8-11-2016

v3NLP Framework: Tools to Build Applications for Extracting Concepts from Clinical Text

Guy Divita
VA Salt Lake City Health Care System and University of Utah School of Medicine, guy.divita@hsc.utah.edu

Marjorie Carter MS
VA Salt Lake City Health Care System and University of Utah School of Medicine, Marjorie.Carter@hsc.utah.edu

Le-Thuy Tran PhD
VA Salt Lake City Health Care System and University of Utah School of Medicine

Doug Redd MS
VA Salt Lake City Health Care System and University of Utah School of Medicine

See next pages for additional authors

Recommended Citation

Divita, Guy; Carter, Marjorie MS; Tran, Le-Thuy PhD; Redd, Doug MS; Zeng, Qing T. PhD; Duvall, Scott PhD; Samore, Matthew H. MD, PhD; and Gundlapalli, Adi V. MD, PhD, MS (2016) "v3NLP Framework: Tools to Build Applications for Extracting Concepts from Clinical Text," *eGEMs (Generating Evidence & Methods to improve patient outcomes)*: Vol. 4: Iss. 3, Article 10.
DOI: http://dx.doi.org/10.13063/2327-9214.1228
Available at: http://repository.edm-forum.org/egems/vol4/iss3/10

The Electronic Data Methods (EDM) Forum is supported by the Agency for Healthcare Research and Quality (AHRQ), Grant 1U18HS022789-01. eGEMs publications do not reflect the official views of AHRQ or the United States Department of Health and Human Services.

v3NLP Framework: Tools to Build Applications for Extracting Concepts from Clinical Text

Abstract

Introduction: Substantial amounts of clinically significant information are contained only within the narrative of the clinical notes in electronic medical records. v3NLP Framework is a set of best of breed functionalities developed to transform this information into structured data for use in quality improvement, research, population health surveillance, and decision support.

Background: MetaMap, cTAKES and similar well known NLP tools do not have sufficient scalability out of the box. v3NLP Framework evolved out of the necessity to scale these tools up and provide a framework to customize and tune techniques to fit a variety of tasks, including document classification, tuned concept extraction for specific conditions, patient classification, and information retrieval.

Innovation: Beyond scalability, several v3NLP Framework developed projects have been efficacy tested and benchmarked. While v3NLP Framework includes annotators, pipelines and applications, the functionalities enable developers to create novel annotators, put annotators into pipelines and scaled applications.

Discussion: v3NLP Framework has been successfully utilized in many projects including general concept extraction, risk factors for homelessness among veterans, and identification of mentions of the presence of an indwelling urinary catheter. Projects as diverse as predicting colonization with methicillin resistant Staphylococcus aureus and extracting references to military sexual trauma are being built using v3NLP Framework components.

Conclusion: v3NLP Framework is a set of functionalities and components that provide Java developers the ability to create novel annotators, place annotators into pipelines, and applications to extract concepts from clinical text. There are scale-up and scale-out functionalities to process large numbers of records.

Acknowledgements

This work is funded by U.S. Department of Veterans Affairs, OR&D, Health Services Research and Development grants VINCI HIR-08-204, CHIR HIR 08-374, ProWATCH grants HIR-10-001 and HIR 10-002. We thank the VA Informatics and Computing Infrastructure (VINCI) for their support of our project. We acknowledge the staff, resources and facilities of the VA Salt Lake City IDEAS Center 2.0 for providing a rich and stimulating environment for NLP research. The views expressed in this paper are those of the authors and do not necessarily represent the views of the Department of Veterans Affairs or the United States Government.

Keywords

Natural Language Processing

Disciplines

Artificial Intelligence and Robotics

Authors

Guy Divita, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Marjorie Carter, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Le-Thuy Tran, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Doug Redd, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Qing T Zeng, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Scott Duvall, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Matthew H Samore, *VA Salt Lake City Health Care System and University of Utah School of Medicine*; Adi V Gundlapalli, *VA Salt Lake City Health Care System and University of Utah School of Medicine*.

eGEMs
Generating Evidence & Methods to improve patient outcomes

v3NLP Framework: Tools to Build Applications for Extracting Concepts from Clinical Text

Guy Divita, MS; Marjorie E. Carter, MSPH; Le-Thuy Tran, PhD; Doug Redd, MS; Qing T Zeng, PhD; Scott Duvall, PhD; Matthew H. Samore, MD; Adi V. Gundlapalli, MD, PhD, MS[i]

ABSTRACT

Introduction: Substantial amounts of clinically significant information are contained only within the narrative of the clinical notes in electronic medical records. The v3NLP Framework is a set of "best-of-breed" functionalities developed to transform this information into structured data for use in quality improvement, research, population health surveillance, and decision support.

Background: MetaMap, cTAKES and similar well-known natural language processing (NLP) tools do not have sufficient scalability out of the box. The v3NLP Framework evolved out of the necessity to scale-up these tools and to provide a framework to customize and tune techniques that fit a variety of tasks, including document classification, tuned concept extraction for specific conditions, patient classification, and information retrieval.

Innovation: Beyond scalability, several v3NLP Framework-developed projects have been efficacy tested and benchmarked. While v3NLP Framework includes annotators, pipelines, and applications, its functionalities enable developers to create novel annotators and to place annotators into pipelines and scaled applications.

Discussion: The v3NLP Framework has been successfully utilized in many projects including general concept extraction, risk factors for homelessness among veterans, and identification of mentions of the presence of an indwelling urinary catheter. Projects as diverse as predicting colonization with methicillin-resistant *Staphylococcus aureus* and extracting references to military sexual trauma are being built using v3NLP Framework components.

Conclusion: The v3NLP Framework is a set of functionalities and components that provide Java developers with the ability to create novel annotators and to place those annotators into pipelines and applications to extract concepts from clinical text. There are scale-up and scale-out functionalities to process large numbers of records.

[i]VA Salt Lake City Health Care System and University of Utah School of Medicine

Introduction

Clinically significant information such as symptoms and other personal details expressed by the patient to the provider are often contained only within the narrative of the clinical note accompanying that medical visit. The fields of clinical informatics and information extraction are devoted to developing methods to accurately retrieve salient information from text for use in quality improvement, research, and population health surveillance. There are limited, open-source common platforms that perform these tasks. We present v3NLP Framework as an open-source suite of functionalities to build such applications.

Background

The medical natural language processing (NLP) field includes seminal contributions from the National Library of Medicine's Unified Medical Language System (UMLS) project[1] and associated extraction tool, MetaMap.[2] The Mayo Clinic, with the advent of the Apache cTAKES project,[3] focused their efforts on the clinical domain. cTAKES—an open-source, clinical-concept extraction tool—was built upon the IBM Unstructured Information Management Architecture (UIMA) platform.[4] The IBM-UIMA project evolved into the Apache UIMA and the scale-out Apache UIMA-AS projects,[5] the technology that underlies the IBM-WATSON projects.[6] The v3NLP Framework utilizes both cTAKES and MetaMap components and is built upon the UIMA platforms.

Clinical Notes Characteristics

There are unique challenges with processing the text that is found in clinical notes as opposed to other types of text or biomedical literature typically processed using NLP. Clinical text written by medical providers includes telegraphic language and semi-structured text. Nursing notes and surveys are replete with check boxes and structured question and answer templates. These structures are the source of the majority of causes for false positive errors.[7] The semantics or truthfulness of the "matches" (the *mentions* found within these structures) has to take into account which, if any, box is checked or what is in the answer, if present. The context of the mention is also relevant. Is the mention about the patient or about someone else, such as a family member? Is the mention about a historical event, such as a surgery performed several decades ago? Is the mention about a hypothetical or conditional event, such as "take this medication if the condition gets worse"? The context of the section of the note wherein a mention is found is important. A medication is a "treatment" when found in the Medications section, but an "allergy" when found in the Allergies section.

The data addressed by v3NLP Framework comes from United States Department of Veterans Affairs (VA) medical facilities. This national system of 150+ hospitals and nearly 800 community-based outpatient clinics uses more than 2,700 note titles, ranging from discharge summaries and nursing notes to educational material and veterans post-deployment health surveys, and includes 50,000 types of note sections.[8]

Natural Language Processing Techniques and Systems

Typical NLP involves the use of a dictionary lookup of sorts, using dictionaries with classifications for each concept or main entry. In the clinical domain, many systems use the the National Library of Medicine's Unified Medical Language System (UMLS).[1] The UMLS includes a metathesaurus.[9] The metathesaurus combines terminologies from controlled medical vocabulary sources, and it groups terms with similar meanings into concepts, thus creating sets of synonyms. Every concept within this resource is assigned a high-level category. The

categories chosen fit within a high-level semantic network, allowing the computation of generalization, if necessary.

Dictionary lookup techniques find, mark, and categorize mentions found in the text that match dictionary entries. This is known as "concept extraction." The resulting annotations can then be aggregated by category to give numeric summaries of how many patients had a given disease or diagnosis. The resulting annotations could be used as supporting evidence for information retrieval purposes, or as features fed to a machine learning algorithm. The National Library of Medicine's MetaMap tool[2] and the Apache cTAKES tool[3] are two well-known open-source, freely available concept-extraction systems.

NLP typically involves modularized processing units that perform a small subtask—like tokenizing, or finding sentences. These are often referred to as "annotators." Annotators can be chained or piped together into pipelines, where the output of one is the input to another. The University of Sheffield's General Architecture for Text Engineering (GATE)[10] and the Apache UIMA[4] are general purpose platforms that create the annotator and pipeline components utilized by many NLP systems, allowing developers to concentrate on the business logic involved with the annotators. The Apache cTAKES project, the IBM Watson technologies, and v3NLP Framework utilize UIMA.

Motivation for the v3NLP Framework

The v3NLP Framework's mandate is to utilize best-of-breed techniques to process clinical notes in the VA electronic medical records to extract information and to provide quality data for health sciences research. Implicit in this mandate is the capability to process big data in reasonable amounts of time. MetaMap and cTAKES, having been well vetted for many tasks, did not have sufficient scalability out of the box. The v3NLP Framework evolved out of the necessity to scale up these tools and provide a framework to customize and tune techniques to fit a variety of tasks including document classification, tuned concept extraction for specific conditions, patient classification, and document information retrieval.

The v3NLP Framework includes best-of-breed annotators where such techniques were well-known, such as the cTAKES part of "speech tagger."[3] The heterogeneity of the data has necessitated some novel annotator development—particularly to handle semistructured text, and the over 35,000 kinds of sections that VA cohorts could include.

In our initial approach to processing large VA data sets, the NLP pipelines used MetaMap and cTAKES. Our initial performance goals were set by the desire to process more than one billion clinical notes with typical pipelines in reasonable and practical time frames such as days or weeks. However, upon benchmarking the performance of these initial pipelines and scale out efforts, it became clear that both tools needed more customization beyond pipeline or process replication wrappers. Sophia, an expedient UMLS concept extraction tool, built using v3NLP Framework components with no pipeline replication, was benchmarked at 18 times the throughput performance of cTAKES, and 7 times faster than the throughput performance of a service around 60 instances of MetaMap.[11] A scaled-out version of Sophia, with 60 replicated pipelines, could process one billion records in 3 years, whereas it would take 2,219 years to process the same records with cTAKES out of the box.

Innovation

The v3NLP Framework is a suite of middleware components that can be used to assemble NLP applications. It contains methods to write data into specific formats useful for aggregate summaries,

data exploration, statistical analyses, machine learning, annotation reuse, and human chart review. The Framework contains extraction tools to retrieve targeted concepts from controlled medical vocabularies and to mine for concepts found in text—such as symptoms, anatomical parts, activities, and psychosocial risk factors.

The Framework contains best-of-breed underlying functionalities to determine if retrieved concepts are about the patient, are negated, and whether they are hypothetical or not. Functionality to decompose text into digestible document elements including slot: values, check boxes, questions and answers where the assertion semantics differ from prose is provided.

Applications built using v3NLP Framework are assembled into pipelines built from sequences of annotators. The v3NLP Framework contains scale-up and scale-out functionality to run pipelines in parallel to increase throughput to handle "big data." The scale-up functionality includes dynamic throttling to maximize CPU resources.

Novel Annotators and Applications

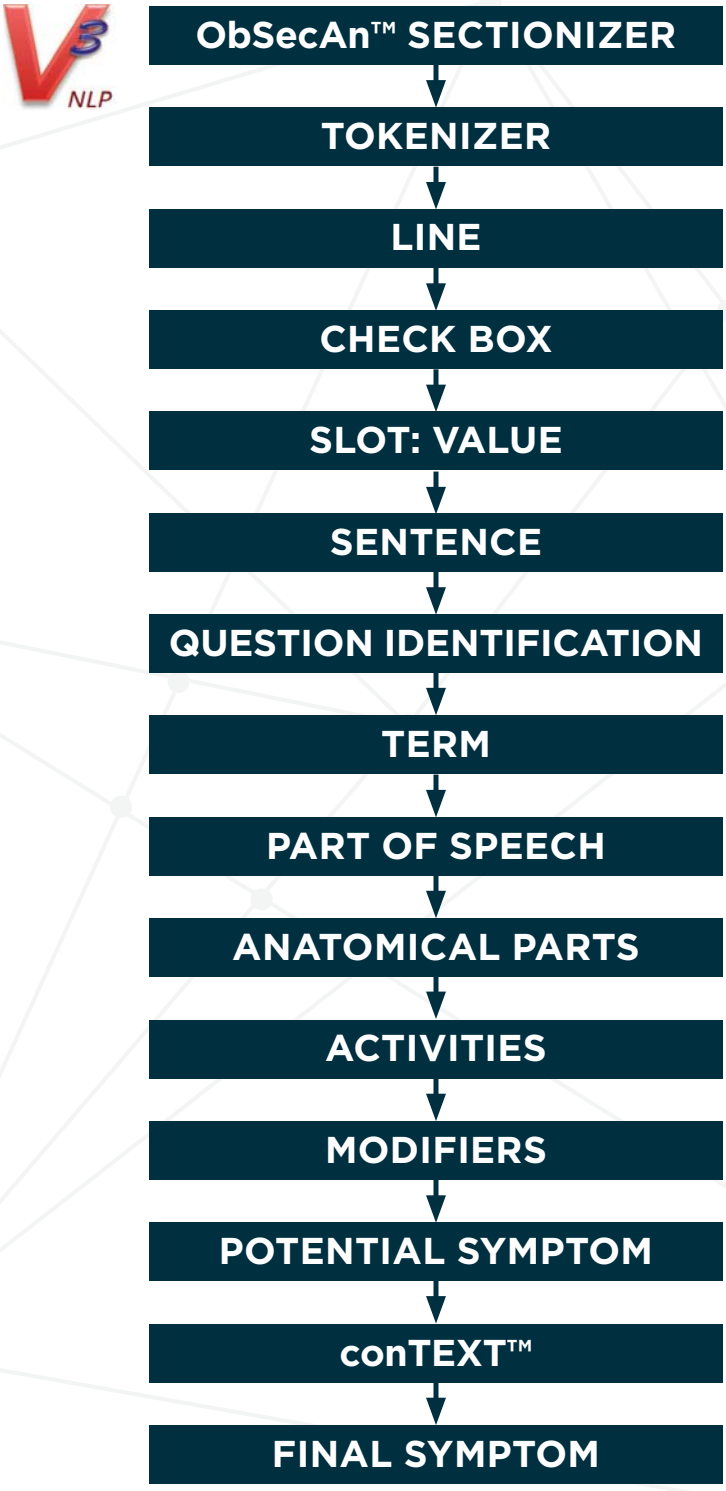
Sophia, an expedient, general purpose UMLS concept-extraction tool,[11] is a good example of an application built from v3NLP Framework components that use best-of-breed annotators such as the cTAKES part-of-speech tagger, along with novel annotators. Sophia includes novel annotators to ameliorate idiosyncrasies found in heterogeneous clinical text. Three annotators ameliorate semi-structured text, slot: value, check box, and question and answer. Each annotator marks the content heading and content value parts.[7]

These annotators create structures that include a generic content header component and a dependent content component. The semantics of whether a concept-mention is found in either location is handled in the question and answer annotator, outside the normal assertion annotator—due to the telegraphic nature of such forms. For check boxes, the values of the dependent content part need to be examined to see if the value has positive or negative polarity. Explicit values such as *N* or *NO* have a negative polarity, and concept mentions in the content heading part receive a negative attribution. The Sophia pipeline also includes a tail end assertion annotator. A multithreaded wrapper was created for this purpose around the widely used conTEXT[12] algorithm to parallelize the assertion computation at the phrase level. The Sophia application has been shown to be as effective as cTAKES, yet is 17 times faster.[11]

Many of the above mentioned v3NLP Framework annotators are also employed in a symptom extraction pipeline (see Figure 1). The symptom extraction pipeline identifies signs and symptoms as distinguished from findings, disorders, and diagnoses. This pipeline includes three novel annotators. v3NLP Framework includes a local dictionary lookup feature, which was employed to utilize a set of 92,000 sign and symptom terms, derived from the UMLS.[13] This set of symptom terms came with organ system categorizations. Features including the surrounding words, the surrounding parts of speech, and the symptom-organ system categorization from symptom mentions were used to create a machine learned model, based on learning from a human symptom annotated cohort. An encapsulating annotator around the machine learned model is included with the symptom extraction tool. The machine-learning component is vital to tease out the plethora of false positive mentions found using just a dictionary lookup. The section in which a sign or symptom mention appears is thought to be a salient feature. An annotator wrapped around the ObSecAn sectionizer[8] is also

Volume 4 (2016) Issue Number 3

Figure 1. v3NLP Framework Symptom Pipeline



employed in the symptom-extraction pipeline to provide section features for the machine learning component to utilize. The ObSecAn sectionizer was developed as an implementation of Denny,[14] with additional features recognized from a database of 35,000 VISTA templates. The symptom-extraction application has been efficacy tested with an F-measure of 0.71[15] for the task of finding signs or symptoms (true positives). When combined with finding what is not a sign or symptom (true negatives), the overall F-measure is 0.87. Prior generic symptom extraction tasks have been benchmarked at much lower rates.[16]

v3NLP Framework Features

To date, the v3NLP Framework contains 33 annotators. Table 1 lists some of the more novel annotators, including annotators that identify activities and modifiers to those activities such as coughing, running, and sleeping. Clinical documents often note normal activities along with a modifier to denote symptoms, such as excessive coughing, difficulty running, or poor sleep. Also listed are validated annotators that identify urinary catheter mentions,[17] scaled-out wrappers around conTEXT,[12] psychosocial risk factors used to predict homelessness,[18] and section zoning.[19]

Table 1. Selected v3NLP Framework Annotators

ANNOTATOR	DESCRIPTION
Activities	Labels UMLS-defined Activities
Anatomical parts	Labels UMLS-defined Anatomical parts
CAUTI Concept	Labels CAUTI mentions
cheapWSD	Disambiguates class A from class B
ConTEXTor	Wrapper around conTEXT
Homelessness	Labels psychosocial mentions
Metamap	Wrapper around NLM's metaMap concept extraction tool
Modifiers	Labels UMLS-defined modifiers
MRSA Concept	Labels MRSA mentions
ObSecAnSection	Labels and decomposes sections
Problem	Labels i2b2 2010 VA Challenge-defined problems
Question and Answer	Labels questions found in text and their answers
Slot Value	Labels semi-structured text in the form of slot: value
Sophia	Labels UMLS concept mentions
Symptom	Labels symptoms
Term	Labels terms from SPECIALIST[20] and locally defined lexica

There are 34 pre-composed pipelines, 13 applications, 5 scaled-up applications, and 5 UIMA-AS Services.

Table 2 shows a selection from the 13 marshallers built thus far. "Marshallers"—packages that include readers and writers—offer the interoperability glue between upstream and downstream parts of workflows that require NLP. These provide flexible ways to read in data and ways that other systems can use processed text. The v3NLP Framework is pure Java, spread across 15 git projects, and is bundled within 200 maven-deployed jars deposited in a nexus repository.

Building New Applications

This framework provides developers with tools and aids to build new annotators and applications compared with the underlying UIMA platform. The UIMA platform includes extensive use of a myriad of configuration files for many of the necessary components, which is a source of developer frustration. The v3NLP Framework obfuscates all but one kind of the configuration files necessary, and that is for a principled reason. The v3NLP Framework utilizes uimaFIT,[21] a tool to automatically generate the configuration files "on the fly." The v3NLP Framework also has the capability to utilize Leo[22] to generate files to take advantage of UIMA-AS capabilities.

A Common Model to Enforce Label Interoperability

The v3NLP Framework requires the use of a Type Descriptor configuration file. This file includes the definitions or schema of the labels that equate to the markup annotations. While Leo[22] and uimaFIT[21] autogenerate this file on the fly, it is kept as a mechanism to explicitly define the attributes of the labels. In addition, the framework requires the use of the Consortium for Healthcare Informatics Research (CHIR) Common Model,[23] an ontology of labels created from the union of labels from existing NLP Systems. The use of labels from a common model enforces a standard between existing and new NLP systems, enabling interoperability among external NLP systems and components. This common model

Table 2. Selected v3NLP Framework Marshallers

MARSHALLER	FROM	TO	DESCRIPTION
Knowtator/eHOST	X	X	Knowtator and eHOST are useful full-featured annotators
BioC	X	X	A minimalistic annotation messaging format
JDBC Database	X	X	Database connector
Multi-Record Files	X	X	Format to bundle multiple records into one file
VTT	X	X	Lightweight portable Annotator distributed by NLM
CorpusStats		X	Instance-based CSV file, and 3 summarization CSV files
Snippet	X	X	Creates snippets around concept instances
String	X	X	Reads in from a passed in string
File	X		Reads in text from a single file

has been rendered into the UIMA type descriptor. New or additional labels are possible through extending from existing classes.

The v3NLP Framework Annotators

The v3NLP Framework annotators are straightforward UIMA annotator classes. As such, each annotator contains an initialization method that gets called once, a process method that gets called for each document, and a destroy method for when all the documents have been processed. The bulk of the business logic is implemented in the process method. There is no deviation from UIMA. This turns out to be one of the elegant parts of UIMA. The v3NLP Framework annotators are interoperable with other UIMA-based pipelines.

Creating a Pipeline

The Framework includes a simplifying class that wraps and hides the details of chaining annotators together, hiding the magic provided by uimaFIT and Leo. Figure 2 shows the code for chaining a set of annotators together to create an example pipeline. Note that the "pipeline.add()" method simply requires the name of the UIMA annotator class. The mechanism for passing in the name of each annotator shown in Figure 2 ensures that each annotator class is included as a compile time dependency to make sure that the referenced classes are in the classpath, rather than waiting until runtime to know that the correct class is being referenced.

Passing parameters into annotators—which involves creating a configuration file for each annotator, making it difficult to change parameters on the fly or to programmatically change parameters from a calling program—is one of UIMA's limitations. The Framework does away with these configuration files, at the expense of including a procedure to transform parameter name=value pairs into a format that passes through to the regular UIMA annotators that function as UIMA parameters. Figure 2 also shows how command line arguments can be passed to each annotator using a method to convert from a string array to the UIMA parameter passing way.

The v3NLP Framework Application

Within the v3NLP Framework, the application class enables a developer to assemble an application that defines a reader, instantiates and attaches a pipeline, and attaches one or more writers prior to running across a corpus. Figure 3 shows an example v3NLP Framework application.

Scale-Up and Scale-Out Capabilities

Scale-up and scale-out capabilities have been incorporated to handle large amounts of text—following v3NLP Framework's mandate.[24] A distinction is made between *scale-up* capabilities, which replicate pipelines using multiple threads within one process, and *scale-out* capabilities, which spread pipelines across different processes and across different machines.

Scale-Up Application

The scale-up application class wraps around instances of framework applications, forking off threads, but with some controls. The initial and maximum number of pipelines can be set. A threshold CPU maximum load can be set. The scale-up application will fork off additional threads until the load has reached the threshold. Pipelines will be destroyed when the CPU threshold is breached, to insure other processes on shared resources do not get starved. Some of the annotators incur a large initialization period. A waiting time was added before spinning up a new pipeline to insure that all initialization periods are complete and that all pipelines are processing documents prior to the

Volume 4 (2016) Issue Number 3

Figure 2. v3NLP Framework Example Pipeline

```
public class ExamplePipeline extends AbstractPipeline  {
  public FrameworkPipeline createPipeline(String[] pArgs) {
    FrameworkPipeline pipeline = new FrameworkPipeline( pArgs );
// ------------------------------------------------+--------------------+-----------------+-----------------+-----------------+
// Convert parameters into UIMA style   |String            | Object          | String          | boolean         | boolean     |
//  Context Parameters                  |parameterName | parameterVal  | paramtrType | isMultiValued | isMandatory|

   UimaContextParameter          argsParameters = new UimaContextParameter( "args",    pArgs,   "String",  true,  true);

     pipeline.add(          TokenAnnotator.class.getCanonicalName()                                  );
     pipeline.add(          LineAnnotator.class.getCanonicalName()                                   );
     pipeline.add(          CheckBoxAnnotator.class.getCanonicalName()                               );
     pipeline.add(          SlotValueAnnotator.class.getCanonicalName()                              );
     pipeline.add(          SentenceAnnotator.class.getCanonicalName()                               );
     pipeline.add(          QuestionAnswerAnnotator.class.getCanonicalName()                         );
     pipeline.add(          TermAnnotator.class.getCanonicalName(),          argsParameters      );
     pipeline.add(          ExampleAnnotator.class.getCanonicalName(),    argsParameters      );
     pipeline.add(          ContextorAnnotator.class.getCanonicalName(),  argsParameters      );
     pipeline.add(          FilterWriter.class.getCanonicalName(),           argsParameters      );
     return pipeline;
  }  // End Method createPipeline() ======================
} // end Class ExamplePipeline
```

Figure 3. v3NLP Framework Example Application

```
public class ExampleApplication {
  public static void main(String[] pArgs) {
        FrameworkBaselineApplication application = new FrameworkBaselineApplication();
        // Create a pipeline; retrieve an analysis engine that uses the pipeline; attach it to the
application
        ExamplePipeline          examplePipeline  = new ExamplePipeline( pArgs );
        AnalysisEngine                              ae = examplePipeline.getAnalysisEngine();
        application.setAnalysisEngine(ae);
        // Create a reader and writer
        application.createReader(FrameworkBaselineApplication.TEXT_READER,          pArgs);
       application.addWriter(     FrameworkBaselineApplication.KNOWTATOR_WRITER, pArgs);
       // Run
       application.process();
  }  // End Method main() ----------------------------------
} // End ExampleApplication Class -------------------------
```

next pipeline being spun up. It was observed that a high-value adopted annotator has memory leaks, which over the span of processing millions of records becomes an issue. A feature was put into place to destroy pipelines that have processed a given number of documents and to spin up a new pipeline, in an effort to reclaim memory.

Scale-Out Capabilities

Leo[22] simplifies the use of UIMA Asynchronous Scale-out (UIMA-AS).[5] UIMA-AS includes a client, broker, and server architecture, where the client is responsible for reading in documents that it sends to a central broker. The broker passes documents to be processed to the next available server. Processed documents from a server are passed back to the broker to be sent back to the appropriate client. Servers wrap one or more pipeline processes. Leo enables a developer to have an easy mechanism to create a client and a server. The v3NLP Framework Pipeline class can be directly instantiated within a Leo server to spin up a UIMA-AS service. This mechanism has been adopted when more than one machine became available for use. Leo services around v3NLP Framework pipelines also have been used for service related annotation.

Software as Service Capabilities

RESTful Services[25] wrapped around v3NLP Framework applications are included in the v3NLP Framework codebase. The RESTful service enables web-based (RESTful) clients to send requests to a service and retrieve the processed document. A thin, minimalistic messaging protocol—the National Center for Biotechnology Information (NCBI)'s BioC[26]—was chosen here to marshal messages between client and service, rather than using UIMA's more verbose protocol.

Discussion

The v3NLP Framework has been utilized in projects beyond Sophia. Projects studying references to homelessness among veterans in the free text of medical notes,[18,27,28] and identifying mentions of the presence of an indwelling urinary catheter,[29] have successfully used this framework. Projects as diverse as predicting colonization with methicillin-resistant *Staphylococcus aureus* and extracting references to military sexual trauma from the narrative of the electronic medical note are being built using v3NLP Framework components.

The framework includes annotators, a mechanism to build pipelines, marshallers to read in data and write out results, and scale-out utilities to replicate pipelines. It provides a label standard to allow interoperability.

These projects are being coupled with machine learning tools such as Waikato Environment for Knowledge Analysis (Weka)[30] and R[31] to develop machine learned modules. The modules will be wrapped as annotators in a pipeline to be used to annotate large cohorts that will be converted into structured data within national repositories for decision support, quality assurance, and downstream research. Such projects also utilize annotation editors like VTT,[32] eHOST,[33] and Knowtator to review and allow human annotation of records.

Next Version

A proof of concept front-end workflow tool, called "Jack," is being developed that allows one to choose a pre-composed pipeline, the input, and the output; to run small cohorts; and to review using a chosen annotation editor. A complementary tool called "SeeMore" is a graphical interface to kick off and monitor back-end v3NLP, UIMA-AS, and RESTful services.

Volume 4 (2016) Issue Number 3

Future Work

Scale-up and scale-out functionality is a major focus. There are opportunities to further decompose work within a document, and to distribute such tasks across multiple threads in a fashion similar to the wrapper around the conTEXT algorithm when CPUs are available to further distribute tasks. Work on generalizing the conTEXT scale-up wrapper to become a generic mechanism for scaling up other annotators is being considered, for those more atomic tasks where work on one part of the document is independent of other parts of the document.

Conclusion

The v3NLP Framework is a set of functionalities and components that provide Java developers with the ability to create novel annotators and to put together annotators into pipelines and applications to extract concepts from clinical text. There are scale-up and scale-out functionalities to process large amounts of records. The v3NLP Framework has been used to create several projects. The v3NLP Framework pipelines and applications have been efficacy- and performance benchmarked.

Availability

In an effort to make our tools available to potential users, we copyrighted the v3NLP Framework, and it is distributed with an Apache License. Versions of v3NLP Framework are distributed from http://inlp.bmi.utah.edu/redmine/docs/v3nlp-framework/index.html.

Acknowledgements and Disclaimer

This work is funded by the United States Department of Veterans Affairs, OR&D, Health Services Research and Development grants VINCI HIR-08-204, CHIR HIR 08-374, ProWATCH grants HIR-10-001 and HIR 10-002. We thank the VA Informatics and Computing Infrastructure (VINCI) for their support of our project. We acknowledge the staff, resources, and facilities of the VA Salt Lake City IDEAS Center 2.0 (CIR 13-414) for providing a rich and stimulating environment for NLP research.

The views expressed in this paper are those of the authors and do not necessarily represent the views of the United States Department of Veterans Affairs or the United States government.

References

1. Lindberg C. The Unified Medical Language System (UMLS) of the National Library of Medicine. J Am Med Rec Assoc. 1990;61(5):40-2.
2. Aronson AR. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. Proceedings / AMIA Annual Symposium AMIA Symposium. 2001:17-21.
3. 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 : JAMIA. 2010;17(5):507-13.
4. Ferrucci D, Lally A. UIMA: an architectural approach to unstructured information processing in the corporate research environment. Nat Lang Eng. 2004;10(3-4):327-48.
5. Apache UIMA-AS.
6. Ferrucci DA, Levas A, Bagchi S, Gondek D, Mueller ET. Watson: Beyond Jeopardy! Artif Intell. 2013;199:93-105.
7. Divita G, Shen S, Carter ME, Redd A, Forbush T, Palmer M, et al. Recognizing Questions and Answers in EMR Templates Using Natural Language Processing. Studies in health technology and informatics. 2014;202:149-52.
8. Tran L-TT, Divita G, Redd A, Carter M, Judd J, Samore M, et al. OBSecAnnot: An Automated Section Annotator for Semi-structured Clinical Documents. JAMIA; 2015.
9. Schuyler PL, Hole WT, Tuttle MS, Sherertz DD. The UMLS Metathesaurus: representing different views of biomedical concepts. Bulletin of the Medical Library Association. 1993;81(2):217.
10. Cunningham H. GATE, a general architecture for text engineering. Computers and the Humanities. 2002;36(2):223-54.
11. Divita G, Zeng QT, Gundlapalli AV, Scott D, Nebeker J, Samore M, editors. Sophia: An Expedient UMLS Concept Extraction Annotator. AMIA Annual Fall Symposium; 2014; Washington D.C.
12. Chapman WW, Chu D, Dowling JN. ConText: an algorithm for identifying contextual features from clinical text. Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing; Prague, Czech Republic. 1572408: Association for Computational Linguistics; 2007. p. 81-8.

13. Tran L-TT, Divita G, Carter ME, Judd J, Samore MH, Gundlapalli AV. Exploiting the UMLS Metathesaurus for Extracting and Categorizing Concepts Representing Signs and Symptoms to Anatomically Related Organ Systems. Journal of Biomedical Informatics. 2015.
14. Denny JC, Spickard A, 3rd, Johnson KB, Peterson NB, Peterson JF, Miller RA. Evaluation of a method to identify and categorize section headers in clinical documents. Journal of the American Medical Informatics Association : JAMIA. 2009;16(6):806-15.
15. Divita G, Gundlapalli AV, Tran L-TT, Workman TE, Carter M, Palmer M, et al. Extracting Symptoms from the Free Text of VA Electronic Medical Notes using Natural Language Processing International Journal of Medical Informatics. 2015(Under Review).
16. Martin L, Battistelli D, Charnois T. Symptom recognition issue. ACL 2014. 2014:107.
17. Gundlapalli A, Divita G, Forbush T, Redd A, Carter M, Gendrett AJ, et al., editors. 873Using natural language processing on electronic medical notes to detect the presence of an indwelling urinary catheter. Open Forum Infectious Diseases; 2014: Oxford University Press.
18. Gundlapalli AV, Carter ME, Palmer M, Ginter T, Redd A, Pickard S, et al. Using natural language processing on the free text of clinical documents to screen for evidence of homelessness among US veterans. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2013;2013:537-46.
19. Tran L-TT, Divita G, Redd A, Carter ME, Samore M, Gundlapalli AV, editors. Scaling Out and Evaluation of OBSecAn, an Automated Section Annotator for Semi-Structured Clinical Documents, on a Large VA Clinical Corpus. AMIA Annual Symposium Proceedings; 2015: American Medical Informatics Association.
20. Browne AC. SPECIALIST Lexicon 1994. Available from: http://SPECIALIST.nlm.nih.gov.
21. Ogren PV, Bethard S. Building Test Suites for UIMA Copmponents. Proceedings of the Workshop on Software Engineering Testing and Quality Assurance for Natural Language Processing (SETQA-NLP 2009). 2009:1-4.
22. Patterson OV, Ginter T, DuVall SL. Large scale clinical text processing and process optimization.
23. Divita G TQea. CHIR Common Model. 2016.
24. Divita G, Carter M, Redd A, Zeng QT, Gupta K, Trautner B, et al. Scaling-out NLP Pipelines to Process Large Corpora of Clinical Notes. Methods of Information in Medicine. 2015(In Press).
25. Lanthaler M, G□tl C, editors. On using JSON-LD to create evolvable RESTful services. Proceedings of the Third International Workshop on RESTful Design; 2012: ACM.
26. Comeau DC, Islamaj Dogan R, Ciccarese P, Cohen KB, Krallinger M, Leitner F, et al. BioC: a minimalist approach to interoperability for biomedical text processing. Database : the journal of biological databases and curation. 2013;2013:bat064.
27. Redd A, Carter M, Divita G, Shen S, Palmer M, Samore M, et al. Detecting earlier indicators of homelessness in the free text of medical records. Studies in health technology and informatics. 2014;202:153-6.
28. Gundlapalli AV, Redd A, Carter M, Divita G, Shen S, Palmer M, et al. Validating a strategy for psychosocial phenotyping using a large corpus of clinical text. Journal of the American Medical Informatics Association : JAMIA. 2013;20(e2):e355-64.
29. Gundlapalli A, Divita G, Forbush T, Redd A, Carter M, Gendrett A, et al. Using natural language processing on electronic medical notes to detect the presence of an indwelling urinary catheter. Open Forum Infectious Diseases. 2014;1(S1):252.
30. Frank E, Hall M, Trigg L, Holmes G, Witten IH. Data mining in bioinformatics using Weka. Bioinformatics. 2004;20(15):2479-81.
31. Ihaka R, Gentleman R. R: a language for data analysis and graphics. Journal of computational and graphical statistics. 1996;5(3):299-314.
32. Lu CJ, Divita G, Browne AC, editors. Development of Visual Tagging Tool. AMIA 2010 Annual Symposium, Washington, DC; 2010.
33. South BR, Shen S, Leng J, Forbush TB, DuVall SL, Chapman WW, editors. A prototype tool set to support machine-assisted annotation. Proceedings of the 2012 Workshop on Biomedical Natural Language Processing; 2012: Association for Computational Linguistics.