**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

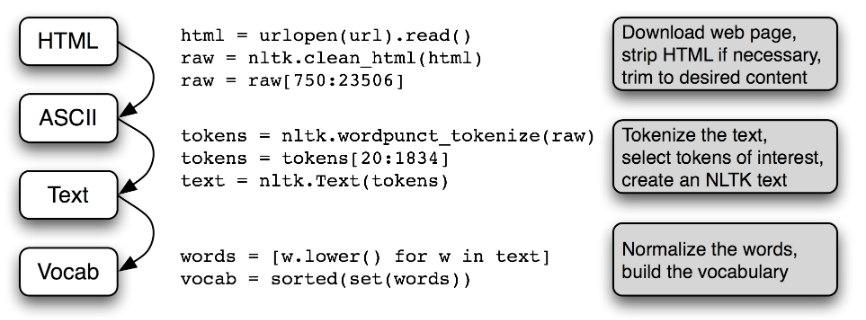
hard/soft classification 🡪 Buch2/9

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.

language

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 four 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.

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

lowercase

stemming lemmatization

The order may influence the result and should therefore be made consciously.

However, 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.

By splitting the string into tokens, 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”.

part of speech

4) Feature extraction

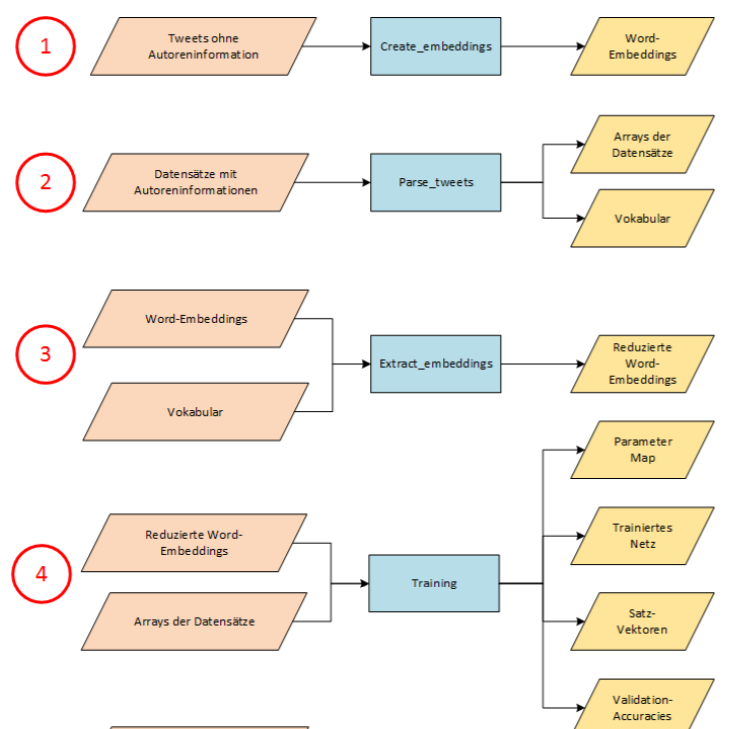
Once it is organized, you will need to calculate various sentiment and polarity scores.

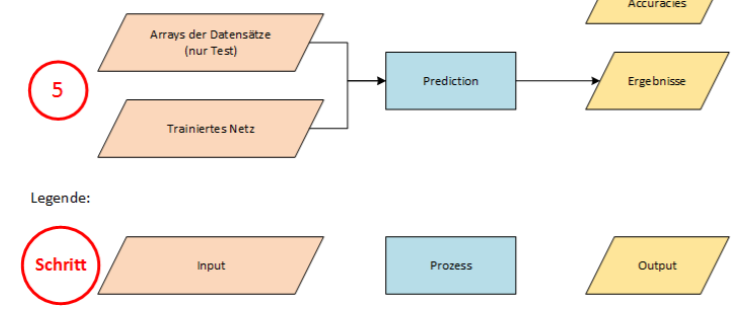
bigram trigram

5) Analyze. The sentiment and polarity scores will be used to subset the comments so that you can analyze the terms used distinctly in positive or negative comments. 4 Sentiment Scoring 89

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?