.

Tokenizer Evaluation

For the evaluation of the Tokenizer configurations, a small test set with a total of slightly more than 2,000 instances is used, which, however, includes all tokens, bigrams and trigrams that occur at least two times in the data set. Thus, the model contains nearly 13,000 attributes and is based on the counts of the terms. The Naïve Bayes classifier was used for the classification, which can consider all attributes without much additional calculation effort. As a further comparison value, the one-against-all classifier (MCC0) with random forest was used. In addition to the move annotation problems (too high computational effort), an attribute selection based on the gain ratio has been applied beforehand. In table~\ref{tab:tokenizer\_statistics} the different tokenizer configurations and the percentage variance of their achieved accuracy in comparison to the one of the standard configuration are listed. Hence, improvements compared to the standard configuration are indicated by a "+".

The removal of the regular expressions for PGN or NAG non-standard codes as well as the removal of those for move- or position-annotating symbols had a slight deterioration of the accuracy as a consequence while the removal of the remis symbol as a token had no effect at all. The removal of the regular expression for abbreviations which also matches move counts, had a negative influence on the accuracy in eight of ten cases. For the regular expression for multiple dots, there were three improvements and three deteriorations, with the deteriorations weighing heavier. In the last three regular expressions for merged words, words and remaining non-whitespace characters, more than half of the accuracy values deteriorated. There are noticeable improvements for the position annotation problem with seven classes by the exclusion of words, i.e. the positions could be determined better by the counts of the letters than by the counts of the words. At the same time, however, this adaptation also led to significant deteriorations in the other problems. Overall, for all comparison configurations, the sum of the degradations is at least as large as the sum of the improvements, which is why the initial configuration of the tokenizer is retained. \\\\

The tokenizer tests were executed before the additional chess files of the Mega Database 2012 were available. For this reason, the ChessBase data sets were not included, which results in a different class distribution, especially in the move vs. position annotation problem. The high values of over 85\% accuracy in the first column can be explained by the fact that in this first test data set about seven eighths of the comments were move annotations. Thus, the baseline is even above the best accuracy. In order to have absolute certainty that the best tokenizer configuration in the tests will also perform best for the new datasets, the datasets would have to be recreated and re-tested.

Data Set Evaluation

After the determination of the tokenizer, the used comment data can be processed uniformly into tokens. For a better understanding of the data, some statistics regarding those tokens are created. In the evaluations in this section, the tokens were converted to lowercase, but not lemmatized. Filtering by language or minimum comment length was also not performed. \\\\

In the first step, the number of comments is analyzed depending on their length in tokens. The statistics are shown in table~\ref{tab:comment\_length} and figure~\ref{fig:comment\_length}. It can be seen that about half of the comments have a maximum token count of nine. 99 of these comments do not even contain a single token, since they consist only of whitespace characters. Of the remaining comments, only a small portion has a length of at least 50 tokens. Due to these circumstances, a length of three to nine tokens was defined for short comments and a length of ten to 49 tokens for long comments.

Likewise, statistics about the tokens themselves can be created. An evaluation of how many tokens, bigrams and trigrams exist with a certain number of minimum occurrences is relevant because it determines the maximum number of attributes for the count- and TF-IDF-based model. This number is given for different numbers of at minimum occurrences in table~\ref{tab:frequency\_tokens\_bigrams\_trigrams}. It should be noted that this number will be smaller if a lemmatization is applied beforehand.

Due to the many possible combinations, there is a very large total number of trigrams, which, however, decreases rapidly when the threshold is increased. In the case of bigrams, this tendency is not so pronounced. Even for a threshold value of 300 there are more bigrams than single tokens. Altogether it can be stated that a threshold of three already reduces the number of possible attributes to about 20\% and a threshold of ten even to 5\% of the original number. From the top row we can conclude that there are only slightly more than 1,000 terms which occur in at least 0.2\% of the comments. This indicates a strong variance in the comments. \\\\

Besides, additional analyses were performed to determine the most common words, bigrams and trigrams. The values can be seen in the tables~\ref{tab:frequent\_tokens},~\ref{tab:frequent\_bigrams}~and~\ref{tab:frequent\_trigrams}. It is not sure whether such frequently occurring terms are meaningful at all, i.e. whether they offer an information gain. If this is the case, however, they can be used very well as learning and decision criteria, because they can be applied to a large number of comments.

Among the most common tokens, bigrams and trigrams are mainly punctuation marks and combinations of those that do not allow intuitive conclusions to be made about certain annotation types. The only exceptions are the bigrams "1/2 -" and "- 1/2" respectively the trigram "1/2 - 1/2", which could indicate a balanced position. If the list is limited to English words, there are numerous stopwords and combinations of these, but also the two tokens "white" and "black" and the phrases "this move", "the game" or "week in chess". If you consider only non-stopwords for the tokens, then with the words "move" and "position" two further interesting tokens are among the top ten, which should be relevant at least for the classification according to move and position annotations.

Model Evaluation

Before the classification is performed, some analyses can already be done on the underlying models. In the two models, where each attribute correlates with a certain term, namely the count-based model and the TF-IDF-based model, meaningful attributes and thus terms that are relevant for classification decisions can be determined by calculating values such as the gain ratio. Table~\ref{tab:most\_informative\_attributes} shows the ten best attributes by gain ratio for the move vs. position annotation problem, the move annotation problem with two classes and the position annotation problem with three classes. In the following, no distinction is made between the count-based model and the TF-IDF-based model, since the values are very similar in both models. \\\\

The best attributes for the move annotation problem all indicate a poor move and are intuitive or at least comprehensible like count(be too). The position annotation problem also includes three intuitively useful attributes: count(white lose), count(balance) and count(equalize). For the rest, it is not obvious at first glance to what extent the classes are differentiated by the attribute. Even less obvious are the statements of the best rated attributes (except the last one) of the move vs. position annotation problem. Though, here the gain ratio values are generally lower.

The attributes shown in table~\ref{tab:most\_informative\_attributes}, however, show partly strong dependencies to each other. If the attribute count(mistake) is already selected in the case of the move annotation problem, the attributes count(mistake .) and count(mistake ,) are redundant and no longer provide any additional information gain. For this reason, the rule-based classifier J48 was used to create a classification model with rules for the three problems mentioned in order to investigate suitable attribute combinations. The learned rules are structured according to the scheme "(condition A) and (condition B) => prediction (true positive count, false positive count) and can be seen in figure~\ref{fig:move\_rules},~\ref{fig:position\_rules}~and~\ref{fig:move\_vs\_position\_rules}. Note that in each case the last rule has no condition and predicts the most frequent class.

As in the tables, the rules for the move and position annotation problem are largely comprehensible, whether intuitive or not. In addition, the comment length seems to be relevant as it is used as part of rules in both models. The rules of the move vs. position annotation problem, on the other hand, are less comprehensible and it is questionable whether they could also be successfully applied in other data sets.

For the word embeddings an evaluation like for the count- and TF-IDF-based models is not useful, because the concrete statements by the attributes of the word embeddings are unknown. Instead, the learned vectors of the vocabulary can be compared directly with each other to check whether the semantics of the comments have been captured meaningfully. For this purpose, the most similar words to the (chess) terms "check" and "blunder" with respect to the cosine similarity are compared for the word embedding learned on the chess comments (see table~\ref{tab:cosine\_similarities\_w2v\_own}) and the word embedding learned on Google News (see table~\ref{tab:cosine\_similarities\_w2v\_pretrained}).

The self-trained word embedding gives as expected a list of related chess terms, while the pretrained word embedding calculates more general terms, which should be irrelevant for the chess comments. For the two words "mistake" and "miscalculation", which occur in both tables, the self-trained word embedding has higher similarity values to the given word "blunder" than the pretrained word embedding. Furthermore, the self-trained word embedding achieves a similarity of 97.78\% for the two words "white" and "black", whereas in the pretrained word embedding it is only 80.92\%. In principle, a similarity of these terms is desired, but a similarity that is too high could make it difficult to distinguish the two words from each other and thus complicate decisions whether a position is in favor of white or in favor of black.

Classification Results

In this thesis five different problems, four different models, ten different classifiers and three different data sets were presented. For all meaningful combinations of these variables, a classification was performed and the accuracy calculated. This section summarizes these results. All accuracies of the first data set are displayed and analyzed in tabular and graphical form for each problem. The best values are then compared with those of the two remaining data sets. \\\\

In all tables the classifiers have the abbreviations as described in section~\ref{subsec:classification\_algorithms}, furthermore the word embeddings are abbreviated with w2v-C for the model learned on the chess comments and w2v-GN for the model learned on Google News. The highest value of each table is marked in bold. In the case of the multiclass classification problems, the accuracies of the two error-correcting output code classifiers (MCC1 \& MCC2) were not determined due to excessive computation time. Therefore, these two classifiers, as well as the Naïve Bayes classifier, whose values fell sharply compared to the rest, are not shown in the diagrams.

On the move vs. position annotations only the classifiers random forest, nested dichotomy and Naïve Bayes are applied because it is a binary classification problem. As seen in table~\ref{tab:total} and figure~\ref{fig:total}, a maximum accuracy of 78.25\% is achieved with the count-based model and the random forest classifier. The same classifiers are also applied to the second binary classification problem of move annotations (see table~\ref{tab:move\_1} and figure~\ref{fig:move\_1}). This time, the classifier nested dichotomy reaches the best accuracy of 76.90\%, again using the count-based model.

All ten classifiers are applied to the three multiclass classification problems. Table~\ref{tab:move\_2} shows the results for the move annotation problem with six classes and a best value of 54.90\% for the count-based model and the one-against-all classifier. It can be seen in figure~\ref{fig:move\_2} that for all classifiers the count-based model achieves the highest accuracies, closely followed by the TF-IDF model. The accuracies of the Google News word embeddings are about five percentage points below these models and the chess word embeddings even about ten percentage points below. This ranking of the models had already been indicated by the first two problems.

As well for the position annotation problem with three classes the count-based model proves to be the best (see table~\ref{tab:position\_1} and figure~\ref{fig:position\_1}). Again, the maximum value is achieved with another classifier, this time the one-against-one classifier with pairwise coupling. The accuracy value of 56.80\% is only slightly above the baseline, which corresponds to a kappa statistic of less than three percent. Previously, maximum kappa values of 56.5\% (move vs. position), 36.5\% (move, 2 classes) and 19\% (move, 6 classes) were reached. For the position annotation problem with seven classes the kappa value is again over 15\%. These values are therefore very low overall, although this is partly due to class imbalance.

The lowest accuracies by far are achieved in the position annotation problem with seven classes (see table~\ref{tab:position\_2} and figure~\ref{fig:position\_2}). The maximum value of 36,20\% is reached in the count-based model by the one-against-one classifier with pairwise coupling as well as by the class balanced nested dichotomy classifier. Again the diagram shows the clear ranking of the models: count-based-model followed by TF-IDF-based model, Google News word embedding and chess word embedding.

Overall, the results of the five tables and figures show that the count-based model is the best, followed by the TF-IDF-based model and the chess word embedding is the worst. Among the classifiers, there was no clear trend which is best. These observations are also confirmed by the other two data sets. The count-based model has the highest average accuracy holding 13 of the 15 best accuracy values (see table~\ref{tab:comparison\_accuracies}). The other two best values are reached by the TF-IDF-based model, which is partially superior to the count-based model in the data set with long comments. The nested dichotomies classifiers achieve the best values overall, whereby the accuracies are only slightly above those of the other classifiers and therefore no conclusion can be drawn. \\\\

However, it is noticeable that the data set with the short comments achieves significantly better accuracies than the mixed-length data set. The only exception is the move vs. position annotation problem. The accuracies for the data set with long comments are on average slightly lower than for the mixed-length data set. These values suggest that either the short comments are generally more meaningful or that the condition that at least three English words must occur in the comment filters out qualitatively inferior short comments, while most qualitatively inferior long comments remain in the data set.

Misclassifications

The classification results obtained in the previous section have so far only been evaluated according to the accuracy value. In this section, the misclassifications are examined and it is checked whether the previously optimal results remain optimal even after weighting the misclassifications with costs. First, the confusion matrices are used to analyze which types of misclassifications frequently occur. The confusion matrices of the ordinal class classifier in table~\ref{tab:confusion\_matrices} are considered as examples; the matrices of other data sets and other classifications show similar patterns. The binary classification problems already result in the same values as with direct application of random forest.

As desired, the confusion matrix of the move vs. position annotation problem has the highest values on the main diagonal, which is also determined by the equilibrium of the instances per class. With the move annotation problem with two classes it looks different; here class 2 (poor move) is rarer than class 1 and is predicted much less often. This even leads to the fact that class 1 is predicted more often than class 2 by the instances of class 2. The same happens with the other three problems: With the move annotation problem with six classes class 2 dominates, with the position annotation problem with three classes class 1 and with the position annotation problem with seven classes class 3 and 4. From this it can be concluded that the classifier for many comments does not find a sufficiently strong decision for any class and therefore chooses the most frequently occurring class. \\\\

However, a closer look at the misclassified instances revealed that some simple mistakes were made in addition to comments that were difficult to classify. Table~\ref{tab:misclassified\_instances} shows six examples in tokenized and lemmatized form that were misclassified by the random forest classifier in all four models. The comments have no particular complications and error-free class assignment is a prerequisite for using the classification model for non-test purposes. However, none of the classification models fulfils this requirement, so that further improvements in preprocessing, model building or classification are necessary.

As already mentioned in section~\ref{subsec:evaluation\_methods}, binary classification problems do not need to be re-examined for optimality if the cost matrix is symmetrical and contains equal values on the main diagonal. For the other three problems, the best two configurations of the accuracies are compared using the absolute and squared costs as defined in section~\ref{subsec:cost\_sensitive\_learning}. The results are shown in table~\ref{tab:comparison\_costs}, the lower costs of both configurations are marked in bold.

In three out of nine cases, the second configuration is better than the first in absolute costs, and in squared costs it is in even five out of nine cases. Of these, all five cost improvements have occurred in problems with more than three classes. This suggests that for these problems the optimization of accuracy occurs at the expense of cost. It also would have been expected that the ordinal class classifier beats the other classifiers taking into account the costs, as is the case for the mixed-length data set with the move annotation problem. It is the only classifier that takes into account the order of classes and should therefore be less likely to confuse classes that are further apart. However, this expectation could not be confirmed in general, as can also be seen in the confusion matrices of table~\ref{tab:confusion\_matrices}.