Launching into clinical space with medspaCy: a new clinical text processing toolkit in Python

Hannah Eyre, MS[1,2], Alec B Chapman, MS[1,2], Kelly S Peterson, MS[2,3], Jianlin Shi, MD, PhD[2], Patrick R Alba, MS[1,2], Makoto M Jones, MD[1,2], Tamára L Box, PhD[3], Scott L DuVall, PhD[1,2], Olga V Patterson, PhD[1,2]
[1]VA Salt Lake City Health Care System; [2]University of Utah, Salt Lake City, UT, USA; [3]Veterans Health Administration Office of Analytics and Performance Integration

Abstract *Despite impressive success of machine learning algorithms in clinical natural language processing (cNLP), rule-based approaches still have a prominent role. In this paper, we introduce medspaCy, an extensible, open-source cNLP library based on spaCy framework that allows flexible integration of rule-based and machine learning-based algorithms adapted to clinical text. MedspaCy includes a variety of components that meet common cNLP needs such as context analysis and mapping to standard terminologies. By utilizing spaCy's clear and easy-to-use conventions, medspaCy enables development of custom pipelines that integrate easily with other spaCy-based modules. Our toolkit includes several core components and facilitates rapid development of pipelines for clinical text.*

Introduction

Retrospective clinical research often relies on data extracted from electronic medical record (EMR) systems using natural language processing (NLP) adapted for clinical language. Despite a wide range of existing solutions, code reuse is challenging because new system development projects face the labor-intensive task of connecting isolated modules into cohesive cNLP pipelines. Over the years, the need for reuse and reproducibility has been met with a number of frameworks and toolkits.[1] Java-based general architectures, such as UIMA[2] and GATE[3], have provided a strong foundation for a number of highly successful cNLP toolkits and general purpose reusable systems such as cTAKES, CLAMP, Leo, and HITEx[4–7]. As a platform-independent and straightforward programming language, Java has been the primary environment for most cNLP applications of the last 20 years even prompting re-implementation of tools previously written in other programming languages to improve accessibility[8,9].

While Python programming language is celebrating its 30-year anniversary, in the last few years its usage has exploded as it had become one of the most popular languages of data science.[10–14] The wider NLP, machine learning, and data science community's shift towards Python is fueled by its capability of interactive environments to explore data and experiment with approaches while avoiding a common cycle of compiling code and reloading data[15]. The machine learning ecosystem in Python is vibrant and growing, including state-of-the-art training methods and deep learning models.

Until recently Natural Language Toolkit (NLTK) has been the leading platform for general text processing in Python.[16] Actively supported by a dedicated team since 2000, NLTK is comprised of a number highly functional NLP libraries for rule-based and machine learning-based text analysis. While NLTK has enabled hundreds of successful research and educational projects, its approach to text processing as lists of strings and lack of a unifying architecture have hindered its ability to scale to large datasets.

In contrast, spaCy* provides a robust architecture for building and sharing custom, high-performance NLP pipelines by taking an object-oriented view of text. It is non-destructive, supports seamless integration of statistical and machine-learning methods with rule-based NLP, and allows for the creation of custom components for specialized tasks. Powered by the strength of Cython, an optimizing static compiler for Python that generates very efficient C or C++ code, spaCy allows achieving exceptional speed of performance. Similar to UIMA approach in Java, spaCy provides a framework for modular plug-and-play construction of custom NLP pipelines. Since its inception in 2015, spaCy universe has gained a robust, highly-active, and growing community of contributors of open source modules, state-of-the-art models, and end-to-end systems developed with or for the framework.

Recognizing the need to adapt processing to different domains, several models and toolkits have been introduced targeting specialized text, among them scispaCy that handles scientific and biomedical sublanguages[17]. While scispaCy includes custom tokenization rules, biomedical concept identification, and a variety of pre-trained dependency parsing

*https://spacy.io/

and named entity recognition (NER) models, the differences between clinical and biomedical language hamper its ability to achieve optimal performance on clinical narrative. To confront the particular challenges posed by medical text, a team at Virginia Commonwealth University created MedaCy, a medical text mining framework built over spaCy to facilitate the engineering, training and application of machine learning models for medical information extraction[18]. Neither scispaCy nor MedaCy provide adequate support for rule-based system development beyond the functionality natively provided by spaCy. Despite a wide range of machine learning applications, rule-based algorithms still comprise a large proportion of successful academic and commercial cNLP projects[19,20].

To bridge the gap between the existing functionality and the needs of the clinical research community for easy-to-use clinical text processing that combines statistical and symbolic methods, a team of enthusiastic cNLP practitioners at the Veterans Health Administration and University of Utah developed a spaCy-based library of core components targeting medical text called medspaCy[†].

Methods

Framework Overview

MedspaCy is designed to utilize spaCy's application programming interface (API) while only minimally expanding upon it where needed. This minimal expansion of spaCy's API ensures each component is usable standalone or as one part of a larger spaCy pipeline that may incorporate components from other sources. Figure 1 illustrates the interconnection of medspaCy components into an end-to-end system through a common API. Once the toolkit reached acceptable maturity, we made it available through PyPI with the command `pip install medspacy`.

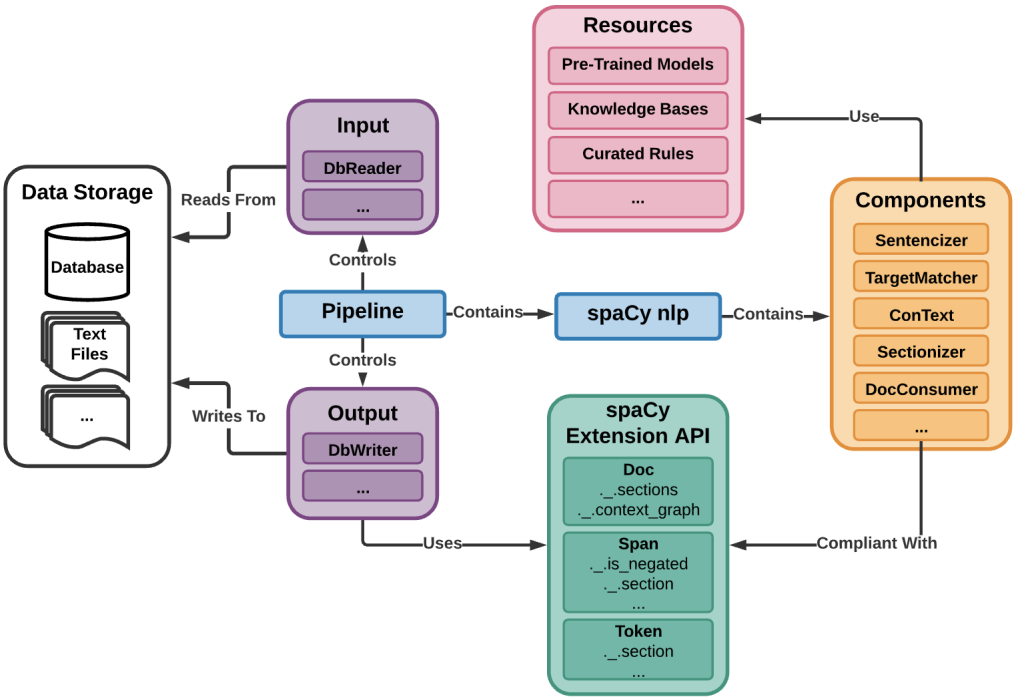

Figure 1: Overview of medspaCy architecture.

All information generated by components is directly accessible through the use of spaCy's attribute extension functionality. For example, a default spaCy attribute of a named entity is accessible using `ent.like_num` attribute, a medspaCy attribute utilizes the custom extensions such as `ent._.is_negated`. Attribute extensions are registered at the `Doc`, `Span`, or `Token` levels, depending on the component to create a variety of useful tools for analyzing data after medspaCy processing is complete.

The components included in medspaCy offer both a default initialization and a large number of optional customization

[†]https://github.com/medspacy/medspacy

parameters. This allows components to be quick to set up, learn, and use to develop prototypes while also being fully customizable for more specific needs. Resources, such as curated rule sets or knowledge bases, may be shared with the community and utilized across projects. Figure 2 is an illustration of an example pipeline using medspaCy. Since the spaCy API allows for modular components, there are many possible pipelines where users could insert alternative components, such as scispaCy AbbreviationDetector or other spaCy-compatible rule-based or machine learning-based modules.

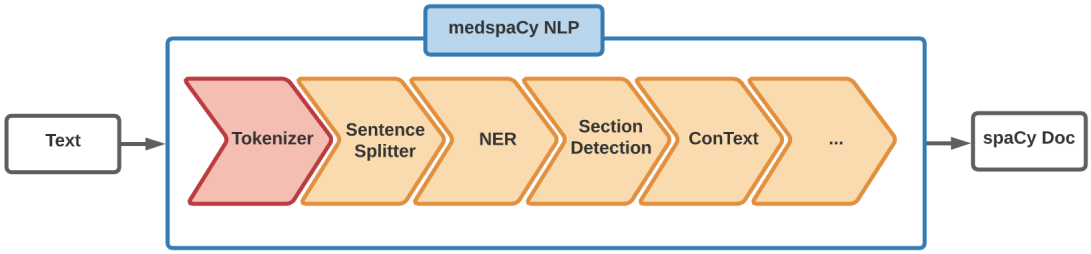

Figure 2: Example of a text processing pipeline that utilizes medspaCy and other spaCy-based components.

MedspaCy Components

MedspaCy has a growing list of integrated components that allow building end-to-end systems for clinical text processing.

Tokenization

The primary advantage of spaCy's approach to tokenization is that it is non-destructive, which preserves all whitespace and punctuation information enabling complete reconstruction of the original text. The major drawback of the default spaCy tokenizer for clinical text processing is that it is not trained on clinical text. Additionally, it has a variety of rules designed to handle text sourced online, including many rules that mitigate excess tokenization of URLs. These rules prevent splitting sequences of alphanumeric characters and punctuation into multiple tokens. However, URLs are relatively uncommon in clinical text but typos and using punctuation to delineate document structures are common. The tokenizer included in medspaCy implemented custom rules to handle punctuation and inconsistent use of whitespace that are common in clinical notes.

Sentence Detection

Due to the limited purpose of clinical language to document and succinctly communicate information about a specific patient, sentences in clinical text are characterized by telegraphic grammar with omitted subjects and the propensity to use long lists for present or absent conditions and prescribed medications[21]. Sentence segmentation in clinical language is hindered by frequent use of templated and tabular text, abundance of concept-value pair statements, and inconsistent use of punctuation. Since the text is meant to be read within a specific user interface of the EMR, space, tab, and new line characters are utilized to indicate sentence or section boundaries as well as for visual indication of grouped text, such as in case of templates and tabular text. Syntactically significant whitespace complicates the traditional approach to text processing that ignores existence and length of whitespace in text. To correctly identify the context of target concepts from clinical text, a clinical-domain sentence boundary detector is essential, because many downstream components work at the sentence level. Commonly used sentence detectors trained on non-clinical text typically do not perform well on clinical text data.[22] To make medspaCy more capable of performing cNLP, we adopted a high-performance, rule-based sentence detector for clinical text – PyRuSH, which is the Python version of Rule-based sentence Segmenter using Hashing (RuSH)[22]. Relying on rules for tokenization allows easy adaptation of the tokenizer to a new setting without extensive retraining.

Section Detection

Clinical documentation has a well established data format for different document types. Logical Observation Identifiers Names and Codes (LOINC) Document Ontology outlines hundreds of document types that vary by subject domains, role of the author, setting, type of service described, and document kind[23]. At each intersection of these axes, a clinical document has a specific structure that is either learned through apprenticeship (medical students learning from practice) or specifically prescribed by guidelines[24,25]. The same statement in different sections of the same or different documents might have very different meanings. For example, a document type called *Progress Note* has Subjective, Objective, Assessment, Plan sections, and *Procedure Report* has Findings and Impressions sections. Depending on which section it is mentioned in, a clinical condition may be experienced by a patient in the past, may be happening at the time when the note is written, or the patient may be at risk for experiencing it in the future. These sections may or may not be explicitly labeled with a variety of section headers fully spelled out or abbreviated.

Breaking documents into relevant sections is often a core part of cNLP, especially as documents grow in length and complexity. medspaCy includes an implementation of clinical section detection based on rule-based matching of the section titles with default rules adapted from SecTag[26] and expanded through practice. The sectionizer contains several other features beyond simple section title identification. If desired, the sectionizer can interact with the attributes of entities, such as changing the temporality attribute of all entities in the "Past Medical History" section. The medspaCy sectionizer is also capable of creating hierarchies of sections and subsections within the document and preserving them in a traversable tree structure.

Concept Extraction

Clinical concept extraction through concept mention detection and concept encoding is one of the most prominent tasks of clinical text processing.[19] Similar to NER, concept extraction employs either manual rules or trained statistical models to identify specific spans of text from a clinical document and labeling them with pre-defined clinical concepts. Several utilities are included in medspaCy for targeted concept extraction which extend existing spaCy rule-based functionality. Rules defining concepts specify the text pattern to be matched, semantic category, and additional metadata about a concept. In addition to integrating existing pattern-matching functionality provided by spaCy, medspaCy incorporates additional regular expression features, such as accepting multi-token regular expressions, and allows directly specifying entity attributes, such as temporality or standard vocabulary codes. Enabling additional regular expression matching ensures greater compatibility with existing cNLP resources and knowledge bases as regular expressions are commonly used in other frameworks and languages.

UMLS Mapping

Clinical text processing systems frequently rely on comprehensive indexing of the narrative text to a standard vocabulary. The Unified Medical Language System (UMLS)[27] is the most comprehensive standard terminology that is typically used as the basis for clinical concept extraction. Aiming to enable UMLS concept extraction with minimal environment configuration, medspaCy adapted QuickUMLS to spaCy framework. The existing system QuickUMLS was selected as a concept mapping solution due to its speed and matching accuracy that are comparable to other existing systems[28]. It allows efficient approximate dictionary matching by leveraging an implementation of the SimString library.[29] Modifications were made to QuickUMLS so that its algorithm could be contained in a spaCy component.

UMLS licensing does not allow redistribution of the databases, therefore, medspaCy is distributed with resources generated from a publicly available UMLS sample.[‡] MedspaCy documentation includes instructions on how to create QuickUMLS resources after installation. An additional contribution of medspaCy is a better support for QuickUMLS and its dependencies for Windows operating system. While our implementation of QuickUMLS in Windows requires Anaconda and some additional installation steps, other operating systems such as Linux and MacOS can perform extractions with the provided small UMLS sample immediately following installation.

[‡]https://www.nlm.nih.gov/research/umls/new_users/online_learning/Meta_006.html

Contextual Analysis

Extracting contextual properties of clinical concepts is an essential step of typical cNLP systems. Negation, temporality, certainty, and experiencer are the most useful contextual features for entities extracted from clinical text. The ConText algorithm[30] asserts attributes by linking entities with linguistic modifiers within a contextual window, typically a sentence. Previously implemented using Python in pyConTextNLP[31], the algorithm was adapted to spaCy API with extended attributes. MedspaCy implements the ConText algorithm with a set of default rules that can be customized as needed. ConText rules take advantage of pattern matching and also allow the user to control the behavior of the modifier by defining properties such as directionality in the text and custom linking logic. Several attributes, such as negation, are registered by default, but attributes can be customized by users to allow for use case-specific semantic categories.

Utilities

Assisting with development of complete end-to-end systems, medspaCy provides a variety of utilities to support deploying pipelines and analyzing output of text processing:

- pre- and post-processing utilities that help developers transform simple methods into components compatible with spaCy pipelines.
- a component that converts a spaCy `Doc` to a dictionary-based format that is directly convertible to Pandas, SQL, CSV, or other tabular format when added to the end of an existing medspaCy pipeline. It supports output for all attributes and spans for entities, sections, and context.
- a pipeline controller for deploying medspaCy on large quantities of data. It facilitates batching documents on both input and output and a wrapper that handles standard database operations through `pyodbc` (Python ODBC bridge), such as creating tables and write queries given a set of desired columns, which allows medspaCy to deploy faster.

Visualization

The visualization tools included in spaCy's have been adapted to display the results of the integrated medspaCy components. There are two types of visualization included (Figure 3): the first highlights extracted entities, contextual modifiers, and section titles in the text. The second represents the ConText algorithm by drawing directed arrows between entities and their linked modifiers. These visualizations can be used to present a model's final output or to explain the logic of a model. During development, visualizing intermediate results can assist with debugging. Updating a model, testing on text, and inspecting the results with the visualizer can be done rapidly when used in an interactive setting such as Jupyter Notebook.

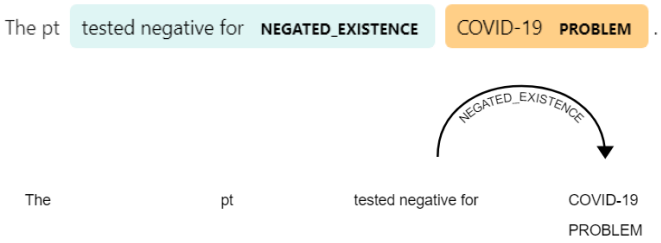

Figure 3: Example of visualization using highlight and context arrows.

Use Cases

Multiple operational and research oriented projects within the U.S. Department of Veterans Affairs (VA) have utilized medspaCy. The rapid development supported by medspaCy has been essential to several projects aiding the VA's COVID-19 pandemic response.

Chief Complaint Surveillance

One of the earliest applications utilizing some of these components in a spaCy pipeline is a system which is used for enhanced syndromic surveillance in VA. This system processes presenting symptoms recorded during emergency department triage such as `"n/v/d"` (i.e. "nausea / vomiting / diarrhea") or `"c/o cp+sob"` (i.e. "complains of chest pain and shortness of breath") to extract UMLS concepts. This pipeline has been running in an operational capacity for biosurveillance since early 2019. It was leveraged in early 2020 in response to the COVID-19 pandemic to identify potential patients under investigation before the existence of laboratory testing for SARS-CoV-2 or the term COVID-19 was adopted. This pipeline performs text pre-processing using a subset of regular expression patterns from the Emergency Medical Text Preprocessor (EMT-P)[32], and maps text spans to standard vocabulary concepts using the QuickUMLS component.

COVID-19 Surveillance

Another operational project used medspaCy to identify VA patients who have were tested for COVID-19 outside of the VA network when no structured lab results were available[33]. This pipeline classifies clinical documents which likely contain a mention of positive COVID testing so that they can be prioritized for manual chart review. Between January 1 and June 15, 2020 the system had processed documents for over 3 million patients and had resulted in over 36% of all identified COVID positive patients being identified by this method alone. An evaluation of the system demonstrated precision and recall of 82.5% and 94.2%, respectively. As terminology related to the virus changed over time, it was imperative to add to concepts and patterns nearly every day. Iterative development of this system with medspaCy permitted putting the system into practice by January 21, 2020 and has been continuously operational throughout the pandemic, having processed over 63 million clinical documents one year after initial deployment. [§]

VA COVID-19 Screening

As the COVID-19 pandemic spread, VA facilities implemented standardized COVID-19 screenings for all visitors and employees. All interactions with Veterans are logged using a specific document format that quickly became one of the most frequent documents entered into the VA EMR in 2020. Veterans in inpatient care or community living centers often had multiple COVID-19 screenings recorded each day.

A medspaCy pipeline was developed to identify the screening as a standalone document or as a subsection of a larger document and then to extract up to 17 different possible screening questions and the associated responses including community- or travel-based exposure to COVID-19, previous COVID-19 tests or diagnoses, or specific COVID-19 symptoms.

This pipeline has processed 6.8 million screening documents for 914,000 Veterans who have received a COVID-19 test. The dataset is available to VA researchers in the VA COVID-19 Shared Data Resource.[34]

Inpatient Nursing Templated Text Processing

Several national VA standard operating procedures have been put into operation during the COVID-19 pandemic mandating use of standardized inpatient nursing shift assessments. While the assessments are presented to users as forms, the data are stored as the raw text conversions of templated questionnaires with their answers, thus, much of the information is not automatically retrievable. The medspaCy sectionizer is being utilized for the automatic extraction of information from these semi-structured documents (Figure 4). Given an empty version of each template, a utility module is used to automatically generate sectionizer rules. The sectionizer applies those rules to identify each question in the template as section header and extracts of all template answers captured as the section body.

Veteran Suicide Risk and Intervention Analyses

Several capabilities of medspaCy were used to rapidly summarize insights from Veterans, family members and organizations who responded to a request for information on how to improve suicide prevention. This survey was performed as part of the President's Roadmap to Empower Veterans and End a National Tragedy of Suicide (PREVENTS) task force to better understand risk factors and approaches to end Veteran suicide[35]. A total of 9,040 open ended responses

[§]https://spacy.io/universe/project/cov-bsv

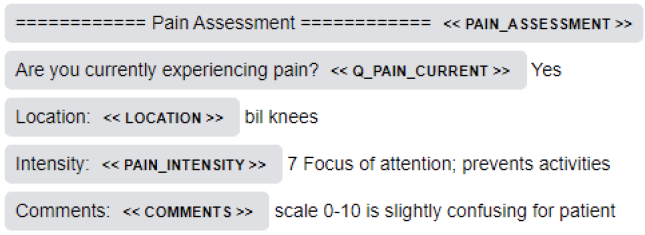

Figure 4: Highlighted section headers delineate template questions with the section body as the responses when processing semi-structured text.

needed to be analyzed and their insights summarized to allow a report of findings to be delivered to the task force within 3 months. Several capabilities in medspaCy made this project possible within the requested deadline, such as iterative rule development, exploratory data analysis, and visualization of text extraction. The project was also able to leverage existing knowledge resources from UMLS, including concepts related to mental health, medication and treatment. These insights were utilized by the task force as material to inform development of the PREVENTS[35].

Homelessness and Housing Stability Identification

Social determinants of health including homelessness are important factors in patient health and are often only documented in free text[36,37]. The VA partners with community organizations through the Supportive Services for Veteran Families (SSVF) program to provide rapid rehousing and temporary financial support to veterans who are homeless or at risk of becoming homeless[38]. A pipeline is being developed using medspaCy which extracts mentions of homelessness and housing stability from clinical texts. This information is then aggregated to infer a patient's housing stability over a period of time and will be used to evaluate long-term outcomes of SSVF participants and identify at-risk individuals who could benefit from enrollment in the program.

Education

Besides supporting operational or clinical application, medspaCy's simple interface and low barrier to entry makes it useful for education. Local, national, and international workshops and tutorials have utilized medspaCy to illustrate clinical text processing. A recent multi-day intensive course on clinical data science targeted to medical students used medspaCy as part of its curriculum[¶]. This course introduces medspaCy in one of the first sessions so that the remainder of the workshop can be devoted to experiential learning where students create rules and modify components to develop systems for processing clinical documentation. A simple installation process and a common programming interface enables students with little programming experience to quickly apply NLP to clinical data.

Future work

There are many additional components and applications possible in the next steps for medspaCy. One potential area of development is releasing medspaCy pipelines which include components, resources, and pre-trained models for specific cNLP tasks. These could also be created from publicly available resources such as MIMIC-III[39], shared tasks such as i2b2[40], or ontologies constructed during previous research. Released models could include a broad range of clinical domains, including adverse drug event detection[41,42], infectious disease surveillance, and radiology reporting.

While most of the existing medspaCy components support rule-based systems, future work could include additional utilities for machine learning. This could include word embeddings loaded as part of a default pipeline or utilities for feature extraction from text. Such improvements would allow users to more fully leverage both statistical and rule-based cNLP.

Conclusion

By improving the accessibility to high-performance NLP solutions optimized for clinical language, we aim to improve the productivity of cNLP development and reduce development burden for projects that require the use of rule-based

[¶]https://github.com/Melbourne-BMDS/mimic34md2020_materials

NLP or the integration of rules with other NLP methods. Unlocking new data elements through clinical information extraction will enable researchers to find answers to questions previously inaccessible for investigation.

Acknowledgements

This work was supported using resources and facilities of the Department of Veterans Affairs (VA) Informatics and Computing Infrastructure (VINCI), VA HSR RES 13-457, the Veterans Health Administration (VHA) Office of Analytics and Performance Integration (API), BASIC (Biosurveillance, Antimicrobial Stewardship, and Infection Control) and by the VA HSR&D Informatics, Decision-Enhancement, and Analytic Sciences (IDEAS) Center of Innovation (CIN 13-414).

The views expressed in this article are those of the authors and do not necessarily reflect the position or policy of the Department of Veterans Affairs or the United States government.

References

[1] Digan W, Névéol A, Neuraz A, Wack M, Baudoin D, Burgun A, et al. Can reproducibility be improved in clinical natural language processing? A study of 7 clinical NLP suites. Journal of the American Medical Informatics Association. 2021 3;28(3):504–515.

[2] Ferrucci D, Lally A. UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering. 2004 9;10(3-4):327–348.

[3] Cunningham H. GATE, a general architecture for text engineering. Computers and the Humanities. 2002;36(2):223–254.

[4] Savova GK, Masanz JJ, Ogren PV, Zheng J, Sohn S, Kipper-Schuler KC, et al. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. Journal of the American Medical Informatics Association. 2010;17(5):507–513.

[5] Soysal E, Wang J, Jiang M, Wu Y, Pakhomov S, Liu H, et al. CLAMP - a toolkit for efficiently building customized clinical natural language processing pipelines. Journal of the American Medical Informatics Association. 2018 3;25(3):331–336.

[6] Cornia R, Patterson OV, Ginter T, Duvall SL. Rapid NLP Development with Leo. In: AMIA Annu Symp Proc. Washington, DC, USA; 2014. p. 1356.

[7] Zeng QT, Goryachev S, Weiss S, Sordo M, Murphy SN, Lazarus R. Extracting principal diagnosis, co-morbidity and smoking status for asthma research: Evaluation of a natural language processing system. BMC Medical Informatics and Decision Making. 2006 7;6(1):30.

[8] Demner-Fushman D, Rogers WJ, Aronson AR. MetaMap Lite: an evaluation of a new Java implementation of MetaMap. Journal of the American Medical Informatics Association. 2017;24(4):841–844.

[9] Aronson AR. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In: Proceedings of the AMIA Symposium. American Medical Informatics Association; 2001. p. 17.

[10] Vizard M. Python Pioneer Assesses the 30-year-old Programming Language; 2021. Available from: https://venturebeat.com/2021/02/19/python-pioneer-assesses-the-30-year-old-programming-language/ [cited 03/08/2021].

[11] Hayes B. Usage of Programming Languages by Data Scientists: Python Grows while R Weakens; 2020. Available from: https://businessoverbroadway.com/2020/06/29/usage-of-programming-languages-by-data-scientists-python-grows-while-r-weakens/ [cited 03/08/2021].

[12] Piatetsky G. Python Leads the 11 Top Data Science, Machine Learning Platforms: Trends and Analysis. 2019; 2019. Available from: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html [cited 03/08/2021].

[13] Github. The State of the Octoverse. GitHub; 2020. Available from: https://octoverse.github.com/octoverse.github.com [cited 03/08/2021].

[14] 2020 Kaggle Machine Learning & Data Science Survey. Kaggle; 2020. Available from: https://www.kaggle.com/c/kaggle-survey-2020/overview [cited 03/08/2021].

[15] Pérez F, Granger BE. IPython: a system for interactive scientific computing. Computing in science & engineering. 2007;9(3):21–29.

[16] Bird S, Klein E, Loper E. Natural Language Processing with Python. 1st ed. Steele J, editor. O'Reilly Media Inc.; 2009.
[17] Neumann M, King D, Beltagy I, Ammar W. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In: Proceedings of the 18th BioNLP Workshop and Shared Task. Stroudsburg, PA, USA: Association for Computational Linguistics; 2019. p. 319–327.
[18] Mulyar A, Mahendran D, Maffey L, Olex A, Matteo G, Dill N, et al. TAC SRIE 2018: Extracting Systematic Review Information with MedaCy. Tac Srie 2018. 2018. Available from: https://github.com/NanoNLP/medaCy [cited 07/05/2021].
[19] Fu S, Chen D, He H, Liu S, Moon S, Peterson KJ, et al. Clinical concept extraction: A methodology review. Journal of Biomedical Informatics. 2020 8;109:103526.
[20] Sheikhalishahi S, Miotto R, Dudley JT, Lavelli A, Rinaldi F, Osmani V. Natural Language Processing of Clinical Notes on Chronic Diseases: Systematic Review. JMIR medical informatics. 2019 4;7(2):e12239.
[21] Ford E, Carroll JA, Smith HE, Scott D, Cassell JA. Extracting information from the text of electronic medical records to improve case detection: a systematic review. Journal of the American Medical Informatics Association : JAMIA. 2016 9;23(5):1007–15.
[22] Shi J, Mowery D, Doing-Harris KM, Hurdle JF. RuSH: a Rule-based Segmentation Tool Using Hashing for Extremely Accurate Sentence Segmentation of Clinical Text. In: AMIA Annu Symp Proc; 2016. .
[23] Hyun S, Bakken S. Toward the creation of an ontology for nursing document sections: mapping section names to the LOINC semantic model. AMIA Annu Symp Proc. 2006:364–8.
[24] Goldsmith JD, Siegal GP, Suster S, Wheeler TM, Brown RW. Reporting guidelines for clinical laboratory reports in surgical pathology. Archives of pathology & laboratory medicine. 2008 10;132(10):1608–16.
[25] T Zeng Q. Characterizing Clinical Text and Sublanguage: A Case Study of the VA Clinical Notes. Journal of Health & Medical Informatics. 2011;04(02):1–9.
[26] Denny JC, Miller RA, Johnson KB, Spickard A. Development and evaluation of a clinical note section header terminology. AMIA Annu Symp Proc. 2008:156–160.
[27] Bodenreider O. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research. 2004;32(suppl_1):D267–D270.
[28] Soldaini L, Goharian N. QuickUMLS: a fast, unsupervised approach for medical concept extraction. Medical Information Retrieval (MedIR) Workshop on SIGIR 2016. 2016.
[29] Okazaki N, Tsujii J. Simple and efficient algorithm for approximate dictionary matching. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010); 2010. p. 851–859.
[30] Harkema H, Dowling JN, Thornblade T, Chapman WW. ConText: an algorithm for determining negation, experiencer, and temporal status from clinical reports. Journal of Biomedical Informatics. 2009 10;42(5):839–51.
[31] Chapman BE, Lee S, Kang HP, Chapman WW. Document-level classification of CT pulmonary angiography reports based on an extension of the ConText algorithm. Journal of Biomedical Informatics. 2011 10;44(5):728–737.
[32] Travers DA. Evaluation of Emergency Medical Text Processor, a System for Cleaning Chief Complaint Text Data. Academic Emergency Medicine. 2004 11;11(11):1170–1176.
[33] Chapman A, Peterson K, Turano A, Box T, Wallace K, Jones M. A Natural Language Processing System for National COVID-19 Surveillance in the US Department of Veterans Affairs; 2020.
[34] DuVall SL, Scehnet J. Introduction to the VA COVID-19 Shared Data Resource and its Use for Research [Webinar]; 2020. Available from: https://www.hsrd.research.va.gov/for_researchers/cyber_seminars/archives/video_archive.cfm?SessionID=3810 [cited 07/05/2021].
[35] US Department of Veterans Affairs. PREVENTS: the President's roadmap to empower veterans and end a national tragedy of suicide; 2020. Available from: https://www.va.gov/PREVENTS/docs/PRE-007-The-PREVENTS-Roadmap-1-2_508.pdf [cited 07/05/2021].
[36] Conway M, Keyhani S, Christensen L, South BR, Vali M, Walter LC, et al. Moonstone: A novel natural language processing system for inferring social risk from clinical narratives. Journal of Biomedical Semantics. 2019 4;10(1).

[37] Gundlapalli AV, Carter ME, Divita G, Shen S, Palmer M, South B, et al. Extracting Concepts Related to Homelessness from the Free Text of VA Electronic Medical Records. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2014;2014:589–598.

[38] Nelson RE, Byrne TH, Suo Y, Cook J, Pettey W, Gundlapalli AV, et al. Association of Temporary Financial Assistance With Housing Stability Among US Veterans in the Supportive Services for Veteran Families Program. JAMA Network Open. 2021 2;4(2):e2037047.

[39] Johnson AEW, Pollard TJ, Shen L, Li-Wei HL, Feng M, Ghassemi M, et al. MIMIC-III, a freely accessible critical care database. Scientific data. 2016;3(1):1–9.

[40] Sun W, Rumshisky A, Uzuner O. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. Journal of the American Medical Informatics Association. 2013;20(5):806–813.

[41] Chapman AB, Peterson KS, Alba PR, DuVall SL, Patterson OV. Detecting adverse drug events with rapidly trained classification models. Drug safety. 2019;42(1):147–156.

[42] Henry S, Buchan K, Filannino M, Stubbs A, Uzuner O. 2018 n2c2 shared task on adverse drug events and medication extraction in electronic health records. Journal of the American Medical Informatics Association. 2020;27(1):3–12.