**1**

This chapter

Problem description

In this paper we want to find a suitable approach for the correct assignment of symbols to given chess annotations which is a specialization of a supervised sentiment classification problem. This problem is split up into four separate questions:

* the correct assignment of two symbols (!, ?) to given chess move annotations
* the correct assignment of six symbols (!!, !, !?, ?!, ?, ??) to given chess move annotations
* the correct assignment of three symbols (+-, =, -+) to given chess position annotations
* the correct assignment of seven symbols (+-, +/-, +/=, =, =/+, -/+, -+) to given chess position annotations

exact meaning of !!, !, … 🡪 Nunn Convention…

\newcommand{\rpm}{\raisebox{.2ex}{$\scriptstyle\pm$}}

**2**

This chapter

Sentiment analysis (also Opinion Mining)

* Text Mining in general
  + Information Retrieval as first step OR database
  + Linguistic analysis, NLP (POS-tagging)
* TM process
  + Structure data
  + find patterns
  + evaluate results
* TM topics
  + text categorization
  + text summarization
  + entity relation modeling
  + sentiment analysis
  + …
* TM concepts?
  + TFIDF

Plutchik

8 created emotions 🡪 Buch6

resources at WordNet

Machine learning

supervised/unsupervised/…

regression/classification

Ordinal classification / Multiple Class Classification

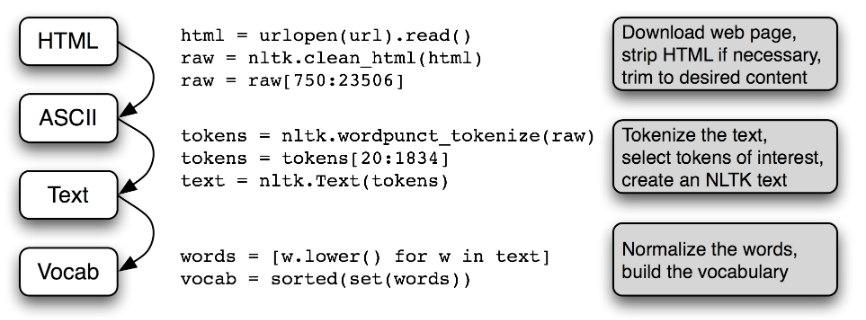
hard/soft classification 🡪 Buch2/9

boosting/bagging 🡪 Buch2/9

cost-sensitive classification in ordinal classification

Absolute cost vectors, Squared cost vectors 🡪 Buch11

NLP & NLTK



further reading chapter 7 (POS-tagging, chunking)

Word2Vec / Word Embedding

neuronal network with two layers

idea: word vector, the more similar words, the more similar the vectors

adding of vectors

comparison of vectors based on angle (0°-90°(?))

CBOW (context 🡪 word) vs skip-gram (word 🡪 context)

glove

**3**

This chapter describes how to solve a general problem of extracting knowledge out of natural language sources and defines which steps have to be considered. Therefore, in the following the focus is set on universal approaches to accomplish such a task. Specific ideas and solutions adjusted to the sentiment classification task for chess annotations will be dealt with in chapter TODO.

Picture processing pipeline of information extraction:

6 Steps (Kwartler)

1 Definition of the problem and goals

In the first step we have to define the problem we want to solve. According to (TODO: Mitchell) a learning problem generally reads as follows: Improve over task $T$, with respect to performance measure $P$, based on experience $E$. The goal is therefore to generalize the experience in a way that allows to improve your performance on the task🡪 Mitchell1997 Machine Learning

T

The task can be formulated by a simple verbal description, e.g. “play the game Connect Four” or “sort the e-mail into the categories spam and no-spam”. A more formally approach requires the exact definition of the input and the output of the task.

The input could be a set of attribute-value pairs, e.g. in relational data, where each input would have the same set of attributes. But since there is no restriction to the input data to be homogeneous the input can be plain text as well. A possible input of natural language could be represented by a whole book, by a (web) page, by a paragraph or just by one sentence. In certain cases, even a single letter is an appropriate input, e.g. for the detection of handwritten letters.

As well as for the input we need to determine the type of output we want to receive. But not only the type, also the precision in the range of values is important for the difficulty of the task. In the case of product reviews already mentioned above the easiest output “good review vs. bad review” could be complicated by using the ten values of a five-star rating or by distinguishing between different ratings for the quality, the price-performance ratio, the delivery etc. The aforementioned examples have in common that the number of possible output values is fixed which means a classification problem is concerned, not a regression problem. In the following the focus will be set on classification tasks.

P

supervised/unsupervised, instance, accuracy, confusion matrix

The straightforward approach for classification tasks would be to count the instances with the correct output and divide this number by the total number of instances. This value is called accuracy. Just as well the complementary probability for misclassified instances, called error rate, can be observed. However, there are also possibilities to weight the classifications. If there are instances that are more important than others these instances can be multiplied or be associated with a weight greater than one. Furthermore, the misclassifications can be considered separately dependent on the correct and predicted class. In spam classification, it is usually significantly worse to classify a no-spam e-mail as spam than the other way around. To map this idea, we can assign to each pair of correct and predicted class, i.e. each cell in the confusion matrix (except of those on the main diagonal), a weight value.

This type of measurement is not appropriate for regression tasks. Dependent on the exactitude of the values the probability that the predicted value is exact the correct one is low. Instead of demanding an exact prediction, we can also use the difference between correct and predicted value as the performance (MAE) or the squared difference (MSE). In the case of an unsupervised problem subjective estimates can be used.

E

The following describes the procedure as generally as possible; we only assume that we face a classification problem in the field of text mining. X is the document space; and a fixed set of classes, C = {c1 DOCUMENT SPACE , c2, . . . , cJ}, We are given a training set D of labeled documents hd, ci,where hd, ci ∈ X × C 🡪 Buch1

Furthermore, the output value should be known, such that tasks of supervised learning can be applied.

2) Identification of data

In the second step we need to find one or several data sources that offer an adequate number of instances, i.e. data sets of the previously defined input and output type. There are four characteristics to be aware of:

Completeness

Each instance of the data should be complete, i.e. we know the input and the output of the instance. All attributes of the input should be filled with a value. If they are not filled, a default value can be used or another way how to manage missing values needs to be defined. Of course, there is also a need of the completeness of the output value. Otherwise, the instance can not be used for training nor for the evaluation.

Format

In addition to the completeness, the data needs to be in the same format. Using data from data sources with different syntactical structure requires a pre-processing, such that the data can be compared and processed in a similar way in further steps. This involves the order and the atomicity or distribution of the data.

Quantity

To be able to discover meaningful knowledge, we need a minimal data volume, usually starting from several hundreds or thousands of instances. This concerns also the absolute and the relative amount for each class or even for the most important attribute values. An upper limit of instances does not exist. However, sufficient computing and storage capacities must be available.

Quality

The quality of the data goes along with the completeness, but it goes one step further. The desired values should not only be existent but also accomplish quality requirements, such as observing minimal and maximal values (numeric attributes) or lengths (linguistic attributes), providing a minimal precision or being available in a specific language. Analogous to checks and other constraints in databases, the data could be validated before using it in the mining process.

After completing the second step we end up with an appropriate data set for the task defined in the first step. The data is in a homogeneous representation, but not necessarily in a structured form.

3) Organization of the text

According to estimates, around 80 % (SOURCE) of all information in the internet exist in the form of natural language. Usually, it is hard to evaluate the information contained in this data, because they are not structured in the same way. For example, in product reviews every customer can write his comment in a different kind, so that there is neither a certain order of the information nor a specification, which information the comment should provide. However, on the basis of an additional star rating it is possible to get a fast assessment of the customer’s attitude towards the product. So, if there is a need of further evaluation, it is helpful to have the data in a structured form instead of an unstructured form.

Often it is not desirable or even impossible to get the unstructured data directly in a structured form, so we have to do the transformation on our own. This process of converting the data from an unstructured into a structured form is called information extraction. It is assumed that the input data is a single string, i.e. a sequence of characters. The sequence can contain just a few characters (e.g. tweets, comments) or thousands of characters (e.g. book contents). Even if in the most common cases this string is natural language, the procedure is similar for other input strings as well. In order not to falsify the data, we have to take into account the character encoding. The goal of this step is to reorganize the input from a string to a collection of tokens, also known as lexing or tokenization. This process takes place in three sub-steps:

Token identification

A lexical token, shortly token, is a string with an identified meaning. The input string is split into substrings where each substring without meaning is discarded and all others are stored as tokens. The definition of a token depends on the use case. In natural language processing, the standard approach is to use blanks or other whitespace characters and interpunctuation symbols as separators of tokens. As a result, there are words as tokens. This general solution can be customized by interpreting whitespace characters and interpunctuation symbols (or combinations of them) as tokens. Words can also be further split at a hyphen or split into characters. The other way around, two or more words can be combined to one token, e.g. for names of persons or places (“New York”) or for usual phrases (“and so on”). All of the parts of the string that do not contain any information should be removed directly. Frequently used words as “and” or “that”, also called stopwords, do not provide added value and can be removed from the token collection. There are lists of stopwords available for common textual resources, but they may be customized for specific data sets. Other challenges in natural language processing include handling spelling mistakes, acronyms and special characters such as smileys \autocite{Kharde2016}.

Normalization

In this step we want to find tokens that should be treated as identical, even if the strings of the token are different. In the simplest case we can make the interpretation of a token case-insensitive by converting all occurrences into lowercase. Before doing so it should be ensured that no information is lost as a result. In sentiment analysis, a large proportion of uppercase letters could indicate rage. Other types of normalization are lemmatization and stemming. Words in natural language can be modified by inflection, most of all by conjugation (modification of verbs caused by person, tense etc.) and declination (other part of speech caused by case, gender, number etc.). Lemmatization brings all parts of speech back into its basic form, e.g. the singular nominative case for nouns or the infinitive for verbs. Stemming reduces all words independent of its part of speech to the word stem, which need not be a proper word. Though, as a consequence information about the original token get lost which can lead to incompatibilities with the further analysis in the next steps. Obviously, the order of normalization and token merging or deletion from the first step may influence the result as well and should therefore be made consciously. Besides, not only the input, but also the output can be normalized. The merging of two or more output values to one joint class can also be considered as a normalization step.

Categorization

This step deals with the syntactic and the semantic analysis of the tokens. Syntactic analysis, also known as parsing, is the process of assigning each token a category describing its function in the context. In classical parsing of code files, categories can be identifiers, keywords, literals etc., in parsing of natural language resources, categories can be nouns, verbs, adjectives etc. Last-mentioned is known as part-of-speech-tagging (POS-tagging). Building on the syntactic analysis we can also assign a meaning to the token in addition to the category. This process is called semantic analysis and can cover difficulties of synonyms and polysemes (ambiguous words). These semantical relations and more for English vocabulary are provided by the lexical database WordNet (TODO: <https://wordnet.princeton.edu/>).

After finishing the chosen approaches of those three steps, we obtain the unstructured text data in a structured collection of tokens, possibly extended or replaced by their category or meaning. By splitting the string into tokens in the beginning, the tokens in the collection will be sorted in the same order as they occur in the input. The collection is in list form and the order is maintained. If the order and the count of tokens is irrelevant for further evaluation, a set can be used instead of a list. However, in most cases just the order is irrelevant but not the count of tokens, such that a multiset is the suitable form of collection. This multiset is known as “bag of words”.

4) Feature extraction

Based on the created output from step three, now the characteristics of the input data are to be figured out by feature extraction. The idea is to calculate various scores, such that we can compare the input data instances with each other, especially concerning the sentiment and polarity. At the end of this step we want to obtain a representation that can be passed as a training set to learning algorithms.

The default feature extraction approach is typically used in web search engines during the indexing step in information retrieval. The input is a list of documents in bag of words-representation or something similar. This results in a document-term-matrix, where each row describes a document and each column the count of a specific term. This concept can be generalized and transferred to our question. The column remains the description of the document as a vector whereas each column is the value of a specific feature. The model is called Vector Space Model (VSM) as it represents each document as a vector of features which simplifies the comparison of two documents by cosine similarity. Possible features with direct relation to the token are:

binary indication

This feature holds the value 0, if the token does not occur, and the value 1 otherwise. This value follows immediately if the collection is represented as a simple set.

term frequency (TF)

This feature holds the count of the token as described in the information retrieval example. This value follows immediately if the collection is represented as a multiset. The value can be normalized by dividing it by the total number of tokens in the document. Another option is to relativize very high term frequency values by using the logarithm function.

term frequency – inverse document frequency (TF-IDF)

In contrast to the term frequency, TF-IDF decreases the weight by taking into account the number of documents in which a token appears. The idea is to reward rarely occurring tokens with a high feature value. If the token appears in all documents, the value will be 0. If the token appears in just one document, the value is maximal.

Features need not be dependent of only one token. There are more advanced features that use accumulation of several tokens like the average token length. The above-mentioned values can also all be calculated if the tokens are grouped by category \autocite{Wang2010}. For problems concerning long input texts in natural languages like author detection another appropriate feature is the lexical diversity. It calculates the ratio between the count of distinct words (vocabulary) and the total word count.

If the collection is passed in list representation, we can additionally create features by using combinations of sequential tokens. These sequences are called n-grams in general; 2-grams (bigrams) and 3-grams (trigrams) are particularly frequently used. With n-grams, the context also flows into the analysis, which is an advantage, for example, when recognizing negations (“not”, “good” vs. “not good”).

Feature selection:

It makes sense to find in the first step as much features as possible while the computing and storage capacities are not exceeded. Thus, we get a large amount of potentially informative features. However, before passing the feature data to the learning algorithm, the dimension should be reduced to avoid redundancy an to accelerate the learning process. This is of particular importance for classifiers that, unlike NaiveBayes, are expensive to train. Second, feature selection often increases classification accuracy by eliminating noise features. A noise feature is one that, when added to the document representation, increases the classification error on new data. Such an incorrect generalization from an accidental property of the training set is called overfitting (TODO 🡪 Buch1, chapter 13.5).

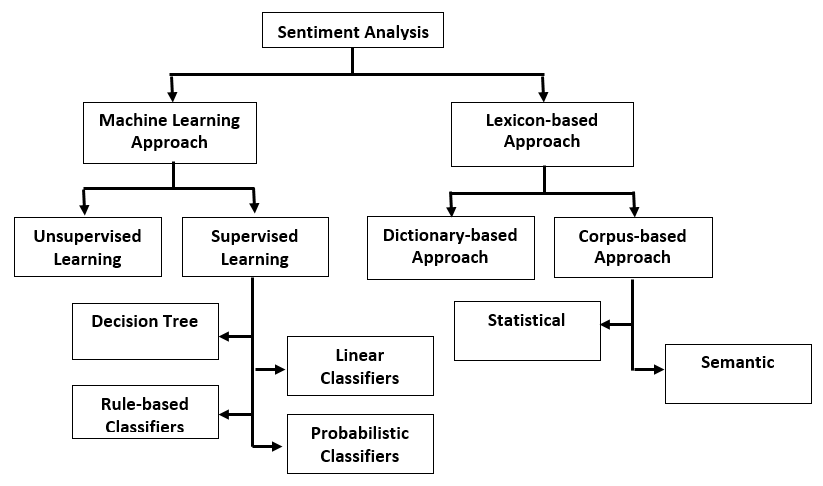
Selecting a subset of features requires a measurement value to compare the utilities of the features. Typical methods are Gini index, information gain, mutual information, \chi^2 or frequency-based feature selection (TODO 🡪 BuchMTD, chapter 6, 2.1). All of those methods are greedy, which can lead to redundant features. There are non-greedy methods that avoid redundancy, but they are rarely used in practice due to increased computing effort (TODO 🡪 Buch1, chapter 13.5). Finally, regardless of the choice of measurement features the features can be selected if they exceed a threshold value or all features are ranked and the best n features are selected.

[picture of formulas]

A different approach for representing text data are word embeddings. The previous approaches are based on the one-hot representation: the feature vector has the same length as the size of the vocabulary. Besides the already mentioned problems due to less meaningful features and overfitting, in a one-hot representation the model cannot handle words that do not appear in the labeled training data \autocite{Turian2010}. In contrast to this, word embeddings offer a distributed representation which is dense, lowdimensional, and real-valued. They are learned on a big general corpus and can therefore associate each word with a corresponding vector. The combination of those vectors yields the document vector. The features and advantages of word embeddings have already been discussed in chapter TODO.

5) Analysis

In the previous chapter, the data was prepared so that we can now apply analysis procedures to the data in vector representation. The goal of this step is to find characteristics, dependencies and rules to be able to make predictions for new input data. In figure TODO there are some possible approaches listed that can be applied to sentiment analysis problems. They can be divided into the two techniques of machine learning approaches and lexicon-based approaches \autocite{Kharde2016}.



<https://www.researchgate.net/figure/Sentiment-Analysis-Source-4-Fig-1_fig1_324360275>

Machine learning approaches use mathematical models built by an artificial intelligence to solve the task given to them. They are split into unsupervised and supervised learning. In unsupervised learning no label of a class is provided, so there is no possibility to compare the calculated solution with the correct one. In this case a common learning approach is clustering. The input data is segmented into a (fixed) number of groups, where instances within a group should have similarities and thus form a category. In contrast, there are procedures that are to determine an output value for a given input. These are in the area of supervised learning. While regression methods can handle the assignment of continuous values, classification methods can only be applied if there is a limited and fixed number of possible output values, known as classes. In the following some such classification learning approaches are introduced, which can be used in the area of text mining.

Irrespective of the choice of classification algorithm, it is recommended to split the evaluation of the procedure into a training set and a test set. Based on the training set, a model can be created that learns the characteristics of a class. Partly the training set is further subdivided, whereby a model is created on the first part of the training set and fine-tuning is performed by the second part of the set (validation set). During the evaluation the correct classes of the test instances are removed. Without prior viewing, now new classes are assigned to these instances by the learned model, which are then compared with the correct classes. If, as in the case described, only one class is assigned to each instance, it is the hard version of classifying. On the contrary, the soft version assigns to an instance a probabilistic value for each class 🡪 TODO chaguaggarwal

Decision Trees

Decision trees are designed with the use of a hierarchical division of the underlying data space with the use of different (text) features. From top to bottom, the partitions become more and more homogeneous by selection of one or several appropriate features for the decision which can be determined by measurement like information gain or Gini coefficient. This procedure can be continued until all leaves are partitions containing only one class. To avoid overfitting and reduce complexity, scarcely informative sections of the tree can be cut by pruning. Finally, a test instance is associated to the class of the partition the decision path starting from the root leads to.

Pattern- or Rule-based Classifiers

Rule-based classifiers are similar to decision trees; more precisely, decision trees can be represented as a set of rules. For each rule the left-hand side is a condition on the underlying feature set (usually expressed in Disjunctive Normal Form (DNF)), and the right-hand side is the class label. A rule should have a high support (absolute number of instances affected by the condition) as well as a high confidence (probability for class if condition is given). In text mining, the rule is typically expressed as a conjunction of terms that have to appear in the instance. The absence of terms is rarely used, because such rules are not likely to be very informative for sparse text data. For any test instance the class of the first rule where the condition is fulfilled will be assigned. That is why the last rule should cover all remaining instances and provide a default class.

Probabilistic Classifiers

Probabilistic classifiers like Bayesian classifiers calculate for each class a probability, whereby the class with the highest value is chosen for the test instance. Naïve Bayes classifiers use therefore the product of all conditional probabilities, e.g. term occurrences in different classes of texts. In comparison to other classifiers, Naïve Bayes classifiers are highly scalable because of its linear complexity. Suitable probabilistic classifiers for text mining are the Bernoulli variate model and multinomial distributions.

SVM Classifiers

Support-vector machines (SVM) are a subgroup of linear classifiers. A linear classifier calculates for a binary classification problem a linear predictor $p=\bar{A}\cdot\bar{X}+b$, where $\bar{X}=(x\_{1}...x\_{n})$ is the feature vector, $\bar{A}=(a\_{1}...a\_{n})$ is a vector of linear coefficients with the same dimensionality as the feature space. A natural interpretation of the linear predictor in the discrete scenario would be as a separating hyperplane between the different classes. The hyperplane with the maximum distance value to any instance, i.e. the one with the maximum margin of separation, is chosen. The SVM approach is quite robust to high dimensionality and ideally suited for text data because of the sparse high-dimensional nature of text \autocite{Joachims1997}.

Neural Network Classifiers

Simple neural networks are also a form of linear classifiers, since the function computed by a set of neurons is essentially linear. The simplest form of neural network, known as the perceptron (or single layer network) are essentially designed for linear separation, and work well for text. However, by using multiple layers of neurons, it is also possible to generalize the approach for non-linear separation. In such a network, the outputs of the neurons in the earlier layers feed into the neurons in the later layers. The training process of such networks is more complex, as the errors need to be back-propagated over different layers.

Proximity-based Classifiers

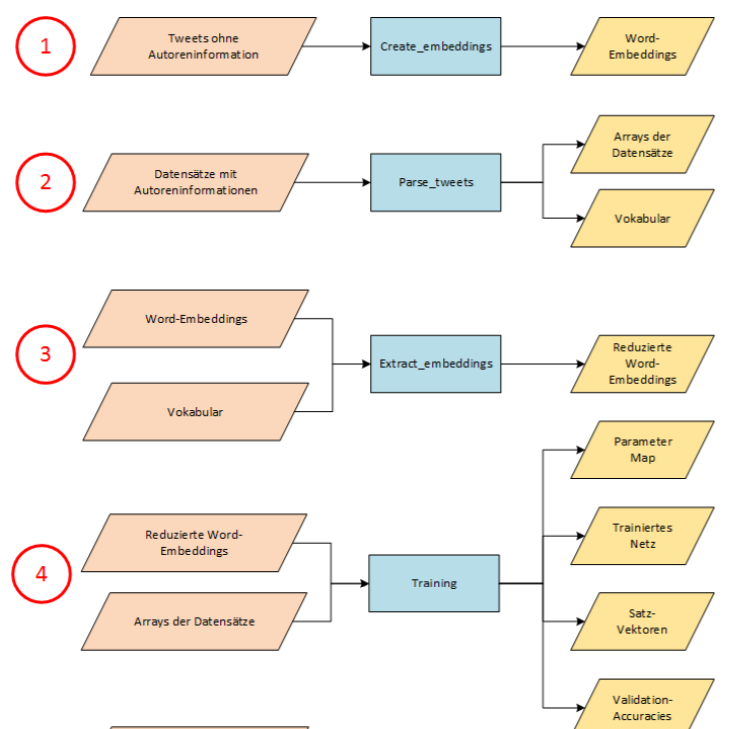
Those classifiers use proximity measures for classification. All training instances are placed in a (high-dimensional) space, proximities of two documents can be calculated by Euclidian, Manhattan or other distance measurements. Then, the most common k-nearest-neighbor classifier identifies for a given test instance the k training instances with the smallest distance values. The most common (or highest-weighted) class among them becomes the class of the test instance.

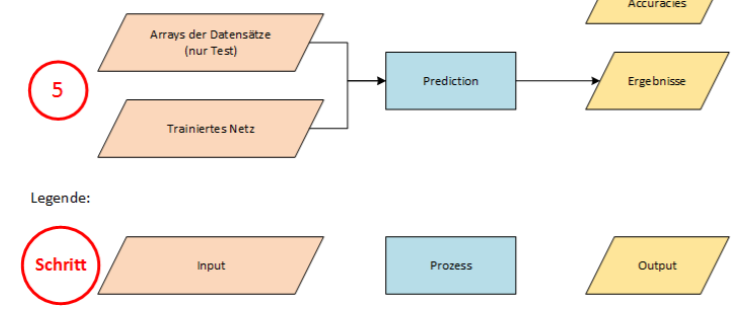
In the simplest version all of the six classifier types are applied to binary classification problems. If there are three ore more classes, the classifier possibly has to be extended to handle this multiclass problem, e.g. for SVM Classifiers. Another possibility is to split the multiclass problem into several binary problems as seen in chapter TODO. The classification is made by several classifiers that are trained to differentiate either between one class and the rest (one-against-all) or pairwise between each two classes (one-against-one). Further concepts for the customization of classifiers are boosting and bagging, the formation of ensembles or the handling of ordinal classes 🡪 TODO chaguaggarwal.

For sentiment analysis problems, apart from machine learning approaches also lexicon-based approaches are suitable. Lexicon-based approaches mainly rely on a sentiment lexicon, i.e., a collection of known and precompiled sentiment terms, phrases and even idioms, developed for traditional genres of communication. To determine the polarity score of a text, the polarity scores of the terms are combined in a certain way, e.g. by addition. There are two subtypes of lexicon-based approaches. The dictionary-based approach uses a list of terms called dictionary, where the collection and scores are both created manually. Manual creation may be time-consuming, but it is usually possible to create optimally customized dictionaries with good results. The corpus-based approach uses already existent dictionaries of a specific domain, which have been constructed based on a big corpus. There are statistic techniques like latent semantic analysis (LSA) as well as semantic techniques using synonyms, antonyms or other relationships from thesaurus like WordNet \autocite{Kharde2016}.

6) Validation and Evaluation

State of the art: accuracies for 2/3/6/7-class sentiment classification problems





**4**

In the context of this work we consider the general problem described in the previous chapter this time in the field of chess annotations.

As data sources a set of files (<http://www.angelfire.com/games3/smartbridge/>) in standard PGN format is used as well as a bundle of commented games that have been extracted from Mega Database 2012 (SOURCE) in ChessBase format. The related user interface offers the possibility to select the desired games and convert them into the standard PGN format.

However, it is not possible to filter the games by the used comment language. For this reason, an additional language detection polyglot (SOURCE) is used to reduce the comments that will be processed to the English ones.

Analyzing a commented file

In summary, we face a supervised sentiment classification problem using

* labeled chess annotations as input
* classes for traditional chess symbols as output
* accuracy as evaluation method

Numeric Annotation Glyphs (NAGs) are used to annotate chess games with assessments of moves or positions in a standard way. They are standard annotation symbols in PGN files, but can as well be used in other chess formats. A NAG is composed of a “$” followed by one or more digits. There are 140 NAGs in total:

* NAG zero is used as a placeholder
* NAGs with values from 1 to 9 annotate the move just played.
* NAGs with values from 10 to 135 annotate the current position.
* NAGs with values from 136 to 139 describe time pressure.

The most common NAGs are listed in table TODO (see chapter 10 of TODO).

As shown in table TODO, the most common NAGs have a corresponding symbol, which has been used traditionally. Those symbols are composed of the signs “!”, “?”, “+”, “-“, “=” and special signs. It should be emphasized that the subjective symbols do not mix up with the objective move symbols for check and promotion because they are used in different combinations.

Token definition by PGN

As we have already seen in chapter TODO, there are different comments in a PGN file.

Since a supervised learning approach is used, we need to know the correct class of a comment in the file. Therefore, the comments which are from importance are those connected to a traditional chess symbol or a NAG.

Weka, short for Waikato Environment for Knowledge Analysis, is an open source software offering a collection of machine learning algorithms for data mining tasks.

Weka offers standard ARFF files to experiment with and to get to know the functionality of the machine learning methods. As well own ARFF files can be imported and used. For this purpose, an ASCII text file needs to be structured like as shown in figure TODO. The file consists of two blocks, the header information and the data information. Before the header information there might be comment lines with information about the author and version or further descriptions.

The first block of the header information contains the keyword “@RELATION” and an arbitrarily name for the relation in the first line. After that for each attribute the relation contains a triple of the keyword “@ATTRIBUTE”, a unique name of the attribute and the data type of the attribute. The data type can be numeric (integer, real), string, date or nominal. For the first three data types, only the type needs to be indicated, whereas nominal attributes require a list of all possible values comma-separated in braces. The output value of an instance is an attribute as well and need to be specified, conventionally as the last attribute. In classification problems the class attribute has a fixed number of values and is represented as nominal, in regression problems it is numeric.

The second block begins with the keyword “@DATA” in the first line. After that for each instance the values of the attributes are listed comma-separated, in the same order as they were declared before. Missing values are indicated by a “?”. Data sets can consist for the most part of zero values, in particular those with attributes used in Information Retrieval. In order to reduce the creation time and the size of the file, in sparse ARFF files (figure TODO) numeric values are zero by default and can be omitted. However, now the instances can consist of a different number of values. For this reason, each instance is represented as a comma-separated list of pairs, surrounded by braces. The first number of a pair is the attribute id (starting from zero), the second one the value. Missing values are not equal to zero and need to be indicated by a “?” further on (TODO).

<https://www.cs.waikato.ac.nz/ml/weka/arff.html>

We can make an ordinal scale out of the NAGs from 1 to 6

Pairwise NAGs White/Black?

Weights in ARFF files?