Chapter 7
Basic Building Blocks for Clinical Text Processing

This chapter will describe the basics for text processing and give an overview of standard methods or techniques: Preprocessing of texts such as tokenisation and text segmentation. Word processing such as morphological processing, lemmatisation, stemming, compound splitting, abbreviation detection and expansion. Sentence based methods such as part-of-speech tagging, syntactical analysis or parsing, semantic analysis such as named entity recognition, negation detection, relation extraction, temporal processing and anaphora resolution.

Generally, the same building blocks used for regular texts can also be utilised for clinical text processing. However, clinical texts contain more noise in the form of incomplete sentences, misspelled words and non-standard abbreviations that can make the natural language processing cumbersome. For more details on the concepts in this section, see the following comprehensible textbooks in computational linguistics: Mitkov (2005), Jurafsky and Martin (2014) and Clark et al. (2013).

7.1 Definitions

Natural language processing (NLP) is the traditional term for intelligent text processing where a computer program tries to interpret what is written in natural language text or speech using computational linguistic methods. Other common terms för NLP are computational linguistics, language engineering or language technology.

Information retrieval (IR) may use NLP methods, but the aim with IR is to find a specific document in a document collection, while information extraction (IE) is to find specific information in a document or in a document collection. A popular term today is *text mining*, which means to find previously unknown facts in a text collection or to build a hypothesis that later is to be proven. Text mining is used in a broad sense in the literature sometimes meaning the use of machine learning-based

H. Dalianis, *Clinical Text Mining*, https://doi.org/10.1007/978-3-319-78503-5_7

methods. The term text mining is also used in health informatics mostly meaning the use of rule-based methods to process clinical or biomedical text.

7.2 Segmentation and Tokenisation

Segmentation in NLP is often the first step. It consists of separating sentences from each other (*sentence segmentation*), but also words from each other, (*word segmentation*). Sentences are mostly separated with a period, question mark or comma, but periods and commas may also be used in numerical values; therefore, the segmentation needs to be undertaken carefully.

In Chinese and Japanese separating words from each other is not a trivial task since there are no spaces between words. A word segmenter is therefore needed in these cases.

A text contains a stream of words and other alphanumerical characters, white spaces, interpunctuations and carriage returns, often called *tokens*. *Tokenisation* is the second step for natural language processing. Tokenisation is to decide what a token (or word) is and what should be analysed in the input stream of characters. A tokeniser is therefore needed to extract the tokens in a sentence or text, before performing the "real" natural language processing.

In Fig. 7.1 we can observe a sentence that has been tokenised. There are various choices a tokeniser must take, should sentence delimitaters be included or not? Should constructions such as *let's* be processed in one unit or not? What about the *400* mg/day? How should dosage *mg/day* be tokenised? How about *x-tra*? Usually a standard tokeniser can be used which is built-in in many natural language processing tools. The built-in tokeniser can be adapted to a new domain or a completely new tokeniser can be constructed.

Usually white spaces, sentence delimiters such as commas and question marks, are useful markers for words and can be used as cues for a simple tokeniser. However, clinical text is very noisy and contain many non-standard expressions and the non-standard abbreviations; therefore, tokenisation can be cumbersome.

In the article by Patrick and Nguyen (2011), the authors explain the problems with tokenisation of clinical text nicely and also show how to solve parts of it. They give an example of *HR 72*, consisting of both an acronym for heart rate measurement

The patient has signs of tuberculosis in his left lung, let's try a new treatment with x-tra Isoniazid, 400 mg/day.

This sentence is tokenised as

"The" "patient" "has" "signs" "of" "tuberculosis" "in" "his" "left" "lung" "," "let" "'" "s" "try" "a" "new" "treatment" "with" "x" "-" "tra" "Isoniazid" "," "400" "mg" "/" "day" "."

Fig. 7.1 Example of a sentence in a clinical text and its tokenisation

and the value of *72*, which need to be treated as one unit *heart rate 72*. Typically also dates such as *3/7/02*, need to be treated as single units.

7.3 Morphological Processing

The next step is morphological processing of each token. Morphological processing is to analyse the morphemes of the words, the different parts as inflections both such as prefixes, infixes or suffixes. Some of the tokens form multiword expressions, that is two or more consecutive tokens which should be processed jointly, for example *heart rate*. Clinical text also contains combinations of abbreviations and full forms making the morphological processing difficult.

7.3.1 Lemmatisation

Lemmatisation is the process of finding the base form of a word. It is specifically useful for inflected languages, such as German, Polish, Russian, Swedish etc. English is on the contrary a language with simple morphology and does need very advanced lemmatisation. The lemmatisater will process the words to their base form or lemma. Lemmatisation makes the variation of the same word with different inflection less frequent, by collapsing the different inflected forms of the same word into one lemma, and makes it easier in many natural language processing systems to process words and also their meaning. There are many different off-the-shelf lemmatisers available to use. Lemmatisers are often built-in in taggers, that decides on the word class or function of a word, see Sect. 7.3.6.

Regarding inflected languages such as Swedish and German, many meaning bearing words such as nouns are inflected, both in plural but also in determined or undetermined form. In German the four noun cases affect the inflection of the noun. Therefore, these should be *lemmatised*, to obtain the base or lemma form.

7.3.2 Stemming

A *stemmer* can sometimes be used instead of a lemmatiser. A stemmer is a more basic or raw form of lemmatisation. A stemmer performs stemming on the word and reduces the word to a stem, which may not be a real word (lemma) but a representation of the word. For example, the inflected words *pathology, pathologies, pathological* and *pathologically* can be stemmed to *pathology*, which actually happens to be real word.

Useful stemmers are found in the Snowball system[1] available on GitHub. The GitHub website for the Snowball stemmer also contains a set of stop word lists for different languages that can be used for preprocessing of corpora. Stop words are non-functional words such as *and, or, in, on, also, with* etc., usually these are high frequency words constituting approximately 40% of the words in a text. There are usually around 200 typical stop words in each language.

Stemming usually increases recall and decreases precision in a retrieval setting, but not always, it may also increase precision (Carlberger et al. 2001).

7.3.3 Compound Splitting (Decompounding)

In compounding languages, such as Swedish and German, some of the tokens should be decompounded and eventually processed to their base form. Compound splitting or decompounding is performed using dictionaries, in combination with rules for decompounding.

For example, in Swedish *diabetespatient* "patient with diabetes" should be decompounded to *diabetes patient*, or the plural form *diabetespatienter* "patients with diabetes" should be decompounded to *diabetes patienter*. The decompounded plural form can then be lemmatised to *diabetes patient*. As *diabetes* is in its base form already the lemmatiser does not need to do anything, but *patienter* is in its plural form and needs lemmatisation to *patient*, its lemma form.

Abbreviations can be compounds of regular words and abbreviations, as in the Swedish clinical abbreviation *lungrtg* meaning *lungröntgen* (chest X-ray). The equivalent in English is *chestx* meaning *chest X-ray*. To perform compound splitting or decompounding one needs dictionaries, and also rules for compounding words. Another issue is how far the decompounding should be carried out not to remove the meaning of the word so it becomes meaningless.

A decompounder for Swedish medical words can be found on the DSV website.[2]

7.3.4 Abbreviation Detection and Expansion

Clinical text contains a proportion of abbreviations ranging from 3% to 10% of the total text, see Sect. 4.4. Many of the abbreviations, up to 33%, are ambiguous; therefore, they need to be disambiguated.

To process abbreviations first they need to be detected and then expanded (or normalised). There have been some studies to solve this. One approach for English

[1] Snowball system, http://snowballstem.org. Accessed 2018-01-11.

[2] Medical Decompounder for Swedish, http://dsv.su.se/en/research/research-areas/health/medical-decompounder-for-swedish. Accessed 2018-01-11.

clinical text was carried out by Xu et al. (2007). One simple method used to detect abbreviations was to match them to dictionaries, both standard and medical dictionaries and taking into account morphological variants of the words. The words that did not give any match were either possible abbreviations or possible misspellings.

Another method is an *heuristic* (or *rule of thumb*) rule-based method based on the form of abbreviations in clinical text, for example words fulfilling one of the following criteria (Xu et al. 2007):

1. If the word contains any special characters such as hyphen "-" or a period ".".
2. If the word contains less than six characters and also contains:

   (a) a mixture of alphabetic and numeric characters.
   (b) capital letter(s), but not followed by a period or first capital letters.
   (c) words with lower case letters, but not contained in any word list.

For Swedish, Isenius et al. (2012) used similar heuristic rule-based methods as Xu et al. (2007) in their prototype *SCAN: A Swedish clinical abbreviation normalizer* to detect abbreviations utilising common abbreviation patterns such as hyphenations etc., but they also decided to check words with a length from three to eight characters as possible abbreviations. The performance of the heuristic rule-based method was measured to a precision of 81% and a recall of 76%, evaluated on 2050 abbreviations in a clinical text manually annotated by a senior physician.

SCAN was developed further for the detection and expansion of abbreviations in clinical text. SCAN obtained an F-score of 0.85 for detection of abbreviations in assessment entries from an emergency department and an F-score 0.83 for detection of abbreviations in radiology notes from a radiology department. Regarding expansions of abbreviations, SCAN obtained 79% correct expansions for the assessment entries and 61% correct expansions for the radiology notes. A set of abbreviation lexicons for Swedish clinical text was created and is available on the DSV website.[3] They are in the form of the abbreviation and the expanded abbreviation (Kvist and Velupillai 2014).

For abbreviation expansion and synonym extraction for Swedish, Henriksson et al. (2014) used a combination of two distributional models Random Indexing and Random Permutation, applied both to a Swedish clinical corpus and also to Swedish medical scientific corpus, *Läkartidningen*. The best results were obtained for a combination of semantic spaces originating from a single corpus. The best results for a list of ten candidate terms, measured in recall, were for the following three tasks: Abbreviations to long forms gave 39% recall, and for long forms to abbreviations gave 33% recall, when evaluating two different terminologies containing these terms.

Tengstrand et al. (2014) also used a distributional semantic approach but combined it with Levenshtein distance to choose the correct candidate among seman-

[3]Swedish Medical Abbreviations, http://dsv.su.se/en/research/research-areas/health/swedish-medical-abbreviations. Accessed 2018-01-11.

tically related words. The domain was Swedish radiology reports and Swedish scientific medical text *Läkartidningen*. Filtering and normalisation with Levenshtein distance gave an improvement of 22% compared with only using distributional semantics which gave correct expansion of the abbreviation in 40% of the cases.

There was also a shared task as part of the ShARe/CLEF eHealth Challenge 2013 to detect and expand abbreviations and acronyms (which the authors called normalising) to aid patients understanding of clinical text, as described in Mowery et al. (2016). The results were state of the art, but acronyms and abbreviations with high ambiguity and two or more meanings were particularly challenging for the normalisation systems.

A Machine Learning Approach for Abbreviation Detection

Xu et al. (2007) also used a machine learning-based method for abbreviation detection. The features used were the same as in their rule-based method. The best results were obtained when using the machine learning algorithm j48 decision tree in the Weka toolkit, extended with external resources from UMLS as well as the average document frequency of a word in the analysed clinical text, giving a precision of 91.4% and a recall of 80.3%.

Wu et al. (2011) used the machine learning algorithm Random Forest trained on 1386 annotated abbreviations in a small English clinical corpora, containing in total 18,225 tokens. The authors obtained an F-score of 0.948, a precision 98.8% and recall of 91.2%.

Regarding expansion or normalisation of abbreviations found in clinical text, Wong et al. (2006) describe a method where they use a combination of an abbreviation dictionary and the Aspell spelling correction algorithm that makes suggestions on possible expansions.

Zeng et al. (2012) used topic models trained on English clinical text to expand queries for an information retrieval task, their topic model approach improved F-score and recall by up to 0.38, but decreased precision when compared with the baseline method.

For more on expansion of terminologies see Sect. 5.6.3.

7.3.5 Spell Checking and Spelling Error Correction

Spell checkers consist of two parts *spell checking* and *spelling error correction*. Spell checking, or spelling error detection is the process to find out if a word is misspelled, if it is misspelled it can then be corrected. This can be done by performing lemmatisation on the inflected word, and matching the lemma to a dictionary to confirm the spelling.

To correct the misspellings the *Damerau–Levenshtein distance*, also called edit error distance, or just edit distance, can be used. Common errors can be corrected using the following four edit errors operations: *insert, delete, substitute* or *transpose*. Each operation changes a single character in a string and the number of operations is counted. The minimal number of operations is counted until a correct match is obtained with the corresponding word in the dictionary. These four operations cover almost 80% of the possible human-made misspellings performed using a keyboard. The edit distance for a misspelled word rarely exceeds two operations.

The Damerau–Levenshtein distance is sometimes called The Levenshtein distance, but then the *transpose* operation is excluded (Damerau 1964; Levenshtein 1966).

Soundex is an older phonetic spelling algorithm used to match words as they are pronounced in English, homophones are encoded similarly so they can be matched. The Soundex algorithm was used mainly for matching similar spelling of names. Kukich (1992) has written a good overview of the spelling correction area.

The next step after spelling error correction is grammar correction, but it will not be discussed in this book since, to our knowledge, it has not yet been carried out for clinical text.

Spell Checking of Clinical Text

Wong and Glance (2011) normalised noisy English clinical progress notes. These contained both misspellings and abbreviations. The authors used web data as a dictionary for their spelling error correction system, together with statistical information derived from the web including the distributional behaviour or what the authors call statistical semantics. The system also calls on the occasional interaction of clinicians, and the system learns and improves from the human interaction. The spelling error correction system obtained 88.73% accuracy.

Ruch et al. (2003) constructed a spelling error correction system for French clinical records. The system is organised in three modules: The first module is a standard context-independent spell checker. The second module tries to rank the candidates from the first module using morpho-syntactic disambiguation tools (part-of-speech tools (POS)) and the third module processes words using the same part-of-speech (POS) tools and word-sense (WS) disambiguation. Finally, as one more improvement the authors added a named entity recogniser to avoid correcting named entities that cause many correction errors. The authors obtained 95% correction of spelling errors.

Siklósi et al. (2016) presented a context-aware system based on statistical machine translation (SMT) to correct spelling errors in Hungarian clinical text. The authors did not have access to any word list with correctly spelled clinical terms or to parallel corpora with correct and misspelled clinical text, so they used SMT as support to select possible correctly spelled candidate terms. The SMT system

assisted in choosing possible context where the term was correctly spelled. The SMT with language model from the medical domain obtained an accuracy of 87.23%.

Grigonyte et al. (2014) developed an approach to detect and correct spelling errors in Swedish clinical text. Since many clinical misspellings are a combination of abbreviations and misspellings, the authors used abbreviation detection combined with lexical normalisation of compounds and misspellings. This spelling error correction approach reached 83.9% precision and 76.2% recall.

The Hunspell spelling system is adapted to languages with rich morphology, and was originally developed for Hungarian. In OpenOffice.org Hunspell supports over 98 languages (Pirinen and Lindén 2010). When using these spell checkers for clinical text medical, or clinical dictionaries need to be added, see Patrick and Nguyen (2011).

Patrick and Nguyen (2011) wrote a nice overview over different studies to create a clinical text spell checking and correction system. In the same article the authors describe the use of SNOMED CT for English as a dictionary for the spelling correction algorithm, matching the SNOMED CT descriptions to clinical text for assigning codes. Using the spelling correction increased the SNOMED CT coded content of 15% and the number unique codes increased by 4.7%.

An approach for spelling error correction of Swedish clinical text was performed by Dziadek et al. (2017), where the authors used the Swedish version of SNOMED CT, MeSH, ICD-10 and NSL[4] as dictionaries for checking the correctness. The study tried out several spelling error correction methods such as Levenshtein threshold, a context-sensitive method based on trigram frequencies and a corpus-based dictionary. The corpus-based dictionary contains words that occur more than twice. Of the detected misspellings, a subset of 571 were manually evaluated by a senior physician, she categorised 222 of these as misspelled. Of the 222 misspellings, 70% were correctly edited by the algorithm. (Note that clinical text contains around 10% misspellings).

A lower Levenshtein threshold gave both higher spelling error correction and mapping precision, while a higher Levenshtein threshold gave more SNOMED mappings.

For details on the approach see the master's thesis of Dziadek (2015)

Open Source Spell Checkers

There are a number of open source spell checkers to use, for example Aspell or Hunspell,[5] often with dictionaries for different languages, to use them in the medical domain specific dictionaries need to be added or created.

[4]The Swedish National Substance Register for Medicinal Products maintained by the Medical Products Agency—Läkemedelsverket.

[5]Hunspell, http://hunspell.github.io. Accessed 2018-01-11.

There is a specific Swedish spell checking system called Stava[6] which can be used for spell checking PDF, HTML, LaTeX and DOC-format documents. Stava is open source and can be downloaded freely.[7]

For a general English spell checker see an implementation in Python of Peter Norvig's spell checker[8] by Nick Sweeting in 2014.

Regarding grammar checkers for clinical text, nothing has so far been reported in academic literature.

Search Engines and Spell Checking

One interesting point is that search engines do not use general dictionaries to perform spell checking on the search questions but uses the index as a dictionary. This means all words contained in the document collection or corpora are indexed and available as a dictionary. The search engine will then propose correct words that are in the document collection to assist the user to find what he or she is looking for. There is no point for the search engine in proposing correctly spelled words that are not available in the document collection.

7.3.6 Part-of-Speech Tagging (POS Tagging)

The next step of the NLP pipeline is *part-of-speech tagging*, which is the process of automatically extracting the function of the words: *determiner, subject, predicate, adjective, adverb, preposition* etc. There are also other types of tagging, such as semantic or thematic tagging.

Each word is usually classified in a word class, such as *noun, adjective, adverb, determiner* and *preposition*, but when analysing a sentence, the words have functions depending on how they relate to each other, for example: *subject, predicate, direct object* and *indirect object*, this is what a part-of-speech tagger will determine.

There are several methods for constructing taggers, both rule- or dictionary based, but machine learning approaches are state of the art. There are several taggers available for use, one tagger for each language, or at least one tagger and one

[6]Stava, (interface in Swedish), http://www.csc.kth.se/stava. Accessed 2018-01-11.

[7]Stava download (in Swedish), http://www.nada.kth.se/~viggo/stava/manual.php. Accessed 2018-01-11.

[8]Peter Norvig's spell checker in Python, https://github.com/pirate/spellchecker. Accessed 2018-01-11.

language model for each language to tag.[9] For English we have, for example, the TnT tagger, which can be trained on manually annotated corpora. For Swedish we have the Granska tagger and Stagger—the Stockholm Tagger.

7.4 Syntactical Analysis

Syntactical analysis or *parsing* is the next step, to find the syntactical structure of a sentence. The syntax determines the order of the lexical items in a sentence. To do so, a grammar describing the language according to the order in which words occur in a sentence is needed. Usually there are grammar rules describing the language and a parser executes the grammar rules. Sentence grammar describes the sentences but also phrases constituting the whole sentence. A text grammar is used for describing a whole text, which constitutes a set of sentences.

Previously, linguists constructed rules manually, usually several thousand rules were necessary to parse natural language correctly. Today, manually annotated text is used as data to train machine learning systems on the behaviour of the language to produce the grammar rules automatically. The grammar model is then used to parse a text. Of course a parser will also use information from a tagger to carry out the parsing.

A parser can work bottom-up, top-down or incrementally with noisy text, or backtracking, left to right, dependency-based etc. The output from a parser is a syntactic tree describing the different parts of the sentence.

The textbook in natural language processing by Jurafsky and Martin (2014) gives a good description of the available parser methods.

One possible parser to use is MaltParser, which is a dependency-based parser.[10] MaltParser has pre-trained models for Swedish, English, French and Spanish (Nivre et al. 2006).

Hassel et al. (2011) used MaltParser with a pre-trained model for Swedish and applied it on Swedish clinical text, they obtained comparably good result for part-of-speech tagging with an accuracy of 92.4%. This result should be considered in the context of clinical text, which is very noisy and in a completely different domain than the pre-trained model of the MaltParser was trained on. Observe also MaltParser needs as input text that has been morphosyntactically disambiguated using a tagger. In Fig. 7.2 we can see the dependency tree from the parsing of a clinical text.

[9]Part-of-speech taggers, https://en.wikipedia.org/wiki/Part-of-speech_tagging#External_links. Accessed 2018-01-11.

[10]MaltParser, http://www.maltparser.org. Accessed 2018-01-11.

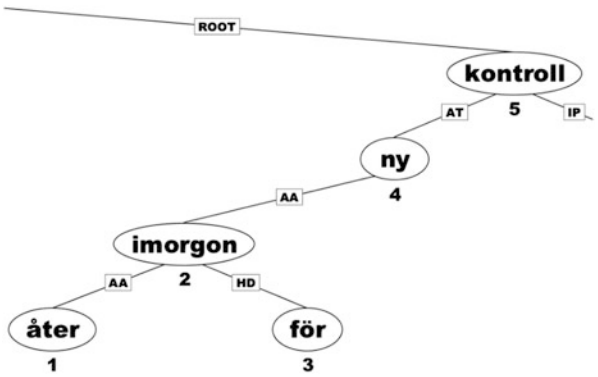

Fig. 7.2 Example dependency parse tree for the Swedish sentence *åter imorgon för ny kontroll* ("back tomorrow for new check-up") from MaltParser (© 2011 Northern European Association for Language Technology (NEALT). Reprinted with the permission of NEALT and the authors. Published in Hassel et al. 2011)

Another useful parser supporting the standard for universal dependencies[11] is the UD-parser[12] that supports over 50 languages (Nivre et al. 2016). The UD-parser is a variant of MaltParser.

7.4.1 Shallow Parsing (Chunking)

A more basic form of parsing is shallow parsing. Shallow parsing (also called chunking, or "light parsing") is something between POS tagging and parsing. The shallow parser or chunker detects constituent parts in sentences in the form of nominal phrases or verb phrases.

7.4.2 Grammar Tools

There are many tools to create grammars. Grammars are used to parse text, or in other words to syntactically analyse text. One such grammar is the definite clause grammar (DCG) form, implemented in the *Prolog* logic programming language. In DCG a grammar can be written in an abstract form and be compiled to an executable Prolog form, see Fig. 7.3. Prolog was constructed in 1972 and was originally created for natural language processing.

[11] Universal Dependencies v2, http://universaldependencies.org. Accessed 2018-01-11.
[12] UDPipe, https://ufal.mff.cuni.cz/udpipe. Accessed 2018-01-11.

```
sentence --> noun_phrase, verb_phrase.
noun_phrase --> det, noun.
verb_phrase --> verb, noun_phrase.
det --> [the].
det --> [a].
noun --> [cat].
noun --> [bat].
verb --> [eats].
```

Fig. 7.3 Example of a toy DCG in Prolog, that can parse the sentence: *The cat eats the bat*, and some variations on this, (from Wikipedia)

The DCG-grammar (and Prolog) can be considered as a set of theorems and the lexical items as facts that have to be proved by using a theorem prover. The theorem prover is the built-in Prolog interpreter. If the facts (the lexical items), in the theorems (the grammar) are proved, corresponding to that the syntax of the tested sentence is correct. One more advantage of DCG is that it can easily be extended to produce a syntax tree that can be used to perform operations on. The DCG is very similar to the *Backus-Naur Form (BNF)* for writing grammars.

Other tools originally developed for building compilers for formal languages, such as programming language is *Lex (Lexical Analysis)* and *Yacc (Yet another compiler-compiler)*,[13] which both are built-in in the Linux operating system. These can also be used to construct parsing tools for natural languages.

7.5 Semantic Analysis and Concept Extraction

Semantic analysis is the task of interpreting the meaning or semantics of entities that were identified. Semantic analysis is also called text analytics. There are several ways to do semantic analysis. One is to analyse the syntactic parse tree and to assign parts of it meaning. Traditionally predicate logic has been used as a representation, but there are many other representations. Other simpler tasks are named entity recognition and negation detection along with more complex tasks such as factuality or uncertainty detection tasks. Relations extraction and temporal processing are also complex semantic tasks as well as anaphora resolution. All these tasks are going to be discussed in the following subsections.

[13]Lex and Yacc page, http://dinosaur.compilertools.net. Accessed 2018-01-11.

7.5.1 Named Entity Recognition

Named Entity Recognition (NER) or Named Entity Tagging was first defined in the MUC 7 challenge as the identification of *personal names, locations, organisations* and *time points* or *dates*, also called TIMEX expressions in newswire text (Chinchor and Robinson 1997), but now it has broadened to include almost anything interesting in a text. In clinical text NER usually means the named entities for de-identifying text, such as *personal names, addresses* and *telephone numbers* etc but also *finding (symptom), disorder (disease), drug* and *body part* when referring to *clinical (named) entity mining*.

To perform NER, traditionally, name lists or so-called Gazetteers, have been used combined with regular expressions: today most of the approaches use machine learning techniques, except for extracting numerical expressions, such as telephone numbers or drug dosages, where regular expressions are more efficient.

Machine Learning for Named Entity Recognition

In the study by Skeppstedt et al. (2014) patient records from a Swedish internal medicine emergency unit were annotated with the named entities *disorder, finding, pharmaceutical drug* and *body structure* by two senior physicians. The inter-annotator agreement (IAA), was calculated to an F-score of 0.77 for disorder, 0.58 for finding, 0.88 for pharmaceutical drug and 0.80 for body structure. The annotated corpus is called the *Stockholm EPR Clinical Entity Corpus*.

A number of features were extracted for the training data. The features were based on the terminology in ICD-10 diagnosis codes, SNOMED CT and MeSH as well as on pharmaceutical drugs from the Swedish FASS,[14] but also on POS tagging and lemmatisation of the input words.

The features were extracted by matching each token from the training text with tokens in the ICD-10, SNOMED CT, MeSH or pharmaceutical drugs description text. These features, the manual annotated data, and all tokens were used as training data utilising the implementation CRF++ of the Conditional Random Fields algorithm. The best results for CRF++ were obtained for lemmas, POS tags and terminology matching for the current token and the previous token as well as compound splitting for the current token. This final model obtained an F-score of 0.81 for disorder, 0.69 for finding, 0.88 for pharmaceutical drug, 0.85 for body structure and 0.78 for the combined category disorder + finding. These results are compared with other researchers' results on clinical texts in Fig. 7.4. Both rule-based and machine learning-based approaches were used.

The system by Skeppstedt et al. (2014) is called *Clinical Entity Finder (CEF)* and has been used as a *pre-annotation system* in Henriksson et al. (2015). In Fig. 7.4 we

[14] Farmaceutiska Specialiteter i Sverige (FASS) is the Swedish version of the American Physician's Desk Reference.

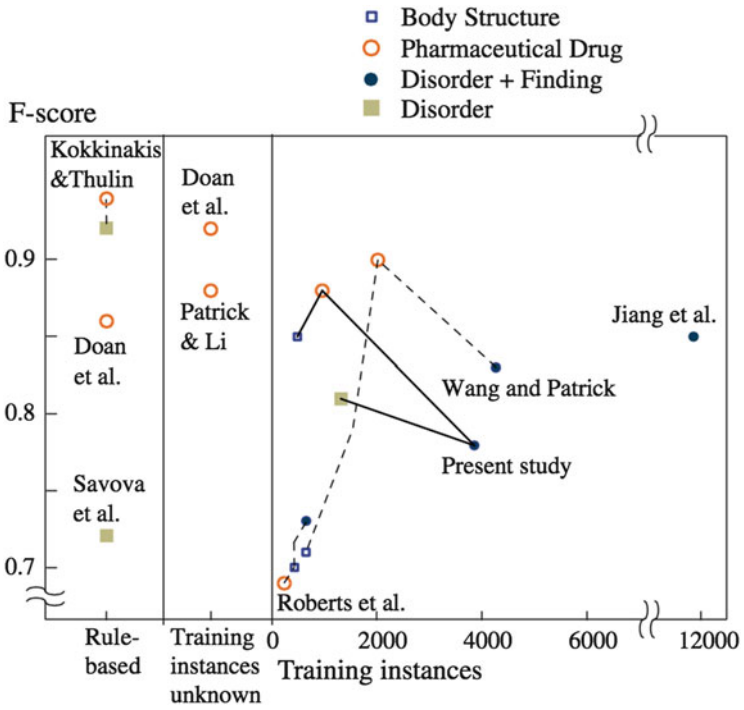

Fig. 7.4 Diagram of clinical entity recognition systems: *Comparison to a number of previous clinical NER studies. Results from the same studies are connected with a line: a solid line for the present study and dashed lines for previous studies. The first column from the left shows the result of three rule-base studies, and the second column shows the result of two machine learning studies for which the number of used training instances were not reported. The rest of the diagram shows the result of a number of machine learning studies for which the number of training instances were reported. The entity names of the present study are used for denoting comparable entity types in previous studies*. Diagram and text cited from Skeppstedt et al. (2014). Present study is the study by Skeppstedt et al. (2014) (© 2014 Elsevier Inc. All rights reserved-reprinted with permission from Elsevier Inc. Published in Skeppstedt et al. 2014)

can see the results compared to other researchers' result. Regarding pre-annotation see Sect. 8.2.2.

In the 2010 i2b2/VA challenge on concepts, assertions and relations in clinical text (Uzuner et al. 2011), there is a nice overview of both entity and relation detection of clinical concepts. The best assertion extraction or clinical named entity recognition system used Conditional Random Fields (CRF) and obtained an F-score of around 0.90.

Regarding medication identification such as *drug names, brand names, dosages, modes, frequencies* and *reason*, there was a *2010 i2b medication challenge*, (Uzuner et al. 2010). Twenty teams participated in the challenge; the ten rule-based approaches scored best; however, the best performing system was hybrid-based. The best performing hybrid system used the machine learning algorithms CRF and

SVM to detect clinical entities, and pattern matching rules to determine whether two entities were related and to detect negations.

Regarding various studies on both rule-based and machine learning-based clinical entity recognition, the PhD thesis by Skeppstedt (2015) is a good place start.

7.5.2 Negation Detection

Negation detection in clinical text mining is the task to detect negations that modify the clinical entities that are affirmed, for example findings, disorders, body parts and drugs. The negation of clinical entities, such as affirmed symptoms and diagnoses, makes them not valid, for example *no cough* or *no fever*. Negation can also be considered as a special case of factuality detection where negation is the weakest factuality consequently pure negation, or that the fact is not valid since it is negated.

Negation Detection Systems

One of the earliest approaches of detecting negations was the construction of *NegEx*. NegEx is a simple regular expression algorithm. NegEx uses three different negation trigger lists and one list containing finding and disorder. The contents of these lists are matched to the input clinical text string to decide if a concept (finding and disorder) in the string is negated or not (Chapman et al. 2001).

Negation Trigger Lists

The first trigger list in NegEx is called the pre-negation list and contains possible trigger phrases, that should be found before the negated word, for example the phrase *no signs of*. The second trigger list is called the post-negation list and contains a possible trigger phrases that should be found after the negated word, such as *unlikely*. Finally, the third trigger list contains possible pseudo-negations, they look like negation triggers but are not, for example *not certain if*.

The contents of these negation triggers are matched to the input text string and when a negation is found the distance is calculated in the form of the number of words from the negation. The distance to the negation should be in the maximum range of six words from the finding or a disorder (disease) in the input string. The list of the findings or the disorders is based on findings and disorders from UMLS.

The original English version of NegEx obtained a precision of 84.5% and a recall of 82.4% when tested on discharge summaries.

Later NegEx was extended to a version called ConText (Harkema et al. 2009) that also detected historical, hypothetical and related persons condition.

NegEx for Swedish

Skeppstedt (2011) adapted the rule-based NegEx to Swedish. The Swedish version of NegEx obtained a precision of 75.2% and a recall of 81.9%, evaluated on the assessment field from randomly chosen Swedish patient records from the Stockholm EPR Corpus. Hence the English version obtained better results than the Swedish version, but the English version was evaluated on a different corpus and language.

Negation trigger list for English[15] and for Swedish[16] can be found online.

NegEx for French, Spanish and German

NegEx was adapted to French by Grouin et al. (2011), the authors also extended NegEx to treat conjunctions and possible negations by adding two more trigger lists. The French NegEx obtained an F-score of 0.863 when evaluated. NegEx was also ported to Spanish, by Costumero et al. (2014), they obtained an accuracy of 84.8%, and also ported to German obtaining an F-score of 0.9 (Cotik et al. 2016).

In an approach by Chapman et al. (2013) the English NegEx was extended to Swedish, French and German and compared. The negation triggers *no* and *not* were similar for all languages. French had the largest diversity on triggers, while German had the least diversity. Agglutination in Swedish and German causes problems for example in Swedish *diabetesfri* means *(diabetes free)*, hence *ej diabetesfri,* means *(has diabetes)*, this can not be processed by NegEx. French negations are inflected depending on gender and number agreement, making it difficult to construct trigger lists.

Machine Learning Approaches for Negation Detection

In a machine learning approach, annotated corpora of Swedish and English negation cues were used as training and evaluation data for the Stanford NER which is based on Conditional Random Fields (CRF) algorithm. The Swedish corpus Stockholm

[15]NegEx trigger list for English, https://github.com/chapmanbe/negex/blob/master/genConText/negex_triggers.txt. Accessed 2018-01-11.

[16]NegEx trigger list for Swedish, http://people.dsv.su.se/~mariask/resources/triggers.txt. Accessed 2018-01-11.

EPR Sentence Uncertainty Corpus (Dalianis and Velupillai 2010a), and the English corpus the BioScope corpus (Vincze et al. 2008) were used. Both corpora were annotated for negations and uncertainty. For Swedish a precision of 87.9% and a recall of 91.7% was obtained for negation cues. For English a precision of 97.6% and a recall of 96.7% was obtained for negation cues. A possible explanation for the better results for English is that the English BioScope corpus is a better and more thoroughly annotated corpora.

Other approaches for detecting negated expressions in English clinical text were investigated by Huang and Lowe (2007), the authors used both parse trees and regular expressions. Their system could find negated expressions both close to and at some distance from the negation cue (signal). The system obtained a precision of 98.6% and a recall of 92.6%.

Mutalik et al. (2001) called their system NegFinder and it was developed by analysing over 40 medical documents for negations patterns and transferring that knowledge to the classical tools for compiler constructions, Lex and Yacc. By using this technique they were able match the negations by complete phrase matching for controlling the negation scope: they used prepositions and conjunctions as well as personal and relative pronouns. NegFinder achieved a precision of 91.8% and a recall of 95.7%.

Rokach et al. (2008) used clinical narratives reports manually annotated for negation a total of 1766 negated instances. Several machine learning algorithms were compared: HMM, CRF, decision trees and AdaBoost, Cascade DTs with longest common subsequence (LCS). Cascade DTs with LCS gave the best results with a precision of 94.4%, a recall of 97.4% and F-score of 95.9%.

7.5.3 Factuality Detection

As mentioned in Sect. 4.6.2, clinical text contains a lot of speculative expressions. These expressions are in a range from completely affirmed through different levels of speculation to completely non-affirmed. It is of great importance to distinguish these expressions. First to detect them and then to grade them. There have been several studies, some initial approaches were mentioned in Sect. 7.5.2, along with pure negation detection.

Szarvas (2008) describes a trial to automatically identify speculative sentences in radiology reports, he used the annotated Bioscope corpus in his approach. The machine learning package Maxent based on the maximum entropy framework gave an F-score of 0.64 and F-scores up to 0.821, when using both advanced feature selection mechanisms and external dictionaries as support.

Morante and Daelemans (2009) obtained a precision of up to 100%, and a recall of 97.5% and an F-score of 0.988 on negation signal, using Tilburg Memory Based Learner (TiMBL) together with a list of manually constructed cue words for training and detecting the negations and speculations. To detect the negation scope, they used

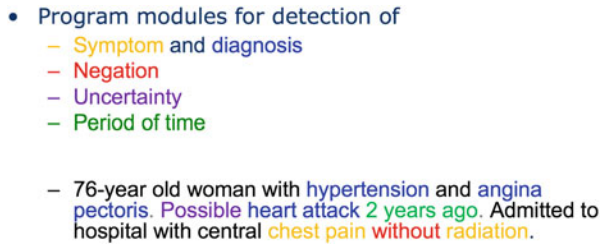

Fig. 7.5 Program modules for clinical named entity recognition (NER) applied on a clinical text. Symptom and diagnosis are sometimes interchangeable. *Possible* weakens factuality, negation remove the factuality of the symptom or diagnosis. Finally, the temporal modifier *2 years ago* takes the occurrence of the diagnosis back in time

three different machine learning systems: SVM, CRF++ and TiMBL, which gave F-scores of 0.893, 0.863 and 0.804 respectively.[17]

Velupillai (2011) defined six different levels, namely *Certainly Positive, Probably Positive, Possibly Positive, Possibly Negative, Probably Negative* and *Certainly Negative*, see Sect. 4.6.2 for details. Velupillai obtained an F-score of 0.699 using all classes, and an F-score of 0.762 using merged classes, with the CRF++ classifier.

Velupillai et al. (2014) carried out and described the porting of the English pyConTextNLP to Swedish pyConTextSwe. pyConTextSwe is dictionary based and can distinguish between four assertion classes (*definite existence, probable existence, probable negated existence* and *definite negated existence*) and two binary classes (*existence yes/no* and *uncertainty yes/no*). pyConTextSwe obtained the following evaluation: F-score 0.81 (overall) and for each of the assertion classes F-scores of 0.88 (definite existence), 0.81 (probable existence), 0.55 (probable negated existence) and 0.63 (definite negated existence). For the two binary classes (existence yes/no) and (uncertainty yes/no), the F-scores were 0.97/0.87 and 0.78/0.86 respectively.

The French version of NegEx (Grouin et al. 2011) mentioned in Sect. 7.5.2 can also process possible negations.

Figure 7.5 demonstrates how these modules will operate on a clinical text. Other clinical entities that can be recognised are drug names, drug doses and body parts.

7.5.4 Relative Processing (Family History)

Findings and disorders mentioned in the patient record may concern relatives of the patient for heredity reasons. Physicians diagnosing a patient search for different

[17]The measurement has changed to a dimensionless quantity from the original percentage in Morante and Daelemans (2009), since F-score is measured as a dimensionless quantity.

His father had high hypertension, and now present with a fever of 39° C lasting two days.

or another possible option is:
Hypertension in family (when it is related to the family of the patient).

or finally:
Had hypertension two years ago (which is a temporal relation).

Fig. 7.6 Examples of excerpts from a clinical texts containing relatives or temporal entities

symptoms, when reading the patient record if one symptom was connected to a family member, or family history, or happened long time ago it may not be valid for the current case, see Fig. 7.6 for different examples of these phenomena.

Clinical text mining to identify relatives and distinguish their symptoms from the patient being treated is sometimes called *relative processing* or *experiencer*.[18] Here a finer grained processing can also be carried out and distinguishing the closeness of the relation to the patient: *1st degree relative* or *2nd degree relative* etc. (South et al. 2009). Relative named entity recognition can also be found in the de-identification of patient records where the category *relative* is used. Entities of the relative class are for example *father, mother, son, daughter, brother, grandfather* and *grandmother*.

7.5.5 Temporal Processing

One other important task in clinical text mining is to distinguish where in a timeline a symptom, disorder or event is placed or occurred. Is the symptom a current one or a symptom that occurred one week, one month or some years ago?

The reason for having temporal treatment of symptoms is related to the way physicians reason about diseases, which symptoms occurred just before the disorder? But also to understand care progression and remove non-relevant symptoms that should not be included in the diagnosis. This is similar to the case with negated symptoms in the text. The negated symptoms are not relevant for the diagnosis.

In Fig. 7.7 we can see some of the aspects of temporality in a clinical text. For example, it is important to know that *the chest pain with radiation* preceded the *angina pectoris*, so the chest pain with radiation is a symptom of angina pectoris.

In the early 1980s within Artificial Intelligence Allen (1984) carried out research in natural language processing and problem solving that included understanding time aspects. Allen defined seven time relations: *before, meets, overlaps, is-finished-by, contains, starts* and *equals*.

[18]Experiencer trigger lists, https://github.com/chapmanbe/negex/blob/master/genConText/experiencer_triggers.txt. Accessed 2018-01-11.

76-year old woman with hypertension and angina pectoris. Possible heart attack 2 years ago. Admitted to hospital with central chest pain with radiation. Oct 23, underwent PCI,[a] and now present with a fever of 39°C lasting two days. (Note that admission date is October 18, 2012 obtained from an administrative data source).

Fig. 7.7 Fictive clinical text showing the different temporal aspects that have to be treated by a computer program. [a]PCI stands for Percutaneous Coronary Intervention usually meaning inserting a stent into one of the coronary arteries (in the heart) that is too narrow, to open it to prevent angina pectoris or save the patient for myocardial infarction (AMI), commonly known as a heart attack

Several approaches from the 1990s and onward have been used to process temporal relations in clinical text. For a nice overview see Meystre et al. (2008) but also Velupillai et al. (2015), and for state of the art specifically in temporal applications see (Sun et al. 2013b).

Zhou and Hripcsak (2007) made an early overview of different approaches to temporal reasoning in what they call medical natural language processing. The authors distinguish three main approaches:

1. Temporal reasoning based on theories and models from Artificial Intelligence.
2. Frameworks based on needs from clinical applications.
3. Resolving issues around temporal granularity and uncertainty.

Zhou and Hripcsak (2007) also discuss how to process absolute and relative time and how to combine structured time points with time points mentioned in unstructured free text.

Jung et al. (2011) have a nice example on building timelines from clinical narrative using a deep semantic natural language understanding, and building and visualising the result.

In Sun et al. (2013b) the authors give examples of typical questions that can be raised in a clinical context, such as:

- What medication was the patient on before the surgery?
- How often did the patient experience headache before the treatment?
- What symptoms did the patient experience after taking Aspirin for 3 days?

In Zhou et al. (2005) there is an approach using MedLEE adapted for temporal tagging of clinical narratives. The authors did not use any standard but they invented the whole temporal processing framework.

Identifying temporal relations needs two basic cornerstones to stand on, first the identification of clinical entities such as findings, disorders and the temporal expressions date, time and duration and secondly determining the order these occurred.

A recent book analysing temporal ordering for events and time points is the book by Derczynski (2017).

TimeML and TIMEX3

One method to annotate temporal expressions is to use the *TIMEX* format from the named entity research area (Chinchor and Robinson 1997), which basically is date and time expressions. TIMEX was developed in the *TIMEX3* version and used in the *markup language for temporal and event expressions (TimeML)*, the TimeML language is described in Pustejovsky et al. (2003).

TimeML is a specification language for temporal expressions in natural language and follows the ISO 8601 standard. TimeML treats four problems in event and temporal expression markup according to Pustejovsky et al. (2003):

- Time stamping of events (identifying an event and anchoring it in time).
- Ordering events with respect to one another (lexical versus discourse properties of ordering).
- Reasoning with contextually underspecified temporal expressions (temporal functions such as *last week* and *two weeks before*).
- Reasoning about the persistence of events (how long does an event or the outcome of an event last).

HeidelTime

Strötgen and Gertz (2010) developed the rule-based system temporal tagger *HeidelTime* (from Heidelberg University, Germany) that extracts temporal expressions from text into the TIMEX3 format, which is part of the TimeML standard. The HeidelTime system is modular and therefore easy to adapt to other languages. HeidelTime is currently available for 13 languages: English, German, Dutch, Vietnamese, Arabic, Spanish, Italian, French, Chinese, Russian, Croatian, Estonian and Portuguese.

i2b2 Temporal Relations Challenge

The Sixth Informatics for the Informatics for Integrating Biology and the Bedside (i2b2), Natural Language Processing Challenge for Clinical Records focused on the temporal relations in clinical narratives: 310 discharge summaries were annotated for temporal information. 18 teams participated in the challenge.

HeidelTime was adapted to English clinical text tagging in the i2b2 challenge and obtained the best results of all systems (Sun et al. 2013a).

In these challenges, annotated data is usually released for the different teams to train and develop on, then *held out test data* (data not used for training or developing) is given to the participating team to evaluate their results.

In the i2b2 temporal relations challenge there were three concepts to identify: the TIMEX3 temporal expressions such as TIMES, DATES and DURATION. FREQUENCY (also called SET), and EVENTs denoting an event or action and TLINKs corresponding to a temporal relation denoting the order (e.g. before, after, simultaneous, overlapping, etc.) of an EVENT and a TIMEX3, or two EVENTs or two TIMEX3s.

This follows exactly Styler et al. (2014) who claim that there are three types of temporal relations:

- Relations between two events.
- Relations between two times.
- Relations between a time and an event.

Some TIMEX3s are the frequency of an event, as for example *twice daily* as in *Claritin* 30 mg *twice daily* or *once a week at bedtime*.

Styler et al. (2014) present their approach where they used the Thyme corpus and extracted 1254 de-identified notes written in English from a large healthcare practice (the Mayo Clinic).

Two independent annotators annotated the clinical notes for the entities, EVENT, TIMEX3 and LINK. In total 15,769 EVENTS, 1429 TIMEX3s and 7927 LINKs were found. Inter-annotator agreement as resolved by F-score was 0.80 for EVENT and TIMEX but 0.50 for LINK.

The authors applied the ClearTK-TimeML system which is based on the SVM machine learning algorithm and trained for the Clinical TempEval 2013 competition, on their annotated Thyme corpus and obtained F-scores of 0.496 for TIMEX3, 0.366 for EVENT and 0.204 for LINK. The low performance can be explained by the difference in domains.

Styler et al. (2014) also discuss a new type of TIMEX3 expressions, called PREPOSTEXP, covering a span of text before or after an EVENT (operation). These temporal expressions are *preoperative, postoperative* and *intraoperative*, meaning before, after an operation and between operations. In Styler et al. (2014) the Thyme corpus is also described. Thyme is an acronym for *Temporal Histories of Your Medical Events* and the corpus contains 1254 de-identified notes, within the brain cancer and colon cancer domains. The colon cancer notes contain both clinical notes and pathology reports. The corpus and the guidelines for the temporal annotations can be found online.[19]

[19]Thyme corpus, http://thyme.healthnlp.org. Accessed 2018-01-11.

Temporal Processing for Swedish Clinical Text

For Swedish, Velupillai (2014) carried out a rule-based approach where she adapted the rule-based system temporal tagger HeidelTime to Swedish and applied it to a small subset of Swedish intensive care unit patient records. The adapted version of HeidelTime obtained a precision of 92% and recall of 66%.

Temporal Processing for French Clinical Text

Hamon and Grabar (2014) adapted the rule-based system temporal tagger HeidelTime to French as well as to English. The adaptation consisted of adding more rules to HeidelTime to deal with complicated clinical text (164 rules in English and 47 rules in French).

Both the French and English adapted version, of HeidelTime performed better in processing clinical text than the general purpose version processing general text. For French the performance increased from an F-score of 0.918 to an F-score of 0.942 for general text and medical text respectively. For English the performance increased more, from an F-score of 0.655 to an F-score of 0.843 for general text and medical text respectively.

Temporal Processing for Portuguese Clinical Text

One piece of research work on Brazilian Portuguese clinical text was carried out by Tissot (2016) in his PhD-thesis. Tissot specifically studied how to detect imprecise temporal expression, for example:

The patient says that he had pneumonia more than 2 years ago. More than 3 months ago. A few days ago. A long time ago. The coming months. When exactly is that?

A very imprecise temporal expression is: *The patient says he had weight loss*. When did the weight loss happen?

The Portuguese clinical corpus used for this study is called the InfoSaude corpus and contains 3360 patient records from general medicine, gynecology, nutrition and psychiatry. It was extracted from the InfoSaude system of the Public Health Department in Florianopolis, Brazil. Tissot also used the English TempEval corpus.

Two systems were developed to identify time expressions, one rule-based called HINX, which was developed using GATE by adapting the standard NLP processing modules to the clinical domain, and one machine learning-based using the SVM algorithm as implemented on LibSVM. Three steps were carried out:

1. Text pre-processing;
2. TIMEX identification; and
3. TIMEX normalisation.

where step 1, is standard in NLP, and step 3, was to determine on the scope of specific imprecise temporal expressions.

Normalisation of imprecise temporal expressions was carried out by interviewing people about how they understand and interpret imprecise temporal expressions and how to normalise them. One approach used was to utilise a fuzzy membership function. The membership function would place an imprecise TIMEX in the timeline to perform the normalisation, for example *3* in the expression *about 3 months* is converted to *90 days*. Part of the normalisation process was to correct spelling errors. The process of building the spelling error detection and correction system was based on both string similarity and a phonetic similarity functionality for Portuguese clinical text.

The HINX system found 503,005 TIMEXs and 52,830 imprecise TIMEXs in the Portuguese InfoSaude corpus among the 3360 patient records; however, the quality of these results was never evaluated since the InfoSaude corpus was not manually annotated.

One finding was that clinical text contained more imprecise temporal expressions, up to 35%, than standard news text or historical text, comprising up to 13%, of the total temporal expressions. The high numbers were found in a British clinical corpus called the SLaM (BRC Case Register) corpus.

7.5.6 Relation Extraction

Relation extraction is a classification problem, is this relation holding between these two entities? First the two entities that will hold a relation need to be identified using named (clinical) entity recognition, then the relation has to be established. NER is a well-established research area with F-scores of 0.90 and above, while relation extraction is more difficult and reaches only an F-score around 0.70 (Uzuner et al. 2011). See Sect. 7.5.1 for more on clinical named entity recognition.

Relation extraction is a common task in biomedical text mining when a relation must be established between two molecules, or two chemical substances. Are these two components reacting with each other? A nice overview of different approaches of relation extractions in the biomedical domain can be found in Luo et al. (2016).

In clinical text mining one task is to detect an adverse drug event, by detecting the relation between a drug and a finding or symptom. Is there an indication for this drug and this finding or symptom?

2010 i2b2/VA Challenge Relation Classification Task

Uzuner et al. (2011) have a nice review article on various approaches for detection of clinical relations; however, in their article they have not really defined what is an assertion or a concept or the difference between them. A relation holds between

two assertions or two concepts or between a concept and an assertion. A concept or assertion seems to be a medical problem, a test, or a treatment.

The best performing system of ten different systems was by de Bruijn et al. (2011), which obtained both best assertion detection with an F-score of 0.936 and best concept extraction with an F-score of 0.852. These are pre-requisites for relation assertion, which they also obtained the second best results with an F-score of 0.731. The assertion classifier and the relation classifier of de Bruijn et al. used an approach called semi-supervised learning through clustering. The Brown clustering algorithm was used on the unlabelled clinical corpora for the unsupervised approach, and applied the features produced for their supervised approach using a semi-Markov model (de Bruijn et al. 2011).

SVM gave the best results for the relation extraction task for clinical text in the 2010 i2b2/VA challenge (Uzuner et al. 2011).

Other Approaches for Relation Extraction

In a study by Henriksson et al. (2015) a Swedish clinical text containing adverse drug events (ADEs) was annotated for named clinical entities as well as ADE relations. For the named entity annotations *finding, disorder, drug, body part* and *ADE cue* the IAA F-scores were around 0.84. For the semantic relation annotations *indication, adverse drug event, ADE outcome* and *ADE cause* the IAA were lower at around 0.65. For the training and detection of the named entities CRF++ was used together with features extracted from an unsupervised distributional semantics approach, specifically Word2Vec (skip-gram model); for detecting named clinical entities the F-score results were around 0.80, while for relation extraction were much lower around 0.45.

In the study by Bejan and Denny (2014) the authors used 6864 discharge summaries that were annotated by two annotators, finding 958 treatment relations and 9628 non-treatment relations. Two different pre-annotation tools were used, first a tool called SemRep and second their own tool known as the MEDI algorithm, which both extract relations but in different ways. MEDI is a UMLS based pre-annotation tool. 25% of the data was double annotated obtaining an F-score for IAA of 0.979 with a Cohen's kappa of 0.86. For the training the authors used LIBSVM and with 5-fold cross validation obtained an F-score of 0.85.

Bejan and Denny (2014) refer to Roberts et al. (2008) as one of the first attempts to extract relations from clinical text. Roberts et al. (2008) annotated 77 oncology narratives with seven categories of relations, obtaining low IAA with an F-score of 0.47. SVM was used for the training and an F-score of 0.72 was obtained over a class of seven relation types.

7.5.7 Anaphora Resolution

Anaphora resolution sometimes also called *co-reference resolution* is the task of resolving which entity in a sentence, a pronoun or a noun phrase refers to.

The *anaphor* is the pointing back reference (the pronoun or the noun phrase). The *antecedent* is the entity that is referred to by the anaphor. These are also called *markables* or *markable*, which are terms used in the anaphora research area.

The task of determining the antecedent of an anaphor is called anaphora resolution or co-reference resolution. For an example of anaphora see Fig. 7.8.

Some anaphora occur within a sentence, and are called intrasentential anaphors and some anaphora are referring over sentence borders and are called intersentential anaphors.

The anaphora resolution usually takes place by gender and number agreement, but there are also many other ways to resolve the anaphora. Sometimes it is not resolved. The research area of anaphora resolution is well studied in computational linguistics though not completely solved, for a nice overview of the state of the art see (Mitkov 2014).

He (2007) wrote a master's thesis in the area of coreference resolution in discharge summaries using 47 hospital discharge summaries written in English, containing in total 4978 lines of text.

The author let two computer science students annotate the corpus with coreference chains and time stamps. The coreference chains contained five different annotation classes depending whether the coreference was about a person, symptom, disease, medication or test. A *coreference chain* is a line between the antecedent and the following possible anaphors referring to the antecedent in an anaphora resolution system.

The inter-annotator agreement measured by Kappa-value averaged 0.836 over all annotation classes, which is considerably high and corresponds to an easy annotation task.

The training set created was comparably small, in total 649 coreferent pairs, and therefore a lot of features were needed to compensate for the scarce training data.

The time stamp was annotated to distinguish whether a symptom occurred a long time ago or not and was used as a feature. In addition to the time stamps the author also used a set of carefully selected features.

> *76-year old woman with hypertension and angina pectoris. Possible heart attack 2 years ago. Admitted to hospital with central chest pain without radiation. She mentioned her daughter had similar symptoms. She was treated and is fine now.*

Fig. 7.8 Fictive clinical text with a pronominal anaphora, *76-year old woman* is the antecedent and *She* is the anaphor. The pronominal anaphora is not resolved, who is the last *she* in the text referring to? The daughter or the patient? Probably the daughter and not the patient

The machine learning algorithm used for training was the supervised C4.5 Decision Tree algorithm and the author obtained a lowest overall B-CUBED precision of 0.983 and a lowest overall B-CUBED recall of 0.947. B-CUBED precision and recall are specific evaluation metrics constructed for evaluating coreference chains as explained in Bagga and Baldwin (1998).

i2b2 Challenge in Coreference Resolution for Electronic Medical Records

A review on different approaches to resolve the anaphoras in electronic medical records is described in Uzuner et al. (2012). This was part of the Fifth i2b2/VA Workshop on Natural Language Processing Challenges for Clinical Records. In total 20 teams from 29 organisations and from nine different countries participated.

The data used in the challenge comprised progress notes, discharge records, radiology reports and surgical pathology reports from Beth Israel Deaconess Medical Center, Partners Healthcare and University of Pittsburgh Medical Center (UPMC), as well as clinical reports and pathology reports from the Mayo Clinic. All written in English and de-identified, in total there were 978 files.

The data was annotated by two independent annotators for coreference pairs, almost 5646 coreference chains were annotated.

The results from the challenge showed that both the machine learning and the rule-based approaches worked best when augmented with external knowledge. 77.75% of the ground truth chains were correctly predicted by all systems and 95.07% of by at least one system.

7.6 Summary of Basic Building Blocks for Clinical Text Processing

This chapter presented all the steps necessary for performing natural language processing on clinical text, starting with word segmentation and continuing with tokenisation and morphological processing, which encompasses, compound splitting, stemming, abbreviation detection, spell checking and correction, part-of-speech tagging followed by a discussion of syntactical parsing and semantic analysis, such as named entity recognition, different negation detection systems, relation extraction, temporal processing and anaphora resolution.

Usually one or more of these steps need to be carried out but not all of them, since usually only some information needs to be extracted from the text for a specific task. For continued reading see the following books on NLP Mitkov (2005), Jurafsky and Martin (2014) and Clark et al. (2013).