Inf Retrieval (2008) 11:499–538
DOI 10.1007/s10791-008-9061-0

Negation recognition in medical narrative reports

Lior Rokach · Roni Romano · Oded Maimon

Received: 18 September 2007 / Accepted: 20 May 2008 / Published online: 7 June 2008

Abstract Substantial medical data, such as discharge summaries and operative reports are stored in electronic textual form. Databases containing free-text clinical narratives reports often need to be retrieved to find relevant information for clinical and research purposes. The context of negation, a negative finding, is of special importance, since many of the most frequently described findings are such. When searching free-text narratives for patients with a certain medical condition, if negation is not taken into account, many of the documents retrieved will be irrelevant. Hence, negation is a major source of poor precision in medical information retrieval systems. Previous research has shown that negated findings may be difficult to identify if the words implying negations (negation signals) are more than a few words away from them. We present a new pattern learning method for automatic identification of negative context in clinical narratives reports. We compare the new algorithm to previous methods proposed for the same task, and show its advantages: accuracy improvement compared to other machine learning methods, and much faster than manual knowledge engineering techniques with matching accuracy. The new algorithm can be applied also to further context identification and information extraction tasks.

Keywords Text classification · Part-of-speech tagging · Negation · Narrative medical reports · Artificial intelligence

1 Introduction and motivation

Information retrieval of free text is now an established and well-known application with vast popularity. The limitations of naïve keyword based information retrieval are also well understood and many research works are focused around this issue.

L. Rokach (✉)
Department of Information Systems Engineering, Ben Gurion University, P.O. Box 653, Beer Sheva 84105, Israel
e-mail: liorrk@bgu.ac.il

R. Romano · O. Maimon
Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv, Israel

The primary application of this work is in improving the information retrieval from medical narratives. Medical narratives present some unique problems that are not normally encountered in other kinds of texts. When a physician writes an encounter note, a highly telegraphic form of language may be used. There are often very few (if any) grammatically proper sentences and acronyms and abbreviations are frequently used. Many of these abbreviations and acronyms are highly idiosyncratic and may not be found in a general dictionary.

Information retrieval from medical narratives has many applications: enrollment of patients into clinical trials; detecting adverse events; modern evidence-based practice; and medical research in general. A typical application scenario may involve a hospital-based medical investigator receiving from a pharmaceutical company a patient profile for a planned clinical trial. The profile includes attributes that cannot be used as is in a structured query of the hospital information systems. Example of such a patient profile is: "*Male and female, 18 years and older; Female must not be pregnant; Location of pain must be low back area; Pain must be present for three months or greater; No surgical intervention in the past 12 months nor plans for surgical intervention for the low back pain during the duration of the study*." Most of the data needed for locating patients meeting the above profile is stored as electronic medical narratives in the hospital information systems.

The medical investigator retrieves such records by a keyword-based search. The keywords primarily include: diagnostic names, symptoms, procedures, medicine, etc. A useful knowledge source designed for resolving medical terms is the Unified Medical Language System (UMLS) (Lindbergh and Humphreys 1993).

The common use-case when searching in discharge summaries is looking for patients with specific symptom, for example, *nausea*. The issue of context is very important. Consider the sentence: "*He complained at admission of headache, nausea, vomiting, and neck soreness*" versus "*The patient denies any headache, nausea, vomiting, blurring vision and fever.*" Both sentences will match a naïve keyword based query containing the term *nausea*. We assume that the person initiating the query is looking for patients with a specific symptom (e.g. *nausea*). For example, the sentence "*The patient states she had fever and chills two nights prior to admission with a nonproductive cough*" taken from a discharge summary report is a positive example for *fever* and *chills* diagnoses, while another sentence from a discharge report: "*The patient denied any cough, chest pain, urinary symptoms or bowel symptoms*" is a negative example for cough, chest pain, urinary symptoms and bowel symptoms diagnoses.

A search for patients with a specific symptom or set of findings might result in numerous records retrieved. The mere presence of a search term in the text, however, does not imply that retrieved records are indeed relevant to the query. Depending upon the various contexts that a term might have, only a portion of the retrieved records may actually be relevant. Therefore, in addition to excluding negated concepts, there are additional contexts we opt to exclude. For example: "*The patient as well as her daughter were given very clear instructions to call or return for any nausea, vomiting, bleeding, or any unusual symptoms.*"; and the sentence: "*The patient could not tolerate the nausea and vomiting associated with Carboplatin*"; "*She is married, lives with her husband and admits to drinking alcohol excessively in the remote past.*"; and more.

This work introduces a new supervised method for inducing a sequence-aware (or sequence sensitive) classifier. First, we automatically discover a set of sequence patterns that are described as regular expressions. Then a classifier is induced to classify instances based on their matching to the discovered set of sequence patterns. We show the

advantages of the new method by applying it to a well-known and well-defined problem in the medical domain. It is estimated that ignoring negations in medical narrative reports may reduce the retrieval performance in about 40% (Averbuch et al. 2004). Thus, the goal is to improve the information retrieval accuracy by correctly identifying the context of the keyword being searched.

The issues encountered in dealing with this problem constitute a special case of context identification in free text, one of the key research problems in the field of text mining. We compare the results obtained using our proposed method to previous works which implement two primary methodologies: knowledge engineering; and machine learning. The knowledge engineering approach is based on handcrafted patterns for identifying the negated context. Such methods yield high accuracy but are labor-intensive, domain specific, and tedious to maintain. The more modern methods are based on machine learning techniques. The bag-of-words is considered as one of the prominent machine learning techniques for classification problems. The negative context detection can be formulated as a text classification problem and solved using bag-of-words. The negation problem is closely related to the part-of-speech tagging (POS) problem, which is properly solved by frameworks for labeling sequential data, such as hidden Markov model (HMM) and conditional random fields (CRF). In this work, we compare our new method to the abovementioned techniques. Our new method is much faster than manual knowledge engineering techniques with matching accuracy. We show that our new method achieves higher accuracy compared to existing methods.

2 Related works

The negation problem in medical reports can be solved in various ways. First, in addition to existing general purpose text classification methods that can be used, there are several information extraction (IE) methods that can also be implemented. After discussing these methods, we survey specific works regarding the negation problem in general and in the medical domain in particular. Finally, we discuss evaluation measures that can be used for estimating the quality of the solutions.

2.1 Text classification

A comprehensive survey of the methods used for text categorization (Sebastiani 2002) describes the recent research trends. The machine-learning paradigm of automatic classifier construction has emerged and definitely superseded the knowledge-engineering approach. Within the machine-learning paradigm, a classifier is built by learning from a set of previously classified documents. The advantages of the machine learning approach as per Sebastiani are a very good effectiveness, a considerable savings in terms of expert manpower, and domain independence.

2.1.1 Bag-of-words

Since texts cannot be directly interpreted by a classifier or by a classifier-building algorithm, it is necessary to uniformly apply a transformation procedure to the text corpora in order to map a text d_j into a compact representation of its content (Sebastiani 2002). In text

categorization (TC) a text d_j is usually represented as a vector of term weights $d_j = (w_{1j}, \ldots, w_{|V|j})$ where V is the set of terms (sometimes called features) that occur at least once in at least one document of T_r, and where $0 \leq w_{kj} \leq 1$ represents, loosely speaking, how many term t_k contributes to the semantics of document d_j. Differences among approaches are accounted for by (1) different ways to understand what a term is; (2) different ways to compute term weights. A typical choice for the first alternative is to identify terms with words. Depending on whether weights are binary or not, this approach is often called either the "*set of words*" or the "*bag-of- words*" approach to document representation.

The following example demonstrates the bag-of-words representation applied to our domain. Consider the two sentences: (1) *The patient was therefore admitted to the hospital and started on <MEDICINE> as treatments for <DIAGNOSIS>*; and (2) *The patient was ruled in for <DIAGNOSIS> and started <MEDICINE> for <DIAGNOSIS>*. Taking case insensitive single words as features and binary weights yields the representation shown in Fig. 1.

One of the main drawbacks of the bag-of-words representation is in its destruction of semantic relations between words; the meaning of word combinations is lost (Bekkerman and Allen 2003). As presented in Fig. 1, this representation loses the meaning of important terms such as "*ruled in*". This bag-of-words limitation is especially important for the negation detection, Averbuch et al. (2004) show that a negation profile contains only ten words and/or phrases, half of these are bi-grams.

2.1.2 n-gram

Another popular choice for text representation is to identify terms with word sequences of length *n*. This n-gram vector text representation method is used to classify text documents (Damashek 1995). Damashek selected the normalized frequency with which the n-gram occurs in the document as the choice of (2) term weight. Each vector identifies a point in a multidimensional space, and similar documents are expected to have points close to each other. Damashek (1995) used the dot product between two histogram vectors as a measure of their similarity, but he pointed out that other measures are possible.

Caropreso et al. (2001) experimented with n-grams for text categorization on the Reuters dataset. They define an n-gram as an alphabetically ordered sequence of n stems of consecutive words in a sentence (after stop words were removed). The authors use both unigrams (bag-of-words) and bigrams as document features. They extract the top-scored features using various feature selection methods including mutual information (see, e.g., Dumais et al. 1998). Their results indicate that in general bigrams can better predict categories than unigrams.

	the	patient	was	therefore	admitted	to	hospital	and	started	on	<medicine>	as	treatment	for	<diagnosis>	ruled	in
First sentence	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	0
Second sentence	1	1	1	0	0	0	0	1	1	0	1	0	0	1	1	1	1

Fig. 1 Bag-of-words representation

2.1.3 Regular expressions

A regular expression is defined as any string that describes or matches a set of strings according to certain syntax rules. Regular expressions are usually used to give a concise description of a set without having to list all elements. The regular expression consists of a letter of the alphabet and special characters. For example, the set containing the four strings: hat, hit, hot and hut can be described by the pattern "h.t" (or alternatively, it is said that the pattern matches each of the four strings). The wild-card (".") denotes a single position that can be occupied by any letter of the alphabet. The curly brackets are used to indicate a match between min and max of the preceding characters. For instance the pattern "without .{0,10} <diagnosis>" can be matched against the following strings "without <diagnosis>", "without any <diagnosis>", "without serious <diagnosis>", etc.

Regular expressions are used by many text editors and utilities to search and manipulate bodies of text based on certain patterns. Many programming languages support regular expressions for string manipulation. For example, Perl and Tcl have a powerful regular expression engine built directly into their syntax. In our work we use the Java regular expression implementation (package *java.util.regex*).

The bag-of-words and n-gram representations are actually a special case of the regular expression representation proposed in this work. A regular expression feature such as "*.{0,500} started .{0,500}*" is actually equivalent to the word feature *started* in the bag-of-words representation. The regular expression feature "*.{0,500} ruled in .{0,500}*" matches the bigram representation of the two words phrase *ruled in.* An additional benefit of our proposed regular expressions compared to bag-of-words is in handling compound sentences that include both positive and negative findings, a limitation noted by Averbuch et al. (2004). For example, the sentence: "*upon admission showed no <diagnosis_1> but did show extensive <diagnosis_2> and <diagnosis_3> but there were no masses noted*". The bag-of-words representation of such sentences is problematic since the same features apply to both negative and positive contexts and the algorithm cannot learn to distinguish between them. The regular expressions representation can represent such structural features using the distance and presence of additional diagnosis.

2.1.4 Part-of-speech tagging

Part-of-speech tagging refers to labeling words in a text as corresponding to a particular part of speech based on both its definition, as well as its context—i.e., relationship with adjacent and related words in the text. POS tagging is hard mainly because some words may have multiple part of speech tags and the correct tag depends on the context. POS tags indicate the basic syntactic function of that token, such as noun or verb, as well as other grammatical information, such as number and tense. POS tagging is a fundamental preprocessing step for many other Natural Language Processing (NLP) applications (e.g., syntactic parsing). Typically, POS tags provide general shallow syntactic information to these downstream applications (Cohn 2007).

Machine learning methods have been shown to be more effective in solving POS tagging than classic NLP methods (Halteren et al. 2001). POS tagging is closely related to our problem. In fact, the negation detection problem can be regarded as a special case of POS tagging—we define a polarity tag (possible values are negative and positive) that is applicable to the <diagnosis> terms only. The following sections present sequences labeling frameworks that have been successfully used for POS tagging.

2.2 Frameworks for information extraction

The research field of IE is related to this work since some of the algorithms developed for IE can also be applied to detection of negative context. A common IE task is to automatically extract entities and relationships from semi-structured or free text. For example, in the medical domain, an IE task is to automatically populate a structured database from a discharge summary report.

Many works in IE propose learning approaches that automatically process free text and overcome the knowledge engineering bottleneck. Califf and Mooney (1999) proposed the RAPIER system that induces pattern-match rules from rigidly structured text. Freitag (1998) described the SRV framework that exploits linguistic syntax and lexical information for corpus based learning while Soderland (1999) proposed the WHISK system for learning text extraction rules automatically. The $(LP)^2$ algorithm described in Ciravegna (2001) learns tagging rules from an annotated corpus. Kushmerick et al. (1997) and Muslea et al. (2001) proposed wrapper induction procedure for extracting structured information from database-like Web pages. These works have shown that wrappers can be automatically learned for many kinds of highly regular documents, such as Web pages. Another algorithm that uses wrapper induction for IE is the boosted wrapper induction (BWI) proposed by Freitag and Kushmerick (2000).

All the above works focus on extracting entities and relationships. There is no emphasis on special contexts, such as negation that might totally divert the meaning of the text. Apparently such cases are rare in the corpora used for evaluating the above works (which is not true when dealing with discharge reports where more than 50% of the findings might actually be negated). More recent IE works are focused on HMM based techniques.

2.3 Frameworks for labeling sequential data

The HMM is a common machine learning technique with published applications in sequential pattern recognition tasks. The HMMs were successfully applied to related problems such as: IE (Seymore et al. 1999); POS tagging (Kupiec 1992; Smith et al. 2004) and many more. Specifically, HMM was successfully applied to POS tagging of bio-medical texts, For example, Smith et al. (2004) trained the MedPost HMM POS tagger and achieved a 97.43% accuracy on the MEDLINE corpus. In their HMM implementation, each POS tag corresponds to a state in the model, and transition probabilities are estimated from tag frequencies in the training set. Another HMM implementation of POS tagging applied to the MedPost data repository is the LingPipe implementation (LingPipe 2007).

We have not found any work that implements HMM to detect negation in free text. Nevertheless, we can utilize HMM POS taggers for solving the negation problem. Applying a HMM POS tagger to the negation detection problem is not a trivial task since there are many possible approaches for structuring the HMM. Figure 2 below illustrates a simple three-state HMM-based POS tagger. The hidden states are the POS tags (e.g. noun, verb, adjective, etc.) and the arrows represent the possible transitions between states. Freitag and Kushmerick (2000) demonstrate that when applying HMM for IE, extraction accuracy strongly depends on the selection of the HMM structure.

Conditional random fields are a newer framework for labeling sequential data (Lafferty et al. 2001). The CRFs define a conditional probability over label sequences given a certain observation sequence. This relaxes the unwarranted independence assumptions about the sequences which HMMs make. Like HMMs, CRFs have been successfully used for POS.

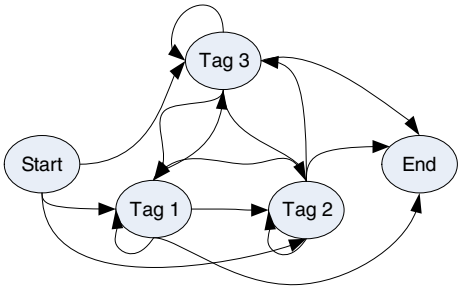

Fig. 2 Sample three-states HMM based POS tagger

A comparative study showed that CRFs outperform HMMs in this application (Lafferty et al. 2001).

2.4 Sentiment analysis

The problem of sentiment analysis is similar to negation recognition. Most work in sentiment analysis classifies documents by their overall sentiment, e.g. determining whether a review is positive or negative (Turney 2002) while some perform phrase-level sentiment analysis, e.g. Wilson et al. (2005). The works on phrase-level sentiment analysis are closer to our work than the works that classify whole documents.

Many of these recent works apply NLP techniques to sentiment analysis. Most are based on machine learning approaches similar to ours. Kim and Hovey (2004) achieved 81% percent accuracy in identifying positive, negative, or neutral sentiments given free-texts. Their approach is based on "sentiment-bearing words" such as: agree, disagree, etc. The lists of seed words are prepared by hand and expanded with words obtained from WordNet. Esuli and Sebastiani (2005) achieved up to 85.4% accuracy when suggesting a novel method that exploits online glossaries in addition to WordNet.

These approaches are different from our work since they rely on seeded words and glossaries for expanding these words. Our approach does not depend of such supervised guidance, the negation terms our automatically detected.

2.5 Identifying negative context in non-domain specific text (general NLP)

Negation is an active linguistic research topic dating 2500 years back with the legacy of Aristotle with ongoing publications, conferences, and workshops (Horn 2001). Negation is considered difficult in NLP due to the overwhelming complexity of the form and the function of sentences with negation. Negation is one of the constants of classical logic and has complex and systematic interaction with the other logical operators, especially quantifiers and modals.

In English grammar, negation is the process that turns a positive statement ("*the patient has <diagnosis>*") into its opposite denial ("*the patient does not have <diagnosis>*"). Nouns as well as verbs can be negated with a negative adjective ("*There is no <diagnosis>*"); a negative pronoun (*no one, nobody, neither, none, nothing*); or a negative adverb ("*he never was <diagnosis>*"). It is easy to identify specific negation words such as: *not*, *neither*, and *never*, as well as for *Not*-negation, e.g., *not, n't*, and *No*-negation. However, in many cases, these specific words are not presented, e.g., *deny*, *fail*, and *lack*. Words in this second category are called inherent negatives (Tottie 1991), i.e., they have a negative meaning but a positive form. An additional morphological form of negation is the

affixal negation. Prefix negations *un-* and *in-*, may create negation words *unhappy, unwise*, and *unfit*. Negations can also be created with suffixes such as *-less*, e.g., *lifeless*. Another complexity arises from double negation, e.g. the sentence "*it is not unlikely*". The neg-raising phenomenon adds additional complexity, e.g. sentences such as: "*I don't believe he is ill*" or "*I don't think he is ill*".

We could not locate any NLP research on identifying negated concepts in specific non-domain areas. However, some NLP techniques such as syntactic and semantic processing can be applied to a negation identification framework, especially part of speech tagging and shallow parsing. These features can be combined into a machine learning classification scheme for negation identification. The effectiveness of such NLP techniques very much depends on the quality of the text, particularly its compliance with grammatical rules. The language used in medical narratives, however, is often grammatically ill-formed. For example, the positive finding cough in the sentence "*the patient reported she was not feeling well due to mild cough*". Thus NLP techniques that rely on grammatical sentences may not be sufficient for identification of negation in medical narratives. Similar observation was noted by other researchers, for example, Java (2007) writes that the complex linguistic structures found in Web blogs require to rely on semantics rather than shallow NLP.

2.6 Identifying negative context in medical narratives

Researchers in medical informatics have suggested methods for automatically extracting information contained in narrative reports for decision support (Fiszman et al. 2000), guideline implementation (Fiszman and Haug 2000), and detection and management of epidemics (Hripcsak et al. 1999). Some of the researches concentrate on methods for improving information retrieval from narrative reports (see, for instance, Hersh and Hickam 1995; Nadkarni 2000; Rokach et al. 2004). A number of investigators have tried to cope with the problem of a negative context, see definition in Sect. 1. These works can be classified into two research domain categories, which are presented in the following two sections.

2.6.1 Works based on Knowledge engineering

The knowledge engineering approach is based on human expert writing rules or patterns. These rules and patterns are designed to capture syntactic and semantic features of the free text. The methods used are mostly from the NLP research field utilizing also deep parsing technologies and sometimes rule engines. These methods are complex and very expensive to develop and maintain, useful mostly when the target text is written according to proper language rules. Examples of knowledge engineering based works are: Friedman et al. (1994), Aronow et al. (1999), Leroy et al. (2003), Mutalik et al. (2001), and Chapman et al. (2001). These works are described in the following paragraphs.

Friedman et al. (1994) developed MedLEE that performs sophisticated concept extraction in the radiology domain. The MedLEE system combines a syntactic parser with a semantic model of the domain. It recognizes negatives which are followed by words or phrases representing specific semantic classes such as degree of certainty, temporal change or a clinical finding. It also identifies patterns where only the following verb is negated and not a semantic class (i.e., "X is not increased"). This method yields is highly accurate. The shortcomings are that it is rigid, not easily adaptable to additional domains and expensive to develop and maintain.

Aronow et al. (1999) developed the NegExpander which uses syntactic methods to identify negation in order to classify radiology (mammography) reports. While NegExpander is simple in that it recognizes a limited set of negating phrases, it does carry out expansion of concept-lists negated by a single negating phrase.

Leroy et al. (2003) developed and evaluated a shallow parser that captures the relations between noun phrases automatically from free text. It uses heuristics and a noun phraser to capture entities of interest in the text. Cascaded finite state automata are capable of structuring the relations between individual entities. The automata are based on closed-class English words and model generic relations not limited to specific words. The parser also recognizes coordinating conjunctions and captures negation in text, a feature usually ignored by others.

Mutalik et al. (2001) used a lexical scanner with regular expressions and a parser that uses a restricted context-free grammar to identify pertinent negatives in discharge summaries and surgical notes. Their Negfinder system first identifies propositions or concepts and then determines whether the concepts are negated. The set of regular expressions is predefined by IT professionals based on input obtained from medically trained observers. Their strategy yields over 95% accuracy in classifying negative medical concepts. Mutalik's algorithm is quite complex and requires other utilities such as *Lex* and *Yacc*. The tools required by Mutalik's algorithm are easily attainable. However, implementing their system in a preexisting indexing tool would be less straightforward.

One of the most extensive studies on negation to date is in the work of Chapman et al. (2001). They developed a simple regular expression algorithm called NegEx that implements several phrases indicating negation, filters out sentences containing phrases that falsely appear to be negation phrases, and limits the scope of the negation phrases. Their algorithm uses a predefined set of pseudo negation phrases, a set of negation phrases, and two simple regular expressions. They use a lexical scanner with regular expressions and a parser that uses a restricted context-free grammar to identify pertinent negatives in discharge summaries and surgical notes. Their system first identifies propositions or concepts and then determines whether the concepts are negated. Their system performed with a sensitivity of 95.7% and a Recall of 91.8% and is fine tuned with rules that apply to particular negation phrases and syntactic structures.

2.6.2 Works based on machine learning

Many of the recent works in the field of text classification are based on the machine learning approach. The work of Averbuch et al. (2003) is an example of detecting negated concepts in medical narratives using machine-learning techniques. In the methodology that Averbuch developed for automated learning of a negative context profile in medical narratives, the profile contains only ten words and/or phrases such as: "*negative for*"; "*had no*"; "*was no*"; etc. It has been shown that this algorithm is superior to traditional classification algorithms that are based on "bag-of-words" representation. Machine learning has proven effective for text classification. The advantages are that such methods are much faster to develop than Knowledge engineering. In addition they are more effective when the text is not written according to proper grammar rules.

Goldin and Chapman (2003) describe an extension of NegEx using machine learning algorithms and demonstrate that machine learning techniques (including decision trees) enhances the performance of their NegEx classifier. They summarize the findings of their study into a simple rule: "*When negation of a UMLS term is triggered with the*

negation phrase 'not,' if the term is preceded by 'the', then do not negate". However, this conclusion is based on manual interpretation of the experimental results. There is no research that tries to learn syntactically rich negation patterns automatically and then uses the discovered patterns to classify medical concepts that appears in unseen text.

Based on the above assumptions, the purpose of this work is to develop a methodology for learning negative context patterns in medical narratives and measure the effect of context identification on the performance of medical information retrieval. While the knowledge engineering approach showed that regular expressions are very effective in identifying negative context, there is no research that tries to automatically learn these expressions from medical narrative reports. Thus the aim of this work is to examine the ability to automatically learn regular expressions from medical narrative report. Moreover, usually all automatic discovery methods used in fields other than medical informatics provides an ordered list of regular expressions. Thus when a new sentence is needed to be examined, this sentence is matched against the list of regular expressions. The outcome of the first positively matched regular expression is then used. In this research we suggest to learn a hierarchical structure. As we will see later this hierarchical structure has important features.

3 The proposed methodology

The methodology we develop enables an effective process of learning useful regular expressions from the training corpus. Section 3.1 below explains the complete process of training a regular expression based classifier from an input of training and test corpora. Sections 3.2–3.5 specify in detail each of the steps. Finally, Sect. 3.6 suggests the concept of cascading several classifiers for improving the performance.

3.1 The process overview

We suggest the following process of training a classifier to predict negation concept using regular expressions patterns. The process includes four steps (see Fig. 3).

1. **Corpus preparation**: A domain-specific task designed to normalize, generalize and tag the free text so that it can be further processed.
2. **Regular expression patterns learning**: The automatic creation of a regular expression patterns from the training set.
3. **Patterns selection**: Applying heuristics and features selection techniques to select the best patterns for correct classification of the concept.
4. **Classifier training**: Training a decision tree classifier.

Fig. 3 The process overview

The following sections describe each of the above steps.

3.2 Step 1: corpus preparation

The objective of the corpus preparation phase is to transform the input discharge summaries data into a usable corpus for the training and test phases. Figure 4 presents the corpus preparation sub-steps. The following sections describe each sub-step.

3.2.1 Step 1.1: tagging

In the first step, we parse all the discharge summaries. All known medical terms are tagged using a tagging procedure presented in our previous work Rokach et al. (2004). Consider for example the following text:

> *This is a 66 year old woman status coronary artery bypass graft in 1989-06-23 with coronary artery disease, hypertension, diabetes mellitus, kidney stones.*

We use the UMLS meta-thesaurus, for tagging the sentence, i.e. replacing medical terms with their concept type. For example, when the parser reaches the term *coronary* it queries the UMLS for terms starting with "*coronary**". The result set includes several terms starting with *coronary*. The parser then uses a sliding window in order to match the longest possible UMLS term to the given sentence. The UMLS terms relevant for the above sentence are listed in Table 1.

Since we are only interested in the generalized form of the sentence (the specific diagnosis or procedure does not matter), the output text following the tagging process takes the following form:

> *This is a 66 year old woman status <Procedure_1> in 1989-06-23 with <Diagnosis_1>, <Diagnosis_2>, <Diagnosis_3>, <Diagnosis_4>, <Diagnosis_5>.*

We are using a simple algorithm for tagging medical phrases in the text based on the UMLS meta-thesaurus. The main idea of this tagger is to efficiently find the longest string

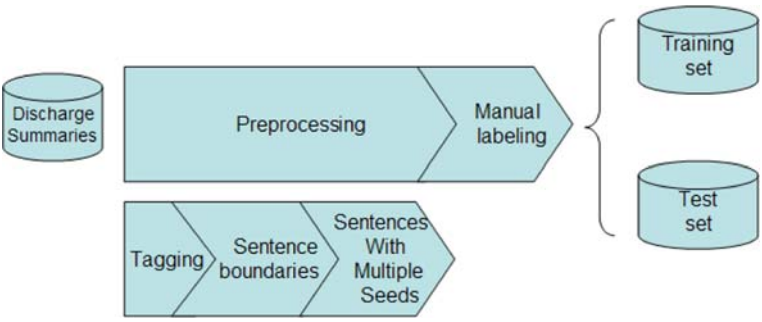

Fig. 4 Corpus preparation

Table 1 Tagging using the UMLS

ID	Term	Type	CUI (concept unique identifier)
1	Coronary artery bypass graft	Procedure	10010055
2	Coronary artery disease	Diagnosis	10010054
3	Hypertension	Diagnosis	10020538
4	Diabetes mellitus	Diagnosis	10011849
5	Kidney stones	Diagnosis	10022650

in the UMLS that a match the given text. This tagger has no capability to resolve ambiguities. Thus if the same string represents two different tags then one of them is arbitrarily selected. These potential errors will be manually corrected during the labeling step.

Figure 5 describes the pseudo-code of the tagger. As the meta-thesaurus is stored in a database table, and this table contains many entries, it will be desirable that the number of queries will be minimized as much as possible. In order to achieve this goal the following measures have been added:

- Stopword—Any word which appears in the Stopwords table (such as "the", "in") will not be searched for. The Stopwords is based on a fixed size hash table. This table is updatable (see below) based on LRU strategy. Nevertheless several basic words can be permanent members of this table and cannot be removed.
- In order to improve database querying performance an attribute that contains the first word and an attribute the represents the phrase length (in characters) have been added to the concept table.
- A caching mechanism is used in order to avoid querying the database with the same query in adjacent times. Each entry in this cache represents a list of phrases beginning with the same token sorted by phrase length. This cache is managed as LRU hash table, where the first token is used as the key and the list of phrases is used as the stored item.

```
INPUT: O - Original Text
OUTPUT: T  - Tagged text
Do
       T ← O /* Create a copy of the original text to be manipulated */
       t <- Next Token in T

       If t ∉STOPWORD then

                  List<- ∅
                     If t ∈ CACHE
                           List<-CACHE(t)
                     Else
                           List <- Select all phrases in UMLS
                                   with first token=t order by length (descending)

                     If List= ∅ then
                            Add t to STOPWORD using LRU strategy.
                      Else
                         Add List to CACHE using LRU strategy.
                      End
                  End
                  For each phrase p_i in List
                         If current position in T contains p_i then
                                  Replace the p_i in T with a tag.
                                  Promote the position in T accordanly.
                                  Exit For
                          End If
                  End For
       End If
Until no more tokens
Retrun RESULT
```

Fig. 5 Pseudo-code for simple tagger

Table 2 Regular expressions to exclude sentence end

(b|t|q)\.i\.d\.?	p\.o\.?	\.([0–9]+)	cc\.
p\.r\.n	q\.d\.?	\. of	\., and
q\.h\.s	mg\.	(Dr\.)(\s?)(\w+)	\sq\.

3.2.2 Step 1.2: sentence boundaries

Physicians are trained to convey the salient features of a case concisely and unambiguously as the cost of miscommunication can be very high. Thus it is assumed that negations in dictated medical narrative are unlikely to cross sentence boundaries, and are also likely to be simple in structure (Mutalik et al. 2001).

An additional processing step includes breaking discharge summaries documents into sentences using a sentence boundary identifier as suggested by Averbuch et al. (2003). The sentence boundary is identified by searching for terminating signs such as {".", "?", "!"}. This approach is not sufficient since periods and other signs are frequently appear inside sentences (for instance: "*Patient was discharged on Lopressor 25 milligrams p.o. b.i.d*".[1] We detect such exceptions using regular expressions (an expression that describes a set of strings) to exclude expressions that might mistakenly be considered end of sentence (Table 2).

3.2.3 Step 1.3: manual labeling

This step refers to the creation of the training corpus. Two physicians reviewed each document and labeled each medical term indicating whether it appears in positive or negative context. If the physicians have noticed that the medical term has been wrongly tagged during Step 1.1, they have first selected the correct tag and then indicated if it appears in positive or negative context.

Since most sentences include more than one diagnosis, it is necessary to label each of them during the manual labeling process. Consider for instance the compound sentence: "*She denied shortness of breath, but did have fever*". In this case "*shortness of breath*" is negative while "*fever*" is positive. Thus, this sentence will be represented in the dataset as two different instances—one for each diagnosis term. Each instance has one label (positive or negative). Since each instance has one label (positive or negative), each has exactly one anchor diagnosis term to which the label refers. This anchor term is tagged as "<DIAGNOSIS>" while any other diagnosis terms in the sentence will be denoted as "<DIAG>". The above procedure is demonstrated in Fig. 6. By doing so, we will be able to obtain different patterns from the same sentence. For instance in this example, the pattern ".* *denied <DIAGNOSIS>*.*" can be learned for identifying negative context, and the pattern ".* *denied <DIAG> but .* <DIAGNOSIS>*" can be learned for identifying positive context.

3.3 Step 2: patterns creation

Instead of using a single regular expression representation for the entire sentence, we use two regular expressions: one for the string that precedes the targeted medical term (the

[1] From Latin: oral administration two times daily.

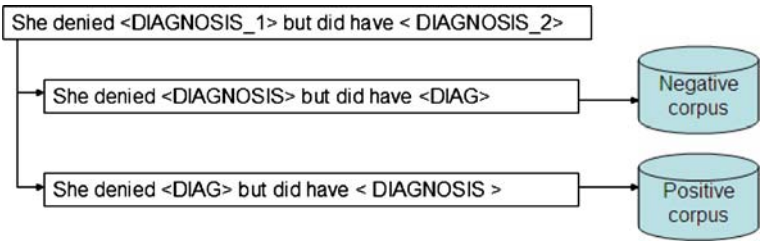

Fig. 6 Handling sentences with multiple seeds

seed) and one for the string that follows it. This split may help to resolve some of the problems that arise in compound sentences that include both positive and negative contexts in the same sentence. Recall the example "*The patient states she had fever, but denies any chest pain or shortness of breath*". In this case the appearance of the verb "*denies*" after the term "*fever*" indicates that the term "*fever*" is left in positive context. The appropriate regular expression will be in this case as follows: "*.{0,200}denies any.{0,200} <DIAGNOSIS>*", where the distance 200 is arbitrary determined per the expected sentence length in the domain.

To learn regular expressions, we have adapted two different algorithms to our task and compared them against each other. The first algorithm, LCS, is commonly used to compare characters in two text files. The second algorithm, Teiresias, was designed for discovering motifs in biological sequences. We describe how we adapted these algorithms to the task of learning regular expressions for negation patterns below.

3.3.1 Learning regular expression patterns using longest common subsequence algorithm

The basis for discovering a regular expression is a method that compares two texts with the same context and incorporates the same concept types (i.e., diagnosis, medication, procedure, etc.). By employing the LCS algorithm (Myers and An 1986) on each part of the sentence (before the targeted term and after the targeted term) a regular expression that fits these two sentences is created. The LCS employs a brute force policy: given a sequence X, determine all possible subsequences of X, and check to see if each subsequence was a subsequence of Y, keeping track of the longest subsequence found. For instance assume we are given the following two sentences:

```
The patient was therefore admitted to the hospital and started on
<MEDICINE> as treatments for <DIAGNOSIS>.

The patient was ruled in for <DIAG> and started <MEDICINE> for
<DIAGNOSIS>.
```

We execute LCS algorithm on the two normalized sentences as presented in Table 3.

Table 3 Longest common subsequence generation

Sentence 1	Sentence 2	Pattern
The patient was	The patient was	The patient was
therefore admitted to the hospital		.{24,35}
	ruled in for <DIAG>	
and started	and started	and started
	on	.{0,4}
<MEDICINE>	<MEDICINE>	<MEDICINE>
as treatments		.{0,15}
for <DIAGNOSIS>	for <DIAGNOSIS>	for <DIAGNOSIS>

Note that the LCS algorithm was revised to compare tokens as opposed to comparing characters in its classical implementation. It should also be noted that whenever there was only insertion (or only deletion) we added a wild card string with a minimum length of 0 and a maximum length of the inserted string (including the leading and trailing spaces). On the other hand, whenever there was simultaneously insertion and deletion, we added a wild card string with the minimum length of the shortest string and maximum length of the largest string (without leading and trailing spaces because they are part of the common substring).

As a result of running the LCS algorithm we obtain the following pattern. This pattern can now be used to classify concept of type medication appearing in positive contexts.

```
The patient was .{24,35} and started .{0,4}<MEDICINE>.{0,15} for
<DIAGNOSIS>
```

3.3.2 Learning regular expression patterns using Teiresias algorithm

The Teiresias algorithm was designed to discover motifs in biological sequences, an important research problem (Rigoutsos and Floratos 1998). The method is combinatorial in nature and able to produce all patterns that appear in at least a (user-defined) minimum number of sequences, yet it manages to be very efficient by avoiding the enumeration of the entire pattern space. Furthermore, the reported patterns are maximal: any reported pattern cannot be made more specific and still keep on appearing at the exact same positions within the input sequences.

Teiresias searches for patterns which satisfy certain density constraints, limiting the number of wild-cards occurring in any stretch of pattern. More specifically, Teiresias looks for maximal <L, W> patterns with the support of at least K (i.e., in the corpus there are at least K distinct sequences that match this pattern). A pattern P is called <L, W> pattern if every sub pattern of P with length of at least W words (combination of specific words and "." wild-cards) contains at least L specific words.

For example, given the following corpus of six negative sentences:

```
no further <diagnosis> was noted
no history of <diagnosis>
no intraoperative or immediate <diagnosis> were noted
no other <diagnosis>
past medical history no other <diagnosis>
patient had no further episodes of <diagnosis>
```

The Teiresias program (L = K = 2, W = 5) discovers six recurring patterns shown in the following file:

```
22     no other <diagnosis>
22     no further
22     of <diagnosis>
33     no . <diagnosis>
22     <diagnosis> . noted
22     no . . . <diagnosis>
```

The first two columns represent the support of the pattern. The dot represents a missing word. Note that the program yields also patterns that do not include the <diagnosis> seed. These patterns are not useful for our purpose and are filtered out. Next we transform the Teiresias patterns to regular expression patterns by replacing each dot (missing word) with a regular expression such as *.{0,L}*, where L is calculated by counting the number of dots and multiplying by the average word length (8 characters as per our corpus).

The resulting regular expression patterns are presented in the following example:

```
no other <diagnosis>
of <diagnosis>
no .{0,8} <diagnosis>
<diagnosis> .{0,8} noted
no . {0,24} <diagnosis>
```

For discovering the regular expressions, we compared the LCS algorithm to the Teiresias algorithm. The usage of Teiresias for creating regular expressions patterns requires the setting of the following parameters: (1) The minimum number of words in a pattern (set to 2); (2) the maximum extent of a pattern (set to 100 words); and (3) The minimum allowed support for a pattern (set to 10).

3.4 Step 3: patterns selection

Obviously there are many patterns that can be created via the LCS (each pair of sentences with the same concept type and context). In fact, initially too many patterns are created and it is essential to keep a manageable number of patterns. For example, a training set of 140 negative sentences and 140 positive sentences yielded 2*(140*139/2) = 19,460 patterns.

3.4.1 Heuristics for pattern selection

Many of the generated patterns differ only in the distance of important keywords from the seed concept. We start by rounding the distances in the regular expression patterns. For example, the pattern "*had no.{12,27} <diagnosis>*" is replaced with the pattern patterns "*had no.{10,40} <diagnosis>*". As a result, patterns such as "*had no.{12,27} <diagnosis>*" and "*had no.{17,32} <diagnosis>*" are replaced with the pattern "*had no.{10,40} <diagnosis>*". Trivial patterns such as "*a.{70,100} <diagnosis>*" are omitted. For example from the original 19,460 patterns, 17,235 were identified as redundant and trivial patterns. After eliminating these patterns, only 2,225 patterns are remained.

3.4.2 Correlation-based feature selection

Feature selection is the process of identifying relevant features in the dataset and discarding everything else as irrelevant and redundant. For this purpose each "regular expression" pattern represents a different feature.

In this work we use a non-ranker filter feature selection algorithm. Filtering means that the selection is performed independently of any learning algorithm. Non-ranker means that the algorithm does not score each pattern but only indicates which pattern is relevant and which is not. Figure 7 below describes the training set matrix before features selection. The rows are training sentences (negative and positive), the first K columns are the regular expression patterns; and the last column is the target class (negative/positive). The cell value is 1 if the regular expression matches the sentence, otherwise it is 0. The matrix described above is the input to the features selection algorithm.

In this work we use the Correlation-based Feature Subset Selection (CFS) as a subset evaluator (Hall 1999). Rather than ranking individual regular expression, CFS ranks the worth of subsets of regular expressions by considering the individual predictive ability of each expression along with the degree of redundancy among them. Subsets of expressions that are highly correlated with the context while having low inter-correlation are preferred.

The CFS first calculates a matrix of expression–context and expression–expression correlations from the training data. Expression–context correlation indicates how much an expression is correlated to a specific context while expression–expression correlation is the correlation between two expressions. CFS then calculates the merit of an expression subset consisting of K expressions:

$$\text{Merit}_s = \frac{K\overline{r}_{cf}}{\sqrt{K + K(K-1)\overline{r}_{ff}}}$$

where Merit_s is the merits of the expression subset, $\overline{r}_{cf}$ is the average of the correlations between the expression and the context and $\overline{r}_{ff}$ is the average expression–expression correlation.

	Pattern 1	Pattern 2	Pattern 3	…	Pattern K	Context
Sentence 1	1	1	0	…	1	Positive
Sentence 2	1	0	1	…	0	Positive
Sentence 3	1	1	0	…	1	Positive
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Sentence M	1	0	1	0	1	Positive
Sentence M+1	0	1	0	…	1	Negative
Sentence M+2	1	1	1	…	0	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Sentence N	0	0	1	…	0	…

Fig. 7 Training set matrix before features selection

As the search space is huge (2^K), CFS starts from the empty set of regular expressions and uses the best-first-search heuristic with a stopping criterion of five consecutive fully expanded non-improving subsets.

The CFS algorithm is suitable to this case, because there are many correlated patterns (for instance, when one pattern generalizes another pattern). For example the 2,225 remaining patterns create a dataset of 280 instances with 2,225 input binary attributes (0 if the pattern does not match the sentence; 1 if pattern matches sentence) and target attribute that represent the concept classification ("Positive" or "Negative). By applying the CFS algorithm on the training set presented in Fig. 7 the number of patterns is reduced from 2,225 to 35.

3.5 Step 4: classifier training

The filtered matrix, together with the manual classification of each concept, is fed into a decision tree induction algorithm which creates a classification decision tree. The J48 algorithm is used as the base induction algorithm for growing a decision tree. J48 is a java version of the well-known C4.5 algorithm (Quinlan 1993).

An illustrative example of decision tree generated is presented in Fig. 8. It describes a classification decision path where pattern "*.{0,200}have.{0,50} <diagnosis>*", learned from positive examples, indicates a positive context with probability P5 in case the sentence does not match the three (negative) patterns: "*.{0,200}without.{0,10} <diagnosis>*"; "*.{0,200}rule out.{0,10} <diagnosis>*"; ".{0,200}had no.{0,10} <diagnosis>" (with probabilities P1, P2, and P3 for negative) but matches the negative pattern "*.{0,200}no.{0,50} <diagnosis>*". Here we denote "negative pattern" as a pattern learned from negative context examples. This demonstrates the power of decision based on matching a sentence with multiple regular expressions.

3.6 Cascade of three classifiers

3.6.1 Overview

It is well known that the classification accuracy of a single decision tree can be significantly improved by growing an ensemble of trees and letting them vote for the most popular class. Analyzing the problem domain, we brought up the hypothesis that it is possible to create a more powerful ensemble structure than the structure obtained from such general purpose ensembles method as Adaboost (Freund and Schapire 1996). Specifically, we noticed that: (1) training set size is a limiting issue due to the computational complexity of the machine learning algorithms used; (2) in the corpus, there are simple

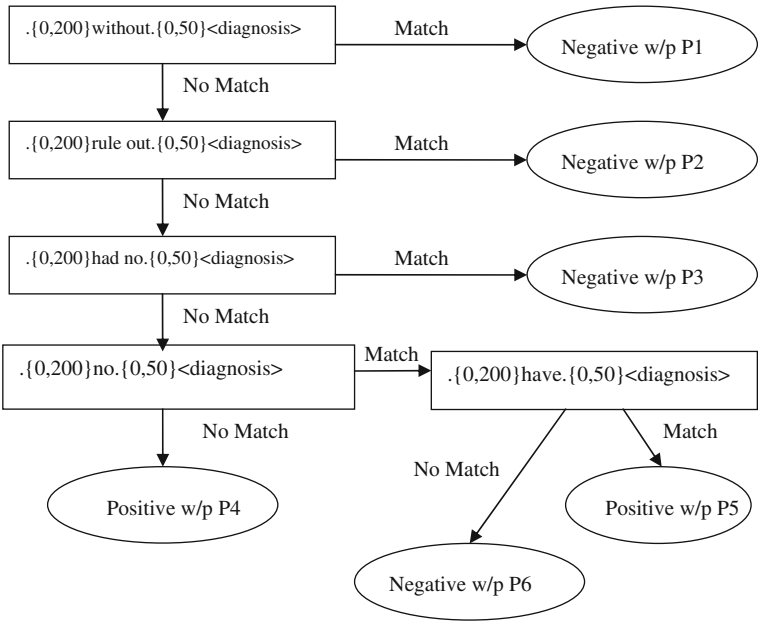

Fig. 8 Example decision tree

sentences versus compound sentences or instructions; (3) Some of the patterns yield very high Precision. This is obvious since for some of the negation terms attached (anchored) to the seed, mean that the seed is negated. For example, in a sentence such as, "... *denied* ***nausea*** ..." the nausea is negated with near 100% probability. Thus, it makes sense to train a simple classifier using only such (anchored) patterns, using it to identify the simple instances with very high Precision. Then, only instances not classified as negative by the first cascade are used to train a second classifier.

These observations triggered the idea of constructing a cascade of classifiers. The idea is to build a cascade of classifiers as shown in Fig. 9. The selection of tree cascades is due to the problem characteristics: the first cascade consists of anchored patterns; the second cascade consists of negative patterns (learned from negative sentences) and the third cascade classifier also includes positive patterns.

Figure 9 demonstrates the cascaded classifier training strategy. The first cascade includes only anchored patterns, ensuring high Precision (very few positive sentences will be classified as negative). Anchored patterns are patterns where the word is anchored (no separating words) to the seed. For example, the following anchored patterns form the first cascade classifier:

- no <diagnosis>
- denied <diagnosis>
- denies <diagnosis>
- not <diagnosis>
- negative for <diagnosis>
- without <diagnosis>
- ruled out <diagnosis>

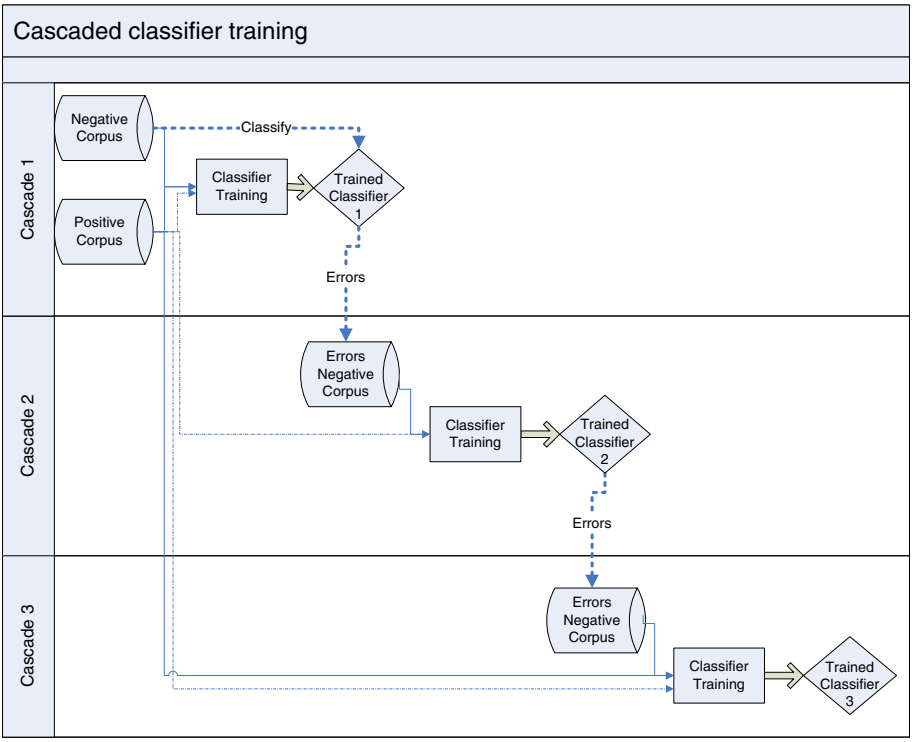

Fig. 9 Cascaded classifier training

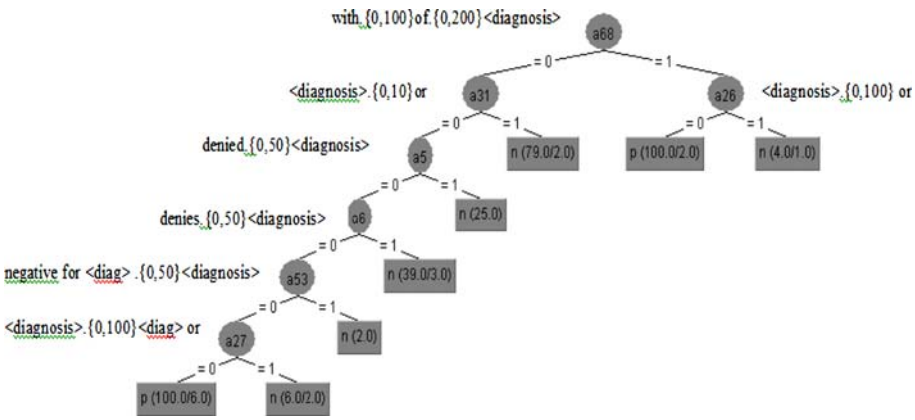

Fig. 10 Second cascade

The training set of negated instances for the second cascade comprises negation patterns that failed to classify as negative by the first cascade "*Trained classifier 1*". The training set of positive instances for the first cascade is used as is in the second cascade. Figure 10 illustrates a schematic description of the second cascade classifier.

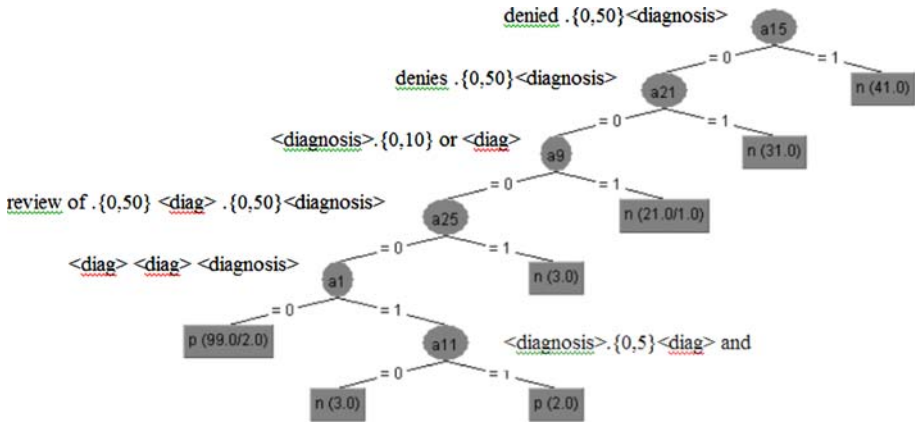

Fig. 11 Third cascade

In the third cascade, we learn patterns from the negative and positive corpora, taking only negative instances which failed to classify as negative by the first and second cascades. As can be seen in Fig. 11, the third cascade classifier includes also positive patterns (patterns learned from the positive corpus). In that sense, these patterns are different from the previous works that rely only on negation patterns.

Figure 12 demonstrates how the cascaded classifiers perform the classification of three unseen sentences. The first sentence "*the patient denied <diagnosis>*" is matched by an anchored pattern "*denied <diagnosis>*" and is classified negative by "Trained classifier 1". The second sentence "*the patient did not experience recent <diagnosis>*" does not match with any of the "Trained classifier 1" anchored patterns, therefore it is fed into "Trained classifier 2" for further classification as negative due to the patterns comprising "Trained classifier 2". The third sentence is classified as negative by the "Trained classifier 3". The last sentence is not classified as negative by all three cascades and is therefore classified as positive.

4 Experimental study

4.1 Experimental setup

We study the potential of the proposed methods in real word datasets. The main training corpus is a set of 1,766 instances parsed from de-identified discharge summaries that were obtained from Mount Sinai Hospital in New York. A shorter version of this corpus was used in our previous work (Rokach et al. 2004). Recall from Fig. 6 that the same sentence can be the source for several instances (if there are several diagnoses in the same sentence).

Experimental evaluation is performed using tenfold cross-validation, the dataset is repeatedly split into training and testing sets so that the statistical significance of the difference between classifiers can be analyzed. The cross validation split was performed on the report level and not on the instance level because different physicians might use different language; If instead the cross validation split was performed on the instance

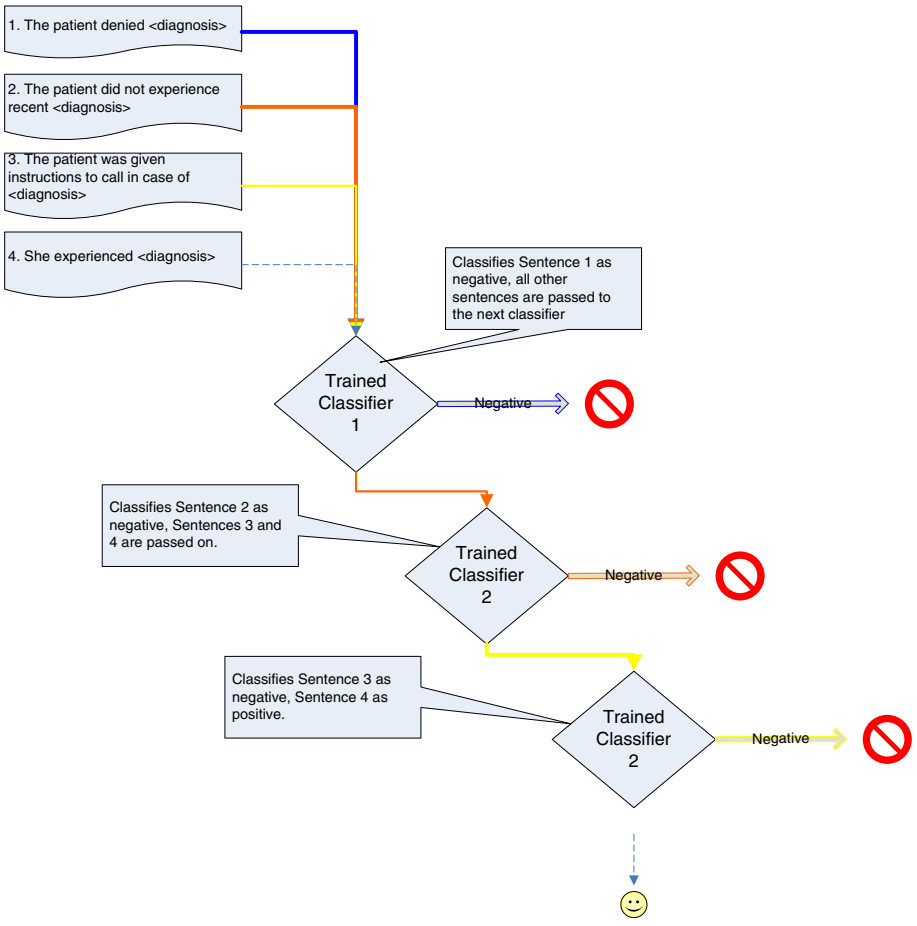

Fig. 12 Cascade classifier classification examples

level the predictive power could be higher than should be expected. The last argument is true, because instance level splitting can generate validation sets which contain instances obtained from the same report that have been used to create the corresponded training set.

Two physicians were asked to label each diagnostic term to either "positive" or "negative". Using Kappa statistics we measured the labeling agreement of the two physicians and concluded that there is almost perfect agreement (Kappa coefficient = 0.968). The few disagreements have been resolved by presenting the cases to a third physician who served as an arbitrator.

The variety of document sources enables us to measure and compare the sensitivity of the classification methods to the text origin. We performed several experiments in order to determine the classifier sensitivity to the following parameters: (a) Different training set sizes; (b) the effect of using feature selection; and (c) the effect of using ensemble of decision trees. We examined here in what extent it can improve the results and whether the new cascading ensemble is more powerful than existing general purpose ensemble methods by investigating (d) the difference between using LCS and Teiresias; (e) the sensitivity to

diagnosis type; (e) the sensitivity to different data sources and documents types (such as operation reports, outpatient, etc.).

In order to examine the predictive power of the examined methods, we will use the following four measures: Negative Predictive Value, Recall, F-Measure and Accuracy. Note that all other measures presented in Sect. 2.5 can be restored from these four measures.[2]

4.1.1 Compared algorithms

Our algorithm has been implemented using the WEKA framework (Witten and Frank 2005). We compared our cascade classifiers to general-purpose ensemble of decision trees. For creating the general-purpose ensemble of decision trees we used the AdaBoost algorithm (Freund and Schapire 1996) with 10 iterations (as explained in Sect. 4.2.5). AdaBoost was also used to create an ensemble of seventy decision stumps.

The NegEx classifier (Chapman et al. 2001) was manually built using knowledge engineering methods in order to specifically resolve negations in medical narrative reports. The NegEx is a fixed classifier and has no learning capabilities. We compared our approach to NegEx in order to examine if an automatic learning method can achieve an equivalent performance.

Our algorithm was compared also to the HMM technique. Our HMM implementation approach is based on the observation that the negation detection problem can be regarded as a special case of part of speech tagging. We define a polarity tag (possible values are Negative and Positive) that is applicable to the <diagnosis> terms only. Our HMM implementation for the negation detection problem is based on the LingPipe/MedPost implementation. The LingPipe implementation accepts a file of tagged sentences as input. An example tagged sentence is: "*The_DD N-terminal_JJ region_NN had_VHD high_JJ homology_NN with_II Ran_NN BP2_NN/_SYM Nup_NN 358_MC ,_, a_DD nucleoporin_NN component_NN ,_, showing_VVG that_CST BS-63_NN was_VBD a_DD member_NN of_II the_DD NPC_NN family_NN ._.*"

These files are then used for training a HMM. The parameters the HMM determine how many characters to use as the basis for the model (8), the total number of characters (256), and an interpolation parameter for smoothing (8.0). According to LingPipe, these are reasonable default settings for English data.

In order to use LingPipe for our purpose, we converted our training corpus files to the format recognized by the LingPipe/MedPost. We implemented two HMM models. In the first HMM model all frequent words are used as states. In the second HMM model, for fair comparison to the regular expressions classifier, all the words that are considered in the regular expressions classifier are used as tags in the HMM implementation. In addition we use three more tags: (1) ni—for words that are not important; (2) neg—tag for negative diagnosis only; (3) pos—tag for positive diagnosis only. For example, the sentence "*the patient does not appear to be with an <diagnosis>*" is transformed to the tagged sentence: "*the_ni patient_patient does_ni not_not appear_ni to_to be_ni with_ni an_ni diagnosis_neg.*"

Our algorithm was also compared to the CRF algorithm. For this purpose, we used the MALLET package (McCallum 2007) for implementing the CRFs. We examined two

[2] The binary confusion matrix has four entries. Thus, by providing the values of these four measures, it is possible to restore other measures presented in Sect. 4.1.2, such as precision and recall.

options; in the first option we used the entire sentence as an input. In the second option, we used only the substring that appeared before the concept to be tagged.

4.1.2 Evaluation metrics

The decision made by the classifier can be represented in a structure known as a confusion matrix or contingency table. The goal of this experiment is examine the ability of classifier to correctly identify negations. Thus, the confusion matrix has been defined with respect to negations and has the following four categories (see Table 4): True positives (TP) are terms that are correctly recognized as negated. False positives (FP) refer to non-negated terms incorrectly labeled as negated. True negatives (TN) correspond to non-negated terms correctly labeled as non-negated. Finally, false negatives (FN) refer to negated terms incorrectly labeled as non-negated.

We use the well-known performance measures *precision* and *recall* (Van Rijsbergen 1979). Because there is a trade-off between the precision and the recall we also report their harmonic mean known as F-Measure. The main criterion for evaluating classifiers is the accuracy which is the proportion of correctly classified instances and the number of instances in the test corpus. Given the confusion matrix presented in Table 4, the above evaluation metrics can be expressed mathematically as following:

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$
$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$
$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FN} + \text{FP})$$
$$\text{F-Measure} = 2\text{TP}/(2\text{TP} + \text{FN} + \text{FP})$$

4.2 Results

4.2.1 Overall results

Table 5 presents the performance measures value obtained by the regular expressions classifiers compared to: (1) "Bag-of-words"; (2) NegEx classifier; (3) HMM classifier and (4) CRF classifier. Moreover, we compare the proposed cascaded algorithm with other regular expression-based classifiers: AdaBoost with C4.5 as inner classifier, AdaBoost with Decision Stump as inner classifier and a single C4.5 decision tree. Experimental evaluation is performed using ten fold cross-validation. The superscript "+" indicates that the performance of LCS Cascade is significantly higher than the corresponding algorithm using paired *t*-test at confidence level of 95%.

The results indicate that "Bag-of-words" obtains moderate Precision scores (the precision regarding the negative label) and moderate Recall scores (recall regarding the negative label). On the other hand NegEx obtains high Recall scores and moderate Precision scores. All regular expressions based classifier obtain both high Precision and high Recall. Moreover, no method has significantly outperformed the cascaded classifiers

Table 4 A binary confusion matrix

	Classified negated	Classified non-negated
Actually negated	TP	FN
Actually non-negated	FP	TN

Table 5 Benchmark results

Method	Precision	Recall	F-measure	Accuracy
Bag of words	$^{+}86.1 \pm 4.1\%$	$^{+}87 \pm 5.3\%$	$^{+}85.2 \pm 3.9\%$	$^{+}86 \pm 3.3\%$
NegEx	$^{+}87.7 \pm 0.4\%$	$99.0 \pm 0.6\%$	$93.0 \pm 0.3\%$	$92.6 \pm 0.3\%$
HMM				
Frequent words as states	$^{+}88.4 \pm 4.1\%$	$^{+}85.5 \pm 3.4\%$	$^{+}86.9 \pm 3.6\%$	$^{+}87.1 \pm 3.2\%$
Cascade words as states	$^{+}87.1 \pm 3.4\%$	$^{+}93.8 \pm 3.5\%$	$^{+}90.3 \pm 2.9\%$	$^{+}90.0 \pm 3.1\%$
CRF				
All words in the sentence	$^{+}88 \pm 4.5\%$	$^{+}89.1 \pm 2.4\%$	$^{+}88.5 \pm 2.7\%$	$^{+}88.6 \pm 2.2\%$
Only words that appear before the concept	$92.7 \pm 4.9\%$	$^{+}90.3 \pm 4.5\%$	$^{+}91.4 \pm 3.7\%$	$^{+}91.5 \pm 3.5\%$
Regular expression				
Single DT with LCS	$^{+}92.3 \pm 3.7\%$	$^{+}85.7 \pm 3.2\%$	$^{+}89 \pm 3.5\%$	$^{+}89.4 \pm 3.1\%$
Single DT Teiresias	$^{+}90 \pm 3.3\%$	$95.1 \pm 2.5\%$	$92.5 \pm 2.7\%$	$^{+}92.3 \pm 2.7\%$
10 AdaBoost DTs with LCS	$94.9 \pm 1.9\%$	$^{+}89.2 \pm 1.4\%$	$92 \pm 1.5\%$	$92.2 \pm 1.2\%$
70 AdaBoost of decision stumps with LCS	$96.3 \pm 1.6\%$	$^{+}83.8 \pm 1.1\%$	$^{+}89.6 \pm 1.4\%$	$^{+}90.3 \pm 1\%$
Cascade DTs with LCS	$94.4 \pm 2.4\%$	$97.4 \pm 2.5\%$	$95.9 \pm 1.9\%$	$95.8 \pm 1.8\%$
Cascade DTs with Teiresias	94.6 ± 2.6	96.7 ± 2.9	95.6 ± 2.1	95.6 ± 1.9

approach. Specifically, the new method outperforms HMM and CRF. Thus, regular expressions outperform other negation algorithms in both F-Measure (F-Measure precision regarding the negative label) and Accuracy. The LCS and Teiresias show similar performance.

Although previous works have shown that CRF significantly outperforms HMM, in this experimental study CRF's improvement is moderate. Maybe it can be explained by the fact that the major advantage of CRF POS tagger versus HMM POS tagger is in the ability of CRF to model orthographic and morphological word features, which is beyond the scope of this paper (all methods examined in this paper use words as features).

4.2.2 The suitability of the first cascaded decision tree

Figure 13 describes the Precision the regular expressions family "*no .{0,D} <diagnosis>*" as a function of the distance D. It is clearly shown that the Precision increases with the decrease of D, maximum achieved when $D = 0$. This explains the cascade strategy that enables high Precision in the first cascade without decreasing Recall. The Recall is not affected since this cascade can only classify negative instances. Instances that are not classified as negative are sent to the next cascade.

4.2.3 The effect of training set size

Figure 14 presents the effect of the training set size on the predictive performance and the number of regular expressions. It shows how the accuracy based on 10 folds cross-validation converges as the training set size is increased. As shown from the graph, the number of regular expressions (postfeature selection) is not directly correlated with the classifier accuracy.

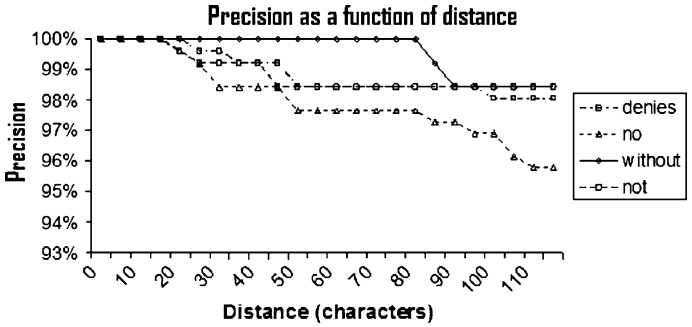

Fig. 13 Regular expression precision versus distance

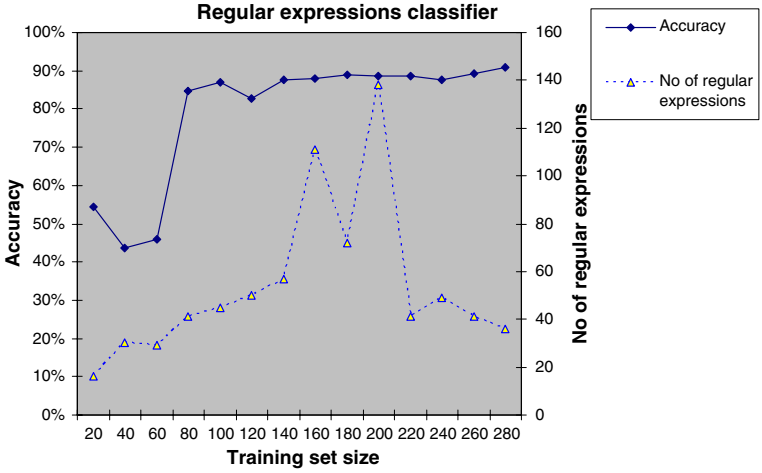

Fig. 14 Regular expressions classifier training

4.2.4 The effect of the feature selection

Table 6 compares the accuracy obtained by the regular expressions classifier before and after CFS feature selection based on 10 folds cross validation. The CFS filter eliminates on average about 95% of the features while achieving substantial accuracy gain. The filtering improves the classifier accuracy by about 5% on average.

4.2.5 The effect of the ensemble size

While the number of classifiers in the cascaded method is set to three, the number of iterations in the AdaBoost algorithm cannot be predetermined and should be tuned to the particular problem.

Figure 15 presents how the ensemble size affects the accuracy when the base classifier is C4.5. The figure presents only the first 11 examined sizes based on 10 folds cross validation. We also checked the ensemble size up to 20. However it has no effect on the accuracy (it remains as the accuracy of ensemble size of 10). It can be seen that the

Table 6 The effect of feature selection on accuracy

Classifier mode	Training set size		
	200	240	280
Cascaded LCS with only heuristic filtering			
Number of regular expressions	2069	2300	2225
Accuracy	81.9%	82.8%	87.0%
Cascaded LCS with heuristic filtering followed by CFS			
Number of regular expressions	138	49	35
Accuracy	88.5%	87.7%	91.1%

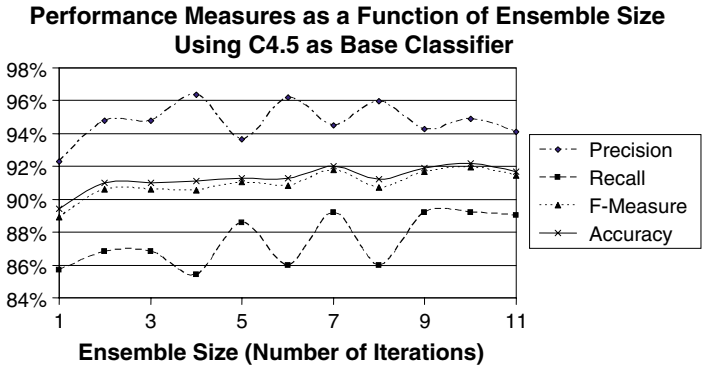

Fig. 15 Performance measures as a function of ensemble size using C4.5 as base classifier

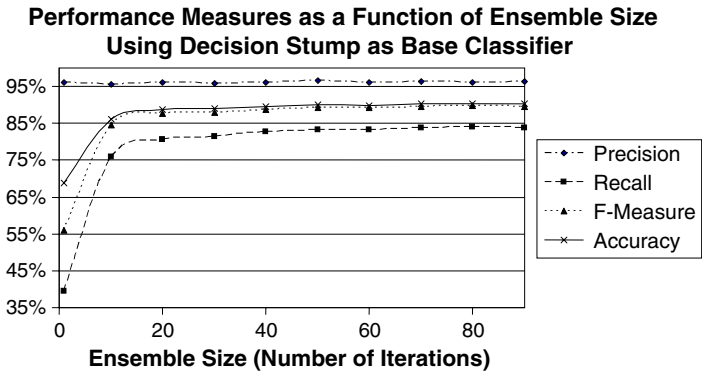

Fig. 16 Performance as a function of ensemble size using decision stump as base classifier

accuracy of the proposed method is improved from 89.39% to 92.20% (in ensemble size of 10).

Figure 16 presents the results obtained for AdaBoost using Decision Stump as base classifier. In contrary to the number of iterations required with C4.5, Decision Stumps requires many more iterations in order to converge to the optimal performance.

4.2.6 Comparing regular expressions with bag of words

The results presented in the previous subsections indicated that the proposed method can obtain higher accuracy than the simple "Bag-of-words". In this section, we are interested in checking if the improved result implies that the regular expression approach covers all predictive expressiveness of the "Bag-of-words". For this purpose we join the two datasets into a single dataset, thus each instance is now characterized by a set of discovered patterns and by a set of words. Table 7 presents the performance of the cascaded classifier based on 10 folds cross validation. It seems that adding the "Bag-of-words" attributes to the regular expression attributes has reduced accuracy.

Table 7 presents also the classifier's complexity. As this work focuses on cascaded decision trees, the classifier complexity was measured as the total number of nodes, including the leaves in all trees. It can be seen that employing regular expression patterns can reduce the classifier complexity in about 20%.

Table 8 presents some of the patterns obtained by the proposed algorithm. Previous works showed few words/phrases that are strong indicators of negative context (Averbuch et al. 2004). In these works mostly two word phrases (e.g. "showed no") were therefore finally considered by the classifier. Terms such as "no" and "not" were not included in their profile because their sole appearance is not a sufficient indication for negation. In this work the pattern learning algorithm learns the two phrase patterns as well as single term patterns such as "no". This is because the term "no" is a strong negation indicator when it precedes the medical concept, or when combined with additional patterns using a decision tree. These examples explain the accuracy improvement in the proposed approach compared to the "bag-of-words" based classifier.

4.2.7 Sensitivity to diagnosis type

Table 9 presents the results of a sensitivity analysis to diagnosis type. The most frequent types of diagnosis were evaluated: Weigh loss; Nausea; Vomit; and Abdominal pain.

In this experiment we used the "leave-one-out" approach in which a certain diagnosis type was used to generate the test set, while the remaining diagnosis types have been used to create the training set. As can be seen, the accuracy for Vomit seems to be less than the

Table 7 Comparing regular expression with "Bag-of-words"

Method	Accuracy (%)	Complexity
Regular expression	95.8 ± 1.8	82
Bag-of-words	87.3 ± 2.4	113
Combined	91.45 ± 2.7	126

Table 8 Example of patterns learned

Negative patterns
.{0,200} no <DIAGNOSIS>
.{0,50}showed no.{0,50} <DIAGNOSIS>
Positive Patterns
.{0,100}patient.{0,5} has.{0,50} <DIAGNOSIS>
.{0,100}history of.{0,100} <DIAGNOSIS>

Table 9 Sensitivity to diagnosis type

Diagnosis type	Accuracy (%)	Number of instances
Weight loss	96.27	134
Nausea	96.58	234
Vomit	91.43	245
Abdominal pain	97.98	247

average. Using the contingency table approach, we checked whether the differences are statistically significant with confidence level of 95%, to conclude that the difference is not statistically significant.

4.2.8 Sensitivity to different sources and document type

In this section, we add to the 1,766 instances that have been used till now, several corpora which totally include 1,492 instances. These instances were obtained from other hospitals; 489 instances obtained from the Epilepsy care center of Missouri; 233 instances obtained from the Staten island hospital; and 770 instances obtained from an anonymous north-American hospital.

Table 10 indicates also the document type distribution of the new instances. While the original training corpus was based only on discharge summaries, the new sources also introduce new document type, such as outpatient and consult reports. The aim of this experiment was to examine the generalization ability of the cascaded design on new data sources or document type. For this purpose we used the "leave-one-out" approach, in which a different data source is used as a test set in each iteration.

Table 11 summarizes the predictive power obtained by the cascaded classifier, CRF and bag-of-words. The results indicate that there is deterioration in the predictive power of both cascaded classifier and CRF regarding the negative class. However the accuracy is kept high. This can be explained by the fact that the proportion of negative instances in the new corpora is much smaller than in the training corpus. The performance of the new method is usually slightly better than the other two methods. As suggested by Dietterich (1998), we used the McNemar's test with confidence level of 95% for examining if the differences in

Table 10 Corpus sentences distribution

Document types	Data source				Total
	Mount Sinai	Staten Island	Epilepsy care center	Anonymous north-American	
Consult		29			29
Discharge	1,766			584	2,350
Inpatient				162	162
Outpatient			489	24	513
Operations		204			204
Total	1,766	233	489	770	3,258

Table 11 Accuracy performance of various corpora

Data source	Cascaded regular expressions				CRF				Bag-of-words			
	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)
Epilepsy care center	80	69.84	74.58	93.87	66.2	77.78	71.53	92.02	74.07	64.72	69.08	63.39
Staten Island	88.2	80.36	84.11	92.70	85.7	75.00	80	91	86.21	76.34	80.97	79.83
Anonymous north-American	82.3	63.64	71.79	97.14	77.7	63.64	70.00	96.88	76.35	60.08	67.25	60.78
Mount Sinai	92.3	91.73	92.78	92.3	87.4	87.06	88.06	87.56	86.96	84.57	85.74	84.94
Weighted average	**87.80**	**80.99**	**84.47**	**93.71**	**81.80**	**79.27**	**80.73**	**90.68**	**82.46**	**75.21**	**78.53**	**75.63**

the accuracy and Recall are significant.[3] The test results indicate that the cascaded classifier method is significantly better than the bag-of-words method in both criteria. However, we could not conclude that the differences between the cascaded classifier method and the CRF method are significant. Moreover, we find out that there is no significant difference between the two methods when summing up the results from all data sources.[4]

Using the contingency table approach, we checked whether the differences in the performance obtained by a certain method on the various data sources are statistically significant with confidence level of 95%, to conclude that there are no significant differences in any of the measures. This means that all methods provide stable performance among different data sources.

Table 12 presents the accuracy obtained for each document type, using the "leave-one-out" procedure. The performance of the new method is usually slightly better than CRF method. We used the McNemar's test with confidence level of 95%, to find out that there is no significant difference between the methods in any of the document types when each type has been checked separately. However as proposed by Demšar (2006), we are using the Wilcoxon Signed-Rank Test, to examine whether the differences between the methods are significant when taking into consideration all document types. We conclude that the cascaded classifier is significantly better than the CRF method in the Precision and the accuracy criteria. Moreover cascaded classifier is significantly better than the Bag-of-Words method in the Recall, F-Measure and accuracy criteria.

Using the contingency table approach, we checked whether the differences in the performance obtained by a certain method on the various document types are statistically significant with confidence level of 95%, to conclude that there are no significant differences in any of the measures.

4.2.9 Sensitivity to the authorship

In this section, we analyze whether the reports' author has an effect on the classifiers' performance. Table 13 presents the distribution of different authors in the datasets. Table 14 presents the mean performance obtained by using a "leave-one-out" procedure such that in each iteration one author has been left out and used as a test set. A two-way analysis of variance (ANOVA) with was performed. The dependent variable was the accuracy. The results of the ANOVA showed that the main effects of the authors $F = 3.106$, $p < 0.001$ and the classification $F = 25.63$, $p < 0.001$ are both significant.

4.2.10 Error analysis

Analyzing the reasons for wrong classifications, suggest the following six main categories of error:

1. Compound sentence—Compound sentences are composed of two or more clauses that are joined by a coordinating conjunction or a semicolon. For example: "The patient denies any chest pain or shortness of breath but admits fever." This sentence is built from two clauses separated by the word "but," which alters the context of the second

[3] McNemar's test cannot be used for the Precision measure or the F-Measure.

[4] The Wilcoxon signed-rank test could not be used here because for sample sizes smaller than 5 there are no possible critical values that would be significant at or beyond the baseline 95% level.

Table 12 Performance of various document types

Document type	Cascaded regular expressions				CRF				Bag-of-words			
	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)
Consult	100	100.00	100.00	100.00	50.00	100.00	66.67	96.55	71.43	52.63	60.61	55.17
Discharge	89.31	91.79	90.53	92.64	86.2	89.6	87.9	90.5	84.32	74.40	79.05	77.45
Inpatient	50.00	50.00	50.00	97.53	40.00	50.00	44.44	96.91	50.00	26.79	34.88	30.86
Outpatient	77.19	69.84	73.33	93.76	66.22	77.78	71.53	92.40	75.00	63.83	68.97	63.16
Operation	88.00	80.00	83.81	91.67	87.23	74.55	80.39	90.20	86.96	65.57	74.77	73.53

Table 13 Test corpus sentences distribution

Data source	Number of distinct authors	Mean reports per author	Standard deviation of reports per author
Mount Sinai	11	160.54	25.72
Staten Island	6	38.83	5.08
Epilepsy care center	3	163	17.63
Anonymous north-American	8	96.25	6.2

clause. The classifier might label all terms after the verb "deny" as negated including the term "fever" based on the regular expression "denies *.{0,60} <diagnosis>*".

2. Future Reference—In this type of sentence, the patient is given instructions on how to react to a symptom he may develop, but currently lacks. For example: "The patient was given clear instructions to call for any worsening pain, fever, chills, bleeding." In this case the patient does not suffer from fever, chills or bleeding and a query for one of these symptoms will mistakenly retrieve the document.
3. Negation indicating existence—although the meaning of a word might be negative, the context in which it is written might indicate otherwise. For example: "The patient could not tolerate the nausea and vomiting associated with Carboplatin."
4. Positive adjective—A sentence is written in a negative form, but an adjective prior to one of the medical term actually indicates its existence. For example: "There were no fevers, headache or dizziness at home and no diffuse abdominal pain, fair appetite with significant weight loss." The adjectives "fair" and "significant" in the sentence indicates that the following symptoms actually do exist.
5. Wrong sentence boundaries—Sometimes the boundary of a sentence is not identified correctly. In this case, one sentence is broken into two, or two sentences are considered as one. For example: "She denies any shortness of breath, dyspnea, chest pain, G.I. bleed, fever or chills." In this case, the terms bleed, fever and chills were not associated with the negation, as the negation phrase was part of the first sentence.
6. Other reasons—which are not covered by the first five categories.

Figure 17 presents the distribution of errors in for the proposed cascaded classifiers. It can be seen that the "compound sentence" category is the origin for most of the errors. Thus, it suggests that breaking the sentences into smaller clauses, may improve the performance.

5 Discussion

The experimental study provides a strong evidence that in the negation problem, regular expressions are better than bag-of-words, in both accuracy and compactness (i.e., obtaining smaller models). In fact, regular expressions can be considered to be a generalization of the bag-of-words representation or any n-gram representation. While there are several IE methods that use regular expressions-like representation, such as WHISK, Soderland (1999), we suggest arranging several expressions into a hierarchical structure which constitutes a decision tree.

Using a decision tree as a base classifier in this case has several advantages: (1) the sentence is not classified according to a single regular expression, but is classified based on

Table 14 Accuracy performance of various authors

Data Source	Author	Cascaded regular expressions				CRF				Bag-of-words			
		Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)
Epilepsy care center	1	78.12	68.17	72.81	91.15	70.46	64.59	67.40	87.83	70.16	60.79	65.14	83.97
	2	76.58	68.75	72.45	90.92	75.69	62.22	68.30	88.68	69.20	60.89	64.78	80.14
	3	74.14	65.85	69.75	88.35	76.70	59.68	67.13	84.09	69.20	54.62	61.05	79.39
Staten Island	1	81.93	74.58	78.08	91.15	75.41	81.77	78.46	83.20	75.19	81.60	78.26	81.18
	2	87.23	73.24	79.62	88.72	82.36	68.31	74.68	92.39	88.42	69.95	78.11	85.88
	3	88.34	70.76	78.58	93.69	83.89	68.05	75.14	91.65	75.62	66.82	70.95	88.02
	4	95.18	84.24	89.37	77.89	93.08	81.03	86.64	71.61	87.61	79.32	83.26	69.75
	5	89.33	64.28	74.76	91.15	81.77	60.77	69.72	90.06	81.84	62.75	71.03	86.47
	6	99.74	92.67	96.08	90.27	90.14	85.31	87.66	82.84	86.86	77.51	81.92	79.03
Anonymous north-American	1	81.17	67.13	73.49	95.14	76.08	64.33	69.71	88.59	69.98	60.41	64.84	81.73
	2	93.17	65.66	77.04	97.00	84.06	71.42	77.23	87.33	82.36	66.72	73.72	81.89
	3	80.36	60.11	68.77	91.42	73.73	54.66	62.78	86.17	72.46	50.37	59.43	79.67
	4	88.30	62.94	73.49	90.73	88.50	62.06	72.96	92.29	94.60	65.55	77.44	95.78
	5	80.06	61.62	69.64	94.65	75.10	57.49	65.12	87.90	71.21	52.30	60.31	79.55
	6	77.09	68.00	72.26	95.19	72.97	64.22	68.31	91.82	79.17	66.14	72.07	91.09
	7	82.77	65.02	72.83	97.98	74.60	60.19	66.62	96.05	72.54	56.11	63.28	95.52
	8	79.73	64.41	71.26	90.62	74.75	63.41	68.62	83.89	74.95	59.83	66.54	79.43

Table 14 continued

Data Source	Author	Cascaded regular expressions				CRF				Bag-of-words			
		Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	Accuracy (%)
Mount Sinai	1	93.17	95.27	94.21	97.00	83.19	93.11	87.87	94.73	81.27	87.96	84.48	90.23
	2	88.55	91.43	89.97	91.42	86.67	88.15	87.40	50.48	73.38	79.14	76.15	77.97
	3	91.16	97.14	94.06	90.73	87.97	85.05	86.49	73.06	83.98	79.16	81.50	69.45
	4	97.13	89.12	92.95	94.65	80.57	86.54	83.45	77.12	76.75	74.20	75.46	75.00
	5	90.02	86.28	88.11	90.54	84.22	76.84	80.36	84.53	82.95	73.12	77.73	81.51
	6	87.21	90.35	88.75	92.82	86.37	87.73	87.04	89.50	81.01	81.11	81.06	85.77
	7	97.02	93.18	95.06	97.16	92.12	44.54	60.05	91.42	86.39	70.60	77.70	75.42
	8	95.12	94.60	94.86	98.14	92.14	48.26	63.34	90.70	78.48	37.45	50.71	68.65
	9	100.00	89.41	94.41	85.60	85.17	89.38	87.23	61.42	67.32	86.76	75.81	59.43
	10	95.48	93.06	94.26	89.06	60.12	83.82	70.02	84.04	37.01	43.77	40.11	50.17
	11	97.89	95.17	96.51	93.98	64.13	92.12	75.62	91.19	84.17	90.12	87.04	76.74

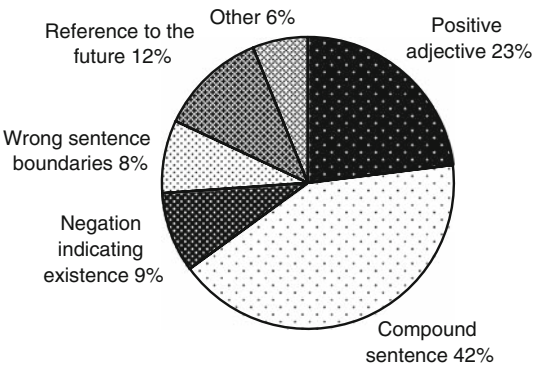

Fig. 17 Distribution of Errors for the cascaded classifiers algorithm

a set of regular expressions, i.e. this classifier can be used to indicate that a sentence is classified to the label "positive" only if it matched two regular expressions and does not match a third regular expression. This is more expressive than the classical approach in which the classification is based on a single regular expression. In this way, instead of searching for complicated regular expressions, we can search for simple regular expressions and "rely" on the decision tree to combine them. In some cases, it is possible to express a tree path comprised of several simple regular expressions as a single complicated regular expression; (2) the hierarchical structure of a decision tree enforces an order (priority) in the usage of regular expressions, i.e., given a new sentence, not all regular expressions should be matched in advance but one regular expression at a time based on the specific branch traversing. Moreover in this way the desired property of lexical analysis known as un-ambiguity (also known as conflict resolution in Expert Systems) which is usually resolved by the longest match and rule priority is inherently resolved here; (3) as opposed to other classifiers (such as neural networks) the decision tree is a white box model whose meaning can be easily explained.

The experimental study strengthens the well-known fact that it is possible to boost the predictive performance by combining several decision trees. Nevertheless, an important drawback of general-purpose ensemble method, such as AdaBoost (Freund and Schapire 1996), is that they are difficult to understand. The resulting ensemble is considered to be less comprehensible since the user is required to capture several decision trees instead of a single decision tree. In addition, the ensemble members might even contradict one another. On the other hand, in the proposed cascaded design, the classifiers do not compete with each other and do not contradict one another, but they are complementing each other. Specifically, we either make a decision in the current cascade or postpone the decision to the next cascade. In any case, the decision is made by a single classifier, and not by some voting mechanism. Moreover, the cascade increases precision by adding additional layers of decision tree classifiers and easily regulates the classifier complexity/precision tradeoff. In addition, the cascaded design needs only three classifiers, as opposed to much larger ensemble size in the case of AdaBoost.

Beside the fact that the proposed method has provided a higher accuracy than the HMM and the CRF classifiers, it can be easily transformed into a maintainable source code. Modern programming languages, such as Java or C#, or script languages such as Perl and Python include inherent support for regular expressions. Any programmer can manipulate these models quite easily as opposed to HMM or CRF models which requires that programmers be familiar with the notion of probability.

As indicated in Sect. 4.2.4 feature selection can be used to improve predictive power. The number of regular expressions (pre-feature selection) is usually greater than linear in the number of instances in the training set. For instance, if the paired LCS approach is used, then for every pair in the training set we obtain a regular expression. At first glance, it seems redundant to use feature selection as a preprocess phase for the training phase. Decision trees inducers, as opposed to other induction methods, incorporate in their training phase a built-in feature selection mechanism. Still, it is well known that correlated and irrelevant features may degrade the performance of decision trees inducers. Moreover, in the way we create regular expressions there are many features that are correlative. This phenomenon can be explained by the fact that feature selection in decision trees is performed on one attribute at a time and only at the root node over the entire decision space. In subsequent nodes, the training set is divided into several sub-sets and the features are selected according to their local predictive power (Perner 2001). Geometrically, it means that the selection of features is done in orthogonal decision subspaces, which do not necessarily represent the distribution of the entire instance space. It has been shown that the predictive performance of decision trees could be improved with an appropriate feature pre-selection phase. Moreover using feature selection can reduce the number of nodes in the tree, making it more compact.

Another way to avoid the "curse of dimensionality" in this case, is to merge several expressions into one expression by generalizing them.[5] However, this increases the risk of over generalization. This is the typical sensitivity-Recall problem. For example if we merge the first and second expressions in Table 8, we obtain the following reasonable expression: *.{0,200} no.{0,50} <DIAGNOSIS>*. On the other hand merging the third and fourth expressions results with the meaningless and over-generalized expression: *.{0,220} <DIAGNOSIS>*.

A criterion for merging regular expressions can be based on existing computational learning theoretical bounds (such as the VC-Dimension) that trade training accuracy with model complexity. Merging regular expressions reduces a model's complexity but at the same time it might also reduce training accuracy (due to generalization). The merging can be performed in any stage: pre-training like feature selection, during the actual training of decision trees (as an extension to the splitting criterion), or post growing as an extension to the pruning phase.

Regular expressions seem quite useful for the examined task. But they do have limitations. For instance, because they do not use syntax but only use words and character length gaps, they can make mistakes due to, for example, a training set that only showed one adjective modifying a negated noun (e.g., no persistent cough) but a test set that has multiple adjectives intervening between the negation phrase and the negated concept. Moreover in order that a negative modifier will be included in the model, it should be repeated at least twice in two different instances in the training set. This is because the regular expressions are created by comparing two strings and identifying the common substring. If the modifier appears only once then it will never be included in any of the regular expressions.

[5] Merging regular expressions that have been created by LCS or Teiresias is straightforward. The merging is performed based on the specific words used in each pattern while ignoring the wild-cards. For instance the "wordy" representation of *.{0,50}showed no.{0,50} <DIAGNOSIS>* is showed no <DIAGNOSIS>. Thus, we first remove the special characters from the two patterns. Then the LCS algorithm is used to create a new generalized expression from these two "wordy" strings. Finally, the wild-cards that have been omitted are reattached to the new pattern.

6 Conclusions and further research

A new pattern-based algorithm for identifying context in free-text medical narratives is presented. It is shown that the new algorithm is superior to previous methods. The new algorithm manages to automatically learn patterns similar to manually written patterns for negation detection, for example in the work of Mutalik et al. (2001), within the same level of accuracy. The advantages of the new method are: accuracy improvement compared to other machine learning methods, comprehensibility of the results and much faster than manual knowledge engineering techniques with matching accuracy.

We suggest an obvious expansion in the medical domain, train the classifier to detect context in general (e.g. in the family), not just the negated context. This requires a sufficiently large training corpus with additional context categories. Another expansion in the medical domain is to implement the method for finding relationships between concepts in medical narratives. This can be useful for applications such as: Process control to identify adverse events in medical treatment; Modern evidence based practices; and also in extending the UMLS with new relations or concepts.

We suggest several research directions to improve the method itself. Cascading several classifiers has shown to raise the classifier accuracy. Further study is needed to obtain sufficient results for suggesting a methodology for cascading classifiers. Classifier complexity versus accuracy is yet another issue for further research, to analyze the tradeoffs between the number of patterns and the classifier performance. Additional studies are required to examine if the proposed method can be efficient in identifying negation in other domains (such as maintenance reports of technicians) and in other closely related problems such as POS.

Acknowledgements The authors would like to thank J. Kannry, M.D. (Center for Medical Informatics, Department of Medicine, Mount Sinai—NYU, New York, USA), T. Karson, M.D. (Departments of Clinical Informatics and Cardiology, Mount Sinai School of Medicine, New York, USA) and M. Averbuch, M.D. (Tel-Aviv Sourasky Medical Center, Israel) for providing the data that have been used in the experimental study and for helping doing the initial prior studies which lead eventually to this study.

References

Aronow, D., Feng, F., & Croft, W. B. (1999). Ad hoc classification of radiology reports. *Journal of the American Medical Informatics Association, 6*(5), 393–411.

Averbuch, M., Karson, T., Ben-Ami, B., Maimon, O., & Rokach, L. (2004). *Context-sensitive medical information retrieval*, MEDINFO-2004, San Francisco, CA, September. IOS Press, pp. 282–262.

Bekkerman, R., & Allen, J. (2003). *Using bigrams in text categorization*. Department of Computer Science, University of Massachusetts, Amherst, CIIR Technical Report IR-408.

Califf, M. E., & Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conf. on Artificial Intelligence*, pp. 328–334.

Caropreso, M., Matwin, S., & Sebastiani, F. (2001). A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In *Text databases and document management: Theory and practice* (pp. 78–102). Idea Group Publishing.

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., & Buchanann, B. G. (2001). A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics, 34*, 301–310. doi:10.1006/jbin.2001.1029.

Ciravegna, F. (2001). Adaptive information extraction from text by rule induction and generalization, In *Proceedings of the 17th International Joint Conference on Artificial Intelligence.*

Cohn, T. A. (2007). Scaling conditional random fields for natural language processing. PhD dissertation, University of Melbourne.

Damashek, M. (1995). Gauging similarity with N-grams: Language-independent categorization of text. *Science, 267*, 843–848.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*(1), 1–30.

Dietterich, G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*, 1895–1924. doi:10.1162/089976698300017197.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pp. 148–155.

Esuli, A., & Sebastiani, F. (2005). Determining the semantic orientation of terms through gloss analysis. In *Proceedings of CIKM-05, the ACM SIGIR Conference on Information and Knowledge Management, Bremen, DE.*

Fiszman, M., & Haug, P. J. (2000). Using medical language processing to support real-time evaluation of pneumonia guidelines. In *Proceedings of AMIA Symposium*, pp. 235–239.

Fiszman, M., Chapman, W. W., Aronsky, D., Evans, R. S., & Haug, P. J. (2000). Automatic detection of acute bacterial pneumonia from chest X-ray reports. *Journal of the American Medical Informatics Association, 7*, 593–604.

Freitag, D., & Kushmerick, N. (2000). Boosted wrapper induction. In *Proceedings of the Seventh National Conference on Artificial*, Austin, Texas, pp. 577–583.

Freitag, D. (1998). Toward general-purpose learning for information extraction. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pp. 404–408.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference*, pp. 325–332.

Friedman, C., & Hripcsak, G. (1998). Evaluating natural language processors in the clinical domain. *Methods of Information in Medicine, 37*, 334–344.

Friedman, C., Alderson, P., Austin, J., Cimino, J., & Johnson, S. (1994). A general natural-language text processor for clinical radiology. *Journal of the American Medical Informatics Association, 1*(2), 161–174.

Goldin, I., & Chapman, W. W. (2003). Learning to detect negation with 'not' in medical texts. In: E. Brown, W. Hersh, & A. Valencia (Eds.), *Proceedings of the Workshop on Text Analysis and Search for Bioinformatics at the 26th Annual International ACM SIGIR Conference (SIGIR-2003)*.

Hall, M. (1999). Correlation-based feature selection for machine learning. PhD thesis, University of Waikato.

Halteren, H., Zavrel, J., & Daelemans, W. (2001). Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics, 27*(2), 199–229. doi:10.1162/089120101750300508.

Hersh, W. R., & Hickam, D. H. (1995). Information retrieval in medicine: the SAPHIRE experience. *Journal of the American Society for Information Science American Society for Information Science, 46*, 743–747. doi:10.1002/(SICI)1097-4571(199512)46:10<743::AID-ASI5>3.0.CO;2-C.

Horn, L. R. (2001). *A natural history of negation*. Stanford, CA: Center for the Study of Language and Information. ISBN: 1575863367

Hripcsak, G., Knirsch, C. A., Jain, N. L., Stazesky, R. C., Pablos-mendez, A., & Fulmer, T. (1999). A health information network for managing innercity tuberculosis: Bridging clinical care, public health, and home care. *Computers and Biomedical Research, an International Journal, 32*(1), 67–76. doi:10.1006/cbmr.1998.1496.

Java, A. (2007). A framework for modeling influence, opinions and structure in social media. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, pp. 1933-1934.

Kim, S., & Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of the 20th international conference on computational linguistics, August 23–27 Geneva, Switzerland*, *International Conference On Computational Linguistics*.

Kupiec, J. M. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language, 6*, 225–242. doi:10.1016/0885-2308(92)90019-Z.

Kushmerick, N., Weld, D. S., & Doorenbos, R. B. (1997). Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 729–737.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, pp. 282–289.

Leroy, G., Chen, H., & Martinez, J. D. (2003). A shallow parser based on closed-class words to capture relations in biomedical text. *Journal of Biomedical Informatics, 36*, 145–158. doi:10.1016/S1532-0464(03)00039-X.

Lindbergh, D. A. B., & Humphreys, B. L. (1993). The unified medical language system. In van Bemmel, J. H. & McCray, A. T. (Eds.), *1993 yearbook of medical informatics* (pp. 41–51). The Netherlands: IMIA.

Lingpipe. Home page, <http://www.alias-i.com/lingpipe/demos/tutorial/posTags/read-me.html>, Accessed 12 March 2007.

McCallum, A. K. MALLET: A machine learning for language toolkit. Home page, http://mallet.cs.umass.edu. Accessed 12 March 2007.

Muslea, I., Minton, S., & Knoblock, C. (2001). Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems, 4*, 93–114.

Mutalik, P. G., Deshpande, A., & Nadkarni, P. M. (2001). Use of general-purpose negation detection to augment concept indexing of medical documents: A quantitative study using the UMLS. *Journal of the American Medical Informatics Association, 8*(6), 598–609.

Myers, E. W., & An, O. (1986). (ND) Difference algorithm and its variations. *Algorithmica, 1*(1), 251–266. doi:10.1007/BF01840446.

Nadkarni, P. (2000). Information retrieval in medicine: Overview and applications. *Journal of Postgraduate Medicine, 46*(2), 116–122.

Perner, P. (2001). Improving the accuracy of decision tree induction by feature pre-selection. *Applied Artificial Intelligence, 15*(8), 747–760. doi:10.1080/088395101317018582.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann, Los Altos, CA.

Rigoutsos, I., & Floratos, A. (1998). Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics (Oxford, England), 14*(2), 229.

Rokach, L., Averbuch, M., & Maimon, O. (2004). *Information retrieval system for medical narrative reports* (pp. 217–228). Lecture notes in artificial intelligence, 3055. Springer-Verlag.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1), 1–47. doi:10.1145/505282.505283.

Seymore, K., McCallum, A., & Rosenfeld, R. (1999) Learning hidden markov model structure for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence: Workshop on Machine Learning for Information Extraction*. Orlando, FL, pp. 37–42.

Smith, L., Rindflesch, T., & Wilbur, W. J. (2004). MedPost: A part-of-speech tagger for biomedical text. *Bioinformatics (Oxford, England), 20*(14), 2320–2321. doi:10.1093/bioinformatics/bth227.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning, 34*, 233–272. doi:10.1023/A:1007562322031.

Tottie, G. (1991). *Negation in English speech and writing: a study in variation*. Academic Press: New York.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews, In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, pp. 417–424.

Van Rijsbergen, C. J. (1979). *Information retrieval* (2nd ed.). London: Butterworths.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver, BC, Canada.

Witten, I. H., & Eibe, F. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco: Morgan Kaufmann.