Chapter 8
Computational Methods for Text Analysis and Text Classification

In this chapter the differences between rule-based systems and machine learning-based systems along with their respective pros and cons will be explained. The principles of machine learning-based systems such as Conditional Random Fields (CRF), Support Vector Machines (SVM) and the Weka toolkit supporting several machine learning algorithms and evaluation packages will be presented. For machine learning feature extraction for improving the machine learning results will be described, feature extraction such as POS-tagging, stemming and lemmatisation, as well as statistical calculations based on tf–idf to filter out relevant words. Active learning is used for selecting the optimal data to be annotated. Different machine learning approaches such as topic modelling, distributional semantics and clustering will be presented. Text is preprocessed into different knowledge representations such as vector space model and word space model etc. These representations are adapted for different computational methods. The results produced from both rule-based and machine learning-based systems will be explained. Ready computational linguistic modules for English clinical text mining, such as MedLEE and cTakes will also be presented, as well as some basic tools such as NLTK and GATE, which need to be adapted to clinical text mining.

8.1 Rule-Based Methods

The rule-based method is the classical programming paradigm. A human programmer or software engineer writes rules to mimic the required behavior of a program. The programmer studies a flow chart of how the program should react depending on the, input data to the program. The programmer may also study the input data and the required output data and try to implement this in the program. The rules can be any type of format, a grammar for parsing text, regular expressions to extract parts of

H. Dalianis, *Clinical Text Mining*, https://doi.org/10.1007/978-3-319-78503-5_8

```
grep -o -P -e "(\d{6}|\d{8}) (-|)\d{4}(\W|$)" personnummer.txt
```

Fig. 8.1 An example of a *regular expression* that would find a Swedish personal identity number, *\d* matches digits, six or eight in a row with a hyphen or not *(-|)* and finally a group of four digits. At the end of the regular expression there is a check for a non-word character *W* and end of the line *$*. The grep command in the Bash shell script of the Linux operating system print lines matching a regular expression applied to a file called "personnummer.txt". This regular expression can be more elaborate by adding features for the century e.g. 1900 and 2000, as well as for 12 months and 31 days, and also checksum for the last four digits

strings, or a number of regular expressions to perform stemming or lemmatisation of words. Rule-based methods are time consuming and require hands-on programming and understanding of the problem to be solved. Rule-based methods are perfect for handling specific problems, but not for processing unexpected input data. See also Sect. 7.4.2 on writing grammars to parse input text and produce output data from a syntax tree. Usually rule-based method obtains high precision and lower recall.

8.1.1 Regular Expressions

Included in rule-based methods are the *regular expressions* also called *regex* or *regexp*.[1] Regexp are very powerful search and string matching techniques to make an exact match on text, words, characters, numerical expressions, non-alphanumeric expressions and parts of them. Regular expressions are available in the Linux operating systems as well as in most programming languages. Regexp are usually used to find, and sometimes replace parts or whole expressions in a text.

An example of using RegExp to find the format of a Swedish personal identity number that contains a six-digit birth date part and four control digits: YYMMDD-NNNN, follows here. A personal identity number is sometimes written in full with four year digits: YYYYMMDD-NNNN, but it can also be the case that it is written without a hyphen: YYMMDDNNNN. Examples of personal identity numbers are *690401-9304, 19690401-9304* or *196904019304*. Therefore, to find all Swedish personal identity numbers in a file called "personnummer.txt" you have to type the following command in Bash shell script of the Linux operating system. Grep in the shell script print lines matching regular expression pattern, see Fig. 8.1.

Of course this can be elaborated, allowing only a maximum 12 months and 31 days, the last four digits are control numbers and these can be also checked.

Regular expressions can be used to find and replace personal identity numbers, telephone numbers and email addresses that are regular and easy to identify for example for de-identification purposes, see Sect. 9.4.

[1]Regular expressions, https://en.wikipedia.org/wiki/Regular_expression. Accessed 2018-01-11.

8.2 Machine Learning-Based Methods

Machine learning is a set of methods that uses previous patterns of behaviour and can generalise a set of rules or behavior from this. Machine learning methods learn from previous patterns. Within machine learning methods we can distinguish between unsupervised methods and supervised methods.

Unsupervised methods utilise training data that is not annotated or preprocessed manually by any human. These unsupervised methods can be different clustering methods, distributional semantics as random indexing or Hidden Markov Models.

Supervised methods use training data that is manually annotated with labels, consequently it is very expensive in terms of human effort and time to produce annotated data for supervised methods. Hence, some supervised methods will first be described, such as Conditional Random Field (CRF) and Support Vector Machine (SVM), and then some unsupervised methods to find semantic relations, such as Latent Semantic Analysis (LSA), Latent Semantic Indexing and Random Indexing as well as text clustering.

Conditional Random Field (CRF) is an efficient machine learning algorithm for detecting named entities in a sequence of data. CRF is related to Hidden Markov Models (HMM) and is a supervised algorithm that needs annotated data. CRF can predict sequences of labels after training. For a thorough description on CRF see Lafferty et al. (2001). There are several implementations of CRF such as CRF++, Stanford NER, and CRF Mallet, to mention some. Stanford NER is a well performing system with an elaborate graphical interface, see Fig. 8.2.

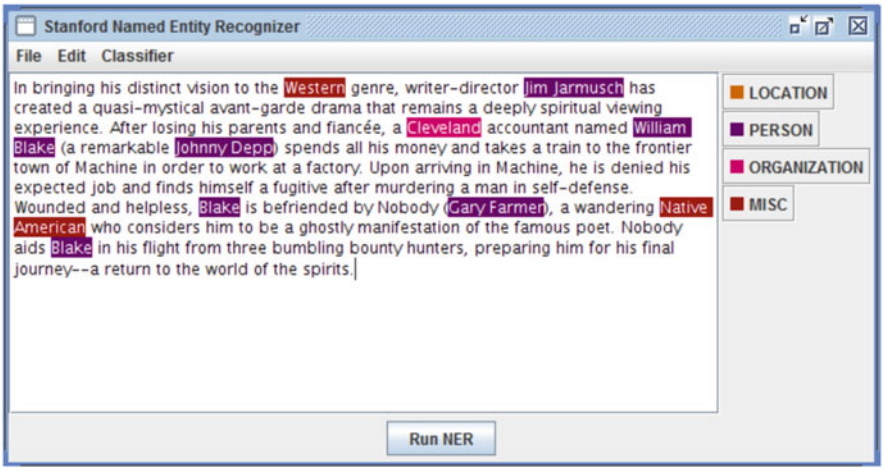

Fig. 8.2 The interface of Stanford NER[a] implementing CRF. [a]Stanford NER, http://www.linguisticsweb.org/doku.php?id=linguisticsweb:tutorials:linguistics_tutorials:automaticannotation:stanford_ner. Accessed 2018-01-11

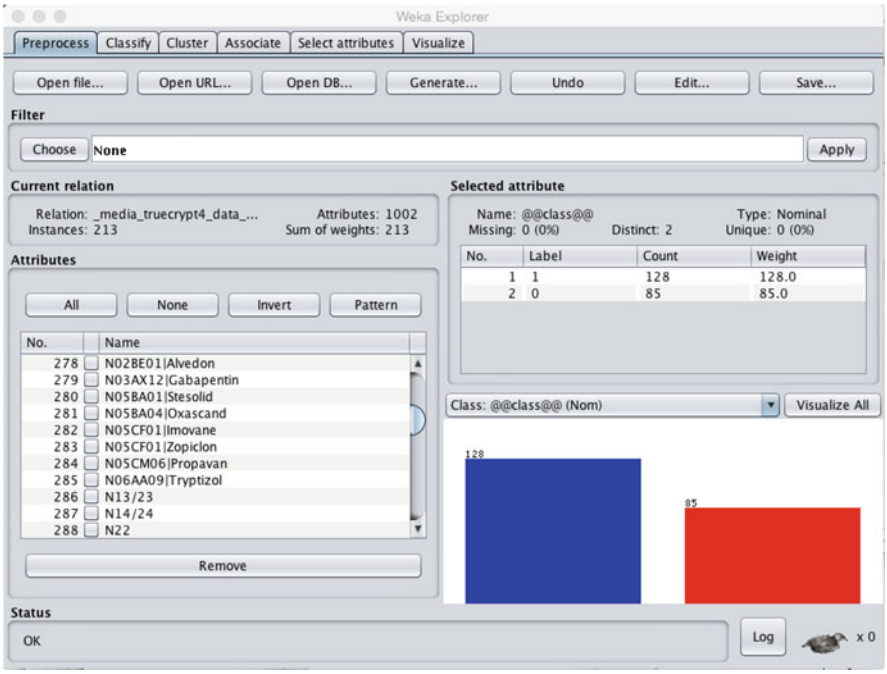

Fig. 8.3 The interface of the Weka toolkit. In this example classifying patient records for healthcare associated infections taken from Ehrentraut et al. (2014). Reprinted with the permission of the authors in Ehrentraut et al. (2014)

Support Vector Machines (SVM) is one of the most effective and popular machine learning algorithms for classification problems. SVM is a supervised algorithm and needs annotated data. SVM can, after training, decide if a concept or a whole document belongs to one class or another.

The Weka toolkit[2] contains several implemented machine learning algorithms including SVM as well as evaluation packages and an elaborated graphical interface, see Fig. 8.3.

Several researchers use the R programming language[3] with its libraries or the Scikit-learn environment[4] which is also based on the programming language Python, both are suitable for performing machine learning-based approaches.

Yet Another Multipurpose CHunk Annotator (YamCha)[5] is an implementation of the SVM algorithm and uses the CoNLL format (which is a standard format for

[2]Weka toolkit, http://www.cs.waikato.ac.nz/ml/weka/. Accessed 2018-01-11.

[3]R Project, https://www.r-project.org. Accessed 2018-01-11.

[4]Scikit-learn, http://scikit-learn.org. Accessed 2018-01-11.

[5]YamCha, http://chasen.org/~taku/software/yamcha/. Accessed 2018-01-11.

NLP training and testing) and is therefore suitable to combine with experiments using CRF++ that also use the CoNLL format.

Machine learning-based methods do not need so much programming effort but time to prepare the data in the right format, in other words, to find the optimal knowledge representation and the most efficient features in the data to be used. Supervised methods require time consuming manual annotation of the data to be used for training (and evaluation).

Machine learning-based methods generally obtain high recall and lower precision. For more details on machine learning see Alpaydin (2014).

8.2.1 Features and Feature Selection

Features represent certain aspects of the training data and are used as input to the machine learning tools. Features are usually produced by different preprocessing tools such as taggers.

Each word's POS-tag such as *determiner, subject, predicate, adjective, adverb, preposition etc.* may be used as a feature, but also features such as if the word contain an initial capital letter, the length of the word, if it is numerical token, the surrounding word's features, etc., for more details see Dalianis and Boström (2012).

Stemmers produce stems and lemmatisers lemmas that may be used as features, for more details see Dalianis and Boström (2012).

Dictionary matching means to select the words and arbitrary features of tokens that have a match with some known dictionary, such as ICD-10, SNOMED-CT, MeSH etc., for more details see Skeppstedt et al. (2014).

Statistical calculations such as the term frequency–inverse document frequency (*tf–idf*) of tokens in the document collection may be used as features, for more details see Ehrentraut et al. (2014).

Stop word filtering means to remove the most common (non-significant) words from the document collections, which account around 40% of all tokens.

Other methods to produce features is to use distributional semantics applied on large unsupervised corpora to extract features from the corpora and apply the features on a smaller subset of annotated text, for more details see Henriksson et al. (2014).

The Weka toolkit has both a built-in feature extraction mechanism as well as feature selection and feature optimisation algorithms.

Term Frequency–Inverse Document Frequency, tf–idf

To find the statistically most significant word in a document collection there is a statistical calculation called *term frequency–inverse document frequency, tf–idf*, it comprises of two separate calculations: term frequency (tf) of a word in a particular

document multiplied with the inverse document frequency (idf) of the same word over the document collection. The product gives the tf–idf weight of a specific word meaning the significance of this word within a particular document. Here follows the definitions of the used terms to calculate tf–idf:

- The *term frequency (tf)* corresponds to the number of times a word occurs in a particular document.
- The *document frequency (df)* corresponds to the number of documents that contains a specific word at least once.
- The *number of documents (N)* corresponds to the number of documents in a document collection.
- The *inverse document frequency (idf)* of a word calculates how unique or common a word is across a document collection. A unique word has a high idf, while a common term has a low idf. Idf for a specific word is calculated by dividing the number of documents *(N)* with the document frequency *(df)* for a specific word in the document collection. The logarithmic function is applied on the result to scale the quote for the length of the documents. Hence, normalising the result and avoiding that words in long documents will obtain high idf. Words in long documents tend to be repeated and consequently obtain high term frequency. For the formula on idf see (8.1) and for the formula on tf–idf see (8.2).

$$idf = log\left(\frac{N}{df}\right) \tag{8.1}$$

$$tf\text{–}idf = tf \times idf \tag{8.2}$$

Words with a high tf–idf weight are more significant than words with a lower tf–idf weight. For further details regarding tf–idf see Van Rijsbergen (1979) and Manning et al. (2008).

One preprocessing method is therefore to filter out words with a high tf–idf weight that are words with high significance, to be used as training data in a machine learning algorithm, while words with a low tf–idf weight usually coincide with stop words, meaning words with low significance.

Vector Space Model

Another similarity measurement between documents apart of tf–idf is the *vector space model* that considers each word in a document to be a vector. All the word vectors summarised gives a measurement for the document. Comparing two documents in the vector space model is carried out by comparing the angle between the two document vectors, if there is a small angle between the two vectors, the

corresponding documents are considered to be closely related. The vector space model is used in information retrieval where one of the vectors is the query vector.

The vector space model can be used both for measuring the similarity between two documents, where the document vector consists of the sum of all word vectors and also for measuring the similarity of two words by comparing the corresponding word vectors.

Cosine similarity is a measurement sprung from the vector space model, indicating the closeness of two vectors by calculating the dot product between them. If the number is 1 or close to 1 then the cosine similarity also shows similarity between the document vectors. If words are compared for similarity, then the word vectors are compared, or more precisely the angle between the word vectors, the smaller angle between the word vectors the more similar words.

8.2.2 Active Learning

Various *active learning* methods for machine learning have been developed. The aim of active learning is to reduce the amount of manual annotation effort needed to obtain training data. Active learning helps to select the most information dense and variated training data to be annotated which contributes to the best and optimal training examples. The process of active learning is often iterative. Optimal data is data not seen or used by the algorithm previously. This optimal data can be selected by a machine or by a human (Settles 2009).

Olsson (2009) has also written a nice overview of active learning within natural language processing. In his PhD thesis Olsson (2008) proposes his method BootMark that includes three steps:

(a) Manual annotation of a set of documents;
(b) Bootstrapping—active machine learning for the purpose of selecting which document to annotate next; and
(c) Mark up of the remaining unannotated documents of the original corpus using pre-tagging with revision.

The BootMark method is proved to require fewer manually annotated documents for the training of a named entity recogniser, and is better than training on randomly selected documents.

Boström and Dalianis (2012) used active learning for a de-identification annotation experiment, and found that both random selection and selecting the most certain examples outperformed the standard active learning strategy of selecting the most uncertain examples. The reason for this can be a skewed class distribution when selecting the most uncertain examples.

Kholghi et al. (2015) used three different active learning algorithms to decide on which unlabeled instances to annotate next. The studied algorithms were: supervised approach (Sup), information density (ID) and least confidence (LC). The LC

algorithm gave the best results. The study reports on a range from 77% to 46% of savings for sequences, tokens, and concepts.

8.2.3 *Pre-Annotation with Revision or Machine Assisted Annotation*

Pre-annotation with revision[6] is related to active learning. A small set of annotated data is used to start the machine learning process. The presented learned data is reviewed and corrected by a human, and the new annotated data is entered again into the machine learning system to improve the system. This is an iterative process (Olsson 2008).

Pre-annotation[7] means to machine-annotate text before the human annotator receives it to support him or her in the manual annotation process. The pre-annotations are manually corrected and missing annotations are added. The pre-annotations may also be corrected by the human annotator if the pre-annotations are wrong. The corrected annotated text is entered into the machine learning system and the performance of the system is hopefully improved.

A study on pre-annotation and revision is presented in Hanauer et al. (2013). The authors use the MITRE Identification Scrubber Toolkit (MIST) for de-identification. They use ten clinical notes for an initial annotation for de-identification and then training the system, then they pre-annotate another ten notes, correct the annotations, train the system, pre-annotate another ten notes, and do that 21 times. At the end they increase the sample with 20 and 40 notes so in total 220 notes were annotated. For each round the annotation time decreased, and the F-score increased to 0.95 from 0.89 with the initial ten notes. In total 8 h annotation time was used for 21 rounds, the initial ten note round took 33 min with the last round just needing 15 min.

Lingren et al. (2014) showed that pre-annotation of clinical trial announcements (documents) made a time saving for annotation in the range of 13.85–21.5% per entity. The annotators annotated 9002 medical named entities, mainly disease/disorder and sign/symptom entities.

Skeppstedt (2013) suggested one approach in pre-annotation inspired by Olsson (2008). Skeppstedt used the CRF system for pre-annotating unlabelled data. Instead of using the standard method of presenting one pre-annotation, the annotator is presented with two possible pre-annotations. The annotator, therefore, always has to make an active choice between two options, which has the potential to reduce bias. The two possible pre-annotations presented are the ones considered as most likely by the trained CRF model. They are, however, presented in a random order, to avoid a bias towards the most likely pre-annotation, see Fig. 8.4. This approach has not

[6] In this book pre-tagging is called pre-annotation.

[7] Pre-annotation is also called machine assisted annotation.

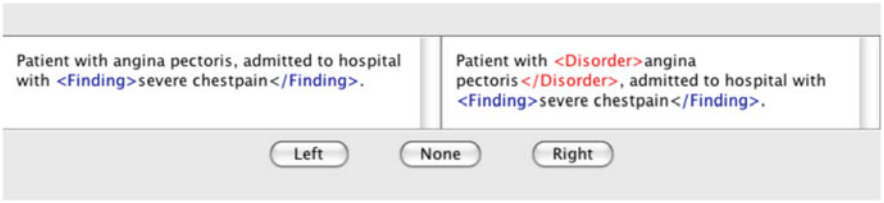

Fig. 8.4 A simple program for choosing between two alternative annotations, one pre-annotated and one manually annotated. The example is a constructed example in English (Figure taken from Figure 1 in Skeppstedt 2013. © 2013 Reprinted with the permission of ACL and the author. Published in Skeppstedt 2013)

yet been evaluated, but the results from Olsson (2008) indicate that pre-annotation is the right way to go.

One other possibility would be to present the annotator's previous annotation and the one the machine proposes to the annotator, without of course informing them which one is human-made or machine generated, and then the annotator can choose which is the correct one, thereby obtaining the optimal annotation.

Skeppstedt et al. (2017) have developed their method of pre-annotation and active learning in a prototype called PAL, Pre-annotation and Active Learning. Where the annotator only is receiving the most optimal data to annotate for each annotation round. PAL is fully integrated with the BRAT annotation tool and is freely available to download from GitHub.[8]

8.2.4 Clustering

Clustering of documents is in contrast to categorisation (or classification) not predefined. Categorisation means to assign documents in predefined categories according to some manual or rule-based process. Clustering, on the other hand, is a completely unsupervised method for grouping documents, that contain similar meaning bearing words, or are similar in some way in the same cluster. Clustering is an indeterministic process not (always) knowing the number of final clusters and their content beforehand. The process is deterministic in the way that the results will be the same each time the clustering starts. Clustering can also produce overlapping clusters, meaning that a document can be assigned in two or more clusters. Clustering needs a similarity measure between documents, the cosine similarity is often used as a measure but also the tf–idf scheme.

There are two main algorithms for clustering: partitioning and hierarchical algorithms. One well-known partitioning algorithm is the *K-means algorithm*. The

[8]PAL, https://github.com/mariask2/PAL-A-tool-for-Pre-annotation-and-Active-Learning. Accessed 2018-01-11.

K-means algorithm is given k random words as seeds to start the clustering process, then calculate cluster centroids (centre of gravity) try to fit them into nearest initial cluster, check for a stopping condition, regroup clusters, one per cluster centroid, let each document belong to the cluster with the most similar centroid, until some final clusterings are selected and a stopping condition is valid as for example the centroids stop moving.

Since the initial partitions are random the final clustering results are also non-deterministic, using the same data but different random k seed words.

Hierarchical algorithms are on the other hand deterministic and create a clustering hierarchy. The hierarchical algorithms can work top-down or bottom-up. One hierarchical algorithm is the *agglomerative clustering algorithm*, which begins by putting each document in its own cluster. The n clusters that are most similar to each other are then merged into one new cluster and the worst cluster is split into n new clusters, the splitting process repeats until a stopping condition is valid. Usually n is equal to 2. For a nice overview of the area see Rosell (2009).

For an open source search results clustering engine see Carrot^2.[9]

8.2.5 Topic Modelling

Topic modelling is, in contrary to clustering, focused on finding topics in one or more documents and then building a model of topics. Of course longer documents may contain more than one topic. Topic modelling is an unsupervised method. The method assumes that words originate from different topics and are used in a mixed way in a document or corpus. The topic modelling algorithm tries to gather all words that encompass one topic and group them in that topic. The process is iterative and continues until a likely distribution of words is put in each topic. One document or corpus can hence contain several topics. There will of course be more words than topics, since each topic contain several words. Topic modelling and clustering are related in such way that the same topics from different documents can be clustered and hence their corresponding documents. One algorithm often used to perform topic modelling is the Latent Dirichlet Allocation (LDA). For a nice overview of topic modelling see Blei (2012).

8.2.6 Distributional Semantics

The basis of the *distributional hypothesis* is that a word is described by its context. Two words are synonyms or more exactly associonyms if they are used in the same or similar context several times in different documents, which is what creates the

[9]Carrot^2, https://project.carrot2.org. Accessed 2018-01-11.

distributional semantics. The semantics of words are described by their context, or their distribution in the corpus. For each unique word in the corpus a context vector is constructed and a word space model is constructed. A *word space model* is a mathematical model of the corpus that contains information about the distribution of the different words. A distribution describes each word's context in form of other words. This method or the result of the method is also called *word embeddings*.

The first theoretical approach for distributional semantics was latent semantic analysis and it was first implemented in *latent semantic indexing*; however, the method was difficult to scale, so a faster method was later implemented called *Random Indexing (RI)* that reduced the dimensionality for the indexing and hence improved performance, see Sahlgren (2006).

An implementation of random indexing by Martin Duneld, can be found online,[10] another possibility is to use the popular *word2vec* implementation of distributional semantics[11] (Mikolov et al. 2013).

8.2.7 Association Rules

One method to reduce the complexity of big data for text mining (and data mining) is to use association rules, which is a method developed by Agrawal and Srikant (1994). Association rules use statistics to find patterns in large amounts of data and replace the data with rules that generalise or associate. The method was used in clinical text mining by Boytcheva et al. (2017a) for 300,000 outpatient records and 1425 health forum postings, both in Bulgarian. The authors tried through association rules to find attribute-value pairs. An example of an attribute is *cardiovascular system* and a value is for example *rhythmic norm frequent heartrate*.

This method generates a great number of rules by performing post processing, the rules with the highest statistical significance are selected. This method identified relations even when the attribute-value pairs were from apart from each other in the text. First as a preprocessing step the authors performed stemming on the text before generating the association rules. The authors used a ready program package called SPMP[12] for the association rules generation. The result is evaluated with something called Lift value, which is a measure on how well these rules manage to associate; the authors obtained a Lift value on 12.21, which is very good, as a Lift value of over 1.1 is considered good.

[10] JavaSDM: A Java Package for Random Indexing, http://www.csc.kth.se/tcs/humanlang/tools.html. Accessed 2018-01-11.

[11] word2vec, https://code.google.com/p/word2vec/. Accessed 2018-01-11.

[12] SPMP, http://www.philippe-fournier-viger.com/spmf/index.php?link=download.ph. Accessed 2018-01-11.

8.3 Explaining and Understanding the Results Produced

Humans trust results if they are explained and if they can be validated. This was the case with the traditional rule-based expert systems of the 1980s.

Generally, rule-based systems or logic-based systems are comprehensible since they are constructed by humans. The programmer can, for each step, explain to the user why something happened and for what reason, while machine learning systems in contrast analyse several thousands examples mathematically or statistically, and then produces a number of rules or behaviours, which in turn give a result. For example, some specific input data give some specific output data.

Machine learning systems are not very good in general of giving explanations or feedback to the user, however some machine learning algorithms such as decision trees can give some explanation of how they reached a result; linear additive systems can also give some abstract explanation, systems such as Naïve-Bayes, logistic regression and linear Support Vector Machines. These algorithms can demonstrate how the weight for each feature impacted the results (Stumpf et al. 2009).

8.4 Computational Linguistic Modules for Clinical Text Processing

There are two off-the-shelf systems that are mentioned in the research literature, one is the *Medical Language Extraction and Encoding System (MedLEE)*, by Friedman et al. (1995) the other one is *clinical Text Analysis and Knowledge Extraction System (cTAKES)*[13] by Savova et al. (2010). Both systems are ready to use with integrated clinical dictionaries, terminologies and classifications (in English).

cTAKES is an open source NLP toolkit based on UIMA and on the Apache OpenNLP toolkit.[14] It has all the basic NLP processing functionalities for English, such as a tokeniser, a POS-tagger, a named entity recogniser, negation detection, machine learning functionality etc. cTakes is currently used at the Mayo Clinic in Rochester, Minnesota, USA.

MedLEE contains a preprocessor, a rule-based parser, a composer and an encoder. The encoder matches entities in the text with entities in UMLS or SNOMED, or other vocabularies. MedLEE was developed academically but is now a commercial tool. MedLEE is in daily use for clinical decision support at the New York—Presbyterian Hospital (Friedman 2005).

A newer off-the-shelf system is *Clinical Language Annotation, Modelling and Processing Toolkit (CLAMP)*.[15] CLAMP is a Java, and Eclipse based annotation

[13] cTakes, https://en.wikipedia.org/wiki/CTAKES. Accessed 2018-01-11.

[14] Apache, OpenNLP , https://en.wikipedia.org/wiki/OpenNLP. Accessed 2018-01-11.

[15] CLAMP, http://clamp.uth.edu. Accessed 2018-01-11.

and NLP toolkit for clinical text. CLAMP has built-in modules for the standard NLP processing steps for English text. CLAMP also has a built-in UMLS encoder.

8.5 NLP Tools: UIMA, GATE, NLTK etc

Here follow a presentation of various ready to use NLP tools. One standard is *Unstructured Information Management Architecture (UIMA)*[16] from IBM for content analytics. It was developed to process unstructured information such as natural language text, speech, images or videos. Another standard that is well-used is *General Architecture for Text Engineering (GATE)*[17] written in Java. GATE was originally developed at the University of Sheffield. Today GATE can process the following languages: English, Chinese, Arabic, Bulgarian, French, German, Hindi, Italian, Cebuano, Romanian, Russian, Danish and Welsh. Another well-known toolkit is the *Natural Language Toolkit (NLTK)*,[18] which was developed in the programming language Python, see Bank and Schierle (2012). Some Java-based NLP tools are:

- LingPipe, http://alias-i.com/lingpipe/
- Stanford CoreNLP, http://stanfordnlp.github.io/CoreNLP/
- OpenNLP Apache, http://opennlp.apache.org/
- Freeling, http://nlp.lsi.upc.edu/freeling/
  (All links accessed 2018-01-11.)

8.6 Summary of Computational Methods for Text Analysis and Text Classification

This chapter presented rule-based methods and continued with machine learning methods such as CRF, SVM and Random Forest, their differences, weaknesses and strengths were compared and explained along with when to use them. For machine learning preprocessing of training data was described, including feature selection. Various knowledge representations of text were presented. Tools as the Stanford NER for CRF and the Weka toolkit supporting a large number of machine learning algorithms were presented. Within machine learning, active learning was discussed, a method for selecting the most optimal data for annotation, continuing with pre-tagging with revision, a method for improving the manual annotation work, with respect to time and quality. Clustering, topic modelling and distributional semantics

[16] UIMA Wikipedia, https://en.wikipedia.org/wiki/UIMA. Accessed 2018-01-11.
[17] GATE, https://gate.ac.uk. Accessed 2018-01-11.
[18] NLTK 3.0 documentation, http://www.nltk.org. Accessed 2018-01-11.

were also explained. Machine learnings algorithms that can explain their results were presented. Various open tools for clinical text mining such as cTakes, NLTK, GATE and Stanford Core NLP were mentioned.