**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 4.

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 data already existing in an unstructured form also in a structured form. 🡪 class

Picture processing pipeline of information extraction

In the first step we have to define the input and the output of the classification task. 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.

**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

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 class of an instance is an attribute as well and need to be specified, conventionally as the last attribute. In many cases the class attribute has a fixed number of values and is represented as nominal.

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?