**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 annotations 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 discretion 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 loose), 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 problems. 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 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.

cost-sensitive: OCC with no better values even if order is taken into account