The Emory System for Extracting Medical Concepts at 2010 i2b2 Challenge: Integrating Natural Language Processing and Machine Learning Techniques.

Akshatha K. Pai[1], Eugene Agichtein, PhD[1], Andrew R. Post, MD, PhD[2], Joel H. Saltz, MD, PhD[1,2]

[1]Emory University Math & Computer Science Department, Atlanta, GA; [2]Emory University Center for Comprehensive Informatics, Atlanta, GA.

Abstract

With the increase in biomedical information available electronically, it is becoming exceedingly difficult to comprehend and identify information which is of relevance to us. The application of natural language processing is useful to facilitate retrieval of this information. This paper focuses on extracting medical concepts – treatments, tests and problems from unstructured data such as patients' discharge summaries and progress notes, as used for the 2010 NLP i2b2 Challenge. Our system extracts an initial set of candidate concepts using a controlled terminology system, labels the concepts found, and is then trained on the provided labeled data, with additional features integrating linguistic, lexical, and structural information, using a supervised sequence learning (conditional random field) algorithm. The outcomes of the experiments show that augmenting natural language processing techniques with supervised machine learning significantly and consistently boosts extraction performance both on the provided training data and on the official test data.

Introduction

In today's world, as the demand for health care is exploding, managing and handling paper based clinical records is becoming a challenge. Many health care institutions have found it beneficial to store patients' medical data in an electronic format, which has opened new avenues to medical researchers to discover treatments, tests, problems and their implications and also has led its way into research in natural language processing on clinical texts.

i2b2 (Informatics for Integrating Biology and the Bedside) is an NIH-funded National Center for Biomedical Computing based at Partners HealthCare System. It has organized a Natural Language Processing (NLP) Challenge to facilitate research in natural language processing for parsing clinical texts and extracting medical information such as problems, treatments and tests that is buried within the unstructured text. As a part of the 2010 Challenge, i2b2 provided discharge summaries and progress notes from about 350 patients. Part of this data was used to develop and train our system, and the remainder was used for validation and final evaluation. The Challenge organizers also provided an evaluation program, used to compute the extraction performance metrics.

Methods

This section describes the general architecture of our system and the implementation details for some of the extraction components, with particular focus on the machine learning-based component.

General Approach and System Architecture

Our system is based on CTAKES[1], a Java-based pipelined concept extraction workflow, that is in turn based on the Unstructured Information Management Architecture (UIMA) framework [2]. UIMA is an open-source platform for handling text and other unstructured information (text, speech, audio, video data). Applications using UIMA are decomposed into components or annotators (e.g., part of speech tagger) which are managed by UIMA. CTAKES leverages the Unified Medical Language System (UMLS) to identify words and phrases in the text corresponding to biomedical concepts and annotating those by assigning unique identifiers to each concept. CTAKES is a rule-based system, which operates primarily by matching the text to the corresponding dictionary and knowledge base resources.

These concepts are then processed using NLP to exploit the grammatical and semantic structure of the text to further refine the extracted information. As the last step, we apply machine learning techniques to augment and filter the candidate concepts, based on the contextual, structural, and lexical information extracted from the original text. Specifically, we applied the Mallet [4] implementation of the Conditional Random Fields (CRF) classification algorithm [5] by encoding the output of the NLP and dictionary-matching steps as *features* provided as input to the CRF algorithm.

System Implementation and Processing Pipeline

[1] https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/OHNLP_Documentation_and_Downloads

We have extended the CTAKES annotation pipeline for our task of extracting treatments, medical problems and tests from clinical documents as illustrated in the Figure 1, and described below.

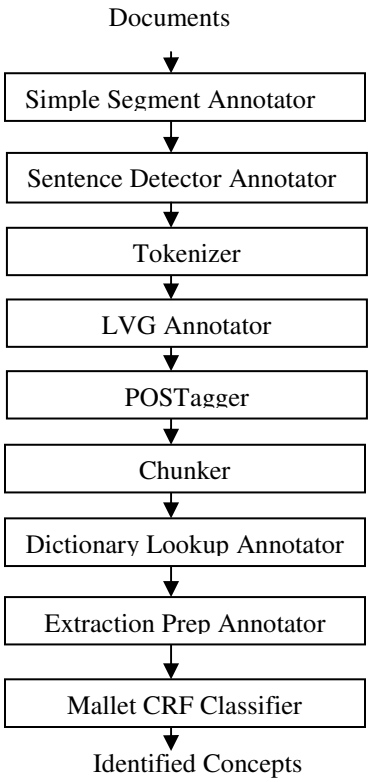

Figure 1: Extended CTAKES Pipeline

The UIMA-based framework (Figure 1) processes each document through the pipeline above. Each of the boxes corresponds to an *annotator*, implemented as follows:

- *Simple Segment Annotator* creates a Simple Segment annotation containing the entire document's text. Its output is used by the later annotators.
- *Sentence Detector Annotator* identifies the sentence boundaries within the text.
- *Tokenizer* identifies all the tokens in the text and creates token annotations.
- *Lexical Variant Generator (LVG)* annotator generates the canonical forms for all the tokens using specialized lexical tools.
- *POS Tagger* tags the parts of speech of all of the tokens using the OpenNLP Library [9] part-of-speech tagger.
- *Chunker* tags all the words in a phrase together using the OpenNLP chunker tool [9].
- *Dictionary Lookup Annotator* produces a look up for all the tokens limited within the window which in our case is a sentence. A search is done on a Lucene-based free text index (http://lucene.apache.org) that contains all the terms obtained from the SNOMED and RXNORM terminologies along with their UMLS semantic types. The lookup hits are then filtered only to extract medical problems, tests and treatments based on their semantic types. Sentences containing the lookup hit are parsed using the Stanford Dependency parser [6], which identifies the grammatical structure of the sentence and gives the dependency (object, subject of a verb phrase for example) between the words. The rules are applied on the parser's output to identify if a surrounding word to the hit needs to be extracted. The rules applied were as follows. If a lookup hit is a modifier in a noun phrase then it would not be marked as a separate concept but marked with the whole phrase. Another rule was applied to identify prepositional phrases and to include up to one modifier phrase. An additional rule was applied to not include any concepts within the headings. The annotations were created with the whole concept along with their span in the document.
- *Extraction Prep Annotator* is the final annotator in the pipeline. It implements feature generation for input to the CRF classifier. The features were selected and generated for each token in the text in a format that is compatible with Mallet's input format. The generated features were appended to a features file containing the tokens in all the documents along with their features. Many target concepts were actually abbreviations. For such cases, we included a feature for a token if it was a known abbreviation to a problem, treatment or a test. Our first attempt to identifying abbreviations was to store the initials of every RxNORM and SNOMED UMLS term in a Lucene dictionary, which was searched for every word in the document whose length was less than 4 letters. This approach resulted in a lot of false positives. Instead, we indexed a list of abbreviations obtained from the Berman's biomedical abbreviation list [8] using Lucene. This approach turned out to work well, and is the procedure that we used to handle feature generation for the words that were abbreviations.

- *Mallet CRF Classifier:* A script was written that uses the appropriate feature and label filesto train (or test) the Mallet classifier.

CRF Implementation

Selecting the features for a machine learning task is the most crucial aspect of training a classifier. The bulk of tuning our system for this task was selecting and implementing features for classification. Table 1 describes few of the important ones that we included apart from the features describing the token itself like is_capitalized, is_number, contains_number, etc.

Table 1: List of important CRF features

Feature name	*Description*
is_heading_name	Heading_name text in the feature was the name of the heading a particular token was associated with. Every token had a feature depending on the name of its heading. As an example a token under the heading HISTORY OF PRESENT ILLNESS would have a feature is_history_of_present_illness.
is_heading	Feature is assigned to only those tokens which were part of the heading in a document.
is_noun	Feature indicating the part of speech for that token. Similarly there are features like is_verb, is_adjective etc. for all the important parts of speech.
is_full_problem	Feature indicating if the token is part of a problem based on its semantic type. Similar feature for treatments and tests were included.
is_abb_problem	Feature indicating if the token was an abbreviation of a problem. Similar feature for treatments and tests were included.
prev_previous_token_name	Previous_token_name text is the word that is previous to the token for which we are generating the features. If there was no word previous to it the feature assigned would be prev_Empty. Similarly another feature was prev_prev_previous_token_name which included the word which was a before the previous word.
next_next_token	next_token had the text of the word next to the token for which we are generating the features. Similarly another feature to indicate the word next to the next_token is also included.
is_umls_feature_treatment	Feature to indicate if the token is a part of the UMLS terminology for treatment or it is identified as a concept as it is part of a phrase including the treatment. Similar feature for problems and tests were included.
is_last_5_char_suffix	Suffix is the last 5 characters of the token which is included as a feature. As an example, a word diskectomy would have a feature ”is_last_5_char_ctomy” set to *true*.

The implementation of Mallet CRF used, *SimpleTagger* (provided as part of the Mallet distribution), implements the linear chain Conditional Random Field (CRF) configuration to label each token as a specific type of concept, or as “other”. Specifically, a pair of labels were used to represent each target class, *Begin-** and *End-** (e.g., *Begin-Problem* and *End-Problem, Begin-Treatment* and *End-Treatment*, etc.), and the label *Other* was used to represent tokens not part of any concept. The features were generated in the same way, when the system is being trained as well as being tested except that the ones in the training phase end with the label.

As an example, Table 2 illustrates the features generated for the tokens in a sentence “he has no bleeding or clotting problems .”. The features are designed to capture both the properties of the token itself, as well as the context (e.g., previous and next tokens in the sentence).

Table 2:Example features for the token sequence for the sentence *“He has no bleeding or clotting problems.”*

Token	*Features*
he	is_review_of_systems is_capitalized is_abb_treatment prev_empty prev_prev_empty next_has next_next_no Other
has	is_review_of_systems prev_he prev_prev_empty next_no

	next_next_bleeding Other
no	is_review_of_systems prev_has prev_prev_he next_bleeding next_next_or Other
bleeding	is_review_of_systems_ is_noun is_full_problem prev_no prev_prev_has next_or next_next_clotting problem
or	is_review_of_systems prev_bleeding prev_prev_no next_clotting next_next_problems Other
clotting	is_review_of_systems is_noun is_full_problem prev_or prev_prev_bleeding next_problems next_next_. Beginproblem
problems	is_review_of_systems is_noun is_full_problem prev_clotting prev_prev_or next_. next_next_empty continueproblem
.	is_review_of_systems is_number prev_problems prev_prev_clotting next_empty next_next_empty Other

Experimental Results

The system's performance was evaluated using a scoring program provided by the Challenge organizers, which calculates the 'Precision' and 'Recall' of the system where 'Precision' is the fraction of retrieved information that is relevant to the search and 'Recall' is fraction of the relevant information that is successfully identified.

The experiments were conducted by splitting the provided training dataset on a 3:1 ratio to choose the best system. The system was trained using 250 documents and tested against 100 documents.

***CTAKES+Rules*:** The extended CTAKES pipeline (using dictionary-match components and applying dependency rules), but without the final machine learning component, served as a baseline to our system.

CTAKES+Rules+basic-features: To improve upon the baseline results we combined the output from the CTAKES pipeline with CRF. The features used in this version were simple token features such as: is_capitalized, is_full_problem, is_full_treatment, is_full_test, is_abb_treatment, is_abb_problem, is_abb_test, is_heading, and features with the previous two and next two words relative to the concerned word.

CTAKES+Rules+extended-features: The next variant in the attempt of improving the results by including additional features like is_umls_feature_test, is_umls_feature_problem, is_umls_feature_treatment, and features including the heading name, parts of speech and last 5 characters of the token.

CTAKES+basic features: Another test was performed with no rules being applied on the Stanford dependency parser's output and directly feeding the output from the dictionary lookup into CRF.

Table 3 reports the Precision and Recall of all variants of our system on the validation (hold-out) part of the training data. These results show that the system with the basic features as input to Mallet exhibited the best performance. Note the dramatic improvement of both precision and recall over the baseline CTAKES system, when the machine learning component is added as the final step to integrate the linguistic, contextual, and lexical evidence.

Table 3: Performance of different variants on the split data

System Version	*Precision*	*Recall*
CTAKES+Rules (baseline)	0.44	0.36
CTAKES+Rules+basic-features	**0.85**	**0.71**
CTAKES+Rules+extended-features	**0.79**	**0.73**
CTAKES+basic features	0.8	0.74

Based on our training experiments (Table 3), we chose to submit to the challenge the system variant ***CTAKES+Rules+extended-features***.

The results of testing our system on the official test data are reported in Table 4. These results indicate that the performance improved dramatically from the baseline after augmenting the NLP pipeline with machine learning. As the variants on real test data show, there is not a large advantage of adding the additional features to CRF for this task and the system performed equally as with basic features. In addition, the results on the real test data show that CRF learned the rules to include phrases based on the training data as the results did not improve very much on application of the rules explicitly on this data.

Table 4: Performance of different variants of our system on the official test data

System Version	*Precision*	*Recall*
CTAKES+Rules (baseline)	0.45	0.34
CTAKES+Rules+basic-features	0.84	0.77
CTAKES+Rules+extended-features	**0.83**	**0.78**
CTAKES+basic Features	**0.84**	**0.77**

Discussion

Our approach of combining the CTAKES pipeline with the CRF classifier resulted in the best performance overall. The specific variant with the best performance on the official test data (with best recall of about 78% and best precision of about 83%), proved to be *CTAKES+Rules+extended-features,* with the variant ***CTAKES+basicFeatures*** exhibiting similar performance.

The basic characteristics of the words (tokens) along with the features describing the context (neighboring words) appear to be sufficient to an extent for the task of extracting medical concepts from the given text. The addition of more sophisticated *(extended)* features did not alter the performance significantly.

The analysis of the results mainly suggests improvement in two areas. One is the problem of associating a complete phrase with an identified concept. Many cases were observed where a phrase was partially extracted leaving us a problem of finding better ways to extract complete phrases. Secondly, the task of assigning ambiguous concepts to the right class after it has been identified as a medical concept could be further improved with additional features representing the provenance (the sequence of rules fired) which could allow the classifier to "correct" the errors of earlier pipeline stages.

Conclusions

Although there is room for improvement on the experiments that we conducted, our results show that natural language processing techniques augmented with supervised machine learning is a promising approach for extracting medical information from clinical texts where feature selection is an important aspect. Training the system on a larger dataset and trying features which are more specific to identifying whole phrases could be few of the experiments that could be conducted to improve on the system before using it in some real tasks.

References

1. Savova GK, Kipper-Schuler K, Buntrock JD, Chute CG. UIMA-based clinical information extraction system. LREC 2008: Towards enhanced interoperability for large HLT systems; 2008.
2. D. Ferrucci, A. Lally, UIMA: an architectural approach to unstructured information processing in the corporate research environment, J. Nat. Language Eng. (2004), in press.
3. Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D267-70.
4. McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu/
5. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the International Conference on Machine Learning (ICML-2001), Williams, MA, 2001
6. Stanford NLP Group Software (2005) http://nlp.stanford.edu/software/lex-parser.shtml
7. Liu S, Wei M, Moore R, Ganesan VAGV, Nelson SANS. RxNorm: prescription for electronic drug information exchange. IT Professional. 2005;7(5):17?23.
8. Berman JJ. Pathology Abbreviated: A Long Review of Short Terms. Archives of Pathology and Laboratory Medicine, 128:347-352, 2004.
9. OpenNLP: http://opennlp.sourceforge.net/