Extracting Patient Identification and Unified Medical Language System (UMLS) Information from UCSF Clinical Notes

Madison J Myers[1] Madison.Myers@ucsf.edu

Abstract— **The evolving study of natural language processing has resulted in a number of tools that are applicable to clinical use cases. By applying and evaluating these tools that were specific to UMLS terminology, an ideal pipeline was achieved for the use of UCSF's clinical nurse notes. In conjunction with these tools, a python script was used to successfully output each note's respective patient ID and UMLS information. This script leaves room for improvement, particularly in the area of negation detection and accuracy in extracting UMLS terms, considering variations in spelling, abbreviations and targeted terms regarding UCSF notes specifically.**

I. INTRODUCTION

Identifying clinical actions and diagnoses in clinical text and unstructured documents has the potential to be a turning point in the field of bioinformatics. The magnitude of impact in using these extracts is huge and ranges from providing a quick summary of patient history to the prediction of future patient issues. This project considered those implications and aimed to build a foundation that could be built upon by future scholars. The goal of this base-project aimed to input an unstructured text document, or in this case a clinical note, and use natural language processing to dissect and extract valuable information that can then be used for further use cases. This project targeted the output of UCSF patient identification numbers and domain-specific information, specifically, terminology from the Unified Medical Language System (UMLS). Clinical notes were chosen not only because they offer invaluable information, but also because they are difficult to work with for large research studies and they often require manual review. Automatic term extraction will bypass this task and not only aid doctors, but also allow researchers to gain information about large sets of clinical notes in a short amount of time.

In this five week-long project, this problem was approached and the details are as follows.

A. UMLS

”The Unified Medical Language System (UMLS) Metathesaurus is a large, multi-purpose, and multi-lingual thesaurus that contains millions of biomedical and health related concepts, their synonymous names, and their relationships. The Metathesaurus is one of the three UMLS components: the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. The National Library of Medicine (NLM) updates the UMLS twice a year in May and November. The Metathesaurus includes over 150 electronic versions of classifications, code sets, thesauri, and lists of controlled terms in the biomedical domain. These are the source vocabularies of the Metathesaurus. Their uses incorporate: patient care, health services billing, public health statistics, indexing and cataloging of biomedical literature, basic, clinical, and health services research” [19]. This Metathesaurus was chosen due to its relevance and due to it being the largest publicly available Metathesaurus of clinical terminology. The UMLS Metathesaurus was downloaded locally and used as a dictionary to extract terms within each clinical note's text. Extracting the terms via the UMLS Metathesaurus is not only becoming relatively standard for similar use cases, but it is also regularly updated, leading to a current place in which terms can be drawn from. Downloading the dictionary locally, rather than using an API also allows the work to remain on the local server and not be at risk for breaking IRB protocol.

B. NLP in a Clinical Context

Natural Language Processing is a budding field which has been applied to diverse areas of study including voice recognition and search and is now being applied to a clinical context. The bulk of natural language processing includes parsing the text into sentences and then words and then tagging parts of speech and categorizing them respectively. It also can include removing ”stop” words, which are characterized by words that are often conjugations. These words are dropped to make topic extraction easier. Natural Language Processing then often divides into multiple approaches including machine learning and text mining. Machine learning approaches look at improving accuracy for identifying patterns in speech as well as topics within the text. Text mining problems include extracting terms and particular attributes from the text. In this use case, both are relevant, though only text mining was applied due to time restrictions.

II. DATA DESCRIPTION

A. Background on Clinical Notes

Clinical notes are the unstructured text written by nurses, MDs and other medical professionals. These notes range from a single sentence describing a patient anecdotally to several paragraphs describing a patient by clinical and medical terms. This may include physical injury sustained before entering a hospital or a long-term investigation into diagnosing a critical illness. This text often includes sensitive

*This work was supported by UCSF's Institute for Computational Health Sciences

[1]As of summer 2017, Madison J Myers is a data science graduate student at UC Berkeley and a bioinformatics programmer at the Institute for Computational Health Sciences, Mission Bay Campus, UCSF

patient information, as well as medical terms, descriptions and values given for certain measures which may or may not include heart rate, blood pressure, medicine and more. These notes are not in any particular format and vary depending on the professional and the time it was written.

B. Notes Used

With IRB approval, access was given to 360 notes on the UCSF Machine Learning server. These notes included the patient ID and unstructured text data that mostly abided by English grammatical rules. Within the text were varied sentence lengths that contained information about the associated patient. This information ranged from context of the patient's medical background, the patient's interaction while in the facility and what the patient was and sometimes was not diagnosed with. These notes also ranged in length from a few words to several pages, with numbers and measures occurring throughout the text. Each note was also diverse in its referral to particular names, but each note was consistently saved by patient ID so that the filename was patientid.txt.

III. APPROACH

The goal of this project was to have a functioning script that parsed through UCSF clinical notes and output the respective patient identification number and UMLS terms and relationships. In the initial findings, there seemed to be a great deal of approaches for similar UMLS problems that had been tried in prior research, including researchers from the University of Wisconsin-Milwaukee who used conditional random fields, MetaMap and Stanford Core NLP jointly [12] and a team from Vienna University of Technology who focused solely on negation detection using NegFinder [11] and many more, but none that specifically addressed our particular problem.

After a great deal of research, as well as trial and error, it was discovered than an array of tools and approaches to natural language processing and UMLS term extraction in a clinical setting were found– overwhelmingly so and which were at various degrees of development. Upon application, many of them were unfortunately found to be lacking in maturity, testing, documentation and a community. Given their nascency, many did not provide adequate resources for troubleshooting and proper application. These various tools also seemed to differ in terms of targeted pipeline, ranging in programing languages and environment requirements. Nevertheless, a decision was made to evaluate which would work best in this particular use case, and build off of it, in an effort to avoid reinventing the wheel. In that decision, the first phase of the project became one focused on testing, trying and eliminating tools rather than developing a detailed script that focused on approaching the problem from scratch. These various tools are listed below in the ”Existing Tools” section.

Once the natural language processing and UMLS classification tools were in place as the first two steps in the pipeline, the project then shifted to focus on the accuracy of the output. Numerous means to improve accuracy were identified throughout the research process, but due to time restrictions, negation detection was chosen as the path forward. This decision was due to negation detection's potential to seemingly improve accuracy by a higher percent than the other identified problem areas. The remaining roadblocks are discussed below in the ”Roadblocks” section and are identified for future improvement in the section ”Conclusions and Future Work”. The negation detection methods that were chosen are discussed in the section ”Negation Detection” below.

A. Existing Tools

Though the list is not exhaustive, the tools that were tried for the UMLS portion of the problem included the NLTK python 3 library [25], the UMLS Metathesaurus download [19], MetamorphoSys [20], QuickUMLS [7], SNOMED [21], KNIME [22], MetaMap, MetaMapLite [23], the UMLS API, CliNER [3], PyMedTermino [4], Apache cTAKES [24], Apache OpenNLP [26], py-UMLS [27] and the Health Vocabulary Rest API [28]. Other tools that were tried were mostly focused on negation detection and included Negation Detection [13], NegEx [18], NegFinder [11], DepND [17], and pyConTextNLP [16], DEEPEN [14], and Negation Resolution [15].

The tools that were used in the pipeline for this project are as follows:

- Python 3 NLTK Library
- UMLS Metathesaurus download, installed by MetamorphoSys
- QuickUMLS downloaded and installed via GitHub

The reasoning behind why the alternate tools were not chosen for this particular project are as follows:

Though SNOMED, an extension of the UMLS Metathesaurus, would be helpful, it was not connected with QuickUMLS, the primary tool used in the extraction of UMLS terms.

KNIME is a helpful tool, with great user interface, that maps out different steps in a pipeline and also conducts the steps for you with the selection of a button. It was not chosen because it was particular in what text was accepted for the pipeline and often resulted in error depending on the format of the files. Because of this, it was difficult to use clinical notes and also connect the results back to a python 3 script.

MetaMap is a tool that is similar to QuickUMLS, but is found on the UMLS website. It ”uses a shallow parser to generate candidate phrases; then, for each candidate phrase, many lexical variations are generated; finally, each phrase is scored based on their distance to concepts in UMLS.” [7] When tried, MetaMap functioned well and connects directly to the Metathesaurus and SNOMED. It was not used for this pipeline as QuickUMLS was faster and also seemed to have higher accuracy in identifying UMLS terms than MetaMap when tested. MetaMap Lite is similar to MetaMap, but is a smaller download and takes up less space. Unfortunately, it did not perform well and was difficult to install. There is also less documentation available for the Lite version, making troubleshooting tiresome. Because MetaMap has some built-in negation detection, I do however recommend MetaMap over QuickUMLS if time is less of an issue.

Like MetaMap and MetaMap Lite, the UMLS API is also available on the official UMLS website. The API works well in a python script with a python wrapper, but only supports a search function. Ultimately, the decision was made to not use the API as the desired pipeline was to parse the text for UMLS terms and conditions rather than the limited API offering to search for one particular term at a time. Because the pipeline also needed to function on a local, secure server, an API was decidedly not the best choice.

Cliner supports Named Entity Recognition (NER) in the clinical domain which ”aims to identify clinically relevant concepts in the provider narrative text of electronic medical records (EMR). Such concepts as diseases/disorders, treatments/medications, and tests have been the focus of clinical NER community challenges” and therefore Cliner focuses on those areas of recognition. Cliner uses concept boundary detection to identify general text features, Genia features and UMLS features. It also uses concept type identification to identify expressions for dates, test results, doctor abbreviations etc. Though Cliner states that ”despite recent advances in clinical NER state-of-the art, no lightweight, easy to set up, open-source implementation is available to date,” Cliner was difficult to set up and little documentation exists to help navigate its use. Due to these reasons, it was not used. I do recommend further exploration, however, as the tool seems quite promising.

PyMedTermino is a generic, open-source, python API ”for a multi-terminology multilingual terminology service”, which uses ICD10, SNOMED CT, MedDRA, CDF, VCM iconic language and the UMLS compendium. [4] While theoretically brilliant in its use of python, the implementation seemed oddly similar to the UMLS API and requires the connection to a SQL server with the UMLS Metathesaurus downloaded. Unfortunately, neither an API nor a SQL server were optimal for this use case and PyMedTermino was not used, though once developed further I believe this will be a useful tool.

The widely acclaimed Apache cTAKES ”matches candidate phrases (as well as their permutations and lexical variations) with concepts in UMLS and a list of concepts maintained by Mayo Clinic.” [7] In this particular trial, Apache cTAKES was not user friendly and did not connect well to a pipeline that focused on having a single python script.

Py-UMLS was not used because it interacts with a SQL-lite database. This seemed unnecessary given the preceding UMLS metathesaurus download and the desire to have a simple, local and efficient pipeline.

Like other tools, the Health Vocabulary Rest API was not used in an effort to keep the script local.

Negation Detection detects negated sentences from xml annotated corpora in TIGER-XML format and annotates them with Negation frames, Scope, Focus and Target tags. [13] Though this seemed like a great tool, it was not relevant to this project’s format.

The NegEx Algorithm is the most cited negation detection algorithm I came across. Indexed by physicians, NegEx had a specificity of 94.5 percent...[and] a positive predictive value of 84.5 percent. [5] It has been adapted into many forms, including python, however, I failed to find documentation to help its implementation and ”due to the failure to consider the contextual relationship between words within a sentence, NegEx fails to correctly capture the negation status of concepts in complex sentences. Incorrect negation assignment could cause inaccurate diagnosis of patients condition or contaminated study cohorts.” [14] Nevertheless, if one had more time, I recommend putting the time in to apply this algorithm to the pipeline.

NegFinder was not publicly available. There were many other softwares similar to NegFinder that were discovered during research, which were written about in papers, but failed to be publicly available or required the purchasing of the software or a subscription.

DepND requires python 2.7 and this project was conducted in python 3. [17]

Negation Resolution is in Python 2. Because it contained smaller scripts, the time was taken to change it to Python 3, but it then could not connect properly to the wrapper it is dependent upon to function.

PyConTextNLP was difficult to install and ultimately ended up being a deciding factor in its use. [16]

DEEPEN ”utiliz[es] Stanford dependency relation to further analyze the negation status of clinical concepts negated by NegEx” and is sold as an ”improvement of [the] NegEx algorithm by decreasing the number of false positives.” [14] Unfortunately the Stanford CORE NLP did not install properly and malfunctioned in this use. DEEPEN itself additionally did not seem to be publicly available.

Negation Resolution is also dependent on Stanford’s CORENLP. While promising in its dive into studying negation for suicide prevention, the tool was not useful for this particular use case. [15]

B. Pipeline

Figure 1 shows the pipeline for this project. Starting on the UCSF Machine Learning server, the UMLS Metathesaurus was downloaded and then installed using the MetamorphoSys tool, which can be found on the same UMLS website as the Metathesaurus. Then, the clinical notes, which were already uploaded to the Machine Learning server, were fed into the python script, via a search for text files in the notes’ folder. Then, using QuickUMLS and negation detection, the script output each note’s respective patient ID and positive UMLS terms, cui, and relationships.

C. Natural Language Preprocessing

Before QuickUMLS or negation detection were used, preprocessing using natural language processing methods was conducted. Generally, many tools and pipelines similar to this project require the text to be separated into chunks, stop words removed, tagged with their respective parts of

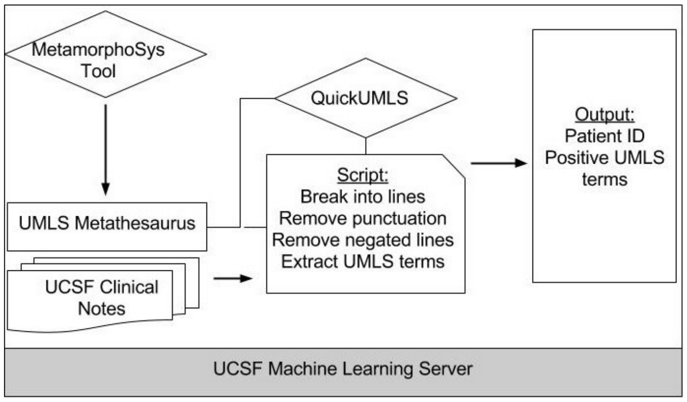

Fig. 1. Project Pipeline

speech and then fed into whatever algorithm they use. For these requirements I used the NLTK library. For the final pipeline using QuickUMLS, however, the required input was a sentence. Due to this simplicity, I added a break between punctuation points, splitting the text into sentences and feeding QuickUMLS one line at a time.

D. UMLS Extraction

The next step in the pipeline was UMLS extraction. UMLS, or the Unified Medical Language System, is a metathesaurus of clinical terminologies used by professionals in the field. This metathesaurus was downloaded and installed using the MetamorphoSys tool provided on the UMLS website. ”MetamorphoSys is the UMLS installation wizard and Metathesaurus customization tool included in each UMLS release.” [20]

”In recent years, tools such as MetaMap and cTAKES have been widely used for medical concept extraction on medical literature and clinical notes; however, relatively little interest has been placed on their scalability to large datasets.” [7] This, combined with its relatively higher accuracy is why I used QuickUMLS: ”a fast, unsupervised, approximate dictionary matching algorithm for medical concept extraction... [that] outperforms MetaMap and cTAKES on a dataset of consumer drug reviews... [and] is up to 135 times faster than both systems.” [7] To a certain extent QuickUMLS also captures variations in clinical terms. Please see the table below for research results:

Method	Prec	Rec	F-1	ms/doc
MetaMap	0.49*	0.48*	0.48*	19,295*
cTAKES	0.71	0.53*	0.61	3,852*
QuickUMLS	-	-	-	-
= 0.6	0.50*	0.75	0.60	1,594*
= 0.7	0.60*	0.66*	0.63	680*
= 0.8	0.63*	0.60*	0.61	332*
= 0.9	0.64*	0.56*	0.60	193*
= 1.0	0.67*	0.54*	0.60	143

Table 1: Results for the i2b2 dataset. cTAKES outperforms QuickUMLS in precision, but QuickUMLS has better recall. QuickUMLS is 2.5 to 135 times faster than MetaMap or cTAKES. * indicates statistically significant differences from best value (Welchs t-test, p ¡ 0.05). [7]

QuickUMLS was installed locally with the same MetamorphoSys tool used to install the metathesaurus. It was used with three lines of code in the python 3 script and proved successful at extracting terms from the clinical notes used.

E. Negation Detection

The following step in the pipeline was negation detection. Negation detection is the identification of negative expressions in language. Examples of this include ’he did not break his arm’; ’After testing I do believe that he is without health’; ’there is no evidence for cancer’. These negative words and phrases are commonly used in language to communicate important judgments and decisions, but the practice is difficult to capture and translate to computer language, especially when the text is clinical in nature. ”Great efforts have been made to transcribe this information into a format a computer can work with. Despite this the processing of negated terms is still an open topic. Negation detection demands a lot of knowledge about language itself to correctly identify negated words or terms in free text. Endeavor on this topic may help to develop computer assisted treatment strategies to improve patient care.” [11] Negation detection is also one of the most promising ways to increase accuracy, but as one can gather from the examples, negation is not always straightforward and the practice of detecting it is difficult and overlaps with sentiment analysis, natural language processing and text mining.

Because negation is heavily used in clinical texts to narrow down potential health issues, negation detection is needed for this project. Ideally the script or tool filters out UMLS terms based on the relevance of the term and its likelihood to be positive (he does have a disease vs he does not have a disease). Unfortunately this problem is rather difficult to navigate considering that there are multiple terms per sentence (and the script is feeding QuickUMLS line by line) and that negation can often be unclear (positive words can be used when negating and negative words can be used in a positive scenario). Due to this challenge, many tools and algorithms were explored. Many are available, but few that fit seamlessly into the pipeline. After weighing the varying approaches and considering the timeline, a short script was developed to help identify key negated words.

To explain this reasoning it is worth mentioning other research in the field. In a previous study, it was found that just seven phrases accounted for ninety percent of negations in clinical text. [5] These include ”no”, ”denies”, ”without”, ”not”, ”no evidence”, ”with no”, and ”negative for”. Therefore, including a few very common negation phrases can capture a large portion of the pertinent negatives. [5] Considering that negation detection could be a project of its own due to syntactical complexity and the difficulty in attaining high accuracy, a simple list of the most common negated words were used as a baseline. Other scholars also analyzed negation detection in the context of extracting whether a patient was deceased or not and its relative cause from free text medical records. [10] This is called the Freetext Matching Algorithm (FMA) which aims ”to extract diagnoses, dates, durations, laboratory results and selected examination findings (heart rate and blood pressure) from unstructured free text". After considering this approach, it was further realized how complex negation detection is and a simple script appropriate for the given time frame was developed. Other scholarly approaches have included using cTakes and UMLS with a bag of words to bag of concepts model [6] using MetaMap, which has some negation detection built in, and also the use of the NegEx algorithm.

Conclusively, for the negation detection step of the pipeline, each note was processed line by line, with punctuation removed, and sorted based on whether the top negated words were present or not. In this limited short script, the lines containing negative phrases are simply removed and the lines with UMLS terms that are not negative, are kept. Issues arise when there are multiple UMLS terms in one sentence and are contrasting in sentiment. Comments on how to take this script further and increase accuracy can be found in the "Conclusions and Future Work" section.

TABLE I
COMMON NEGATED PHRASES IN CLINICAL TEXT

Phrase	Percent of Negated Terms
no	51
denies	15
without	8
not	6
no evidence	5
with no	2
negative for	2

IV. ROADBLOCKS

Throughout this project, roadblocks were prevalent. Errors with tools were frequent and frustrating and because most of these tools are in their nascent stages or were created and then not worked on further, there was a great deal of under-documented steps and error-filled installations that were sometimes out of date. Roadblocks, however, were not limited to the tools used. They also included different versions of python, UMLS extraction accuracy, varying abbreviations and spellings, IRB approval, and space needed on a machine. Specifically, many tools available in python were in version python 2.7 or earlier, greatly limiting current work using python 3. UMLS extraction accuracy also greatly varied depending on the tool used and how many UMLS terms were in a given sentence or chunk. Abbreviations also varied depending on the author of the clinical note. Due to time restrictions, an extended dictionary of likely abbreviations and approximate alternate spellings were not included. Machine Learning could also be used as a solution. This problem, along with the negation detection issues before-mentioned could lead to a curse of dimensionality which should be noted [6]. IRB approval was additionally needed to work with personal clinical notes. This ended up creating a time problem and took up several weeks of the given twelve-weeks' time for the project. Lastly, these tools and the metathesaurus in particular take up a great deal of space. I recommend testing one tool at a time and then removing it and uninstalling it once it has been ruled out so that space is maintained on the machine or server.

V. CONCLUSIONS AND FUTURE WORK

For this iteration of the solution, I successfully navigated through the problem I set out to solve: inputing UCSF clinical notes and outputting respective patient IDs, as well as UMLS terms, cui's and relationships. In order to attain this output I tried out a great deal of tools and eliminated the majority based on how they fit in with the pipeline as well as their performance with the clinical notes. After the evaluation of over twenty tools, I moved forward with QuickUMLS which drew from the UMLS metathesaurus. Due to time restraints and lack of an effective operating negation detection tool for the time frame, I added lines to the python script that would remove any line that included the most common negated words, reportedly making up ninety percent of negated terms in clinical text.

Originally this project was assigned twelve weeks, but after four weeks of working on a completely different project, focused on a separate natural language processing problem, two weeks waiting for IRB approval, and one week given to an emergency situation, this project was left with five-weeks' time to evaluate the cited tools and construct a script outputting the before-mentioned information. Due to this time restraint, much was left unexplored.

I believe there is great value in the potential of this project and therefore recommend this project be explored further in an effort to increase the accuracy of the output. Ways to do this include refining the output to include only terms or summaries of the notes; add additional dictionaries beyond the metathesaurus; further negation detection by implementing machine learning, tailoring it for UCSF notes in particular, where pre- and post- UMLS negated terms are considered, as well as multiple terms per line; and then map these outputs to SPOKE, the hetionet web that hosts mapped diseases, symptoms and more on Neo4j. Implications for this project and its future iterations are huge. Doctors could operate faster with the summary of notes being available for each patient, adding to their medical history and prediction could allow flags of potential diseases and illnesses. With these flags, preventative medicine could prevail over reactive medicine, creating a much more effective means of care.

ACKNOWLEDGMENT

Acknowledgements to my mentors and supervisors at UCSF. Charlotte Nelson of Butte and Baranzini Lab, Sharat Israni at the Institute for Computational Health Sciences and Sergio Baranzini at the Baranzini Lab, thank you for your guidance and for the opportunity to explore this problem.

REFERENCES

[1] Wendy W. Chapman PhD, Will Bridewell BS, Paul Hanbury BS, Gregory F. Cooper MD PhD, and Bruce G. Buchanan PhD, Evaluation of Negation Phrases in Narrative Clinical Reports, Center for Biomedical Informatics, Computer Science/Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA.

[2] William Long, PhD, Extracting Diagnoses from Discharge Summaries. CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA.

[3] William Boag, Kevin Wacome, Tristan Naumann, Anna Rumshisky, CliNER: A Lightweight Tool for Clinical Named Entity Recognition, Text Machine Lab for Natural Language Processing, UMass Lowell Clinical Decision-Making Group, MIT CSAIL.

[4] Jean-Baptiste LAMY, Alain VENOT, PyMedTermino: an open-source generic API for advanced terminology services, LIMICS, Universite Paris 13, Sorbonne Paris Cite, Universite Paris 6, INSERM UMR S 1142, 74 rue Marcel Cachin, 93017 Bobigny, France, European Federation for Medical Informatics (EFMI), 2015.

[5] Chapman WW, Bridewell W, Hanbury P, Cooper GF, Buchanan BG, A simple algorithm for identifying negated findings and diseases in discharge summaries., Center for Biomedical Informatics and Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15213, published online May 9, 2002.

[6] Ryan Cobb, Sahil Puri, Daisy Wang,Tezcan Baslanti, Azra Bihorac, Knowledge Extraction and Outcome Prediction using Medical Notes, Department of Computer and Information Science, Engineering and Department of Anesthesiology, University of Florida, Gainesville, FL.

[7] Luca Soldaini, Nazli Goharian, QuickUMLS: a fast, unsupervised approach for medical concept extraction, Information Retrieval Lab, Georgetown University.

[8] Martine Enger, Erik Velldal, Lilja vrelid, An open-source tool for negation detection: a maximum-margin approach. University of Oslo, Department of Informatics, Proceedings of the Workshop Computational Semantics Beyond Events and Roles (SemBEaR), pages 6469, Valencia, Spain, April 4, 2017.

[9] Stephane Meystre, Peter J.Haug, Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation. Journal of Biomedical Informatics Volume 39, Issue 6, December 2006, Pages 589-599. 2006.

[10] Anoop D Shah, Carlos Martinez and Harry Hemingway, The free-text matching algorithm: a computer program to extract diagnoses and causes of death from unstructured text in electronic health records. BMC Medical Informatics and Decision Making https://doi.org/10.1186/1472-6947-12-88 Shah et al.; licensee BioMed Central Ltd. 2012.

[11] Stefan Gindl, Negation Detection in Automated Medical Applications. Vienna University of Technology, Institute of Software Technology and Interactive Systems, October 2006.

[12] Omid Ghiasvand, Under the Supervision of Rohit J. Kate, PhD, DISEASE NAME EXTRACTION FROM CLINICAL TEXT USING CONDITIONAL RANDOM FIELDS. The University of Wisconsin-Milwaukee, 2014.

[13] Darmin Spahic, Robert Sass, https://github.com/darminspahic/negation-detection, MIT License, 2016-2017.

[14] SaeedMehrabia, AnandKrishnana, SunghwanSohnd, Alexandra M.Rochb, HeidiSchmidtb, JoeKestersonc, ChrisBeesleyc, PaulDexterc, C.Max Schmidtb, HongfangLiud, MathewPalakal, DEEPEN: A negation detection system for clinical text incorporating dependency relation into NegEx. Journal of Biomedical Informatics Volume 54, Pages 213-219, April 2015.

[15] George Gkotsis, Sumithra Velupillai, Anika Oellrich, Harry Dean, Maria Liakata and Rina Dutta, Dont Let Notes Be Misunderstood: A Negation Detection Method for Assessing Risk of Suicide in Mental Health Records, IoPPN, Kings College London, School of Computer Science and Communication, KTH, Stockholm, Department of Computer Science, University of Warwick.

[16] Brian Chapman, Wendy W. Chapman, Glenn Dayton, Danielle Mowery, pyConTextNLP. https://github.com/chapmanbe/pyConTextNLP. University of Utah. 2017.

[17] DepND, https://github.com/zachguo/DepND, Apache License, 2007.

[18] Chapman, Wendy W.; Bridewell, Will; Hanbury, Paul; Cooper, Gregory F.; Buchanan, Bruce G, A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. University of Pittsburgh, May 9th, 2002.

[19] https://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html.

[20] UMLS Reference Manual, https://www.ncbi.nlm.nih.gov/books/NBK9683/.

[21] https://www.nlm.nih.gov/healthit/snomedct/.

[22] KNIME, https://www.knime.org/.

[23] MetaMap and MetaMap Lite, https://metamap.nlm.nih.gov/, 2016.

[24] Apache cTAKES, http://ctakes.apache.org/, The Apache Software Foundation.

[25] Steven Bird, Ewan Klein, and Edward Loper, Natural Language Processing with Python Analyzing Text with the Natural Language Toolkit, Published online, http://www.nltk.org/book/.
[26] Apache OpenNLP, https://opennlp.apache.org/.
[27] UMLS for Python, https://github.com/chb/py-umls.
[28] Health Vocabulary Rest API, https://github.com/chintanop/health-vocabulary-rest-api.