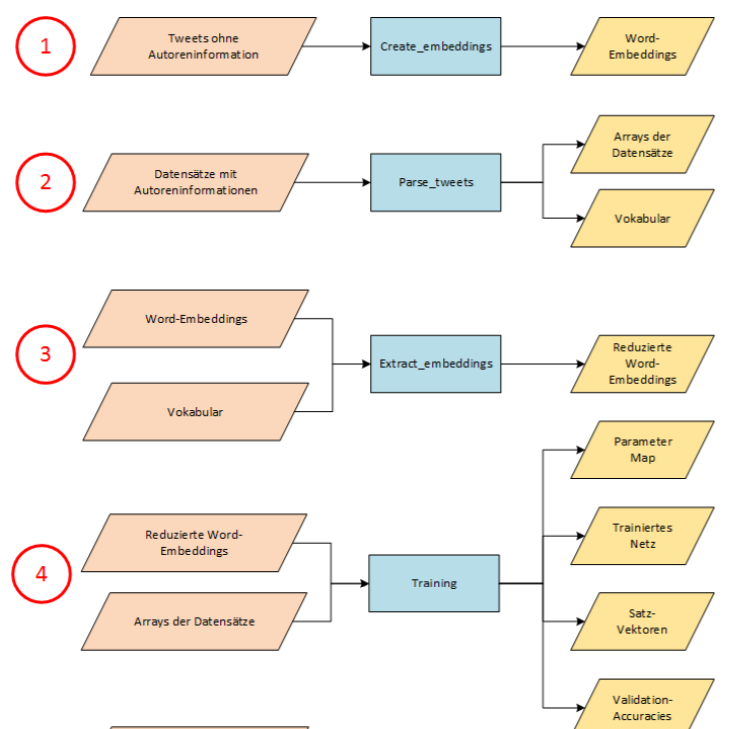
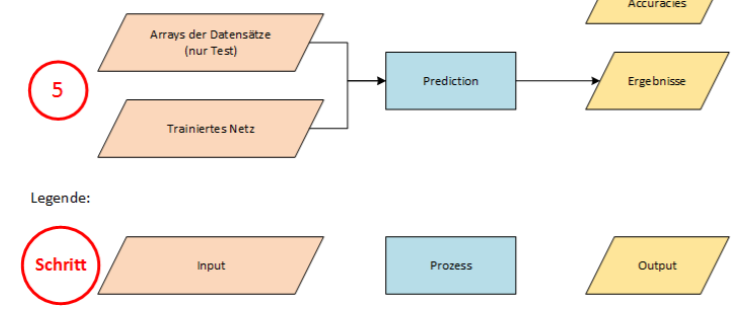
Weka Knowledge Flow for representing process

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





**4**

In this chapter we consider the general problem described in the previous chapter this time in the field of chess annotations. Therefore, in the beginning of this chapter, the basic chess game format PGN and the corresponding annotations NAGs are introduced. Afterwards we will follow the six steps on which the process is based on. In the following all necessary definitions and tools are presented and some statistics of the data are given. The results of the analysis and their evaluation are discussed in section~\ref{sec:evaluation\_of\_results}.

Problem Description

Chess games can be recorded in plain-text-files with the aim of reviewing and analyzing the game later on. Especially in professionally chess tournaments it is common to note not only the player data and the moves, but also some additional information about the game dynamics like an interim position or decisive good or bad moves. There are two ways to describe these game dynamics and annotate the chess game; either by comments in any natural language or by standardized codes and symbols like NAGs. Combinations of both variants are also common. \\\\

For the evaluation of chess games and their machine processing the unified and structured form given by the standardized codes and symbols is preferable to the unstructured form of the comments in natural language. For chess games which only contain annotations in commentary form, it would therefore be helpful to also provide them with standard codes. This results in the problem of converting the comment into an appropriate code. Using the scheme of \citeauthor{Mitchell1997} presented in subsection~\label{subsec:definition\_of\_the\_problem\_and\_goals}, we have:

\begin{itemize}[noitemsep]

\item Task $T:$ determine the correct standard code for a given chess annotation comment

\item Performance measure $P:$ percentage of correctly assigned codes in all code assignments (accuracy)

\item Experience $E:$ database with tuples of comments and correct codes

\end{itemize}

Before further specifying the input and output of the task $T$ in subsection~\ref{subsec:problem\_specification}, we will first take a closer look at the structure of the chess data. The data sets used for the experience $E$ will be presented in subsection~TODO and the performance measure $P$ will be adapted to the problem in subsection~TODO.

PGN Format

PGN is "Portable Game Notation", a standard designed for the representation of chess game data using ASCII text files. PGN is structured for easy reading and writing by human users and for easy parsing and generation by computer programs \autocite[Section~1]{Edwards1994}. A sample game in PGN notation is shown in figure~\ref{fig:sample\_pgn\_game}.

A PGN game contains first a list of tuples with general information of the game (“tag pairs”). Seven of those tags are mandatory (Seven Tag Roster: Event, Site, Date, Round, White, Black, Result), the other tags are optional. Afterwards the “movetext” section starts. The chess moves themselves are represented using SAN (Standard Algebraic Notation). A move pair (one move of white and one of black) starts with the move pair number followed by a dot and a blank, then the move of white, another blank and the move of black, e.g.

\begin{quotation}

7. Bg5 a6.

\end{quotation}

Each move contains the piece by a single upper-case letter except of the pawn (see table~\ref{tab:basic\_chess\_notations}) followed by the square the piece is moved to (see figure~\ref{fig:square\_names}). Hence, the example describes the seventh move of both players in the game; white moves his dark-squared bishop to the square g5 and black moves his a-file-pawn to a6. If a piece of the opponent is placed on the destination square, this piece is captured and in the move an "x" is inserted immediately before the destination square. In this case, if the capturing piece is a pawn, the lower-case letter of the previous file of the pawn is used at the beginning of the move, e.g. "exd5". Whenever a move pair is interrupted by a comment, the move of black is prefaced by the move pair number, an ellipsis and a blank:

\begin{quotation}

Nxf4 \$2 \{doesn't work because of\} 27... exf2+

\end{quotation}

Additionally, there are some further moves with a special notation (see table~\ref{tab:basic\_chess\_notations}). In cases of disambiguation of pieces, an additional letter for the file or a number for the rank is used. In summary, a move can contain between two and seven signs in SAN \autocite[Chapter~8]{Edwards1994}.

Parts of the moves are annotated using comments in braces. A comment can contain information about the opening of the game, about a single move or about the current position. In the last two cases the comment is often prefaced by one or several NAGs (see subsection~ref{subsec:nags}) or the corresponding chess symbol. Since there is no restriction on the exact position of a comment, comments may refer to the move before or after itself. A comment can also connect two or more moves with each other. On the contrary, a comment can be interrupted by a move such that it is split into two parts, which may only make sense when seen together. All in all, there are four possibilities of comment-move combinations shown in the examples of table~\ref{tab:comment\_move\_combinations}.

Besides, by convention there should not be nested braces, however, sometimes nested braces are used to comment different move variants separately. Those variants need not be part of a comment and are written down in parenthesis. The enumeration of the moves proceeds within a variant and is set back before a new variant starts or the game itself continues.

NAGs

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 standard NAGs in total:

\begin{itemize}[noitemsep]

\item NAG zero is used as a placeholder

\item NAGs with values from 1 to 9 annotate the move just played.

\item NAGs with values from 10 to 135 annotate the current position.

\item NAGs with values from 136 to 139 describe time pressure.

\end{itemize}

The NAGs with values from 140 to 255 are partially defined and used unofficially. The most common NAGs are listed in table~\ref{tab:meaning\_of\_nags} (see \autocite[Section~10]{Edwards1994}).

As shown in table~\ref{tab:meaning\_of\_nags}, 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.

Problem specification

Now the structure of PGN chess files and the corresponding NAGs being clarified, we can identify use cases in which a sentiment analysis of chess annotations might be useful. In PGN files, comments are often assigned to the NAG that precedes this comment. If no NAG is given - which is the case for more than half of all comments (see table~TODO) - we could assign the correct NAG automatically if we would have a reliable learned model. Thus, we will collect the data of already correctly mapped comments and NAGs and recognize contained patterns therein. Concretely, the following problems will be discussed:

\begin{itemize}

\item Classification into move and position annotations

As already seen in table~\ref{tab:meaning\_of\_nags}, the NAGs are subdivided into NAGs annotating moves and NAGs annotating positions (those describing time pressure are rarely used and therefore negligible). Both annotation types are used at the same place in the PGN file, in particular directly after a move. Therefore, we need to recognize and learn other patterns in order to distinguish these two types of annotations. For this learning problem, the input space $\mathbb{X}$ and output space $\mathbb{C}$ are defined as follows:

\begin{quotation}

$\mathbb{X}:=$ set of chess comments without annotation \quad $\mathbb{C}:=\{1,2\}$

\end{quotation}

The output class $1$ is used for move annotations and the class $2$ for position annotations.

\item Classification of move annotations

Among the move-annotating NAGs there are basically two groups of annotations; positive and negative ones. It should be noted that positivity and negativity does not refer generally to white or black, but from the viewpoint of the player with the move directly before the NAG. We can formulate the classification problem on the same input space in two degrees of difficulty:

\begin{quotation}

$\mathbb{X}:=$ set of move comments without annotation \quad $\mathbb{C}\_{1}:=\{1,2\} \quad \mathbb{C}\_{2}:=\{1,2,3,4,5,6\}$

\end{quotation}

In the first output set, the class $1$ is assigned to all positive move annotations (i.e. \$1, \$3, \$5) and the class $2$ to the negative ones (i.e. \$2, \$4, \$6). In the second output set, each of the six NAGs gets an own class ranked by their “positiveness”. This converts the binary classification problem to an ordinal classification problem with the following mapping of NAGs to classes (1 = most positive, 6 = most negative):

\begin{quotation}

$1: \$3$ \quad $2: \$1$ \quad $3: \$5$ \quad $4: \$6$ \quad $5: \$2$ \quad $6: \$4$

\end{quotation}

\item Classification of position annotations

With the position-annotated NAGs we have a similar situation, but with the decisive difference that a neutral class also exists. So even the simpler classification problem already contains three classes:

\begin{quotation}

$\mathbb{X}:=$ set of position comments without annotation \quad $\mathbb{C}\_{1}:=\{1,2,3\} \quad \mathbb{C}\_{2}:=\{1,2,3,4,5,6,7\}$

\end{quotation}

In the first output set, the class $1$ is assigned to all position annotations with an advantage of white (i.e. \$14, \$16, \$18), the class $2$ to the balanced position annotations (i.e. \$10, \$11, \$12, \$13) and the class $3$ to the annotations with an advantage of black (i.e. \$15, \$17, \$19). In the second output set, classes $1$ and $3$ are each divided into three subclasses, which makes a total of seven ordered classes (1 = best for white, 7 = best for black):

\begin{quotation}

$1: \$18$ \quad $2: \$16$ \quad $3: \$14$ \quad $4: \$10,\$11,\$12,\$13$ \quad $5: \$15$ \quad $6: \$17$ \quad $7: \$19$

\end{quotation}

\end{itemize}

Note that in all cases only one of the output values can be assigned, i.e. we only face single-label problems.

\subsubsection{Python NLTK}

RegExp parsing, tokenization, extraction of comment and class

\subsection{Prepare data for classification}

\subsubsection{Chessbase-DB}

Transformation to PGN, language detection using polyglot

\subsubsection{Feature extraction}

simple features: count(word), advanced features: tf-idf, bigrams, trigrams

\subsubsection{Generating training instances}

structure of arff-file

a brilliant counterattack of white

a big mistake of black

\clearpage

\section{Experiment setup}

to count total NAGs: $

to count total symbols: \d\w\*\+\*\s\*[!\?]+

class statistics: note that if same symbol usually NAG of is used, e.g. 36 instead of 37

\subsection{Selecting classes}

distribution how many instances per class, splitting into several problems: 2 classes (!,?), 6 classes (!!,!,!?,?!,?,??), 2 classes (+,-), 7 classes --> introduction of dictionary

difference if even or odd number of classes ("neutral class")

\subsection{Tokenizer tuning}

punctuation, special chess notations (\#ce etc.) and non-standard nags

\subsection{NLTK Parameters}

stopwords, stemming, threshold(hapax), lowercase, bigram, trigram

\subsection{Classifiers}

which classifier to use? --> MCC (x3), OCC, RF, NBM

\clearpage

Classification algorithms

Cost-sensitive evaluation

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

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?