August 2019

Clinical Language Engineering Workbench (CLEW) Technical Report

Contents

The findings and conclusions in this report are those of the authors and do not necessarily represent the official position of the author's agencies (CDC, FDA). The authors have no conflicts of interest related to this work to disclose.

CLEW Team Members

Team Member Organizations	Team Members
Center for Biologics Evaluation and Researcher Food and Drug Administration	Taxiarchis Botsis – Project Co-Lead Mark Walderhaug – Project Co-Lead Kory Kreimeyer Abhishek Pandey Matthew Foster Richard A. Forshee
Cancer Surveillance Branch Division of Cancer Prevention and Control Centers for Disease Control and Prevention	Sandy Jones – Project Co-Lead Joe Rogers Wendy Blumenthal Temitope Alimi
Northrop Grumman	Steve Campbell – Project Manager Fred Sieling – Project Manager Marcelo Caldas Sanjeev Baral
Health Language Analytics Global (sub-contract with Northrop Grumman)	Jon Patrick
Engility Corporation	Wei Chen Guangfan Zhang Wei Wang
Vassar College (sub-contract with Northrop Grumman)	Keith Suderman

List of Abbreviations

AE .....Adverse event
AI .....Artificial intelligence
API .....Application programming interface
CDC .....Centers for Disease Control and Prevention
CDE.....Common data element
CLEF.....Conference and Labs of the Evaluation Forum
CLEW .....Clinical Language Engineering Workbench
EHR.....Electronic health record
ES.....Environmental scan
ETHER.....Event-based Text-mining of Health Electronic Records
eMaRC Plus .....Electronic Mapping, Reporting, and Coding Plus
FAERS .....FDA Adverse Event Reporting System
FDA.....Food and Drug Administration
GUI .....Graphical user interface
HL7 .....Health Level Seven
HTML .....Hypertext Markup Language
i2b2 .....Informatics for Integrating Biology & the Bedside
IAA .....Inter-annotator agreement
JSON.....JavaScript object notation
MedDRA.....Medical Dictionary for Regulatory Activities
NLE.....Natural language engineering
NLP .....Natural language processing
PCOR…….Patient-Centered Outcomes Research
PCORnet .....National Patient-Centered Clinical Research Network
PCORTF…….Patient-Centered Outcomes Research Trust Fund
SemEval .....Semantic evaluation
SER .....Semantic entity recognition
ShARe.....Sharing Annotated Resources
SNLP.....Statistical natural language processing
UIMA.....Unstructured information management application
UMLS .....Unified Medical Language System
VAERS .....Vaccine Adverse Event Reporting System
XML.....eXtensible Markup Language

Overview of Report

This report is the fourth deliverable in the *Development of a Natural Language Processing (NLP) Web Service for Structuring and Standardizing Unstructured Clinical Information*, a collaborative project between the Centers for Disease Control and Prevention (CDC) and the Food and Drug Administration (FDA). The Workbench (CLEW) will provide federal agencies, public health agencies, Patient-Centered Clinical Research Network (PCORnet) participants, and others with access to advanced NLP tools and pipelines with capacity to assemble their own pipelines. The pilot applications developed using these tools convert unstructured clinical information such as cancer and safety surveillance data into structured and standardized coded data.

This report presents a detailed technical description of the core NLP approach of the prototype version of the Workbench and two pilot applications developed using the Workbench. In doing so, the core motivating use cases for NLP across a range of clinical domains were explored. The tools and constituent components that are available as part of the prototype are described.

The intrinsic versatility of our approach revolves around the ability to develop customized processing pipelines from unique compositions of well-developed containerized services. This can enable software engineers to establish the best NLP pipeline for a given task. In developing a generalized framework for pipeline, generation based on applications, our methodology moves from the research activities of NLP performed by computational linguistics to the functional requirements of large engineering practice. To reflect a more generalized and versatile framework, it is best considered a Clinical Language Engineering Workbench (CLEW). Refer to section 3 of this document for a detailed review of NLP to natural language engineering (NLE).

1 Project Objectives

The primary objective of this project is to provide a mechanism to "translate" free text data into a structured form that enables the harnessing of clinical prose for surveillance and research purposes. Patient-Centered Outcomes Research (PCOR) researchers, federal agencies, and public health agencies will have access to NLP tools and shared services to translate clinical textual information into standardized formats to enable the use of classical data analytics methods. The following objectives were set in the statement of work for this project:

- Conduct an environmental scan. The project reviewed published and existing NLP architectures, NLP methods, NLP tools, common data elements (CDEs), and SDC activities that are in use by federal agencies, public health agencies, academic centers, commercial vendors, and PCORnet that are freely available for inclusion on the CLEW Web Services. The Workbench will provide an environment to support and encourage agencies to build and provide NLP services that return standardized data needed for analytics or other work processes, such as the CDC cancer surveillance and FDA safety surveillance domains.
- Design the pilot CLEW web services and, pending Office of the Secretary (OS) Patient-Centered Outcomes Research Trust Fund (PCORTF) Resource Center design review and the Assistant Secretary for Planning and Evaluation (ASPE) approval, proceed with the development of the pilot version.
- Design and construct a language engineering environment that serves the needs of three communities of users:
  - Common users who want a simple mechanism to investigate a small number of documents.
  - Information technology developers who want to use a service to automate operational data processing.
  - NLP expert users who want to create new services by experimenting with the configuration of an NLP pipeline to optimize its performance.
- Include NLP functional modules and a mechanism for linking them together into a coherent pipeline that enables design and testing of any combination of modules to create usable services.
- Provide a mechanism for deploying a developed service onto the CLEW so that it is available for use by any appropriate public authority.
- Provide a catalogue of publicly available services whether they installed in the CLEW itself or on a third party site.
- Pilot the CLEW web services on CDC's Innovation Research and Development Laboratory Environment.
- Evaluate the effectiveness of the CLEW for building NLP services.

- Evaluate the performance of two pilot web services built for processing cancer pathology reports and safety surveillance data.
- Provide directions and illustrations for third parties to create their own services by developing pilot applications involving:
  - Annotation of corpora of sample reports using text datasets drawn from CDC and FDA resources to form a gold standard of target content. This can inform the learning process and assess the learning capabilities of the developed demonstration services.
  - Building of functional NLP pipelines using the annotated corpora.
  - Conversion of the pipelines into services and deploy them in the CLEW.
  - Documentation of the complete processes used to create the services as an illustration of end-to-end language engineering (LE).
- Release the system for use by federal agencies, public health agencies, and PCORnet participants.

2 Background and Significance of Pilot User Systems

In the United States, central cancer registries collect, manage, and analyze longitudinal data about cancer cases and deaths. These data are collected from multiple sources such as hospitals, laboratories, physician offices, and independent diagnostic and treatment centers. While there have been strides through Meaningful Use and other activities to implement standardized electronic health record (EHR) systems, parts of the medical record, laboratory reports, and other clinical reports still use free-form narratives. Information in this format cannot be used in any form of statistical analysis. Similarly, a considerable amount of clinical information submitted to FDA's spontaneous reporting systems is either not coded at all or not linked to the appropriate Medical Dictionary for Regulatory Activities (MedDRA) terms.

The 2009 American Recovery and Reinvestment Act included multiple measures to modernize our nation's infrastructure, one of which is the Health Information Technology for Economic and Clinical Health (HITECH) Act. The HITECH Act supports the concept of EHR Meaningful Use, an effort led by the Centers for Medicare & Medicaid Services and the Office of the National Coordinator for Health IT. HITECH proposed the Meaningful Use of interoperable EHRs throughout the United States' health care delivery system as a critical national goal. Because of these initiatives, the clinical care community has moved to EHR systems that utilize standardized common data elements (CDEs). Despite this move, a significant amount of clinical information in the EHR continues to be unstructured textual data, which limits both the primary and secondary usage of the contents of these reports.

Hospital reporting of cancer cases has been highly standardized over the past decade but as patient care has expanded beyond the hospital setting, central cancer registries have had to incorporate data from non-standard systems containing large amounts of unstructured data. Over the past decade, CDC has worked closely with the College of American Pathologists to develop

synoptic reports for standardized reporting of pathology and cancer biomarker data. The use of these templates would standardize the majority of cancer data that pathology and genetic laboratories report to central cancer registries. However, an incentive does not currently exist for pathology laboratories to change their laboratory information systems to use these templates; thus, the anatomic pathology and cancer biomarker data reported to central cancer registries continues to be free-form narrative reports.

The data contained in these unstructured pathology reports, biomarker reports, and physician EHR reports are critical for maintaining a complete cancer surveillance system for analysis of population health and adequate patient care. However, the process of abstracting cancer data is labor intensive and expensive, requiring human intervention to abstract critical pieces of information such as histology, primary site, behavior, grade, laterality, and more. The use of NLP tools could reduce the resources needed for analyzing unstructured data and increase the completeness, timeliness, and accuracy of cancer surveillance data, as well as offer potential of reaping a much more substantial variety of data from these reports.

The FDA Adverse Event Reporting System (FAERS) and Vaccine Adverse Event Reporting System (VAERS) are spontaneous reporting systems in which pharmaceutical manufacturers, medical practitioners, patients, and their representatives submit data regarding the safety of drugs, vaccines, and biologics. The reported information supports important surveillance tasks such as the examination of safety concerns related to marketed products, the evaluation of manufacturer compliance to reporting regulations, and multiple research activities both within and outside FDA. The adverse event (AE) description in VAERS and FAERS data are coded as MedDRA codes. In both systems, a considerable amount of clinical information in the AE narrative is either not coded at all, such as the medical and family history, or not linked to the MedDRA codes, such as the date of onset. In particular, the exact time information for each event (and the code(s) that represent it) is not fully captured from the AE narrative and stored in a structured VAERS or FAERS data field. Both systems also often have multiple reports for the same event, which can affect surveillance. Manual review of each report is often the only way to trace these duplicates. However, after unstructured fields have been transformed into structured data, detection and accounting for duplicate data samples becomes trivial. It is therefore important to develop services for getting the most from the VAERS and FAERS narratives, structuring the unstructured data, improving its quality, and better supporting data requests from the research community.

This CDC and FDA collaborative project can result in the development of a Language Engineering (LE) workbench to enable any public authority team to assemble an NLP service for their own needs that accepts and processes unstructured textual information and returns standardized semantic entities of their own definition. The CLEW reports a variety of clinical terminologies as an added tool for service developers. This project completed two pilot implementations on the CLEW implemented by CDC using cancer data, and by FDA using surveillance data for blood products and vaccines. Guidance is provided to other federal

agencies, public health agencies, and PCOR Institute/Clinical Effectiveness Research participants on how to include their domain-specific terminologies and coding rules, such as the National Library of Medicine MetaMap tool for mapping text to terminologies in the Unified Medical Language System (UMLS). The CLEW is open-source with a modular programming approach that is implemented to allow flexibility for future expansion or modification of its NLP functionalities.

The pilot projects also adapted the existing NLP functionalities of FDA's Event-based Text-mining of Health Electronic Records (ETHER) system for clinical text, and CDC's Electronic Mapping, Reporting, and Coding (eMaRC) Plus text mining functionality to operate as services on the Workbench. A requirements gathering process was completed prior to development of the architectural design of the Workbench's NLP components to ensure that a broad range of needs are included for any future users of the web service.

3 Introduction to Natural Language Engineering (NLE)

The construction of CLEW is to enable the "productization" of text mining and statistical NLP (SNLP) methods to be developed readily and distributed as web services, in support of the broad range of health organizations encompassed by public administration.

This section:

- Discusses the justification for taking a LE strategy for completing the project.
- Positions the various types of NLP functions and their contributions to analyze text successfully.
- Discusses the various implementation directions a practical solution could take.
- Details the steps required to assemble a team of appropriate professionals to build the pipeline using the Workbench and commission it into service.
- Defines the strategy we have adopted to implement the CLEW.

Researchers have addressed NLP since computing was invented. Initially, it was identified as a type of artificial intelligence, but soon emerged as a separate discipline known as computational linguistics. In its first 30 years, computational linguistics focused on converting linguistics, particularly grammar, into algorithms. Since the 1990s, the role of linguistics has taken a back seat to the methods of computer scientists, particularly machine learning algorithms. The term SNLP encompasses these methods that form the foundation of much of the research in the field and a backdrop for the CLEW.

The development of Internet search engines led to another line of language computational development. The field of information retrieval, better known as text mining, ignores linguistics and focuses on the statistical distribution of independent words within texts. In a recent development, the surrounding context of words as well as the word itself are considered.

Text mining methods are fast at processing millions of documents to give a relatively coarse representation of their macroscopic lexical characteristics. The advance of text mining has led to the general belief that a fully generalizable NLP technology can be built. This is not the case. The success of general search engines obscures the difficulty of creating high-precision NLP systems where speed and volume of documents hide the lack of semantic precision in the results they produce.

Computational linguistics is a lengthier process targeted at representing the detailed semantics of texts. It takes longer to develop applications and process documents, but gives more detail and captures the subtleties of the text. More time and effort is required to attain higher accuracy.

The CLEW is founded on the principles in computational linguistics because the crude methods of text mining are not normally sufficient to achieve a level of accuracy that is productive and safe for carrying out the critical research and analytics of the health and PCORI communities. Therefore, use of the CLEW to build NLP pipelines requires assembling a team with the competencies to carry out a complex project over a significant period of time. The CLEW provides some of the technology and a solid framework with which to design and plan a sophisticated language-processing objective better and more easily. However, it is not a method for manufacturing a quick and easy text mining application. It will be an environment that supports nascent NLP aspirations for many health organizations and researchers by providing working services either for experimentation or to perform routine operational processing for data analytics objectives.

With the emergence of computational linguistics into a widespread discipline of interest, particularly in the European Union, an effort to systematize the development of NLP applications, rather than just do research in it, has arisen. This work is now identifiable as natural language engineering (NLE) and best represented by the General Architecture for Text Engineering (GATE) project. The CLEW is an effort to introduce LE into the broader health and PCORI communities, following earlier research efforts best represented by the various participants in the AMIA NLP User Group, the i2b2 NLP Challenges and the release of open source software such as cTAKES, as well as the precedence established by GATE.

For the purposes of this report, LE is defined as the systematic process of using NLP and SNLP methods to deliver useful processing outcomes for a defined function. While NLP is classically a pipeline of processors that produce some particular content identification, LE aims to systematize the process of creating NLP pipelines and using their output for user defined value propositions. It is a production line for SNLP and NLP based use case applications.

The advantages of conceptualizing this project as a CLEW are:

- It is the best position for fitting the project into ASPE’s generalization objectives.
- It defines the project for research activities by pushing it into a functional activity for the broader health community.

- It identifies the project as a distinct activity separate from the work of the many research groups engaged in the field.
- It provides a platform and philosophical pillar to separate the project from commercial artificial intelligence (AI) operators who are building applications using text mining approaches by asserting the separation of NLP and SNLP is a language engineering choice.
- It demonstrates the complimentary roles of text mining NLP and SNLP in the pilot solutions we provide in eMaRC Plus and ETHER.

4 Building an NLE Service Application

4.1 The Importance of Accuracy

A core understanding for creating and using an NLP pipeline is the importance of accuracy and the effects of errors. As information is generated throughout a pipeline of processes, errors at earlier stages have a multiplicative effect downstream. Hence, it is vitally important that even small processing weaknesses are identified and eliminated in every stage of processing. This requires attention to the minutiae of both processing algorithms and the texts to which they are applied. In the health and PCORI contexts, accuracy is paramount due to the effect errors can have on peoples' lives.

Many applications are touted in the public arena as NLP, where text is processed in simple ways such as searching for words and phrases using string-matching algorithms. These methods are easy to implement and effective in many contexts, such as web searches. However, other applications, particularly health applications, need higher accuracy and serious consideration of the target concept's semantic context. For example, it is important to understand the distinction between a diagnosis that is confirmed, negated, or compared to another diagnosis. This processing requires a sophisticated solution that captures semantics with high accuracy. The full power of computational linguistics are applied; text mining methods rarely suffice.

The CLEW is an environment where different techniques can be harnessed for resolving specific use cases. Accuracy is attained by choosing and applying those tools thoughtfully, not by the convenience of using the Workbench. In many settings, no services are available to perform the NLP functions required for a particular task. The Workbench will make it easier to build that functionality.

4.2 The NLE Pipeline Architecture

The process of building an NLP application requires consideration of many variables. A linear pipeline is essential and there is a small selection of publicly available methodologies that parallel this approach. The pipeline's core processes include:

- Tokenization.
- Word recognition.

- Sentence boundary detection (SBD).
- Semantic entity recognition (SER).

Other processes may contribute to a given application. Project objectives will determine these enhancements.

Tokenization is the process of identifying each of the objects, known as *tokens,* between whitespace. This string is most often thought of as a word, but many tokens are not words, such as numbers, abbreviations, and acronyms. A tokenizer separates words from punctuation, which is treated as a token. This task becomes complicated with the use of punctuation to indicate both the beginning and end of pertinent sections of a document, especially in headings, where non-alphanumeric characteristics can be used for highlighting. In scientific settings, such as health, tokenization is complicated by tokens that combine letters, digits, and other characters, and where capitalization of letters adds meaning.

Word recognition is the task of identifying an alphanumeric string as a known word. It also includes the task of collecting other attributes of that word for semantic entity recognition processing downstream. A word is recognized by comparing it to a predefined lexicon of words with attached attributes. Common attributes include its dictionary representation, known as a *lemma*, its notional part of speech, and any appropriate semantic categories such as body part, disease, or medication.

The *orthography* or case of a word can also be important in word recognition. For example, the string “all” would have the common English meaning, while the string “ALL” would mean <acute lymphocytic leukemia> in a cancer context, and possibly something entirely different in another context. (Note: Quotation marks indicate strings and angle brackets indicate semantic meaning.) In downstream processing, the proper recognition of <ALL> would convey the knowledge in the lexicon that it is a member of the semantic class <cancer> with a sub-type of <lymphatic cancer>.

The word recognition processor can also perform spelling correction. Missing or extra white space, letter omission or addition, and misordering of letters can cause spelling errors. The ability to correct misspellings either automatically or by offering a selection to a user is severely limited because of the computational cost of computing all possible corrections. Most applications can only offer rectification of up to two letter errors and hardly ever encompass whitespace errors. Spelling correction might also be provided as a post processing to word recognition acting as a separate module in the pipeline.

Sentence boundary detection, also known as sentence splitting, is the process of recognizing the beginning and end of sentences. We are all used to sentences being defined by a full stop, whitespace, and new word beginning with an upper-case letter. While this is the most common method, it is not unique. Sentences can be terminated by the newline character “\n” or by a colon

":" or semicolon ";". In text that is massaged by electronic processing, normal sentence markers can be lost altogether, especially if all the text is in upper case. While it is possible to obtain high accuracy in sentence boundary detection, a small amount of uncertainty can remain. These errors can lead to serious misrepresentations of meaning. For example, the text

"- invasive ductal carcinoma – high grade intraepithelial hyperplasia" leads to the <ductal carcinoma> being assigned the value of <high grade> if the sentence boundary is not properly identified.

Semantic entity recognition (SER) identifies entities of interest to the NLP use case. SER is different from the more commonly understood named entity recognition (NER), which is the task of recognizing proper names in texts and usually divides into the groups of persons, organizations, locations, and other names. In health settings, NER is valuable, but SER is the dominant need.

SER is the single largest computational function in any health NLE application. SER can be implemented in many ways, from simple methods to elaborate machine learning methods. In some instances, a simple method may reach the desired accuracy needed for the purpose. However, a more elaborate method may be required to reach the highest rate of accuracy that is achievable.

4.3 Rule-Based Modeling

Gazetteers: SER begins with a gazetteer of words and phrases relevant to the professional community of practice for which the application is being developed. This is in addition to a general language lexicon of the host community (English in our case). The gazetteer can be acquired from public sources or manually developed by the project team. In the latter case, a systematic method of acquisition is applied using software to annotate the source texts so there can be re-investigation of the origins of the terminology. Importantly, the gazetteer contains coherent noun phrases without verbs. Recording more information about the phrases and storing them all appropriately can achieve the extension of the gazetteer into a lexical database. The added information may be recognized acronyms, abbreviations, semantic categories, and synonym phrases.

The gazetteer list method tends to be efficient for data collection and can be applied to target texts using computationally simple methods such as regular expressions. The methods are precise in that they find all the examples stored in the gazetteer. However, their major drawback is that the gazetteer may not find anything to fit the stored form exactly. Some variation can be achieved by using Regular Expressions with wildcard items but this substantially increases search cost, is liable to produce many false positives, and is not a systematic approach to solving the general SER problem.

Regular expressions: This is a mechanism for searching for literal strings in a text. Searching for literal text strings is effective at finding all known examples. However, it doesn't enable

finding similar strings that have inconsequential differences such as the difference between a singular or plural form of a noun. It does incorporate a number of devices to generalize a string search to make it more expansive, so wildcard elements can be inserted. Regular expressions overcome the problem of slight variation in word morphology as an expression could use a wildcard as the last character so singular or plural forms would be found. The difficulty with regular expressions is they quickly overgeneralize leading to a surfeit of false positive results. Furthermore, they cannot find anything that does not fit the deterministic definition of the regular expression, so there is little idea of the false negatives in a search.

4.4 Statistical NLP (SNLP) Modeling

The most advanced method for SER is the approach of SNLP. This method uses machine learning algorithms of the type known as *supervised classifiers,* which use a set of data objects with known classes for training a classifier to predict the classes of data objects (words and phrases) in unseen texts. A data object is an object with many defined attributes and the values of those attributes collected. Once the set of objects are compiled and their classes assigned they constitute the "training set", so a statistical computational model of how to classify the data objects can be trained using Machine Learning algorithm. In the case of NLP, the data objects are each token in a collection of texts and the attributes known about those tokens, such as their semantic class, their neighboring words and the semantic class of those neighbors. The model built using these attributes is the <language model> in NLP settings.

The attributes are compiled by passing a text through a series of processing modules forming a pipeline, each of which computes defined characteristics of the tokens in the text. A huge number of attributes can be compiled about each token, and part of the skill of building an efficient application is to discover quickly and efficiently the best attributes that produce the most accurate results. Accuracy is estimated by holding out 10% of the training set and applying the trained model to them to compute their tokens' classes. This is called the *test set.* The differences between the gold standard classes of the tokens and their model classes is a measure of the accuracy of the model. This process is repeated 10 times on 10 different segmentations of the corpus and the results of the 10 tests averaged to arrive at an estimate of the mean accuracy. It also acts as a guide as to what needs to be improved in the model by either adding extra features or reassessing the upstream processing modules to provide different attributes for the tokens. The investigation into the optimal model for the application consists of iteratively testing one attribute set for its accuracy and then changing the attribute set to overcome the deficiencies of the current attribute set.

Identifying the classes that need to be recognized in a given application can also be an intellectual challenge. In the eMaRC Plus pilot application the user requirement is to extract from cancer pathology reports five basic features of the clinical case: {site of the cancer, its histology, behavior, grade, laterality}. These are the core classes in the application. For the

training process to continue, each token in the training documents must be assigned values of one of those 5 semantic classes; otherwise it is assigned to other, that is, unclassified.

However, this simple representation of the use case may not produce an optimal result. Cancer can spread, so the investigation site may be different from the original disease site. Where this information is provided in the report it requires another class, that is, <Origin of Cancer Primary>, so the class set is larger than the target list of attributes. In many use cases, the number of classes that need to be computed may be many more than the core content required for the primary use case defined.

All NLP methods can be considered methods for generating analytics. They are not normally the end of processing, but rather a step to obtain something of higher value. The end use of the NLP outputs is key to deciding the type of a language model to build.

Expanding the feature set: The attribute set in machine learning is known as the “feature set” in NLP circles. The optimal feature set could be explored so that the combination of classes and source texts are best matched. There are well-known heuristics for a starting feature set, such as the neighboring tokens to the left and right of the current target token, their parts of speech, and their grammatical role. The latter two features are available only if processors assign to parts of speech and a grammatical roles to tokens earlier in the pipeline. The features used in the language model are determined by the pre-processors in the pipeline that assign feature values to each token. These pre-processors differentiate NLP from text mining techniques, and allow for higher accuracy and richer semantic interpretation of the texts.

In clinical texts, some of the classes of most interest are diseases, body sites, medications, procedures, social history, and clinical events. A common gazetteer source that can supply more features is the Metathesaurus provided by the National Library of Medicine. Some other features used are MESH terms, SNOMED CT categories, and time categories.

Supervised classifiers: Many classifiers are available from the machine learning open source community. The most commonly used in NLP are Support Vector Machines and Conditional Random Fields. The CLEW does not incorporate any machine learning functions and requires service developers to use their own resources to build their training sets and language models. Given the rapid advances being made in the machine learning community, the CLEW allows developers to write custom modules to incorporate application-specific machine learning methods if desired. The Workbench provides a catalogue of NLP processors suitable for use in an NLE pipeline, and the service platform on which the final service can be installed for public use.

Preparing a training set: Select and prepare training set meticulously. First, choose a set of documents that precisely represents the use case under development. The documents can represent all of the variables with the linguistic variety a real-world data set has. Then define the primary class set and the operational class set. This set of semantic classes forms the tag set by

which the texts need to be annotated, as the annotations represent the training set of examples the algorithm learns from. The annotations are applied manually using annotation software that can be selected from the Workbench or other sources that have data formats compatible with the Workbench formats.

The annotations represent the gold standard from which the language model is constructed and later applied back to, so as to assess its accuracy. The pre-processing pipeline computes the features of all of the tokens, and when brought together with the annotations, which define the classes of the tokens, constitutes the training data for the classifier algorithm.

The annotation process can be problematic, as it requires humans to learn the meanings of the semantic classes and apply them consistently. Early iterations of the language modeling are as much about identifying inconsistencies in the manual annotation as about modeling the classification task.

The final pipeline: The pipeline is complete when the language modeling operates at sufficient accuracy to warrant installation of the service. The development team decides when it has exhausted its efforts to gain more accuracy through statistical methods and to use post-processing rules to polish results containing idiosyncratic language not amenable to statistical analysis.

The executable service is compiled together as a pipeline sequence of processing that prepares all of the tokens in a text for their features, submits the tokens with features for classification, and receives the outputs. That content and the format for delivery depend on the primary use case.

5 Specifications for the CLEW

Based on the discussion in the above section on NLP, the CLEW provides the following functionality:

- Pilot web services that compute defined functions for, minimally, cancer pathology and safety reports.
- Identification of candidate service consumers and providers.
- Documentation for conducting the stages of the language engineering task sufficient to use the web development environment.
- Documentation on how to set up a client and service to use the Workbench.
- Documentation for creating the pipelines using the services in the web environment.
- Documentation for conducting the stages of the language engineering task sufficient to assemble and test a new application and install for operation.
- Demonstration on how to assemble and test these components using the Language Application Grid (LAPPS) platform.
- Open-source code modules that would be constituents in a statistical NLE application,

including:
  - Annotator tool.
  - Corpus management tool.
  - Core NLE processors: Tokenizer, word recognition, sentence boundary recognizer, and semantic entity recognizer.
  - Machine learning algorithm.
- An environment that enables development of an NLP pipeline in three stages:
  - Building a gold standard corpus as the source of training data to create the service.
  - Building an NLP pipeline.
  - Training the language model as an off-Workbench task.
- An environment that enables the development of an NLP pipeline by combining the existing services.

6 A Practical Way Forward

The range of conceptual solutions have been presented above to set the scene for describing a practical pathway forward. We now define in more detail the nature of the total production process, what the CLEW can provide, and what developers would provide.

6.1 The Architecture of a Language Engineering Production Line Incorporating SNLP Methods

a. Compose a training set of data:

- Define the computational objectives of the endpoint processing.
- Define the semantic entity set for extraction.
- Compile a representative set of documents
- Annotate the documents with the semantic entity tags
- Revise the annotations based on the accuracies of the language modelling

b. Build an SNLP or NLP processing pipeline:

- Choose the processors for inclusion in the pipeline that provides suitable features for identifying semantic entity classes (tags)
- Join the pipeline processors so they can pass outputs of one as inputs to the next
- Generate a set of features at the end of the pipeline for input into the Machine Learning algorithm

c. Build a language model from the training set:

- Select a Machine Learning algorithm that matches the classification task
- Accept the input into the Machine Learning algorithm
- Compute the language model using the selected algorithm

- Evaluate the accuracy of the computed language model
- Return to the annotation process and correct identified missing or incorrect annotations.
- Repeat the evaluation and language model generation iteratively until the error model is stable and sufficiently small.

d. Assemble the SNLP pipeline and the final language model as a service in a production environment.

6.2 Using the Service

All services provided by the Workbench are provided as-is. All data submitted to any service are free from personally identifiable information (PII) or any sensitive data. The services are mostly stateless and transient, where no data is stored on the servers except for some basic logging functionality. Other third-party tools (for example LAPPs/Galaxy) may require users to authenticate and submit data that are be stored on such servers. This is beyond the scope of the Workbench. The Workbench itself does not require authentication, since all of the services and information are publicly available. If more security is needed, download the needed projects to a separate environment and secure it as appropriate.

7 CLEW Architectural Design

7.1 Leveraging the Environmental Scan

In the NLP environmental scan (ES), we combined the results from a systematic literature review with a comprehensive multi-channel review covering researchers and institutions, NLP challenges, and government activities. We selected the eligible tools on the basis of availability (a tool is open-source, downloadable, and source code exists) and relevance (a tool supports the processing of clinical texts, such as the ones included in adverse event or pathology reports; it generates standardized and/or coded output; or is equipped with advanced capabilities). These tools were subsequently categorized into complete systems, applications, and NLP components and further evaluated on three aspects (development activity, popularity, and framework used) representing the importance and applicability of the tools (See Appendix D).

The development of the pilot version of the CLEW was guided by the ES findings and the pilot use cases, with initial focus on selected tasks around the processing of safety surveillance and cancer data. The CLEW prototype contains tools identified in the ES, identified by stakeholders, and/or revealed during our ongoing monitoring of the NLP community.

7.2 Users - Key Competencies Assumed Knowledge

The CLEW provides a wide range of functionality and flexibility depending on the end-users capabilities and expertise with using NLP and Machine Learning techniques. CLEW supports three roles: the non-NLP expert user, the information technology (IT) developer, and the NLP expert user, described below.

- Non-NLP expert: These users have a high-level conceptual understanding of NLP that can interact with existing services offered by the Workbench.
- IT developer: These users have detailed technical knowledge to interface programmatically with pre-defined shared services on the CLEW, submit data to a service, and process expected outputs back to the local application on behalf of its users.
- NLP expert: These users can build NLP solutions and pipelines. They can have the following competencies to build an effective NLP application:
  - Computation linguistics.
  - Linguistics for semantic class design and annotation.
  - Clinical knowledge related to the specific domain being addressed.
  - Software engineering expertise in pipeline development and application service creation and installation.
  - Machine learning methods.

7.3 The Architecture

The CLEW is establishing a framework that enables the use and sharing of NLP tools, modules, and methods to develop, evaluate, and deploy web services that support domain-specific use cases. See Figure 1 below for a high-level conceptual architectural design of the CLEW. Components of the LAPPS project, funded by the U.S. National Science Foundation, were leveraged in development of the CLEW architecture. The LAPPS Grid project has made significant progress towards interoperability of NLP tools and data, as well as creating a comprehensive network of web services and resources within the general NLP community[1]. LAPPS has not yet been used for any clinical applications, and has not been proven that it has the functionality to support development of clinical NLP use cases.

The CLEW is a service-oriented architecture and supports Simple Object Access Protocol (SOAP) and REpresentational State Transfer (RESTful) services to utilize various NLP tools and modules to engineer clinical NLP pipelines. The clinical NLP pipeline and associated services can be catalogued on the Workbench for use as a public service, or, if desired, downloaded and implemented within the user's local environment.

It is important that users of the CLEW understand the impact errors in NLP processing can have on the output. As information is generated throughout a pipeline of processes, errors at the earlier stages have a multiplicative effect downstream. Therefore, it is important that even small processing weaknesses are identified and eliminated in every stage of processing. This requires attention to the minutiae of both processing algorithms and the text the algorithms are applied to. Tremendous care and precision is required to build a successful application. There are many different understandings of NLP in the public arena and each results with outcomes of varying quality. Prior to implementing any NLP processing, the user assesses what quality is adequate for the work they need to complete. This assists the user in determining the best approach to take in developing an NLP pipeline that is appropriate to complete the task.

The CLEW supports multiple combinations of the existing components as well as generation of more than one pipeline for a use case.

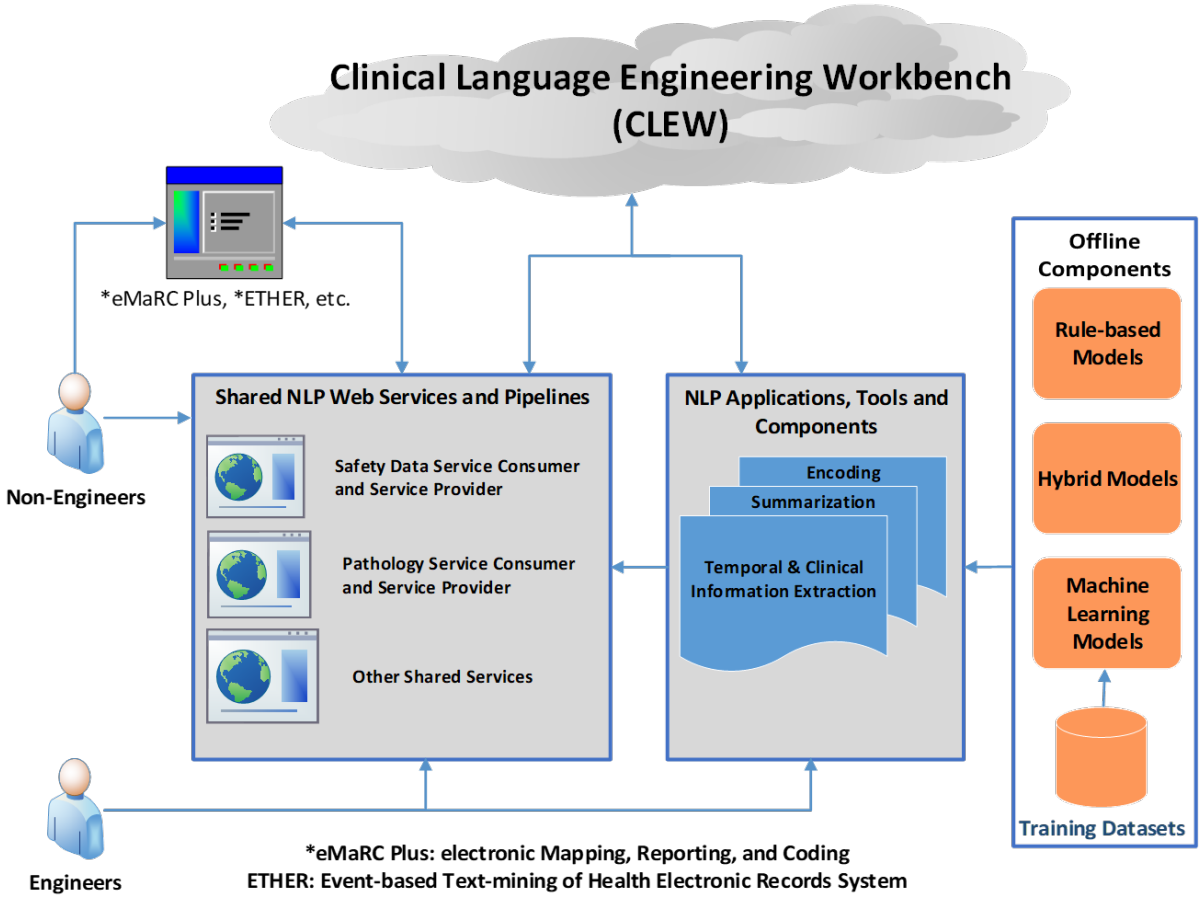


Figure 1. High-level architecture design of the CLEW.

7.4 Security and Protection of Personally Identifiable Information (PII)

The user is responsible for ensuring that no personally identifiable information (PII) is uploaded to the CLEW even if limited de-identification tools may be provided. Warning messages are included at key decision points to ensure the data do not include PII.

If there is a concern that PII has not been removed, download and implement the services in a local environment.

7.5 Governance and Administrative Oversight

The CLEW is intended to provide a clearinghouse of NLP tools and services that can be used and shared among researchers and PCORnet participants. The CLEW prototype is developed with a minimal set of tools and services, with the intent that other NLP experts request to include their specific tools, pipelines, and services on the CLEW.

In order to maintain a high level of quality and organization of the CLEW, it is necessary to create a formal governance that oversees the review and inclusion of tools and services. This

governance could include participants from government, academia, and industry to ensure that oversight is handled appropriately.

Determine issues around the provision of data for public display and reuse before the Workbench goes live. Make data contributors aware of the manner in which data are used and complete MOUs and DUAs.

A detailed process documents how NLP researchers can contribute their tools and services to the CLEW.

7.6 Tools Catalogue

The ES identified 54 existing open-source tools. Links to a range of these tools are provided in the CLEW Tools Catalogue (see Figure 2). Users can read about each tool and download the tools for use in their own local environment. Refer to Appendix D for the full ES list.

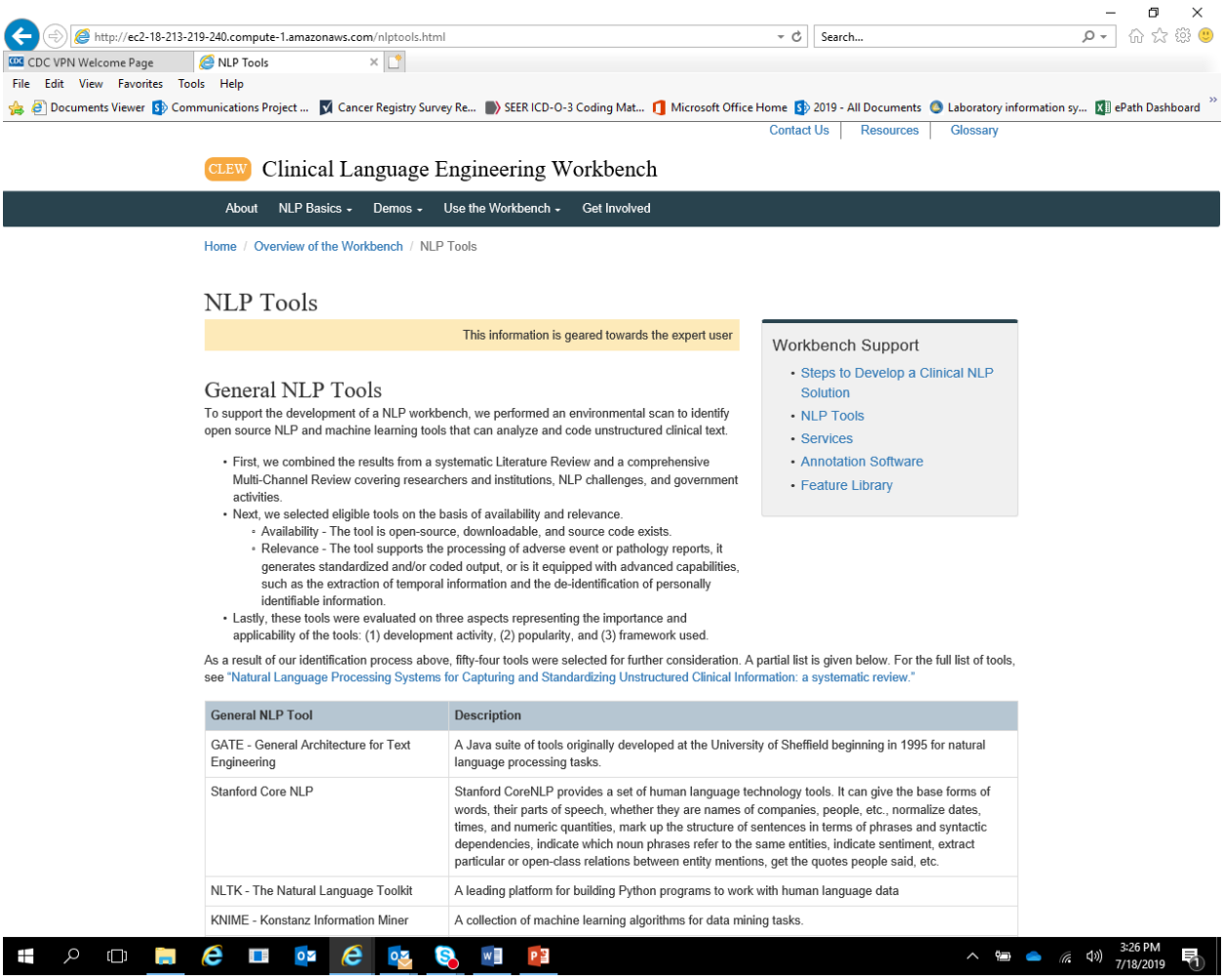


Figure 2. Screen shot of the CLEW NLP Tools Catalogue.

7.7 NLP Services Catalogue

CDC and FDA have developed NLP services using tools or certain modules from the tools identified in the ES to address the use cases described in sections above. NLP services have been combined in pre-defined ways that are wrapped into application programming interface (APIs) to achieve maximum interoperability and portability. The APIs can be used by:

- IT developers in local clinical applications.
- The CLEW to demonstrate functionality.

Apart from the pre-defined pipelines, the users can use the CLEW to combine the NLP services and generate pipelines for their use cases on the fly.

The NLP Services Catalogue maintains a list of shared services for utilization by any user to analyze the specific type of data of the service. New services can be added to the CLEW Services Catalogue for use by others. The new services may support more combinations and pipelines.

7.8 NLP Feature Library

To train a machine learning model, identify natural characteristics, or features, for each token. These features include, for example, word length, part of speech, and relative location in the sentence.

With the feature library, one can readily identify a set of features for each token in a training corpus, a manually annotated set of documents. These identified features and documents are then input into the pipeline. Next, analyze the pipeline output. If the pipeline performed accurately, the model is trained and the pipeline is ready for production. Any taggings of the corpus that do not match the manual annotations in the gold standard represent a disparity in the process. Go back and adjust the features and repeat the process.

The CLEW has built a Feature Library software by which features can be invoked to be used in the development of a language model. It is primed with features commonly used in SNLP applications, but is expected to be enhanced over time by new users who develop applications that require specialized feature sets. Having these feature generating applications in one location can increase the resources available to the language engineering community.

7.8.1 How to Use the Feature Library

The Feature Library can be used inside LAPPS for a single file, or as an external service for a folder of files.

Inside LAPPS: The Feature Library is a service in LAPPS that can be inserted at the end of an NLP pipeline the user has assembled. After inserting the Feature Library in the pipeline, when a file is submitted for processing, the BIO file is generated at the completion of all the processing. It can then be downloaded to the user’s desktop using the LAPPS download function.

For bulk file processing: When a folder of files is processed, the Feature Library can be accessed via the CLEW. Below are steps for using the Feature Library for bulk file processing:

1. Open the feature library to use the Feature Library for bulk file processing.
2. Select required parameter values.
3. Use the LAPPS upload function.
4. The CLEW then accesses the files and process them through the Feature Library and produce an output BIO file that is stored in the users Downloads folder.
5. Users can then directly input the BIO file into their preferred machine learning algorithm.
6. The output is a language model that is then available for tagging files.

7.8.2 Feature Library Example

To train a machine learning model for the safety surveillance data, the following types of features for each token in the text were identified and implemented:

- Word length: the length of the word.
- Part of speech (POS): grammatical tagging or word-category disambiguation.
- Features extracted from MetaMapLite, such as semantic types.
- Features extracted from NCBO web service, such as preferred terms.
- Features extracted from ETHER lexicon (dictionary mapping in the lexicon).
- Features of adjacent tokens: POS/pattern/Stem features of tokens before/after the current token.
- Heuristic features extracted based on ETHER rules. As an example, a token of interest next to Dx/Diagnosis is treated as Primary Diagnosis, while a token of interest next to “cause of death” or “COD” is a good indicator of cause of death.
- Word stem: the stem of the word.
- Pattern extracted based on the Unicode categories of each of the character in the text (see http://www.unicode.org/reports/tr49/tr49-2.html).
- Relative location in the sentence.

Among these features, we selected and evaluated the performance of different combinations of categories.

7.9 Data Exchange Formats

In order to create a pipeline that utilizes different NLP tools and modules, the CLEW provides a mechanism to exchange data between the different tools and modules that make up any NLP pipeline.

After exploration of different NLP frameworks, we found that there have been various approaches to facilitate data exchange among different components of NLP pipelines as a key part of a complete NLP framework, including, but not limited to:

- Apache Unstructured Information Management Architecture (UIMA) Common Analysis System (CAS).
- General Architecture for Text Engineering (GATE) Plug-In.

- Natural Language Toolkit (NLTK).
- Apache OpenNLP.

In order to make use of various third-party software tools and ready-to-use components, efforts to explore the integration of heterogeneous NLP components and the ability to standardize different data formats for exchanging annotated language data among tools such as:

- DKPro.
- Ready-to-use software components based on the Apache UIMA framework.
- NLP components developed by different parties as GATE Plug-Ins.
- LAPPS Grid standard formats: Web Service Exchange Vocabulary (WS-EV), LAPPS Interchange Format (LIF), and JavaScript Object Notation-based serialization for Linked Data (JSON-LD).

The LAPPS Grid has already developed translators that allows tools that use different frameworks for exchanging inputs and outputs.

Based on the findings of the environmental scan and review of the LAPPS Grid, LAPPS LIF, JSON-LD, and WS-EV were chosen to exchange information between different NLP tools, modules, and web services.

Self-contained frameworks invite selection as a single strategy for implementation as they simplify the movement of outputs from one tool into another tool further down the pipeline. However, they create barriers to using software built in other data sharing paradigms. The group that has made significant progress to overcome this siloing barrier is LAPPS, so it has been chosen as the general mechanism for assembling pipelines.

This does not, however, exclude any development team from selecting a particular paradigm to constrain its own development pathways.

Based on the findings of the environmental scan, the early experimentation with the Workbench prototype includes several components that can communicate internally by using the UIMA framework (see Appendix A). UIMA has both advantages and disadvantages, so it is not a requirement for components in this project, but it has been beneficial at this early stage for a number of reasons:

- The common use and acceptance of UIMA in research communities and application domains, as observed in the environmental scan.
- The potential ease of integration of existing or new UIMA-based third-party applications, components, and tools.
- The programming languages that may adequately support UIMA-based development.
- The portability of UIMA-compliant software to various development environments and operating systems.

Moving forward, the resolution of UIMA formats into the LAPPS paradigm are needed before including them in CLEW..

7.10 CLEW Instance of the LAPPS Grid

The LAPPS grid is a development environment created by a team at Vassar College to enable a process of experimentation with different open-source NLP pipelines. LAPPS is built using the Galaxy platform.

Galaxy is an open, web-based platform for data intensive biomedical research. The Galaxy team is at Penn State, and the Biology and Mathematics and Computer Science departments at Emory University. The Galaxy Project is supported in part by National Human Genome Research Institute (NHGRI), National Science Foundation (NSF), The Huck Institutes of the Life Sciences, the Institute for CyberScience at Penn State, and Emory University.

The user interface in LAPPS Grid makes it easy for an NLP researcher to plug and play with different components from different NLP open source projects. It encourages development and experimentation by project teams wanting to engage in engineering NLP production lines. The service model created in the CLEW encourages the sharing of the production lines with other user communities.

New pipelines can be developed to compare the quality of output from using different combinations of tools, components, and features. The CLEW integrates the use of the LAPPS Grid Interchange Format (LIF) to support integration of different clinical tools, components, and features across NLP frameworks (such as UIMA and GATE). For example, a tokenizer from one system can be paired with a POS tagger from another system. This interoperability is achieved by the creation of the LIF standardized data sharing format so that outputs from one module are converted into a LIF format ready for use in any other module. On involving another module the LIF format is converted into the input format needed for that module. Hence LIF represents a common language for the transfer of data from one module to another without needing special translation tools.

The intended primary users of the CLEW instance of the LAPPS Grid (see Figure 3) are IT specialists and NLP expert researchers. These specialists can have expertise in machine learning, computational linguistics, and software engineering. The team can also include an expert with clinical subject matter expertise.

Figure 3: Screen shot of the CLEW instance of LAPPS Grid that provides a list of the tools and pipelines included.

8 Pilot Use Cases to Demonstrate Use of the CLEW

This project's long-term goal is to support NLP in multiple clinical domains through the modular addition of new tools, ontologies, and methods. Progress can be made on many fronts. NLP tools have already been developed that focus on many of the important tasks that need to be performed to make use of clinical free text. In a previous deliverable, *A Report of the Natural Language Processing Environmental Scan Results*, we have identified and categorized a large number of tools useful for clinical NLP across a wide range of domains.

This section reviews the cancer pathology and safety surveillance use cases that were used to design, develop, and test the CLEW prototype. The initial focus, particularly for the prototype, was in supporting the identification of clinical information from cancer and safety data as well as time information and time relations from safety data. The final version supports the creation of NLP pipelines for the use cases described below and are made available to end users for experimenting and developing new pipelines for use in other domain specific data or applications. We are also in the process of presenting our approach to other agencies and identifying more use cases based on their particular needs that could be supported by the CLEW.

8.1 Safety Surveillance Use Cases

8.1.1 Identification of Clinical Information in Text

One of the first steps in the review of safety surveillance reports is the identification of the outcome of interest (such as diagnosis or cause of death), the time to onset, and other alternative explanations (such as drug, medical, and family history) from the free-text narratives in the post-market reports. "Symptom" and "rule out diagnosis" information is also evaluated in this process. The example below shows the free-text narrative from a vaccine report that was submitted to FDA's Vaccine Adverse Event Reporting System (VAERS). Features of interest such as the vaccine name (bright green), the primary diagnosis (blue), symptoms (gray), medical history (yellow), and cause of death (pink) have been highlighted.

> *"A 33 year-old man with past medical history significant for dizziness/fainting spells received the following vaccines on 10 March 2001: VAX1 (lot number not reported); and VAX2 (lot number not reported either). Ten days after vaccination, he developed shortness of breath and chest pain and was subsequently diagnosed with myocarditis. On Day 20 (30 March 2010) post vaccination, the following tests were performed: an electrocardiogram which was reported to be normal and troponin I levels were measured and found to be 12.3 ng/ml (abnormal). Patient died on 02 April 2010. COD: heart failure. List of documents held by sender: None."*

Safety surveillance reports contain an abundance of clinical features, the extraction of which is essential for clinical NLP tasks. Obviously, the identification of clinical information in medical texts is a major milestone in all biomedical domains.

8.1.2 Normalization/Coding to Medical Terminologies

Comparisons of clinical information across different patients, different institutions, or even different notes for the same patient are substantially easier to interpret when using a common, standardized terminology. Many such terminologies exist, and the encoding of clinical information into these formats is an important task in developing the NLP pipelines, since these normalizations facilitate many forms of automated processing.

8.1.3 Identification of Temporal Relations

The processing of temporal information and its association with clinical information is of paramount importance in the NLP field[2, 3, 4, 5]. Temporal information in clinical texts is mainly in

the form of *temporal expressions*, which consist of time modifiers (such as "before" or "after"), units days, weeks, or years), and numerical tokens. Although temporal expression can be recognized easily in the text, the assignment of these expressions to clinical features and the identification of overarching temporal relations can be complex and challenging process, even for humans. Many clinical NLP challenges have attempted to address these difficulties by incorporating specific tasks aimed at identifying temporal relations. The 2012 Informatics for Integrating Biology & the Bedside (i2b2) Challenge on Temporal Relations[2], the 2015 Semantic Evaluation (SemEval) Challenge (Task 6: Clinical TempEval)[6], and the 2016 SemEval Challenge (Task 12: Clinical TempEval)[7] all addressed the identification of temporal relations in clinical texts. Although results from this work have been successful, there are many unresolved challenges to address.

The same free-text narrative from VAERS is shown below. In addition to the clinical features, temporal expressions including absolute dates (dark yellow) and relative time statements (red) are also highlighted.

> *"A 33 year-old man with past medical history significant for dizziness/fainting spells received the following vaccines on 10 March 2001: VAX1 (lot number not reported); and VAX2 (lot number not reported either). Ten days after vaccination, he developed shortness of breath and chest pain and was subsequently diagnosed with myocarditis. On Day 20 (30 March 2010) post vaccination, the following tests were performed: an electrocardiogram which was reported to be normal and troponin I levels were measured and found to be 12.3 ng/ml (abnormal). Patient died on 02 April 2010. COD: heart failure. List of documents held by sender: None."*

In this example, both vaccines are assigned the "2001-03-10" timestamp, corresponding to the absolute time "10 March 2001." While the assignment of timestamps to some features is straightforward, for other features this requires calculations based on temporal expressions from previous sentences. The primary diagnosis "myocarditis" is assigned the "2001-03-20" timestamp, corresponding to the relative time statement "Ten days after" which refers to the vaccination date "10 March 2001" in the previous sentence.

8.1.4 Automated Case Summarization

Case summarization aims to reduce the size of potentially large documents by retaining only the key information from the original documents. In the safety surveillance domain, automated case summarization involves the organization of key clinical features such as primary diagnosis, and any corresponding time information into a succinct paragraph. It is often necessary to combine features retrieved from the free-text narratives with data from various structured fields, such as patient age, to summarize the information in the safety surveillance report accurately.

The goal of case summarization in the safety surveillance domain is to assist medical reviewers to retrieve key information by enabling them to review reports in a more effective and efficient manner. Baer et al. showed that case summarization of VAERS reports can considerably reduce the time and effort spent on the actual review of safety surveillance reports[8]. Botsis et al.

described the main elements of an abbreviated textual summary generated by an automated decision support environment at the Center for Biologics Evaluation and Research at the FDA[9]. Their summary combined age, sex, product name, medical and family history, and primary diagnoses, along with any applicable timestamps, to summarize safety surveillance reports. For example, the sample VAERS report presented above can be summarized as follows:

> *"A 33 year-old male **was treated with** VAX1 and VAX2 on 2010-03-10. He **was diagnosed with** myocarditis 10 days later. The **cause of death** is heart failure."*

The summary contains 27 words and 124 characters (without spaces) versus the 100 words and 535 characters (without spaces) in the VAERS report. Although this is a short post-market report, the 73% reduction in the size of the text that has to be reviewed by the medical experts creates considerable efficiency. Case summarization is significant in other settings too, such as the clinical environment, where physicians have to process large amounts of free-text data. Some previous efforts discussed and presented some approaches to address this task[10, 11].

8.1.5 Case Deduplication

Reported cases and patient records can become duplicated in a particular database if information from multiple overlapping sources is brought together. Identifying and addressing these duplicates can ensure high data quality. Historically, only structured record information has been compared to identify duplicates[12, 13, 14], but free text comparisons can also be valuable. In the safety surveillance domain, we have applied a method combining both structured information and extracted text that has helped create high-quality lists of potential duplicates[15].

8.1.6 Semi-Structured or Templated Text

Some text sources can be considered semi-structured, meaning they use a specifically mandated template but have large variations in filling out content within that template. One excellent example is the Structured Product Label (SPL) standard for printed medical product labels required by FDA. It designates sections and section ordering that must be included in all product labels, but the manufacturers define the specific content and structure within the sections. Processing the text from these labels can quickly identify listed potential adverse effects, as several studies have attempted to do[16, 17, 18] .

8.2 Cancer Pathology Use Cases

8.2.1 Annotation of Clinical Semantic Entity Recognition (SER) in Pathology Reports

Pathology reports contain a wealth of clinical SERs such as primary site, behavior, grade, laterality, and histology. The example below presents the free-text portion of a pathology report found in one of the cancer registries supported by CDC. SERs representing histology (blue), primary site (pink), behavior (yellow), laterality (gray), and grade (bright green) information have been highlighted. Identifying the target SERs in this example is a simple task, but the general problem is much more difficult when multiple specimens, organs, and histologies are present in the text, all needing to be tied together in correct relationships.

"Right breast tissue with seed localization, lumpectomy specimen: [Blank]. In situ component: Ductal carcinoma in situ. Architectural pattern: Solid and comedo. Nuclear grade: High grade. Necrosis: Present. Extent: 3 mm residual focus. Margins of resection: Negative. Closest margin: Inferior, 3 mm. Calcifications: Present. Lymphovascular invasion: Not identified. Lymph nodes: Not applicable. Tumor staging: pTis pNX pMn/a. Prognostic markers: Not applicable. Additional pathologic findings: Prior biopsy/seed site including inflammatory and repair reaction changes....

8.2.2 Coding of Clinical Annotations from Cancer Pathology Reports to International Classification of Diseases for Oncology, 3rd Edition (ICD-O-3)

The identification of SERs of clinical terms in the pathology report is only one part of the process that is required to complete the reporting necessary for cancer surveillance. The next step is to map the SERs to a nationally adopted coding standard, such as the ICD-O-3. This also can have its complexities when there are dependencies between the SERs that determine the correct codes, such as when the correct site for lymphoma is a function of both the specimen site and its tissue type.

The use of standard coding systems can enable researchers and PCORnet participants to analyze the information captured from the text-based pathology reports. The CLEW can provide a service to codify the clinical annotations that result from the use case described in 8.2.1 above.

9 Categories and Components

A wide range of NLP tools and approaches can satisfy clinical NLP tasks across the domains mentioned in the use cases of Section 4. The environmental scan identified and classified a large number of NLP tools that were applicable for these types of problems. We have prepared a number of these tools to run as part of the CLEW prototype, either on their own or as part of a combination pipeline that integrates multiple tools.

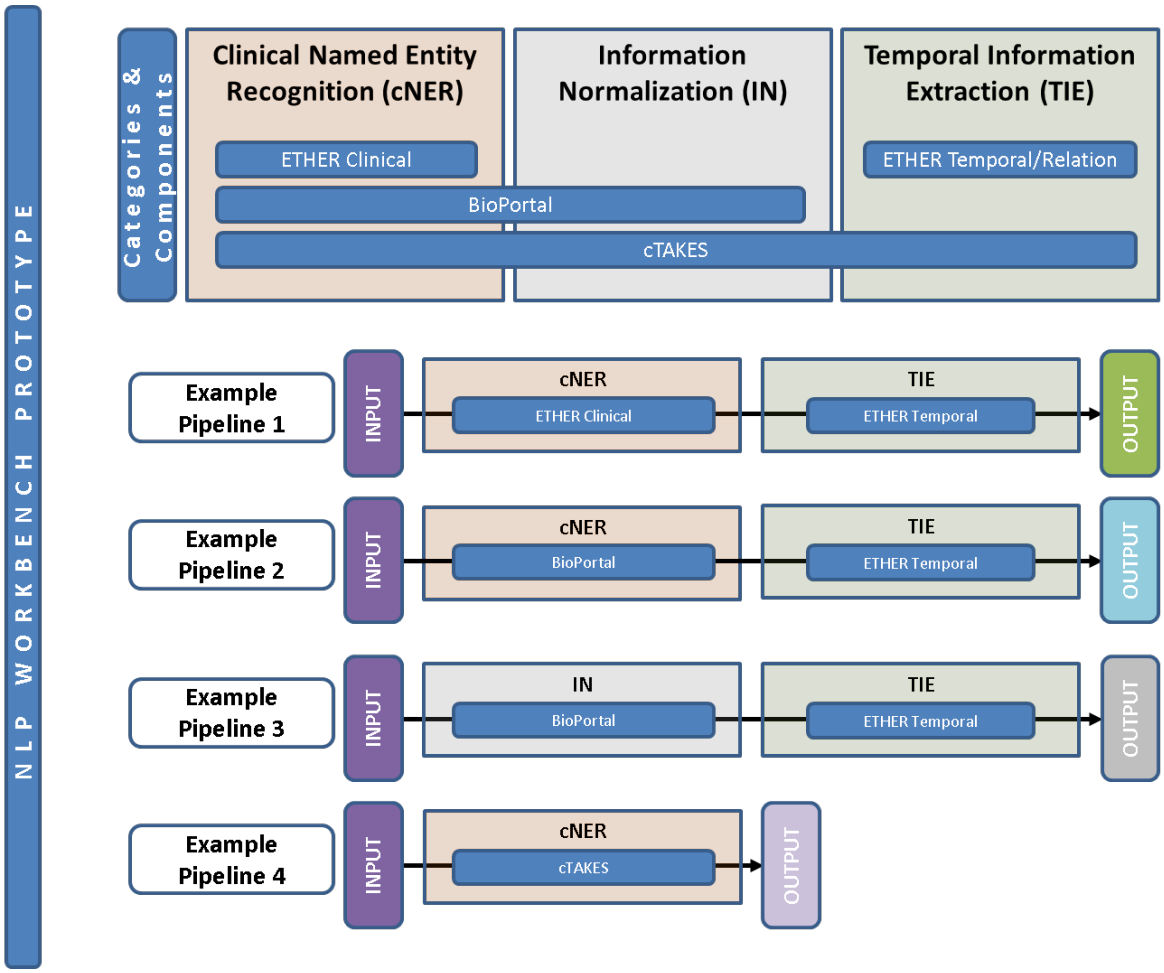


Figure 4. An outline of the CLEW prototype. The integrated tools are shown in the top boxes, where they have been categorized according to the functionality they offer. Four example pipelines are also shown that demonstrate some of the possible combinations of tools that can be made in the CLEW prototype to process input text and produce annotated output.

Figure 4 above shows the tools that are currently incorporated into the CLEW prototype, the capabilities of those tools, and a number of sample multi-tool pipeline configurations. The tools cover several different functionalities, and we have made efforts to separate multi-purpose tools into their constituent parts, so that pipelines can be constructed by combining components from more than one tool. Depending on the specific clinical need, not all of the listed tasks are required, as demonstrated in the example pipelines.

The development of the services in the CLEW is inspired by the ES findings, but we take the next step here and, in some of the cases, utilize certain functionalities only from each tool. For example:

- We have split the original ETHER into two pieces: one for the clinical NER and one for the TIE.
- We are using the cTAKES engine and its temporal module (nothing else in this phase).

This strategy allows for the combination of components and the creation of pipelines on the fly. In some cases, this kind of development is not necessary. For example, BioPortal has an API we used directly with minimum development.

9.1 Integrated Tools and Services

9.1.1 ETHER

The FDA's ETHER tool extracts key clinical and temporal information from safety surveillance reports or other free-text sources[8]. This tool was selected for inclusion in the CLEW prototype because it supports multiple use cases, and members of the team are familiar with the tool and its code, meaning that integration could be performed smoothly. See Appendix G.

9.1.2 cTAKES

The Clinical Text Analysis Knowledge Extraction System (cTAKES) (see http://ctakes.apache.org/) is an Apache Software Foundation project. It supports multiple NLP tasks, such as the recognition of clinical named entities with a number of contextual attributes. It additionally utilizes the UIMA architecture and was therefore considered an appropriate solution for early integration in the CLEW prototype. See Appendix H.

9.1.3 BioPortal

The CLEW prototype includes a functional link to some of the BioPortal services provided by the National Center for Biomedical Ontology through the BioPortal REST API (see https://bioportal.bioontology.org/). The BioPortal Annotator service encodes text into standardized terms from selected ontologies. Users can restrict output to a list of chosen ontologies and a list of chosen UMLS Semantic Types. See Appendix I.

9.1.4 Pathology SER

The SER processing for the eMaRC Plus service sends a document to the service and receive back the texts that form the five entities of interest, including primary site, histology, behavior, grade, and laterality.

9.1.5 Pathology Codification

The SEs extracted from the pathology reports are sent to a service that computes the ICD-O-3 codes for the entities. This service can be of use to anyone needing to obtain ICD-O-3 codes for their texts.

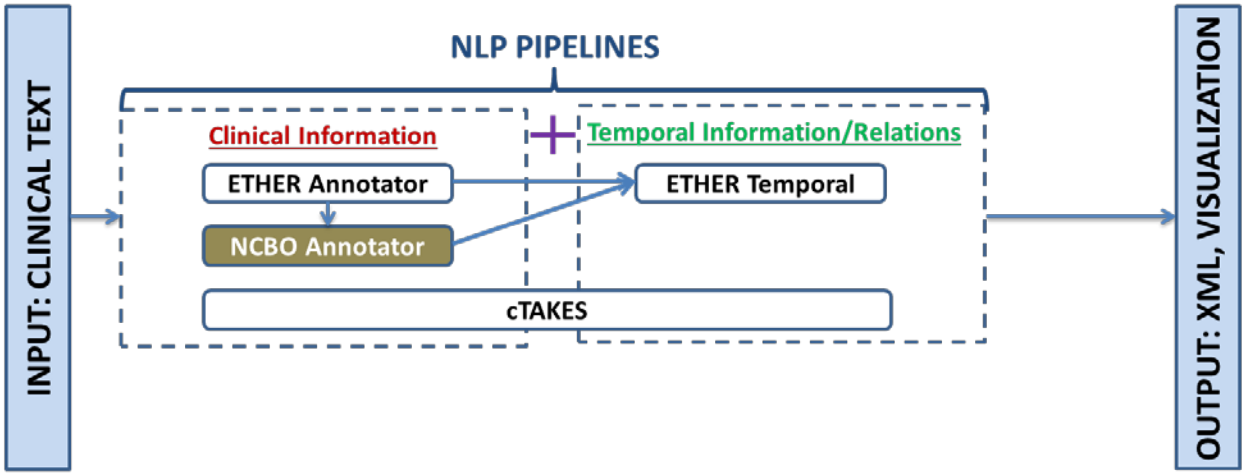
9.2 Future Expansion of Tools as Services

The environmental scan identified many more tools suitable for addressing use cases in clinical domains. By incorporating additional tools over time, the CLEW functionalities could expand and new configurations of pipelines could be available. It may be also possible to support the generation of more pipelines on the fly as the developed services would have the data exchange mechanism in place. However, any third-party tools may not necessarily be readily integratable into the NLP installed base of tools if they do not have data exchange mechanisms already in

place, in which case they can be installed as services. Some of the more widely useful tools that might be prioritized include:

- TARSQI Toolkit (TTK): A tool for identifying both temporal expressions and temporal associations within text.
- Med-TTK: An adaptation of the TARSQI Toolkit to provide better coverage of various date formats commonly found in clinical text.
- MetaMap: A system for encoding text into a structured clinical terminology, namely the Unified Medical Language System (UMLS).

Figure 5 shows a general outline of the tools currently available in the prototype version, and Figure 6 illustrates the components that may support the synthesis of NLP pipelines for the extraction of clinical and time information as well as the identification of temporal relations. Those are some of the use cases described above. Again, the final version of the Workbench can include more components and support the creation of multiple pipelines.



NCBO: National Center for Biomedical Ontology; ETHER: Event-based Text-mining of Health Electronic Records system; **cTAKES**: clinical Text Analysis Knowledge Extraction System

Figure 5. The tools and available connections between tools in the current pilot version of the core NLP approach.

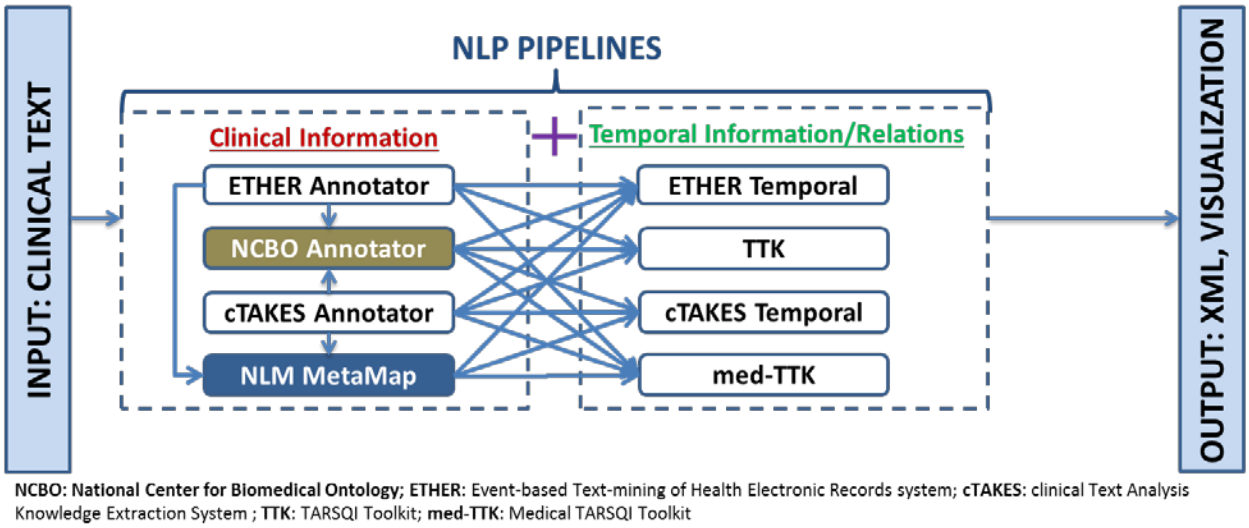


Figure 6. The tools and available connections between some of the tools planned to be developed in the second year for the core NLP approach.

10 Example NLP Pipelines

The following list includes all the possible pipelines using one or multiple tools that are supported in the CLEW prototype. Some pipelines may only contain tools from a subset of categories but still meet specific clinical needs.

- ETHER Clinical Feature Extraction
- BioPortal Annotator (using one or more ontologies)
- cTAKES Clinical Feature Extraction
- cTAKES Clinical Feature Extraction + cTAKES Temporal
- ETHER Clinical Feature Extraction + ETHER Temporal
- BioPortal Annotator (using one or more ontologies) + ETHER Temporal

In addition to these examples, the BioPortal Annotator in the CLEW prototype can be parameterized (it supports the coding to more than 500 ontologies/terminologies), making the number of possible configurations much larger.

A few of these possible pipelines are demonstrated in Appendix J by applying them to a vaccine safety report and a pathology report excerpt. Two different pipelines with similar goals are run on each sample text, and the resulting output is compared.

11 Discussion

This report presents the requirements, architecture, and current implementation of the core NLP approach. The approach is flexible for adapting and incorporating NLP tools for many clinical activities across different domains, which have been identified through a comprehensive

environmental scan. A subset of these tools, chosen for their applicability to specific use cases for safety surveillance and cancer pathology processing, are now prepared for use. They have been tested and made available through a CLEW prototype, which allows for the construction of multiple NLP pipelines that can combine components of separate tools. All of the current tools are also accessible through straightforward Java APIs that allow them to be programmatically accessed by developers of other software tools and applications.

The ES has provided a long list of tools that are potentially useful in building pipelines for various clinical NLP domains. We will add support for additional tools and looking especially at the processing of cancer reports with tools like caTIES, MedKATp, and others. The analysis of pathology reports can be advanced by investigating a variety of pipeline configurations that best identify pertinent SERs. We will also be further exploring methods for visualizing results from the CLEW to identify the best methods for certain use cases.

In terms of evaluating tools and pipelines, the best-performing NLP pipelines were identified for a small set of cancer and safety surveillance use cases only. The corresponding evaluation for all use cases described in this report and others that may be supported by the CLEW is beyond the scope of this project. End users (federal agencies, public health agencies, academic centers, and PCORnet) will be using the Workbench to create, evaluate, and select the best pipelines for their own work.

Acknowledgements

This work was supported in part by the appointment of Matthew Foster, Abhishek Pandey, and Kory Kreimeyer to the Research Participation Program administered by ORISE through an interagency agreement between the U.S. Department of Energy and the U.S. FDA.

References

1. Ide N. et al., (2016) The Language Application Grid. In: Murakami Y., Lin D. (eds) Worldwide Language Service Infrastructure. WLSI 2015. Lecture Notes in Computer Science, vol 9442. Springer, Cham
2. Sun, W., A. Rumshisky, and O. Uzuner, Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. J Am Med Inform Assoc, 2013. 20(5): p. 806–813.
3. Sun, W., A. Rumshisky, and O. Uzuner, Annotating temporal information in clinical narratives. J Biomed Inform, 2013. 46 Suppl: p. S5–S12.
4. Styler IV, W.F., et al., Temporal Annotation in the Clinical Domain. Transactions of the Association for Computational Linguistics, 2014. 2: p. 12.
5. Wang, W., et al., A new algorithmic approach for the extraction of temporal associations from clinical narratives with an application to medical product safety surveillance reports. J Biomed Inform, 2016. 62: p. 78–89.
6. Bethard, S., et al. SemEval-2015 Task 6: Clinical TempEval. in Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). 2015. Association for Computational Linguistics.
7. Bethard, S., et al. SemEval-2016 Task 12: Clinical TempEval. in Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). 2016. Association for Computational Linguistics.
8. Baer, B., et al., Can Natural Language Processing Improve the Efficiency of Vaccine Adverse Event Report Review? Methods Inf Med, 2016. 55(2): p. 144–150.
9. Botsis, T., et al., Decision support environment for medical product safety surveillance. J Biomed Inform, 2016. 64: p. 354–362.
10. Pivovarov, R. and N. Elhadad, Automated methods for the summarization of electronic health records. J Am Med Inform Assoc, 2015. 22(5): p. 938–947.
11. Hirsch, J.S., et al., HARVEST, a longitudinal patient record summarizer. J Am Med Inform Assoc, 2015. 22(2): p. 263–274.
12. Registry Plus™ Link Plus. January 13, 2015 [Accessed August 18, 2017]; Available from: www.cdc.gov/cancer/npcr/tools/registryplus/lp.htm.
13. Norén, G.N., et al., Duplicate detection in adverse drug reaction surveillance. Data Mining and Knowledge Discovery, 2007. 14(3): p. 305–328.
14. Tregunno, P.M., et al., Performance of Probabilistic Method to Detect Duplicate Individual Case Safety Reports. Drug Safety, 2014. 37(4): p. 249–258.
15. Kreimeyer, K., et al., Using Probabilistic Record Linkage of Structured and Unstructured Data to Identify Duplicate Cases in Spontaneous Adverse Event Reporting Systems. Drug Saf, 2017. 40(7): p. 571–582.
16. Duke, J.D. and J. Friedlin, ADESSA: A Real-Time Decision Support Service for Delivery of Semantically Coded Adverse Drug Event Data. AMIA Annu Symp Proc, 2010. 2010: p. 177–181.
17. Fung, K.W., C.S. Jao, and D. Demner-Fushman, Extracting drug indication information from structured product labels using natural language processing. J Am Med Inform Assoc, 2013. 20(3): p. 482–488.
18. Kuhn, M., et al., The SIDER database of drugs and side effects. Nucleic Acids Res, 2016. 44(D1): p. D1075–D1079.

Appendix A: UIMA Framework

The Apache UIMA framework is a comprehensive architecture for text analysis and interpretation with a specification that has been approved by the Organization for the Advancement of Structured Information Standards consortium. It has been used extensively in the development of many NLP tools, as found in our environmental scan. UIMA provides a means for independent components to communicate and operate as part of a single pipeline by sharing a common data structure.

UIMA data can be transmitted easily in XML Metadata Interchange (XMI) format, which standardizes a structured XML document for containing the textual information, any type of data/feature type, and annotation results. The XMI format can be used easily for internal communication with other tools that have not been following the UIMA framework.

Annotations must be based on a Common Analysis Structure (CAS) that defines the parameters available for the annotations. All feature structures, including annotations, are represented in the UIMA Common Analysis Structure (CAS)[1]. The CAS is the central data structure through which all UIMA components communicate.

The first step in developing an annotator based on the UIMA framework is to define the CAS Feature Structure types. This is done in an XML file called a type system descriptor, which may be tailored to various domain applications by including domain-specific definitions.

A VAERS Data Type System descriptor has been initially defined and developed for the CLEW (see Appendix B). This data type system is based on the UIMA primitive data types and contains additional data types and features in the safety surveillance domain that have been defined hierarchically.

One important benefit of the hierarchical structure of the type system is the possibility for straightforward extensions via inheritance to represent data types in other domains. The existing data types can be extended to allow new attributes and new annotation types. More details about inheritance and expanding the type system can be found in Appendix B.

As specified above, the VAERS Data Type System is able to represent the VAERS safety surveillance domain. This type system can be combined with the data specifications of other general NLP components like tokenizers or taggers to further enable data exchange among the NLP components for domain-specific NLP pipelines. As an example, the output for the report text "*102.3 Fever (lasted 2 days). 1/1/2016 Rash on trunk and face*" may be stored in an XML file with proper namespace definitions and feature type definitions as shown below.

[1] UIMA Tutorial and Developers' Guides. https://uima.apache.org/

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:vaers="http:///gov/hhs/fda/srs/annotation/vaers.ecore"
xmlns:tcas="http:///uima/tcas.ecore" xmlns:cas="http:///uima/cas.ecore">
    <cas:NULL xmi:id="0"/>
    <tcas:DocumentAnnotation language="en" xmi:id="1" sofa="6" end="61" begin="0"/>
    <vaers:Symptom xmi:id="2" sofa="6" end="11" begin="6"/>
    <vaers:Symptom xmi:id="3" sofa="6" end="60" begin="38"/>
    <cas:Sofa xmi:id="6" sofaString="102.3 Fever (lasted 2 days). 1/1/2016 Rash on trunk
and face." sofaNum="1" sofaID="_InitialView" mimeType="text"/>
    <View sofa="6" members="0 1 2 3"/>
</xmi:XMI>
```

Appendix B: The VAERS Data Type System

The VAERS Data Type System created for the CLEW is defined hierarchically. In other words, every data type is defined based on inheritance from a super-type. All new types inherit the attributes of their super-type. For example, the UIMA base annotation *uima.tcas.Annotation* is the super-type of the highest abstract type in the VAERS Data Type System, *gov.hhs.fda.srs.annotation.vaers.VaersFeature*. The other types in the VAERS Data Type System are then all defined as nested sub-types of that type. The first column of Table B1 shows the level of abstraction of a particular type name in the VAERS Data Type System; the second column shows the name of the data type; and the third column lists the super-type of the data type.

Table B1: The VAERS Data Type System defined for the CLEW prototype.

Level	Data Type Name	Super-Type
1	gov.hhs.fda.srs.annotation.vaers.VaersFeature	uima.tcas.Annotation
2	gov.hhs.fda.srs.annotation.vaers.ClinicalFeature	gov.hhs.fda.srs.annotation.vaers.VaersFeature
	gov.hhs.fda.srs.annotation.vaers.RelationFeature	
	gov.hhs.fda.srs.annotation.vaers.TemporalFeature	
	gov.hhs.fda.srs.annotation.vaers.VaersSummarization	
3	gov.hhs.fda.srs.annotation.vaers.CategoryCauseOfDeath	gov.hhs.fda.srs.annotation.vaers.ClinicalFeature
	gov.hhs.fda.srs.annotation.vaers.CategoryDiagnosticFeatures	
	gov.hhs.fda.srs.annotation.vaers.CategoryFamilyHistory	
	gov.hhs.fda.srs.annotation.vaers.CategoryMedicalHistory	
	gov.hhs.fda.srs.annotation.vaers.CategoryMedicalProduct	
	gov.hhs.fda.srs.annotation.vaers.CategoryStatus	
4	gov.hhs.fda.srs.annotation.vaers.Drug	gov.hhs.fda.srs.annotation.vaers.CategoryMedicalProduct
	gov.hhs.fda.srs.annotation.vaers.Vaccine	
4	gov.hhs.fda.srs.annotation.vaers.PrimaryDiagnosis	gov.hhs.fda.srs.annotation.vaers.CategoryDiagnosticFeatures
	gov.hhs.fda.srs.annotation.vaers.RuleOutDiagnosis	
	gov.hhs.fda.srs.annotation.vaers.SecondLevelDiagnosis	
	gov.hhs.fda.srs.annotation.vaers.Symptom	
3	gov.hhs.fda.srs.annotation.vaers.Date	gov.hhs.fda.srs.annotation.vaers.TemporalFeature
	gov.hhs.fda.srs.annotation.vaers.Frequency	

Level	Data Type Name	Super-Type
	gov.hhs.fda.srs.annotation.vaers.Duration	
	gov.hhs.fda.srs.annotation.vaers.Relative	
	gov.hhs.fda.srs.annotation.vaers.Time	
	gov.hhs.fda.srs.annotation.vaers.Weekday	
3	gov.hhs.fda.srs.annotation.vaers.FeatureTimeRelation	gov.hhs.fda.srs.annotation.vaers.RelationFeature

Appendix C: Lessons Learned Working with cTAKES

The cTAKES system provides many useful functionalities for text analytics, but this also makes it a complex system with a daunting learning curve. The available documentation does not always adequately prepare readers to use the system properly, and at least one major software bug presented significant challenges for incorporating cTAKES into our project.

The integration of cTAKES in the CLEW involved preparing a Java API to be accessible as a web service or as part of a domain-specific Java application. To accomplish this, we wanted to incorporate cTAKES into a Maven project via a simple POM dependency. However, an ongoing bug in cTAKES versions 4.0.0 and 4.0.1 prohibited it from being included as a Maven POM dependency, causing a “URI is not hierarchical” Java exception when an application attempts to load cTAKES through the POM mechanism.

To resolve this issue for the current development of the CLEW, we implemented a two-part solution that included making minor modifications to cTAKES source code and generating a new JAR file to replace the distributed version, and placing additional resource files into the “resources” directory of the target project structure. With these two changes, the current CLEW project was able to incorporate cTAKES through a POM dependency. This is not necessarily a general workaround, and other applications may require different approaches.

A revision to the cTAKES source code would allow third-party developers to leverage cTAKES for integration with their domain applications.

Appendix D: Environmental Scan List of Tools

The combination of the literature and the multi-channel review resulted in a final list of 54 tools (16 complete systems, 26 applications, and 12 components). As shown in Table D1 below, significant development activity around the GATE and UIMA frameworks was noted, and all tools supported have either multiple or very specific functions. The tools also spanned a wide range of methodologies, including rule-based, machine learning, and hybrid approaches. A complete discussion on the ES findings can be found in the corresponding deliverable.

Table D1. The final list of 54 tools identified in the environmental scan. Each tool belongs to one of the three categories (complete system, application, or component). Tools may provide more than one function, and several tools have been developed to be consistent with a specific framework.

Tool Acronym	Tool Category	Function	Framework
ClearTK	Complete System	***Multiple***	UIMA
cTAKES	Complete System	***Multiple***	UIMA
GATE	Complete System	***Multiple, SER***	GATE
GENIA Tagger	Complete System	***Multiple***	GATE
HITEx	Complete System	***Multiple***	GATE
IXA pipes	Complete System	***Multiple***	
KNIME	Complete System	***Multiple***	
Leo	Complete System	***Multiple***	UIMA
MedEx	Complete System	***Multiple***	
MetaMap	Complete System	***Multiple***	
Neji	Complete System	***Multiple***	
NERsuites	Complete System	***Multiple***	
NLTK	Complete System	***Multiple, SER***	NLTK
OpenNLP	Complete System	***Multiple***	OpenNLP
Stanford CoreNLP	Complete System	***Multiple, SER***	
Weka	Complete System	***Multiple***	
ABNER	Application	NER	
BioEnEx	Application	NER	
BioLemmatizer	Application	S/L	UIMA
BioMedICUS	Application	NER	UIMA, OpenNLP
Bio-SCoRes	Application	CR	
BioSimplify	Application	Sentence Simplification	
caTIES	Application	NER	GATE
CliNER	Application	NER	
CRIS-IE-Smoking	Application	NER	GATE
i2b2	Application	***Multiple***	GATE
Lancet	Application	NER	

Tool Acronym	Tool Category	Function	Framework
Maveric Annotator	Application	NER	UIMA
MedCoref	Application	CR	UIMA
medKATp	Application	NER	
MedTagger	Application	NER	UIMA
MedXN	Application	NER	UIMA
MIST	Application	De-id	
NLP Pipelines	Application	NER	UIMA
PEP	Application	NER	UIMA
PubMed-Ex	Application	NER	
pyConText	Application	NER	
RapTAT	Application	NER	GATE
TagLine	Application	NER	
THYME	Application	NER & TER	
TTK	Application	TER	
V3NLP	Application	NER	
BADREX	Component	Abbr Norm	GATE
brat	Component	Annotation	
ConText/NegEx	Component	Neg Detection	
eHOST	Component	Annotation	
Genia SS	Component	Chunking	GATE
GUTime	Component	TER	
HeidelTime	Component	TER	UIMA
Knowtator	Component	Annotation	
MedTime	Component	TER	UIMA
MSTParser	Component	Parsing	
Noun Phrase Chunker	Component	Chunking	
TreeTagger	Component	POS Tagging	GATE

TER: Temporal entity recognition; **POS:** Part of speech; **Neg:** Negation; **Abbr Norm:** Abbreviation normalization; **NER:** Named entity recognition; **CR:** Coreference resolution; **S/L:** Stemming/lemmatization; **De-id:** De-identification; **SER:** Semantic entity recognition.

Appendix E: Cancer Pathology Pilot Project

E.1 Services

Four pathology services were developed using Stanford, OpenNLP, Gate, and cTAKES. The services address the needs of cancer registries and other clinical researchers to convert text into categorical data or codes. The Clinical Entity Recognition (CER) service identifies the same information that may be written differently in pathology reports. Organizations can use the Pathology Coding Service to complete the conversion from text to code. The four services can be used to demonstrate differences between different pathology datasets using different pipelines.

E.2 Identified Semantics

The pathology services are designed to supply codes for five key data elements:

- Body site of the cancer.
- Histology.
- Grade of disease.
- Behavior of the cancer.
- Laterality or side of the body where it is located.

The CER service supplies 22 different entities of information including structural information about the document, which is necessary for finding the best location to extract the desired content.

The different clinical entities identified in the CER service are filtered down to those that are most useful for deriving the wanted codes.

E.3 Approach to Address the Use Case

This use case applies to organizations that want to extract content from pathology reports and code that content. The service is split into two services so that the CER can be assessed independently from the Pathology Coding Service process.

E.4 Cancer Pathology Demonstration

Four cancer pathology pipeline demonstrations are provided in the CLEW. They illustrate how NLP rule-based and machine learning solutions work. Figure E1 shows how the input of unstructured data is analyzed along with the output produced.

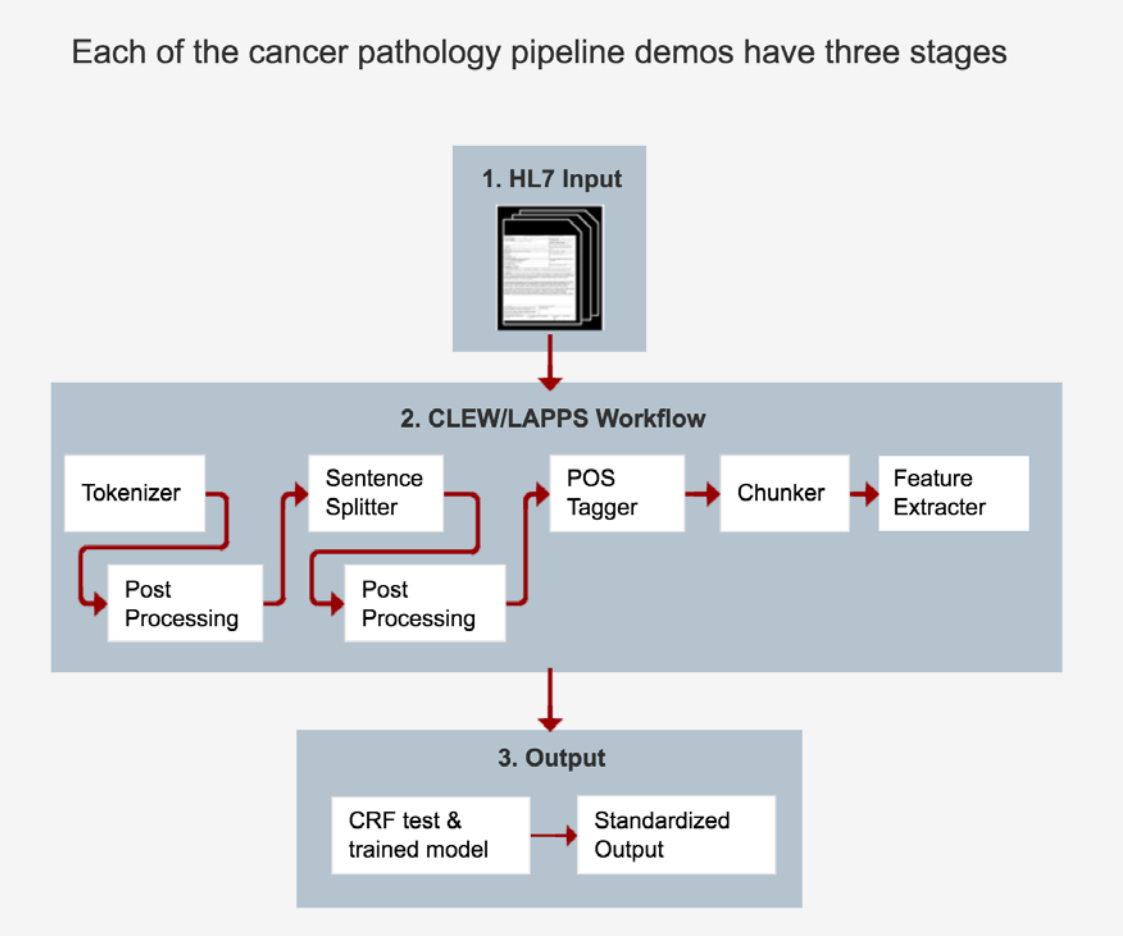


Figure E-1: Stages of the pathology demonstrations

1. HL7 Input: Convert the HL7 input file into a standardized (LIF) format.

2. CLEW/LAPPS Workflow: Use CLEW/LAPPS Grid tools below to generate a standardized file (BIO file) with post-processing tools to modify the output of some of the pipeline steps for the clinical domain.

    - Tokenizer
    - Sentence splitter
    - POS tagger
    - Chunker
    - HLA feature extractor

3. Output: Use Conditional Random Field (CRF) and the previously trained model and output the standardized annotated content TXT file with:

    - Semantic tag name
    - Entity text, start
    - End position of entity text

E.5 Sample Cancer Pathology Demonstration Using Stanford Pipeline

In the **Input** field, select a sample cancer report provided that contains important clinical and temporal cancer patient information, or select **My data input** to enter input data.

HL7 data for selected input

Note that metadata is at the top of the file, but not processed through the pipeline. It is used by the pathology laboratory to supply information about the content needed for the management of the pathology that does not refer to the clinical content itself. It contains identification of the patient, the pathology service, and time and date stamps for different actions taken in completing the report, plus a brief summary. Different laboratories insert different information in the metadata, so it has a variable structure.

```
document_text=OBX|1|TX|22638-1^Comments^LN||Comment||||||F|||20150901172804
OBX|2|TX|22636-5^Clinical History^LN||ClinicalInformation||||||F|||20150901172804
OBX|3|TX|22633-2^Nature of Specimen^LN|1|Colon- Random biopsy (Sample
2)||||||F|||20150901172804
OBX|4|TX|22634-0^Gross Pathology^LN|1|Specimen is received in formalin, labeled as
"Random Colon" with the patient's name and consists of 26 pieces of soft pink tissue
measuring from 0.1 x 0.1 x 0.1 up to 0.6 x 0.2 x 0.1 cm. All submitted (6 Blocks).
Blocks #1-3- 5 pieces each Blocks #4-5- 4 pieces each Block #6- 3
pieces.||||||F|||20150901172804
OBX|5|TX|22637-3^Final Diagnosis^LN|1|Colonic mucosa showing a few lymphoid
aggregate. No cryptitis, no crypt abscesses, no glandular distortion and no dysplasia
seen.||||||F|||20150901172804
OBX|6|TX|22633-2^Nature of Specimen^LN|2|Colon-Sigmoid biopsy (Sample
3)||||||F|||20150901172804
OBX|7|TX|22634-0^Gross Pathology^LN|2|Specimen is received in formalin, labeled as
"Sigmoid Polyp" with the patient's name and consists of 1 piece of soft pink tissue
measuring 1.4 x 1.0 x 0.9 cm. Also received a stalk measuring 0.4 x 0.4 cm. Sectioned
and entirely submitted (5 Blocks).||||||F|||20150901172804
OBX|8|TX|22637-3^Final Diagnosis^LN|2|Fragments of tubulovillous adenoma with
intramucosal adenocarcinoma/high grade dysplasia. The stalk margin is free of high
grade dysplasia and intramucosal carcinoma.||||||F|||20150901172804
```

Select the Pipeline

After choosing the input to process, select a NLP pipeline of interest to perform various clinical NLP tasks on the input.

Pipelines to select from include:

- Stanford
- OpenNLP
- Gate
- cTAKES

Stanford Pipeline Description

1. Convert the HL7 file into an extracted TXT file.
2. Convert the TXT file into LIF format.
3. Run Stanford Tokenizer in LAPPS.
   The tokenizer identifies each individual word and non-word string. It can be confused by various punctuation and numeric content and unusual orthography.
4. Run HLA Post Tokenizer Corrector for Stanford.
   The corrector step modifies some of the output to correct for certain known gross errors that it makes on clinical texts.
5. Run Stanford Sentence Splitter in LAPPS.
   The sentence splitter separates the text into sentences. It can be misled by various format layouts, unusual use of punctuation, and lack of punctuation.
6. Run HLA Post Sentence Splitter Corrector for Stanford.
   The corrector unit corrects some of the known gross mistakes the splitter makes on clinical texts.
7. Run the Stanford POS Tagger in LAPPS.
   The POS tagger assigns a part of speech to each token. It marks any word not recognized as a noun. POS tagging enables some recognition of the relationships between words, but has some uncertainty because words can be used as different parts of speech without changing their morphology; for example, the verb “weeping” can be used as an adjective in “weeping sore.”
8. Run the GATE Chunker in LAPPS.
   The chunker aggregates multiple tokens into a single phrase. This assists in recognizing clinical descriptions.
9. Send GATE chunks to MetaMap for initial concept recognition.
   MetaMap is a system for recognizing clinical concepts and assigning them code values from a range of clinical classification systems. These codes are attached as features to the tokens in the text chunk created in step 8.
10. Run the HLA Feature Extractor for Stanford.
    When all of the NLP processing is complete, the computed values for each token are assembled as a set of features. These are used to generate a file with one token per line where each token has all its attributes/features assigned. The last step is to attach the semantic class of the token iterated in the annotation step and assign it to the beginning of the annotation (B-class), an intermediate position (I-class), or not tagged at all (O-class). This file is called the BIO file and is input directly into the machine learning algorithm.
11. Generate the BIO file.
12. Send the BIO file to the service.
13. The service returns the extracted clinical entities with semantic tag name, entity text, and start and end position of entity text in the TXT file.

Cancer Pathology CER Service Demonstration Output

After the CLEW processes the text through the workflow defined in the selected pipeline, the compiled information in Table E-1 below serve as output that can be used at this point or can be passed as input to the Cancer Pathology Coding Service to convert the text to standard ICD-O-3 codes for use. See Appendix F to see how the output from the Pathology Coding Service is received from the CLEW in eMaRC Plus.

Table E-1. Cancer pathology CER service demonstration output from the CLEW Stanford Pipeline.

Tag Name	Content	Start	Stop
Final Diagnosis Heading	Final Diagnosis	608	623
Final Diagnosis Heading	Final Diagnosis	757	772
Clinical History Heading	Clinical History	1071	1087
Cancer Histology Subtype	tubulovillous	787	800
Macroscopic/Gross Description Heading	Gross Pathology	31	46
Macroscopic/Gross Description Heading	Gross Pathology	338	353
Cancer Histology Type	adenocarcinoma	827	841
Cancer Histology Type	carcinoma	930	939
Comments Heading	Comments	1051	1059
Specimen Identifier	Sample 2	985	993
Specimen Identifier	Sample 3	1039	1047
Organ/Body Structure	Colon	963	968
Organ/Body Structure	Colon	1017	1022
Neoplasm Behaviour	intramucosal	814	826
Nature of Specimen Heading	Nature of Specimen	943	961
Nature of Specimen Heading	Nature of Specimen	997	1015
Relative Location	Sigmoid	1023	1030

Appendix F: eMaRC Plus

Central cancer registries use eMaRC Plus to receive and process cancer pathology and biomarker data in unstructured narrative format in HL7 version 2 messages. Currently, eMaRC Plus provides integrated rule-based text mining functionality to translate text to code. While this has been useful in the cancer community and with the improvements in processing unstructured data in the NLP community, expansion in the eMaRC functionality would allow for the implementation of machine learning methodologies to improve the quality of its coding.

F.1 Scope

eMaRC Plus was enhanced to interface with the CLEW web services to process unstructured pathology data and return coded data for primary site, histology, behavior, grade, and laterality.

F.2 Description of System Enhancements

eMaRC Plus allows users to use either the rule-based text mining method, the CLEW web services for SNLP and coding methods, or both methods of coding the attributes during import of pathology reports. Adding this option allows current eMaRC Plus users to continue using the application to process pathology reports that can contain any type of cancer cases, while the CLEW web service developers target and fine-tune the language model to target specific types of cancers, including lung, breast, prostate, and colorectal. See Figure F-1 below for a glance at what the cancer registry user sees when processing pathology reports within eMaRC Plus.

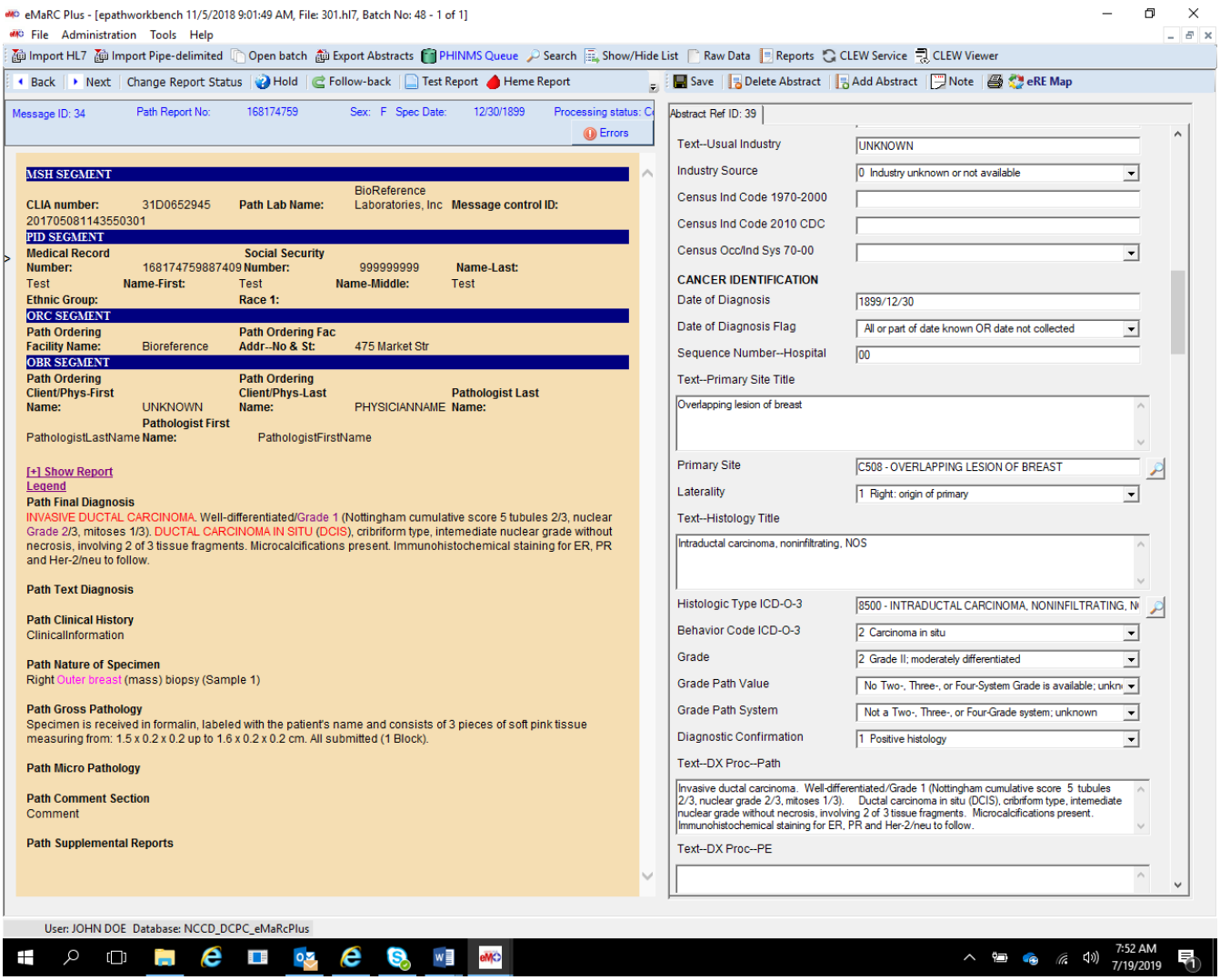


Figure F-1. Screen shot of eMaRC Plus that shows the pathology report text on the left side and the resulting coded data for primary site, laterality, histologic type, behavior code, and grade on the right.

When both coding options are enabled, eMaRC Plus uses results from both methods to highlight terms in the pathology text and for coding the five attributes, and provides a way to compare results of both methods. Figure F-2 below shows the CLEW annotation and coded values returned to eMaRC Plus. The highlighted terms in the user interface have different color borders around them to indicate whether the current text-mining method or CLEW SNLP method found them.

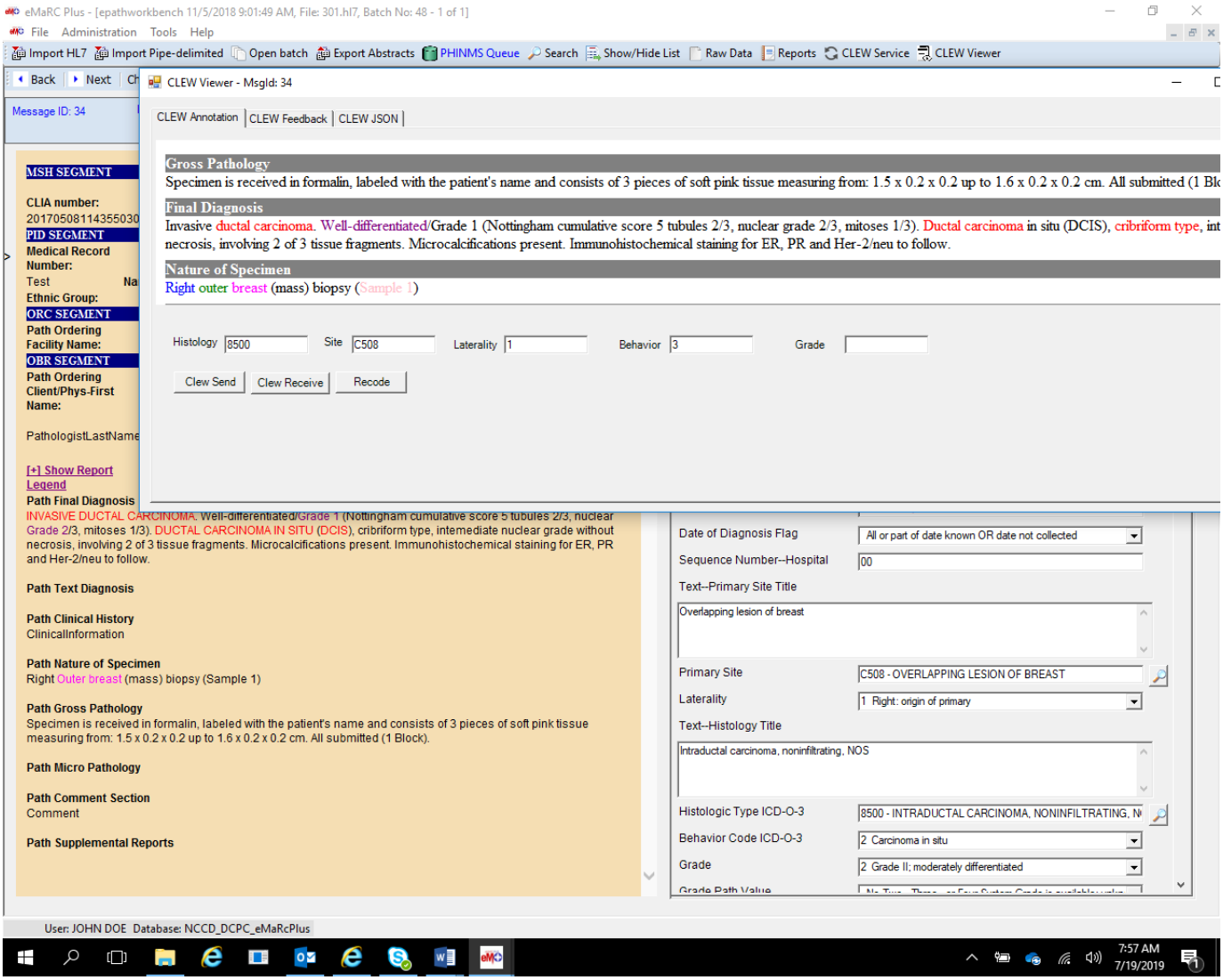


Figure F-2. Screen shot of eMaRC Plus with resulting CLEW annotations and suggested coded values from processing the sample pathology report.

Figure F-3 shows where users can provide their inputs to create a feedback loop to further train the language model so that it improves accuracy in the next iteration.

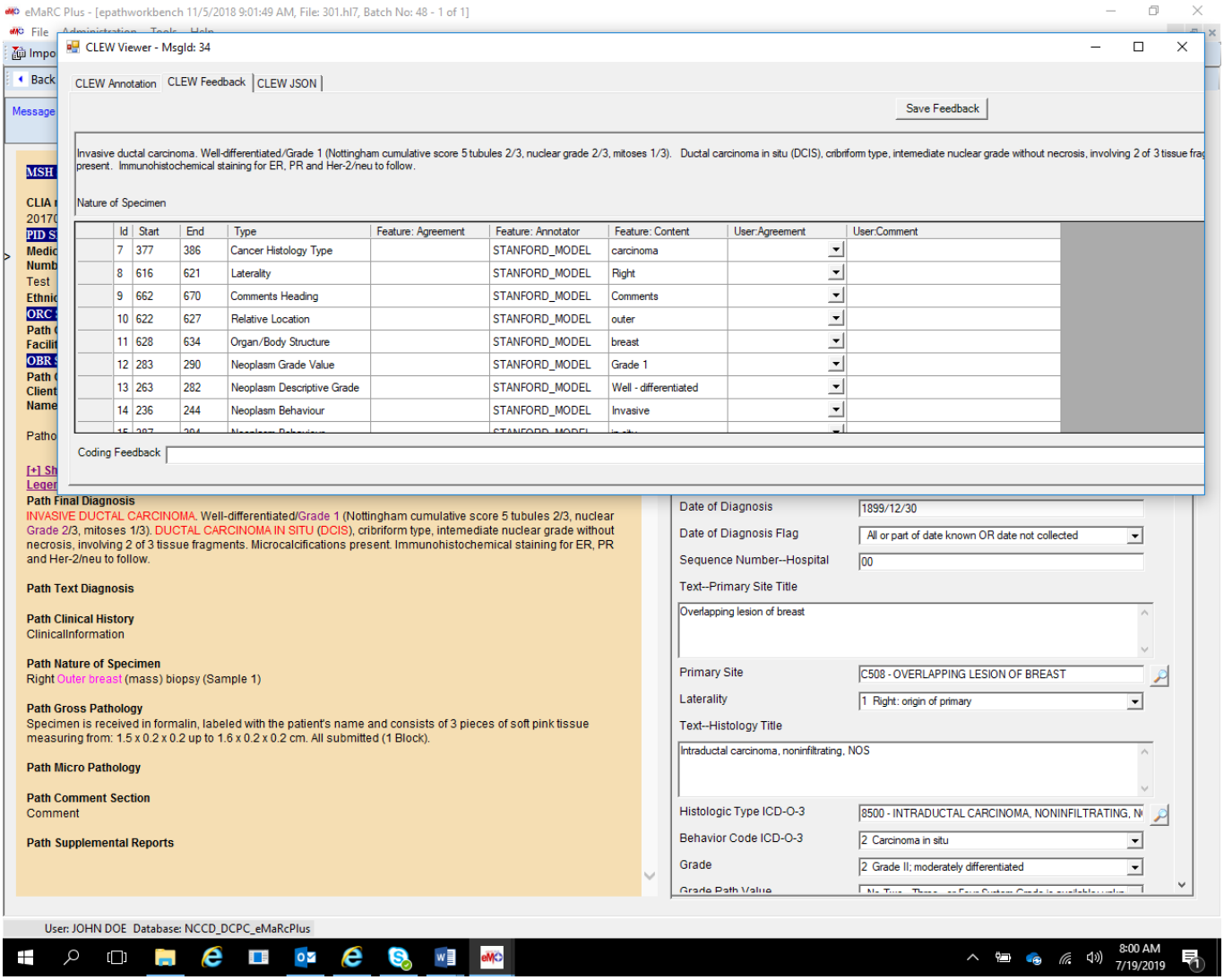


Figure F-3. Screen shot of CLEW Feedback screen in eMaRC Plus where the user can provide feedback on the accuracy of annotation and coding elements of the pathology report.

Appendix G: ETHER

ETHER was modified to be compatible with the CLEW architecture by separating some of the key components of the tool, so they could be executed independently. The separate services include the extraction of clinical features, the extraction of temporal expressions, and the creation of temporal associations between features and expressions. Each service requires a plain text input and returns an XML output file listing the identified features, expressions, or associations that conform to the VAERS Data Type System discussed in Section 6.

The ETHER components have been modularized as different functions and can be executed with the following commands:

(1) To extract clinical features:
ETHERNLP.exe –e textInput.txt featureOutput.xml
(2) To extract temporal features:
ETHERNLP.txt –t textInput.txt timeOutput.xml
(3) To extract clinical-temporal association features, in short, relation features:
ETHERNLP.txt –r featureOutput.xml timeOutput.xml relOutput.xml

For the parameters of the commands, “textInput.txt” is the input file that contains the text to be processed, “featureOutput.xml” is the name of an output file to contain the annotation results of extracted clinical features, “timeOutput.xml” is the name of an output file to contain the annotation results of extracted temporal expressions, and “relOutput.xml” is the name of an output file to contain the annotation results of extracted clinical-temporal associations. The actual filenames for processing are supplied by the user.

G.1 Use Cases Satisfied

ETHER supports the use cases 4.1 Identification of Clinical Information in Text, 4.3 Identification of Temporal Relations, and 4.4 Automated Case Summarization.

G.2 Use of ETHER in the CLEW for End Users

The ETHER components for the clinical, temporal, and relation functions have been integrated into the CLEW prototype (FDA’s development environment) as shown in Figure G-1.

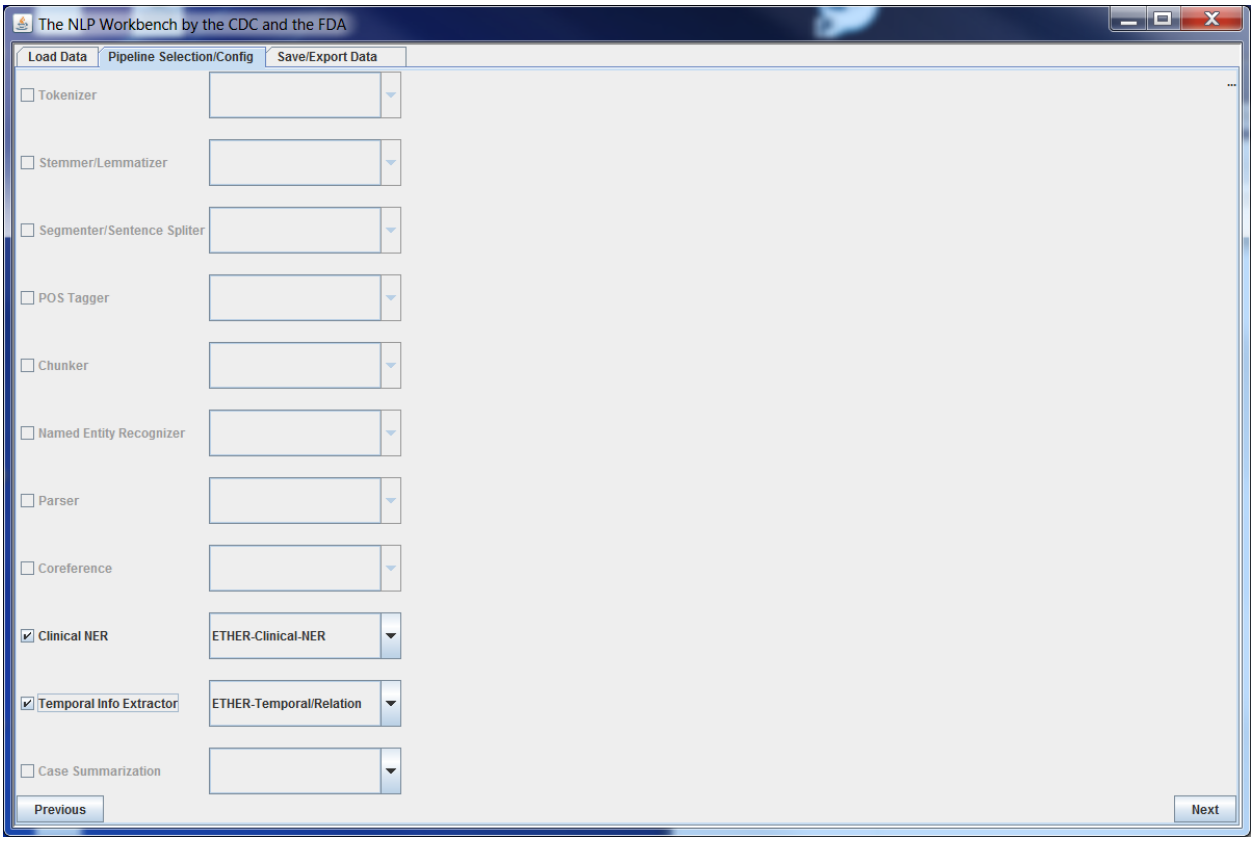


Figure G-1. A screen shot of the selection of the ETHER modules in the CLEW prototype.

After loading the data (first tab; process not shown in the screenshot), a user can select the Clinical NER and Temporal Info Extractor components. As shown in Figure G-1, "*ETHER-Clinical-NER*" and "*ETHER-Temporal/Relation*" components have been selected from the "*Clinical NER*" and "*Temporal Info Extractor*" categories, respectively. Following the synthesis of the NLP pipeline that supports the retrieval of the clinical and temporal information, the user can click the "Next" button located at the lower-right corner of the screen. The CLEW executes the synthesized NLP pipeline and display the results in the "*Save/Export Data*" tab of the window (not shown in Figure G-1).

Notably, users can select the "*ETHER-Clinical-NER*" component to carry out the clinical NER alone. The same applies to "*ETHER-Temporal/Relation*" that can support a single-component pipeline as well.

G.3 Integration of the ETHER Functions in Domain Applications for Developers

The ETHER modules can be currently employed by end users in the CLEW prototype running locally in the FDA's development environment. They can be further used by system or software developers to implement high-level applications tailored to specific subdomains in a local or a web environment.

The ETHER modules have been wrapped as very straightforward Java APIs for achieving maximal interoperability and portability. They can be used by software developers in clinical applications and are utilized by the CLEW prototype to realize the functionalities shown in Figure G-1.

Specifically, the ETHER clinical, temporal, and relation functionalities have been wrapped as three Java functions that can be called in a target Java application class. The inputs and outputs for each function are specified below.

1. **String processETHERClinical(String input)**
   This function accepts a string containing the input text and generates a string that contains the annotation results of clinical features using the VAERS Data Type System.

2. **String processETHERTemporal(String input)**
   This function accepts a string containing the input text and generates a string that contains the annotation results of temporal features using the VAERS Data Type System.

3. **String processETHERRelation()**
   This function does not accept any parameters. Instead, it makes use of the clinical and temporal results from the previous two functions, and generates a string that contains the annotation results of association relationship features using the VAERS Data Type System.

In a target application, if a pipeline is intended to obtain clinical features only, the first function needs to be called alone. If a pipeline is intended to only obtain temporal features, the second function needs to be called alone. However, if a pipeline is intended to obtain the association relationships, both of the above functions need to be called, and after these two functions finish, the third function also needs to be called.

A Maven project has been created to include the ETHER modules. Key Maven commands have been tested via JUnit test cases, such as '*mvn clean install*' and '*mvn test*'.

In terms of code distribution and maintenance, there is a single requirement to place the ETHER distribution package under the "*resources*" directory of the target program. There is no requirement for any other software installation (except for the latest versions of Java and Maven) or environmental variable setup to execute the ETHER modules. The ETHER distribution is a zipped package, named "ETHERNLP.zip"[2]. A developer can download it, extract the package into suitable directory, and copy the directory along with the underlying directory hierarchy and all the underlying files into the "*resources*" folder of the target program.

[2] Available at the official FDA and CDC GitHub pages (after the release of the CLEW prototype).

Appendix H: cTAKES

FDA integrated cTAKES version 4.0 into their version of the CLEW prototype and exposed its functions to retrieve clinical named entities and temporal information through our API infrastructure. Specifically, the main cTAKES clinical and temporal pipelines have been wrapped as Java functions that accept textual clinical notes as input and generate UIMA CAS structures as output.

H.1 Use Cases Satisfied

cTAKES supports the use cases 4.1 Identification of Clinical Information in Text, 4.2 Normalization/Coding to Medical Terminologies, and 4.3 Identification of Temporal Relations.

H.2 Use of cTAKES in the CLEW for End Users

The cTAKES clinical and temporal extraction capability has been integrated into the CLEW prototype (FDA's development environment) and can be executed easily by end users as shown in Figure H-1.

After loading the data (first tab; process not shown in the screenshot), a user can select either "*cTAKES-Clinical-NER*" from the Clinical NER category (not shown) or "*cTAKES-Temporal*" from the Temporal Info Extractor category (as shown in Figure H-1). Following the synthesis of the NLP pipeline that supports the retrieval of the clinical and/or temporal information, the user can click the "Next" button located at the lower-right corner of the screen. The CLEW executes the synthesized NLP pipeline and display the results in the "*Save/Export Data*" tab of the window (not shown in Figure H-1).

Notably, users can select the "*cTAKES-Clinical-NER*" component to carry out the clinical NER alone. Selecting the "*cTAKES-Temporal*" component executes both the clinical and temporal aspects of cTAKES in a single pipeline.

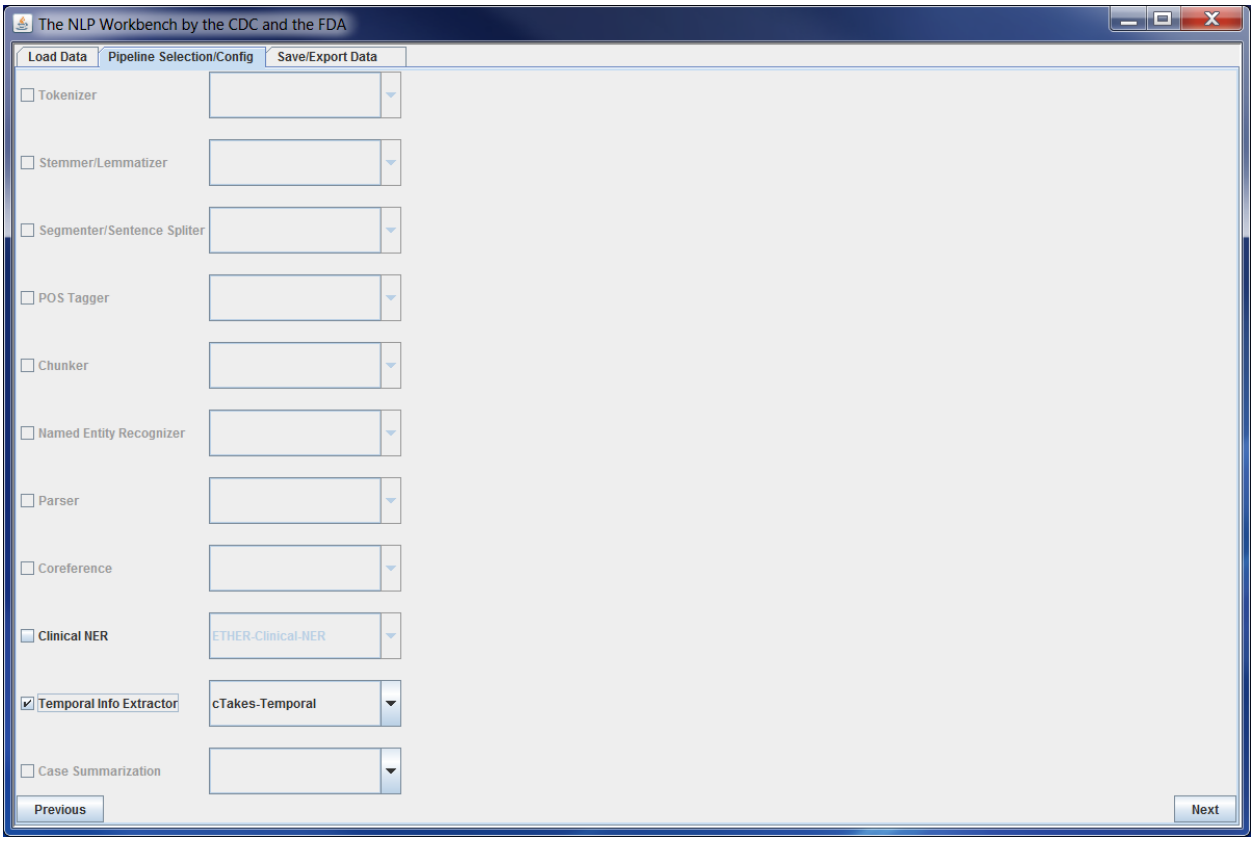


Figure H-1. A screen shot of the selection of the cTAKES Temporal module in the CLEW prototype.

i. Integration of cTAKES in Domain Applications for Developers

The cTAKES modules can be currently employed by end users in the CLEW prototype running locally in the FDA's development environment. They can be further used by system or software developers to implement high-level applications tailored to specific subdomains in a local or a web environment.

The cTAKES modules have been wrapped as very straightforward Java APIs for achieving maximal interoperability and portability. They can be used by software developers in clinical applications and are utilized by the CLEW prototype to realize the functionalities shown in Figure H-1.

Specifically, a Java utility class named "*CTakesAnnotatorForVAERS*" has been developed to accommodate the initialization and use of cTAKES. The Java utility class is instantiated in order to initialize cTAKES and enable calls to the specific functions for cTAKES functionality. The inputs and outputs of each function, as well as the sample code for the instantiation step are specified below.

1. To get an object of the *CTakesAnnotatorForVAERS* utility class:

   CTakesAnnotatorForVAERS ctakesAnno = new CTakesAnnotatorForVAERS();
   This statement instantiates a new Java object of the utility Java class.

2. The cTAKES temporal pipeline initialization function:
   void initializeCTAKES();
   This initialization process carries out suitable configurations and loads necessary resources for the full cTAKES temporal pipeline. This Java function does not take any input and does not return any value as output.

3. The cTAKES clinical pipeline initialization function:
   void initializeCTAKESClinicalNER();
   This initialization process carries out suitable configurations and loads necessary resources for the full cTAKES clinical pipeline. This Java function does not take any input and does not return any value as output.

4. The cTAKES execution function:
   CAS cTAKESAnno.processDocument(String input);
   This function takes an input string of clinical text and generates annotation results of extracted temporal features, as well as clinical features based on the initialization function executed.

A Maven project has been created to include the cTAKES modules. Key Maven commands have been tested via JUnit test cases, such as '*mvn clean install*' and '*mvn test*'.

In terms of code distribution and maintenance, there is no requirement of any software installation (except for the latest version of Java and Maven) or environmental variable setup to execute the cTAKES functions. A few environmental variables have been specified in the executable file (such as the run.bat file included in the CTAKESService package distribution) that enable the execution of cTAKES from the command line. For software development, the Maven project allows the inclusion of related projects and libraries in Java to be taken care of by the Maven dependency management mechanism. For example, if a target application is developed using Java and Maven, the POM.xml file contains the dependencies, and the relevant libraries can be downloaded and included automatically.

Appendix I: BioPortals

The CLEW prototype (FDA's development environment) provides parameter configurations for the BioPortal Annotator, including the ontologies and UMLS semantic types. The BioPortal contains 588 biomedical ontologies and related datasets, as shown in Figure I-1. These cover a broad variety of clinical domains and include many of the most widely used clinical terminologies. The completeness of the provided ontologies nor the veracity of the coding process have not been evaluated. Evaluations related to the performance of specific pipelines were conducted in Year 2.

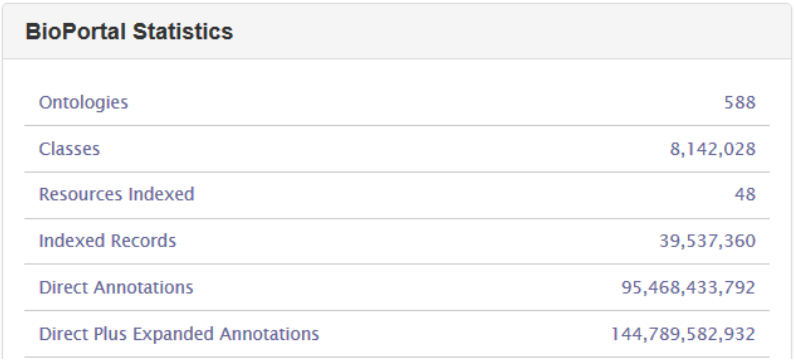


Figure I-1. A screen shot of the BioPortal statistics.

I.1 Use Cases Satisfied

The BioPortal Annotator supports use cases 4.1 Identification of Clinical Information in Text and 4.2 Normalization/Coding to Medical Terminologies. It may also support other activities as part of a larger pipeline.

I.2 Use of BioPortal in CLEW for End Users

The BioPortal service has been integrated as part of the CLEW prototype (FDA's development environment) as shown in Figure I-2.

Users can access the biomedical ontology coding service with some of the same features provided in the BioPortal web interface in the CLEW prototype. After loading the data (first tab; process not shown in the screenshot), a user can select "*NCBO-Clinical-NER*" from the Clinical NER category and then add one or more ontologies and UMLS types (as shown in Figure I-2). The "*NCBO-Clinical-NER*" is currently the only component in the CLEW prototype that requires some configuration, which is supported by the right panel. The user must click on the "*Start Coding*" button first, and the software runs the BioPortal coding service automatically. After the completion of the coding process, the user may click the "*Next*" button located at the lower-right corner of the screen. The CLEW executes the synthesized NLP pipeline and displays the results in the "*Save/Export Data*" tab of the window (not shown in Figure I-2).

The "*NCBO-Clinical-NER*" can be used alone or in combination with other components and can support both clinical NER and information normalization (as shown in Figure I-2).

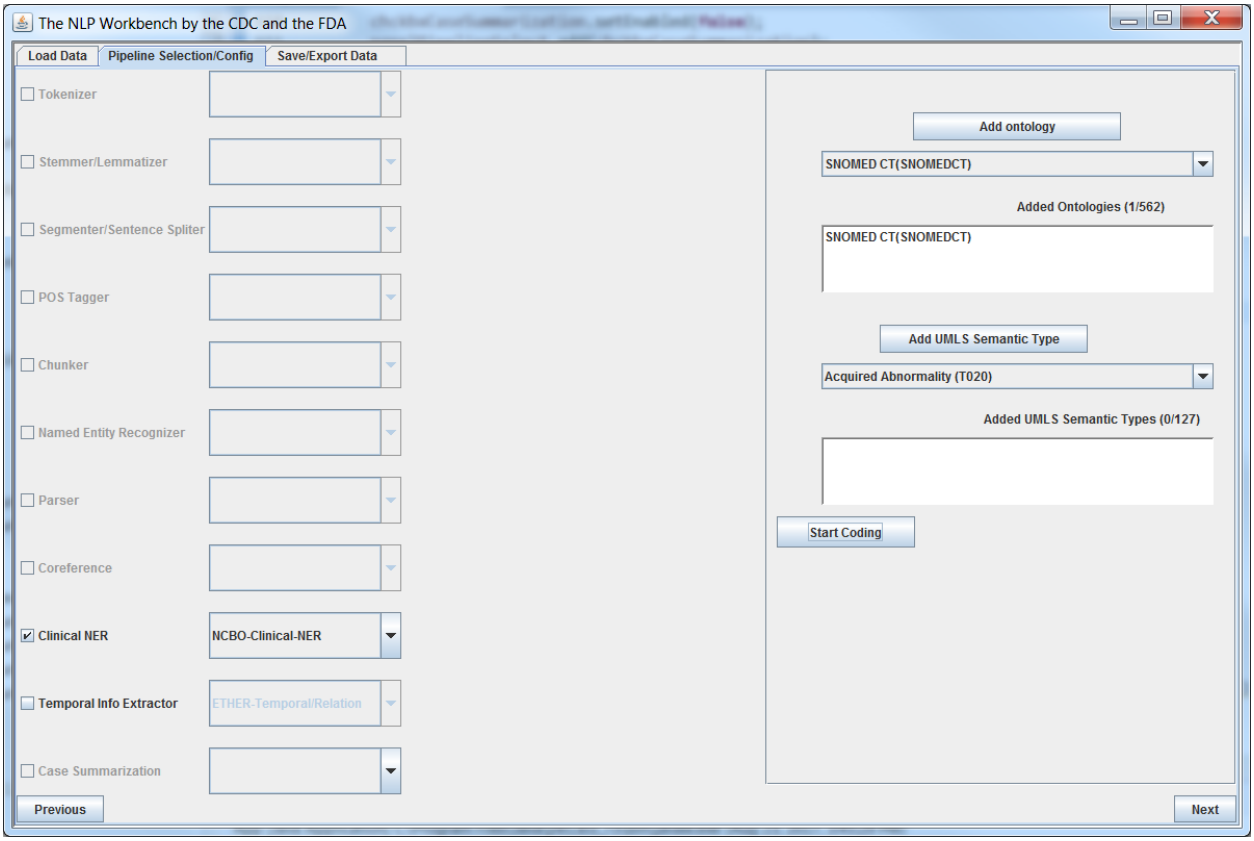


Figure I-2. A screen shot of the selection of the NCBO BioPortal Clinical Annotator using the SNOMEDCT ontology parameter in the CLEW prototype.

I.3 Integration of the BioPortal Annotation Function in Domain Applications for Developers

The BioPortal coding service is available to end users in the CLEW prototype running locally in the FDA's development environment. System or software developers can use it to implement high-level applications tailored to specific subdomains in a local or a web environment.

The BioPortal Rest API has been wrapped as a very straightforward Java API for achieving maximal interoperability and portability. This can be used by software developers in clinical applications and is utilized by the CLEW prototype to realize the functionalities shown in Figure I-2.

Only one initialization and one processing statement is used the coding features in a target Java class; inputs and outputs are specified below. There is also one utility function.

- Initialization:
  EncodeText et = new EncodeText();

- Processing:
  exec(String rawText, ArrayList<String> selectedOntologies, ArrayList<String> selectedUMLS)
  - *Inputs:*
    - *rawText: text to process*
    - *selectedOntologies: array list containing selected ontologies (acronyms), such as "MEDDRA" and "ETHER"*
    - *selectedUMLS: selected UMLS semantic types ("T200" for example)*
  - *Return: HashMap<String, Term>; a hash map between word (words) of interest (lower case) and a TERM class*
    - *Hash map key: {word(s)_fromPosition}. For example: pain_49 indicates a word pain (starting position in the text is 49).*
    - *Hash map value: Term class structure.*
      - *word: word(s) of interest*
      - *from: starting position in text (>=1)*
      - *to: ending position in text*
      - *strType: a term being mapped to*
      - *strCls: preferred label*
      - *ontology: ontology being used*
      - *context: the context for the word(s), determined by the range parameter*
- A utility function: ***getNCBOOntologyList()***: return the object of the OntologyList structure, in which a user can access the list of ontology names, the list of acronyms or the mapping between them. For example:
  - *et.getNCBOOntologyList().name <*** et is an instance of the EncodedText class mentioned above. ***>*
  - *et.getNCBOOntologyList().acrononym*

Please note that there are variations when initializing the EncodeText class:

- If a user wants to download the ontology list from the BioOntology web site, call "*new EncodeText(true)*". By default, the list has been downloaded and saved as *resources\ontology\OntologyList.txt.*

- To change the size of the context window around the word(s) of interest for display, provide an integer value to the constructor (default is 10) to indicate how many characters before/after the current word(s) are to be shown by calling
  - *new EncodeText(int range)* or
  - *new EncodeText(boolean updateNCBOOntology, int range)*

In terms of code distribution and maintenance, there is no requirement of any software installation (except for the latest version of Java and Maven) or environmental variable setup to execute the BioPortal coding function. For software development, a Maven project allows the inclusion of related projects and libraries in Java to be taken care of by the Maven dependency management mechanism. For example, if a target application is developed using Java and Maven, the POM.xml file contains the dependencies, and the relevant libraries can be downloaded and included for compilation and execution purposes automatically.

Appendix J: Examples of Shared NLP Pipelines and Web Services

The following list includes all of the possible pipelines using one or more tools that can be made and that are fully supported in the CLEW prototype. Some pipelines may only contain tools from a subset of categories, but still meet specific clinical needs.

- Pathology Annotator
- Pathology Coder to ICD-O-3
- ETHER Clinical Feature Extraction
- BioPortal Annotator (using one or more ontologies)
- cTAKES Clinical Feature Extraction
- cTAKES Clinical Feature Extraction + cTAKES Temporal
- ETHER Clinical Feature Extraction + ETHER Temporal
- BioPortal Annotator (using one or more ontologies) + ETHER Temporal

In addition to these examples, the BioPortal Annotator in the CLEW prototype can be parameterized (it supports the coding to more than 500 ontologies/terminologies), making the number of possible configurations much larger.

A few of these possible pipelines are demonstrated in the following subsections by applying them to a vaccine safety report and a pathology report excerpt. Two different pipelines with similar goals are run on each sample text, and the resulting output is compared.

J.1 Safety Surveillance Example

For this example, we are using the sample vaccine safety report from Section 4.1 and extracting clinical and temporal features using two different pipelines. The CLEW prototype is used to run both of these examples. The first pipeline is:

- ETHER Clinical Feature Extraction + ETHER Temporal

First, we load the text of this report into the CLEW prototype, as shown in Figure J-1. Then we make the appropriate tool selections to apply the separate pieces of the ETHER tool, as shown in Figure J-2. Then, after running, the CLEW prototype produces the output XML files containing the marked annotations and the temporal associations between clinical features and temporal expressions.

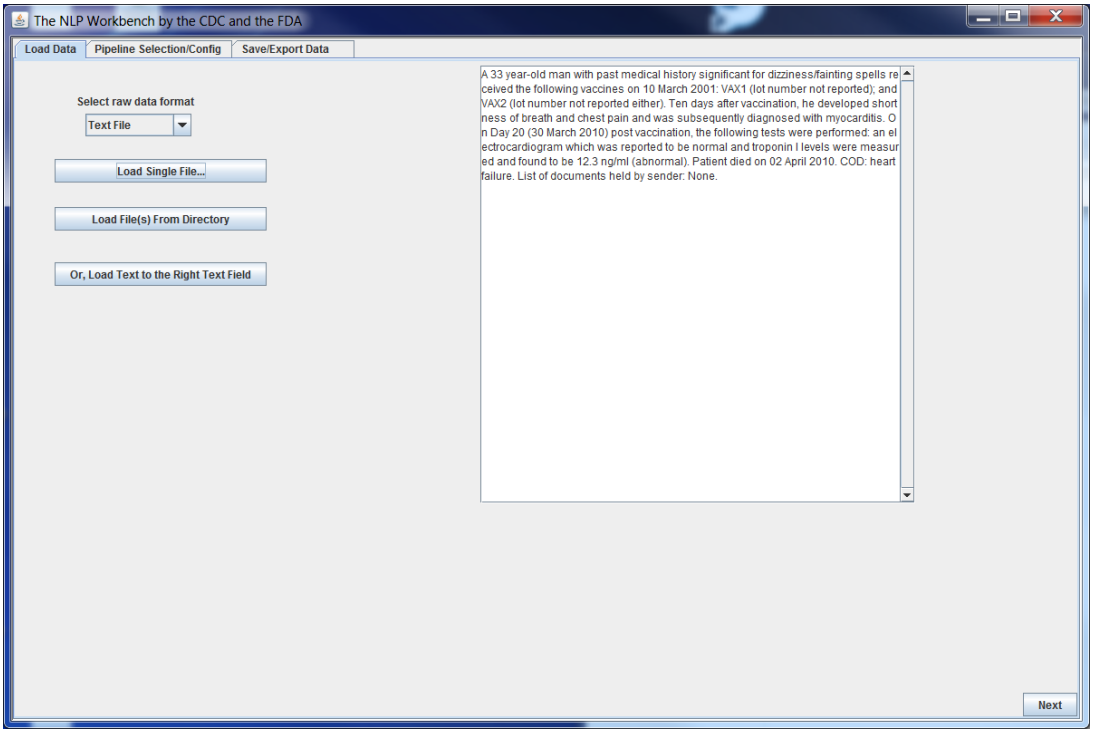


Figure J-1. The sample vaccine safety report being loaded as input text in the CLEW.

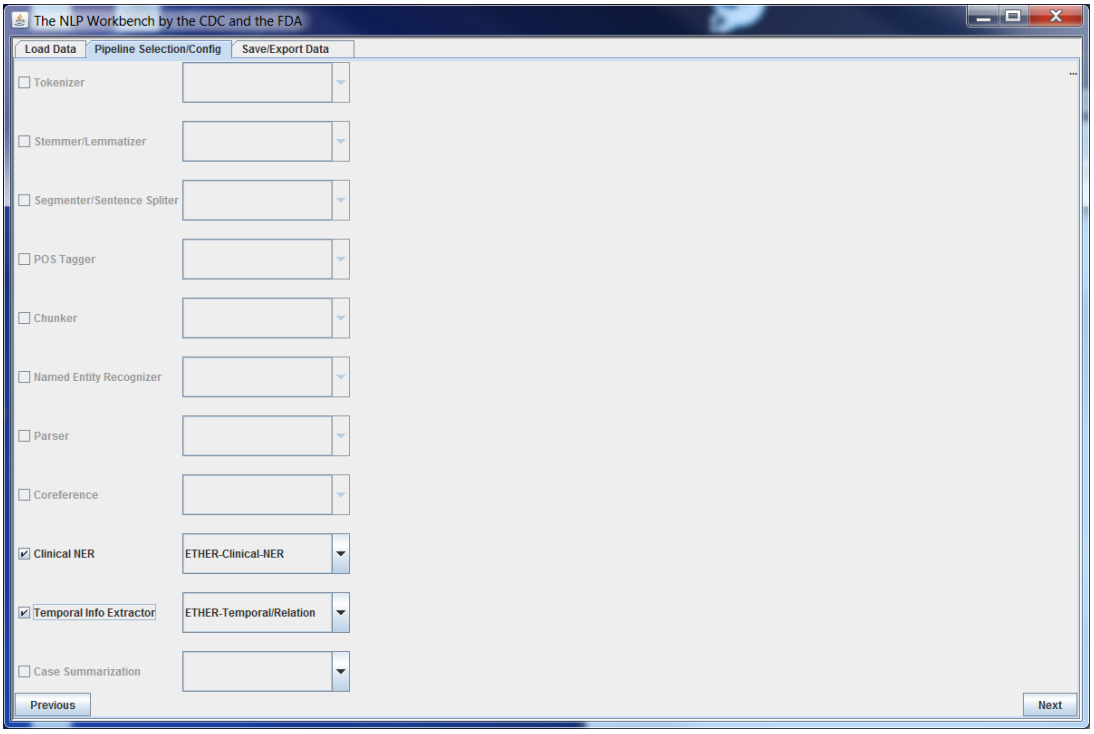


Figure J-2. The ETHER Clinical Feature Extraction and ETHER Temporal selections are chosen for processing the text.

The second pipeline for the sample vaccine safety report is:

- BioPortal Annotator (using MedDRA ontology) + ETHER Temporal

For this pipeline, we load the text into the CLEW prototype in the same way, then we set the Workbench to use the BioPortal Annotator with a single ontology, MedDRA. The temporal relation process is performed in the same way, using the ETHER tool. These selections are shown in Figure J-3. After running this pipeline, a similar set of output XML files with annotations and temporal associations is created.

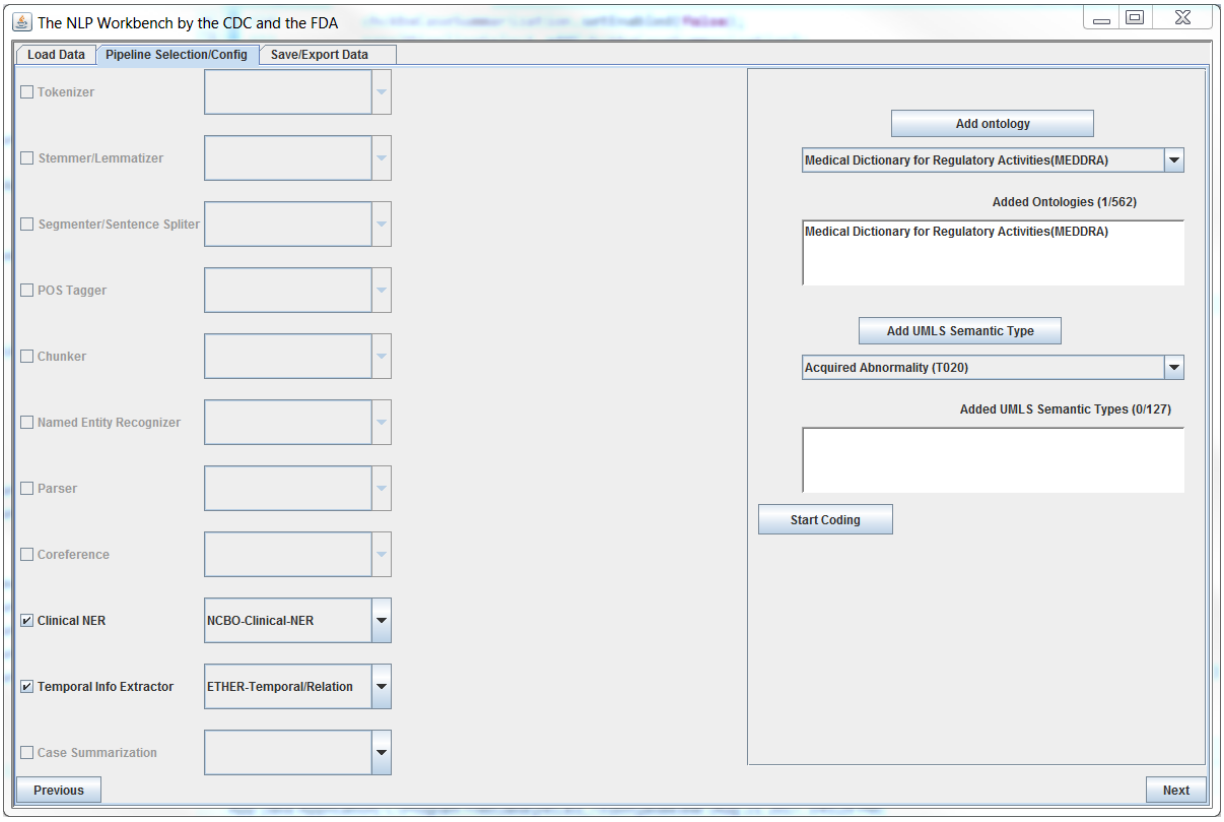


Figure J-3. The BioPortal Annotator using the MedDRA ontology and the ETHER Temporal selections are chosen for processing the text.

By comparing these two outputs, as in Figure J-4, we see that the two approaches have captured some of the same information, but that there are also differences in the annotated text. This demonstration simply shows the difference in outputs between pipelines and not performance evaluation.

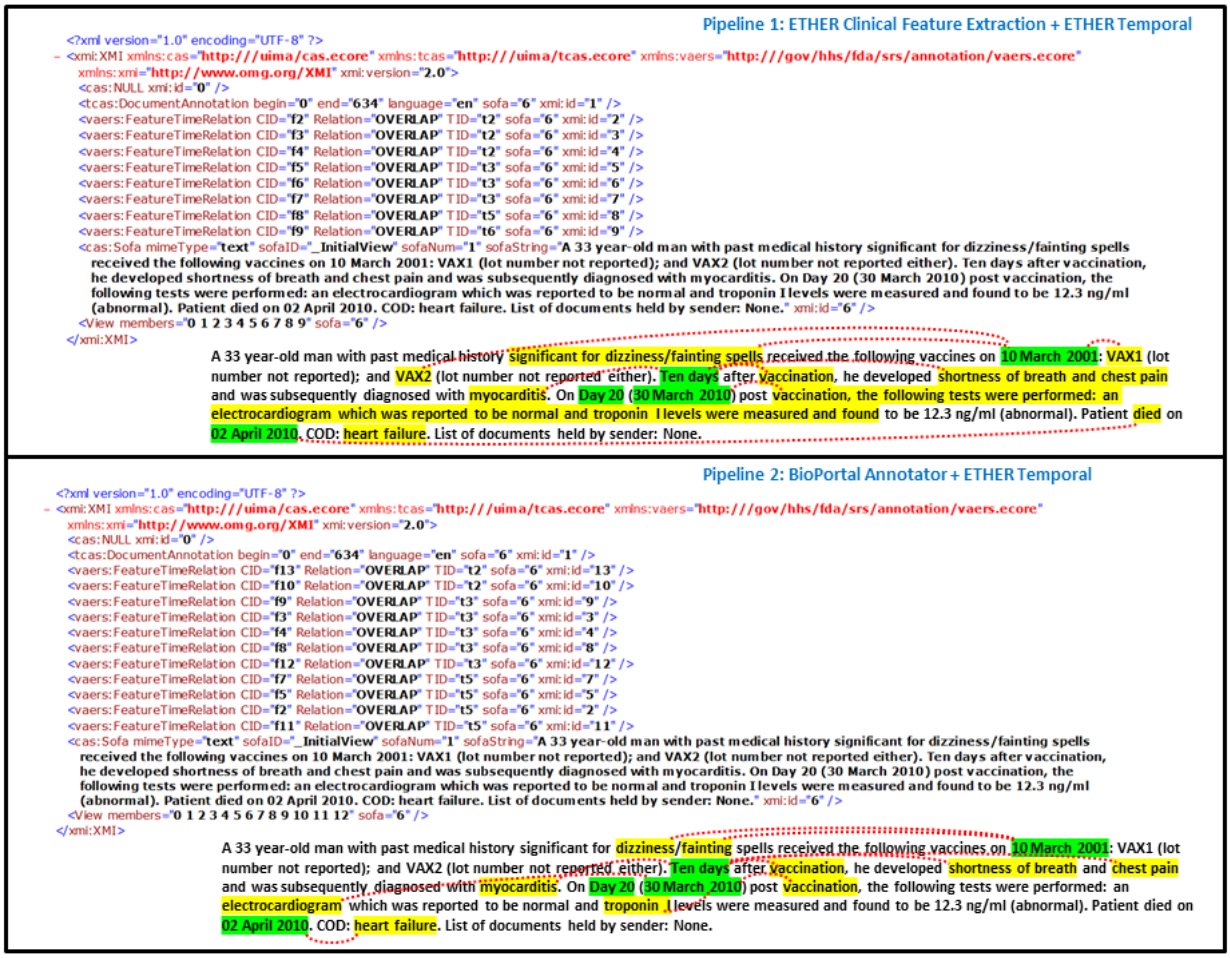


Figure J-4. A comparison of the extraction and temporal relation output from the two pipelines for the sample vaccine safety report. The output XML for the relations is shown for both pipelines, along with the full text of the report with highlighted text. The yellow highlighted text indicates features identified by clinical NER, and the green highlighted text indicates identified temporal expressions. The red dotted lines connect clinical features to temporal expressions, as described in the XML output for each pipeline.

J.2 Pathology Example

In the second example, we are identifying clinical information from a portion of the text of the sample pathology report provided in Section 4.1. We use only the first few sentences in the report because we expect the tools to mark many extra words and phrases throughout the report. The tools are not optimized for these reports, and the remaining text is not needed for this demonstration purpose. The first pipeline for the pathology report is:

- BioPortal Annotator (using SNOMED-CT ontology)

First, we load the text from the pathology report into the CLEW prototype, as shown in Figure J-4, then we set up the Workbench as shown in Figure J-5 to use the BioPortal Annotator with the SNOMED-CT ontology only. The Workbench then runs and produces an output XML file with annotated terms from the text.

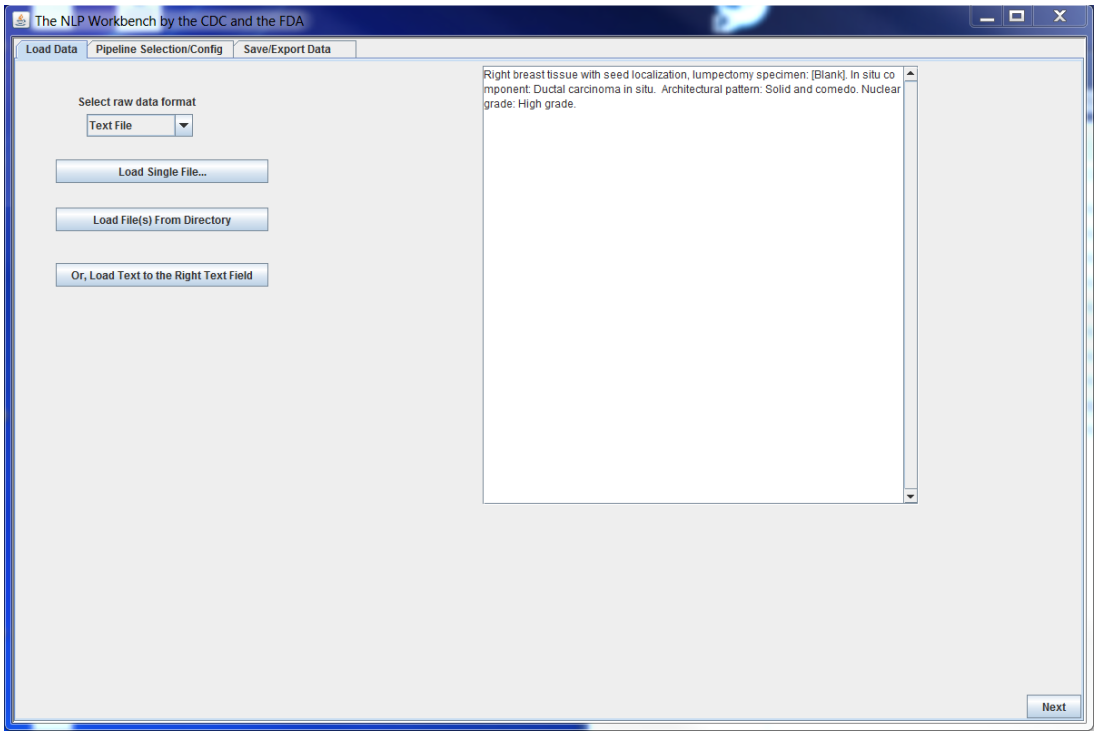


Figure J-5. The sample pathology report text being loaded as input text in the CLEW prototype.

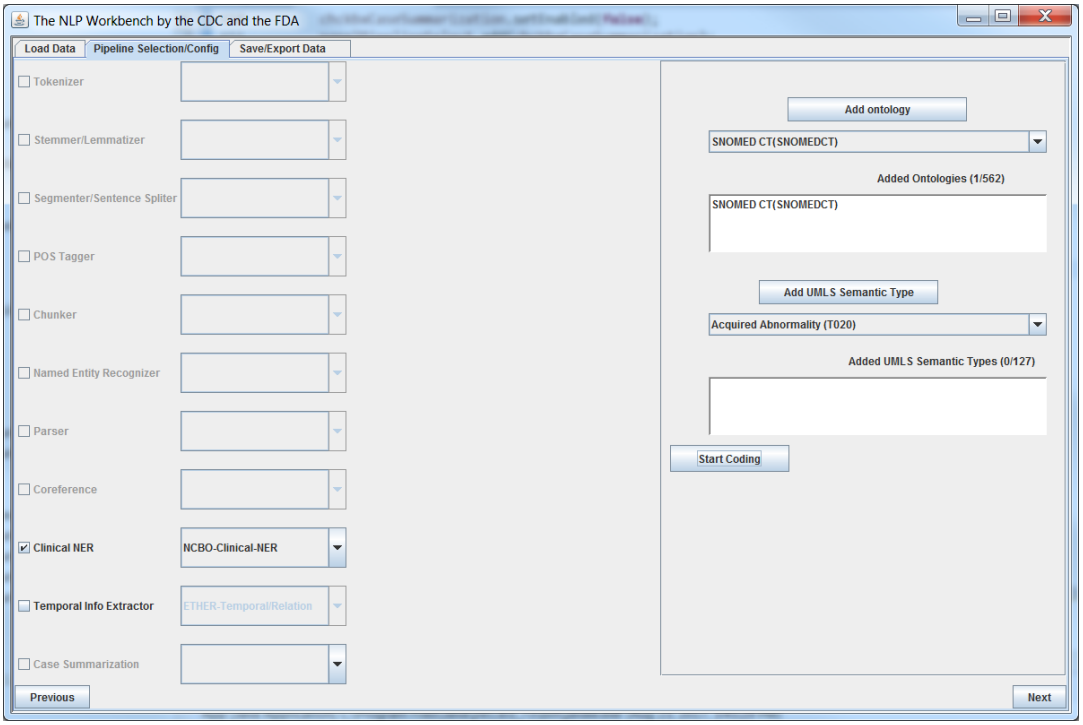


Figure J-6. The BioPortal Annotator using the SNOMED-CT ontology is chosen for processing the text.

For the sample pathology report text, the second pipeline is:

- cTAKES Clinical Feature Extraction

We again load the text from the pathology report into the CLEW prototype in the same manner. Then we choose the cTAKES tool to process the text, as shown in Figure J-7. When this pipeline is run in the Workbench, the cTAKES output showing the annotated output is produced.

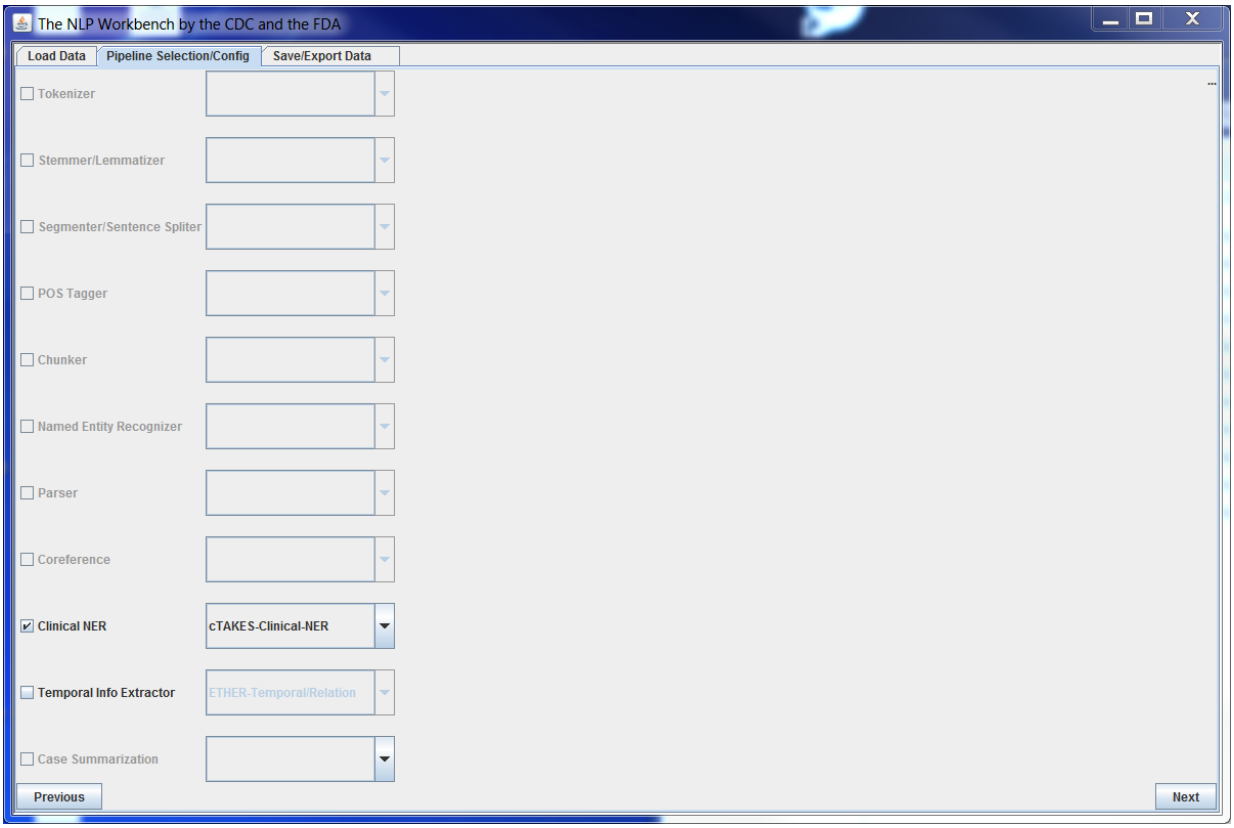


Figure J-7. The cTAKES selection is chosen for processing the text.

The outputs from these two pipelines are not in the same format, but they can still be compared to observe similarities and differences. Figure J-8 shows some of the text differences in what has been identified by the two different pipelines. Note that the performance of the various tools and pipelines has not yet been evaluated.

Figure J-8. A comparison of the feature extraction output from the two pipelines for the sample pathology report excerpt. The output formats are very different, but the extracted features from the reported are represented with highlighted text on the right side of each box.