



Beyond: the Usable Enterprise

Sean Finan, Timothy Miller, Chen Lin and Guergana Savova
Boston Children's Hospital and Harvard Medical School

1 Introduction

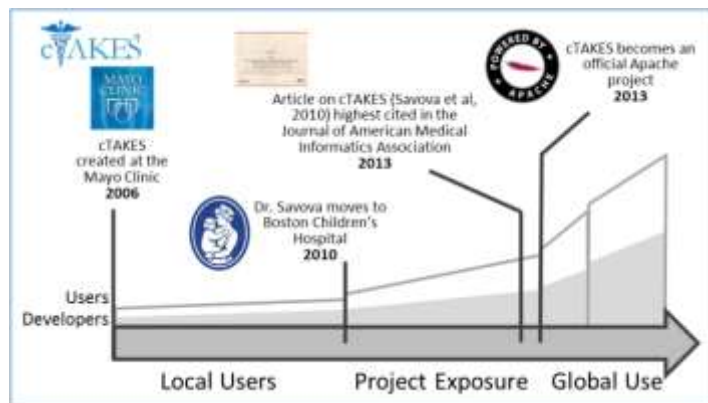
This pamphlet accompanied a hands-on training session for Apache clinical Text Analysis and Knowledge Extraction System (cTAKES; ctakes.apache.org) with a focus on usability. The intended audience includes Natural Language Processing (NLP) researchers, clinical researchers, software developers and anyone considering the use of NLP software for clinical research or other purposes.

Audience members can have any level of experience with cTAKES or NLP in general, as the session covers several topics and start from a basic foundation knowledge.



2 Background

cTAKES focuses on extracting knowledge from clinical text through NLP techniques. cTAKES can supply commonly extracted biomedical concepts such as symptoms, procedures, diagnoses, medications and anatomy with attributes and standard codes. cTAKES is engineered in a modular fashion to make extensions easy, and worldwide research investigations are continuously adding the latest, leading edge rule-based and machine learning probabilistic methods to cTAKES. These powerful components can perform tasks as complex as identifying temporal events, dates and times – resulting in placement of events in a patient timeline. For more details on the capabilities of cTAKES see section 10.7.



3 How to use this manual

This manual is not a *cTAKES for Dummies*, nor is it an *Expert's Guide to cTAKES*. It is an amalgamation, usable by three expertise levels based loosely upon the user's background and purpose.

- 1) **Beginner:** Curious about NLP, developing a project or using clinical notes for research.
- 2) **Intermediate:** Between levels 1 and 3
- 3) **Expert:** Using natural language processing, machine learning , cTAKES or UIMA.

Special terms used in this text can be found in the Glossary.

The style used in this manual follows.

Workflows:

- 1. General action and information.
- 2. **Software Action Button** or **GUI Label**.
- 3. *directory/path/, file/path, code.package.*

Command lines:

```
command option parameter_value
```

File contents:

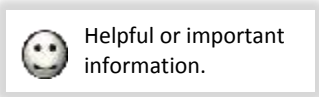
```
command parameter_value
```

```
parameter_name=parameter_value
```

Code snippets:

```
Class object = new Class( value );
object.method();
```

Table Title		
	Column Name	Column Name
Row Name	value	value
Row Name	value	value



Contents	
1	Introduction1
2	Background1
3	How to use this manual2
4	Installation3
4.1	Binary Release.....3
4.2	Dictionary Releases.....3
4.3	Code Repository.....4
4.4	cTAKES-the-API4
5	Pipelines.....5
5.1	Default Pipeline5
5.2	Custom Pipelines5
6	Dictionaries7
6.1	Dictionary Creator GUI7
6.2	BSV Dictionaries.....8
7	Analysis Engines8
8	Models8
9	Glossary.....10
10	Appendix.....11
10.1	Piper Files.....11
10.2	Collection Readers12
10.3	Analysis Engines.....12
10.4	Cas Consumers.....13
10.5	Dictionary Specification13
10.6	State of the Art14
10.7	Current Efforts15
11	Participation.....15
11.1	Get Involved.....15
11.2	The Authors16
12	References17

4 Installation

“I wouldn’t want them to operate a plane I was on with software that happened to be the latest greatest release.” - Nathan Myhrvold

4.1 Binary Release

The Binary release is a pre-built image of a cTAKES installation. It includes cTAKES and third party libraries, as well as a small number of sample notes. The current version is 3.2.2.

1. Visit <https://ctakes.apache.org>
2. **Click Download.**



 Prerequisites titles link to web pages.



3. Ensure you meet **Prerequisites: All Users.**
4. **Click User Installation** for your system.
5. **Unzip** the downloaded file.


“Words matter.” - John Kenney

4.2 Dictionary Releases

There are two primary downloadable bundles of Unified Medical Language System (UMLS) dictionaries usable by cTAKES. The first contains several very large databases, and is compressed to a single 650MB file. It is required if you wish to use the original Dictionary Lookup module (see Section 6). The second bundle contains a 15MB compressed file with a database that is only usable by the newer Fast Dictionary Lookup module. We strongly recommend using the Fast Dictionary Lookup module which is much faster at no loss of recall/precision. The smaller bundle does contain everything that is traditionally used in the standard cTAKES clinical pipeline, it has been streamlined with excess unused data removed.

1. Ensure you meet **Prerequisites: Dictionary Users.**
2. **Click UMLS Dictionary** for your desired version.
3. **Unzip** in your binary installation
resources/org/apache/ctakes/dictionary/lookup
4. For the **Fast Version**, **unzip** in subdirectory
fast/ctakesnorx/



 For the latest dictionary downloads, visit <https://sourceforge.net/projects/ctakesresources/files/>

“There is no intelligence where there is no need of change.” - H.G. Wells

4.3 Code Repository

A great number of fixes and new features have been added to the working code online Subversion repository (SVN repo).

Check out the latest and greatest.

1. Ensure you meet **Prerequisites: Developers**.
2. **Execute**

```
svn co https://svn.apache.org/repos/asf/ctakes/trunk
```



“Prosperity is always just around the corner.” - Herbert Hoover

4.4 cTAKES-the-API

The binary installation of cTAKES and the code repository involve a large amount of cTAKES and third party code and resources. Avoid the large disk footprint and update times by including only the modules that you need in your NLP project. Include cTAKES-the-API as a dependency in your project and uncomment needed cTAKES modules within its *pom.xml*.

1. Ensure you meet **Prerequisites: Developers**.
2. **Execute**

```
svn co https://svn.apache.org/repos/asf/ctakes/sandbox/ctakes-the-api
```



cTAKES Installation Types			
	Binary Installation	Code Repository	cTAKES-the-API
Intended User	Non-developer	Developer	Developer
File Types	Compiled libraries	Java code	Compiled libraries
Versatility	Resources	Editable Code	Selectable Modules
Features	Released Only	Latest, Greatest	Latest, Greatest
Bugs	Bugs documented	Documented bugs fixed	Documented bugs fixed
Disk Footprint	630 MB	3,850 MB	40 MB

5 Pipelines

“Without strategy, execution is aimless.
Without execution, strategy is useless.” - Morris Chang

5.1 Default Pipeline

The default clinical pipeline performs Entity Recognition and identifies Entity Properties.

The default pipeline is a great first step for beginners, and advanced capabilities of cTAKES can be added after experience.

The patient underwent a CT scan in April which did not reveal lesions in his liver.

Entity Recognition	CT scan Procedure UMLS ID: C0040405	Lesion Disease / Disorder UMLS ID: C0022198	Liver Anatomy UMLS ID: C0023884
Entity Properties	Negated: no Subject: patient	Negated: yes Subject: patient	Negated: no Subject: --

Run the default pipeline via command line.

1. Execute

```
bin/runClinicalPipeline -i inputDirectory
                        --xmiOut outputDirectory
                        --user umlsUsername
                        --pass umlsPassword
```

Browse annotations and properties with the CVD.

1. Execute

```
bin/runtakesCVD
```

2. Select File > Read Type System File.

3. Select *TypeSystem.xml* in

resources/org/apache/ctakes/typesystem/types/

4. Select File > Read XMI CAS File.

5. Select any *.xmi* file in your *outputDirectory*.



“You shape your own destiny.” – Chet Atkins

5.2 Custom Pipelines

Create custom pipelines to extract more information than is available through the Default Clinical Pipeline.

Special Analysis Engines are in various cTAKES modules. Analysis Engines can be removed or added to pipelines to obtain desired results.

There are four methods available to create custom pipelines.

The patient underwent a CT scan in April which did not reveal lesions in his liver.

UMLS Relation	Lesions LOCATED AT liver	
Temporal Relations	CT scan WITHIN April	Lesions WITHIN CT scan
Coreferences	patient SAME AS his	

1. XML descriptor files are the original method used to create pipelines in UIMA. They are verbose and editing is error prone.



2. UimaFit enables creation of pipelines through Java code.

This greatly simplifies unit testing and experimentation.

```
JCas jcas = JCasFactory.createJCas();
CollectionReader reader = CollectionReaderFactory.createReader(
    FilesInDirectoryCollectionReader.class,
    FilesInDirectoryCollectionReader.PARAM_INPUTDIR,
    "my/input/dir" );
AggregateBuilder builder = new AggregateBuilder();
builder.add( ClinicalPipelineFactory.getTokenProcessingPipeline() );
builder.add( ExampleHelloWorldAnnotator.createAnnotatorDescription() );
SimplePipeline.runPipeline( jcas,
    reader,
    builder.createAggregateDescription() );
```


3. **PipelineBuilder** is a facade for UimaFit factories and objects.

```
PipelineBuilder builder = new PipelineBuilder();

builder.readfiles( "my/input/dir" )
    .add( ClinicalPipelineFactory.getTokenProcessingPipeline() )
    .add( ExampleHelloWorldAnnotator.class )
    .run();
```

4. Piper files are a modern equivalent of the descriptor XML files.

Piper files list basic commands and parameters.

```
readFiles my/input/dir
load DefaultTokenizerPipeline.piper
add ExampleHelloWorldAnnotator  HelloWorld.piper
```

Piper files can be run programmatically.

```
PiperFileReader piperReader = new PiperFileReader( "HelloWorld.piper" );
PipelineBuilder builder = piperReader.getBuilder();
builder.run();
```

Piper files can be run from the command-line.

```
bin/runPiperFile -p HelloWorld.piper
```

Pipeline Customization Options			
	Best Use	Edit	User
XML Descriptor	Run Existing	Xml File	Non-developer
UimaFit Builder	Create New	Code	Developer
PipelineBuilder	Create New	Code	Developer
Piper File	Run / Create	Text File	Non-developer

“The faster you go, the shorter you are.” – Albert Einstein

6 Dictionaries

There are two dictionary lookup modules in cTAKES. The original dictionary lookup module (Old) has a large disk footprint and is very slow. A newer dictionary lookup module (Fast) has improved speed, decreased disk footprint, and the same or better accuracy. It also is more customizable.

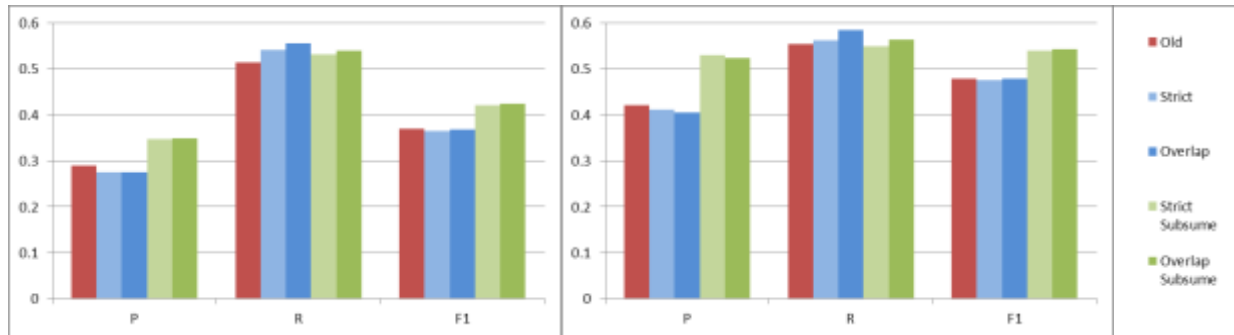
The Fast dictionary can run strict or overlap span matching.

It can also run with subsumption of smaller into larger semantically-alike spans.

Seconds per Note		
	Share	Sharp
Old	10.843	6.515
Fast	0.019	0.012

Disk Size	
	MB
Old	602
Fast	53

Configuration can be further defined, but that is outside the scope of this document.



Precision, Recall, F1-score of dictionary lookup configurations on Share and Sharp corpora.

Strict span matching is performed by the default annotator. Use `OverlapJCasTermAnnotator` to enable overlap matching. Subsumption is not performed by the `DefaultTermConsumer`. Use `PrecisionTermConsumer` to enable semantically-alike subsumption.

“A User Interface is like a joke. If you have to explain it, it’s not that good.”

6.1 Dictionary Creator GUI

The Fast Dictionary Lookup module can use custom dictionaries created by the cTAKES Dictionary Creator.

This requires a local copy of UMLS .rrf files.

1. Execute

```
svn co https://svn.apache.org/repos/asf/ctakes/sandbox/dictionary-gui
```

2. Compile the code.

3. Execute class

```
org.apache.ctakes.dictionary.creator.gui.CreatorGui
```

4. Select a cTAKES Installation directory.

5. Select a UMLS Installation directory.

6. Select Source and Target Vocabularies.

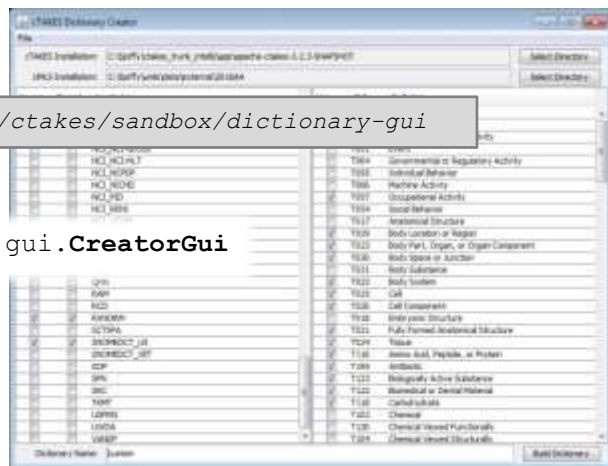
7. Select Semantic Type TUIs.

8. Type a Dictionary Name. Use all lower case.

9. Click Build Dictionary.

10. Set fast dictionary parameter DictionaryDescriptor to

```
resources/org/apache/ctakes/dictionary/lookup/fast/DictionaryName/DictionaryName.xml
```



“Simplicity is the ultimate sophistication.” – Leonardo Da Vinci

6.2 BSV Dictionaries

Create small custom dictionaries with simple text files. A bar-separated-value (pipe-delimited) file with two, three, or four columns can be used.

```
// code|text
1|arm
2|left arm
3|right arm
```

```
// code|tui|text
5|21|foot
6|21|left foot
7|21|right foot
```

```
// code|tui|text|preferred text
8|21|end of forearm|hand
9|21|digit|finger
3|21|volar|palm
```

The tui must be specified as its integer value or that value with the prefix T. The code can be represented by anything that ends with an integer. *1*, *C001* and *Appendage1* are all valid codes.

Add a BSV dictionary with the dictionary specification .xml file.

“Analysis does not transform consciousness.” – Jiddu Krishnamurti

7 Analysis Engines

Analysis Engines are the pieces of functionality that make up a pipeline. Create them using UimaFit.

1. Extend `org.apache.uima.fit.component.JCasAnnotator_ImplBase`.
2. Implement `process(JCas jcas)`.
3. Use `ConfigurationParameter` annotations if parameters are required.
4. Implement `initialize(UimaContext context)` if engine setup is required.

```
class MyAE extends JCasAnnotator_ImplBase {

    @ConfigurationParameter( name="word", description="Word to look for" )
    private String word;

    public void process( JCas jcas ) throws AnalysisEngineProcessException {
        if ( jcas.getDocumentText().contains( word ) ) {
            log( "Found " + word );
        }
    }
}
```

“Train yourself to let go of everything you fear to lose.” – Yoda

8 Models

Previously, almost all modules were wrappers for Apache OpenNLP components, and used OpenNLP tools to train the models. Many of our modules still work this way, but we have introduced new ways to train machine learning models that offer more choice to developers.

Many of the newest components use ClearTK APIs to interface with machine learning libraries. ClearTK provides a uniform API for things like features, training instances, and learning algorithms while linking to many common machine learning libraries like LibSVM, LibLinear, Mallet, etc.

The ClearTK paradigm is to write analysis engines that inherit from `CleartkAnnotator`, which inherit a Boolean method `isTraining()`. The analysis engine then extracts features, and if it is training time, uses

an inherited **DataWriter** to write the instance features and its label. If it is not training time, it uses an inherited **Classifier** to classify the instance features. The returned object from the classification is then converted into a UIMA typesystem object and added to the CAS in the usual way.

ClearTK features typically just map from UIMA types in the CAS to a **Feature** object. For example, to classify the current token's part of speech, we could query the CAS for the previous **BaseToken** and create a feature with `new Feature("PrevToken", prevToken.getCoveredText());` Both the relevant **DataWriter** and **Classifier** methods take a **List of Features**. It is common to create separate feature extractor classes and call something like `features.addAll(extractor.extract())`. Again, ClearTK provides feature extractors for many common feature types. The sentence detector demo contains some of the simplest possible feature extractions. See a class like `org.apache.ctakes.temporal.ae.TimeAnnotator` for an example of a more sophisticated feature extraction.

Creating a new classifier requires creating a pipeline that reads gold standard data and instantiates the annotator in training model. There are several ways of storing and reading annotations -- xml formats (Anafora, Knowtator), offset formats (Conll), and ad hoc formats. cTAKES has readers for several formats and pipelines for re-training many of its modules. The code for building and evaluating models can also make use of ClearTK workflows – an **Evaluation_ImplBase** class that defines methods one must override for building training and testing pipelines. For building these pipelines, we most commonly use **UimaFIT** as it allows for programmatically creating pipelines, allowing for internal logic, e.g., when evaluating with different feature sets.

Modules with Trainable Components			
Module name	Components	Readers/Writers	API
Assertion	Negation, uncertainty, historyOf, generic, conditional, subject	I2B2 Challenge, MiPACQ, Negex	ClearTK
Chunker	Shallow parser (chunker)	OpenNLP Chunker model	OpenNLP
Constituency Parser	Deep syntactic parser	OpenNLP Parser model	OpenNLP
Core	Sentence segmenter	SHARP reader	OpenNLP
Coreference	Entity linking (coreference resolution)	CoNLL Writer	ClearTK
Dependency	Syntactic dependency parser	ClearNLP models	ClearNLP
POS	Part of speech tagger	OpenNLP POS tag model	OpenNLP
Relation Extractor	LocationOf, Severity relation extractors	SHARP reader	ClearTK
Temporal	Events, Time expressions, Container relations	THYME readers (Anafora, Knowtator, Treebank) TempEval writer	ClearTK

9 Glossary

Term	Definition
UIMA	Unstructured Information Management Architecture. The software framework used by cTAKES. See uima.apache.org
Cas, JCas	Contains all of the information gleaned from the document. In the beginning it is empty. At the end of the pipeline it is full of informative goodness.
Collection Reader (CR)	Inputs text for a pipeline.
Analysis Engine (AE)	Performs one or more actions within a pipeline.
Cas Consumer (CC)	Outputs information from a pipeline.

10 Appendix

10.1 Piper Commands

Piper Commands			
Command	Parameter 1	Parameters 2-n	Description
load	Piper file path	-	Load external piper file
set	name=value	<name=value ...>	Add global parameters
Cli	name=char	<name=char...>	Add a global parameter based upon command line character option value
addPackage	Package specification	-	Add to known packages
add	AE or CC class name	<name=value ...>	Add AE/CC to pipeline
addDescription	AE or CC class name	<value ...>	Add AE/CC with .createDescription() method to pipeline
addLogged	AE or CC class name	<name=value ...>	Add AE/CC to pipeline with Start/Finish logging
addLast	AE or CC class name	<name=value...>	Add AE/CC to end of pipeline. Useful if the pipeline is meant to be extended
reader	CR class name	<name=value ...>	Specify collection reader for input data
readFiles	<input directory>	-	Use Files in Directory collection reader
writeXmis	<output directory>	-	Write XMI files to output directory
// or #	Comment Text	-	Line comment

10.2 Collection Readers

Collection Readers		
Name	Description	Parameters
FilesInDirectoryReader	Reads text files in a directory	
FileTreeReader	Reads text files in a directory tree	
JdbcCollectionReader	Reads text in a database	

10.3 Analysis Engines

Analysis Engines		
SimpleSegmentAnnotator	Creates a segment enclosing the entire document	
RegexSectionizer	Uses Regex lines in BSV file to create segments	
ParagraphAnnotator	Uses Regex or multiple \n to create paragraphs	
ListAnnotator	Uses Regex to create Lists	

“The secret of being boring is to say everything.” – Voltaire

10.4 Cas Consumers

CAS Consumers		
XmiFileWriter	XMI files usable by the CVD.	
PrettyTextWriter	Decorated note ASCII text.	
PrettyHtmlWriter	Decorated note html.	
PropertyTextWriter	Property list text.	

10.5 Dictionary Specification

```
<lookupSpecification>
<ictionaries>
  <dictionary>
    <name>MyCustomWords</name>
    <implementationName>org.apache.ctakes.dictionary.lookup2.dictionary.BsvRareWordDictionary</implementationName>
    <properties>
      <property key="bsvPath" value="mycustomfile.bsv"/>
    </properties>
  </dictionary>
</ictionaries>
<conceptFactories>
  <conceptFactory>
    <name>MyCustomCodes</name>
    <implementationName>org.apache.ctakes.dictionary.lookup2.concept.BsvConceptFactory</implementationName>
    <properties>
      <property key="bsvPath" value="mycustomfile.bsv"/>
    </properties>
  </conceptFactory>
</conceptFactories>
<dictionaryConceptPairs>
  <dictionaryConceptPair>
    <name>MyCustomPair</name>
    <dictionaryName>MyCustomWords</dictionaryName>
    <conceptFactoryName>MyCustomCodes</conceptFactoryName>
  </dictionaryConceptPair>
</dictionaryConceptPairs>
<rareWordConsumer>
  <name>TermConsumer</name>
  <implementationName>org.apache.ctakes.dictionary.lookup2.consumer.DefaultTermConsumer</implementationName>
  <properties>
    <property key="codingScheme" value="custom"/>
  </properties>
</rareWordConsumer>
</lookupSpecification>
```

📄 Dictionary Specification File

10.6 State of the Art

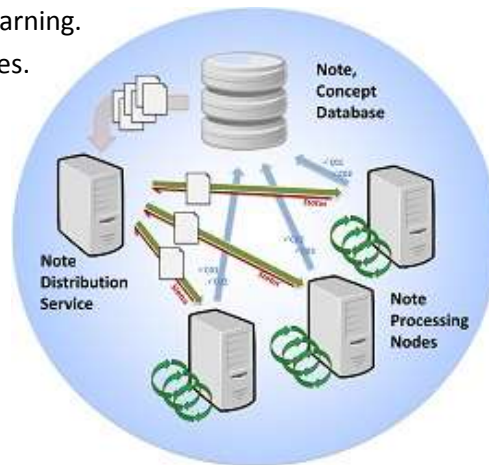
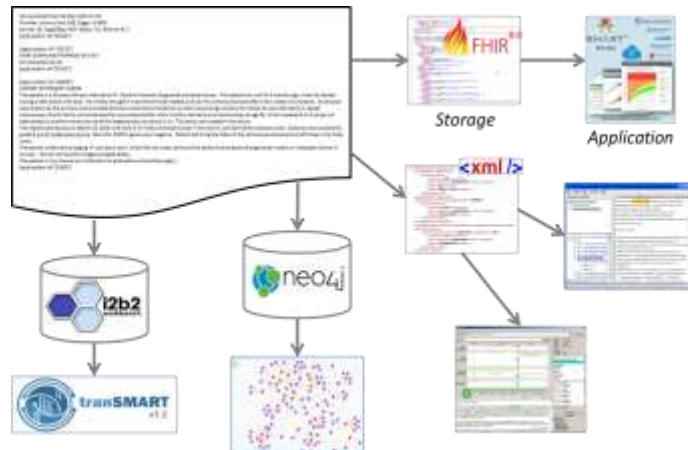
cTAKES implements a full stack of critical components. These powerful components perform duties as simple as locating sentence breaks and as complex as the extremely important task of identifying temporal events, dates and times – resulting in the absolute and relative placement of events in a patient timeline. cTAKES is also unique in that components are trained on gold standards from the biomedical as well as general domain. This affords usability across different types of clinical narrative (e.g. radiology reports, clinical notes, discharge summaries) in various institution formats as well as other types of health-related narrative (e.g. Twitter feeds), using multiple data standards (e.g. Health Level 7 (HL7), Clinical Document Architecture (CDA), Fast Healthcare Interoperability Resources (FHIR), SNOMED-CT, RxNORM). For cTAKES integration with popular tools/formats, see Figure 1. For details on some of the projects where cTAKES is being extended see www.thyme.healthnlp.org , www.cancer.healthnlp.org , www.share.healthnlp.org .

cTAKES Component or Function	Score	Score Type
Sentence boundary (1)	0.949	Accuracy
Context sensitive tokenizer (1)	0.949	Accuracy
Part-of-speech tagging (1) (2)	0.936 – 0.943	Accuracy
Shallow parser (1)	0.952 ; 0.924	Accuracy ; F1
Entity recognition (1)	0.715 / 0.824	F1 ¹
Concept mapping (SNOMED CT and RxNORM) (1)	0.957 / 0.580	Accuracy ¹
Negation NegEx (3) (1)	0.943 / 0.939	Accuracy ¹
Uncertainty, modified NegEx (3) (1)	0.859 / 0.839	Accuracy ¹
Constituency parsing (4)	0.810	F1
Dependency parsing (2)	0.854 / 0.833	F1 ²
Semantic role labeling (2)	0.881 / 0.799	F1 ³
Coreference resolution, within-document (4)	0.352 ; 0.690 ; 0.486 ; 0.596	MUC ; B [^] 3 ; CEAF ; BLANC
Relation discovery (5)	0.740-0.908 / 0.905-0.929	F1 ⁴
Events (publication in preparation)	0.850	F1
Temporal expression identification (6)	0.750	F1
Temporal relations: event to note creation time (7)	0.834	F1
Temporal relations: on i2b2 challenge data (7)	0.695	F1

Table 1: Evaluation of cTAKES components by established standard metrics. ¹ Exact/Overlap span; ² Unlabeled/labeled attachment; ³ Argument/with classification; ⁴ locationOf/degreeOf

10.7 Current Efforts

- Deep phenotyping.
 - see deepphe.healthnlp.org
- Cross-document coreference.
- Cross-document timeline creation.
 - see thyme.healthnlp.org
- Adverse Event detection.
- Question-Answering.
- Social media integration.
- Asynchronous scale-out.
- i2b2 database reader, writer.
- Deep Learning libraries (e.g. keras).
- Performance improvement with Deep Learning.
- Ontology web language (OWL) dictionaries.
- Word-sense disambiguation.
- Computable phenotypes.
- Docker Containerization.
- GPU utilization.



11 Participation

"Go vote!" - President Barack Obama

11.1 Get Involved

cTAKES is open source software, and developer contribution is welcome. cTAKES is also software built with purpose, and user contribution is encouraged. Contributions can be improvement suggestions, bug reports, documentation, and questions as they provide developers information and direction.

1. Visit <https://ctakes.apache.org>
2. **Select Resources > Get Involved.**

cTAKES has active mailing lists used to post questions and answers.

1. Visit <https://ctakes.apache.org>
2. **Select Resources > Mailing Lists.**



11.2 The Authors

Contact us at FirstName.LastName@childrens.harvard.edu -- Sean Finan, Timothy Miller, Guergana Savova.

Sean Finan is a lead software developer at Computational Health Informatics Program (CHIP) located in Boston Children's Hospital with over twenty years of experience architecting software for academic research and professional use. He has participated in a number of large scale software projects and developed several software tools – some under the Apache cTAKES umbrella and others outside it. He has created advanced topic tutorials for large software companies, trained professional developers and is a member of numerous local and international groups on Java, several focusing on esoteric topics such as high performance, concurrency and thread safety. He has received expert level training in and used many techniques and tools for agile development; is an early signatory of the manifesto for agile software development, as well as a member of user groups devoted to agile development and principles.

Dr. Timothy Miller is faculty at Computational Health Informatics Program (CHIP) located in Boston Children's Hospital. He works in natural language processing (NLP) of clinical text, extracting information from medical records to facilitate clinical research and make the healthcare system more efficient. Dr. Miller is trained as a computer scientist. In the general domain, natural language processing (NLP) is usually applied to standard corpora including financial newswire text and a few other canonical sources, to the extent that the whole field is probably overtrained on these few data sources. Dr. Miller is interested in doing clinical NLP research to broaden the usage and development of NLP models to new domains, and the domain of clinical research is especially exciting because of the direct impact it can have on people's lives. Dr. Miller is interested in applying statistical models of human language to data in the electronic health record. Specifically, he is currently interested in tree kernels for support vector machines, constituency parsing and features derived therefrom, clinical domain adaptation, generative (Bayesian) models of text generation, and coreference resolution.

Mr. Chen Lin is a trained computer scientist with machine learning expertise. He is an informatician at Computational Health Informatics Program at Boston Children's Hospital. Mr. Chen has been investigating cutting edge machine learning methods and applying them to complex NLP tasks such as temporal relation extraction. He has generously contributed the best methods to cTAKES temporality module. Recently Mr. Chen has been fascinated by the resurgence of neural networks (a.k.a. deep learning) and has been researching varied architectures for higher level tasks – a topic to which neural networks have not been explored so far. He is also interested in using the NLP output to tasks of interest to the clinical investigators such as automatic discovery of disease activity from the electronic medical record, automatic discovery of medication adverse events from the electronic medical record, quality metrics mining.

Dr. Guergana Savova is Associate Professor at Harvard Medical School and Computational Health Informatics Program at Boston Children's Hospital. She is the Principal Investigator of the Natural Language Processing Lab. Before joining Boston Children's Hospital and Harvard Medical School in 2010, Dr. Savova was faculty at the Biomedical Statistics and Informatics Department, Mayo Clinic (2002-2010). Her research interests are in natural language processing (NLP) and information extraction especially as applied to the text generated by physicians (the clinical narrative). Dr. Savova has been creating gold standard annotated resources based on computable definitions and developing methods for computable solutions. The focus of

Dr. Savova's research is higher level semantic and discourse processing of the clinical narrative which includes tasks such as named entity recognition, event recognition, relation detection and classification including coreference and temporal relations (thyme.healthnlp.org; share.healthnlp.org; cancer.healthnlp.org). The methods are mostly machine learning spanning supervised, lightly supervised and completely unsupervised. The result of Dr. Savova's research with her collaborators has led to the creation of the clinical Text Analysis and Knowledge Extraction System (cTAKES; ctakes.apache.org). Dr. Savova has been the principal of cTAKES since its inception.

12 References

1. Savova GK, Masanz JJ, Ogren PV, Zheng J, Sohn S, Kipper-Schuler KC, et al. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *J Am Med Inform Assoc JAMIA*. 2010 Oct;17(5):507–13.
2. Albright D, Lanfranchi A, Fredriksen A, Styler WF, Warner C, Hwang JD, et al. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *J Am Med Inform Assoc JAMIA*. 2013 Oct;20(5):922–30.
3. Chapman WW, Bridewell W, Hanbury P, Cooper GF, Buchanan BG. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*. 2001 Oct;34(5):301–10.
4. Zheng J, Chapman WW, Miller TA, Lin C, Crowley RS, Savova GK. A system for coreference resolution for the clinical narrative. *J Am Med Inform Assoc JAMIA*. 2012 Aug;19(4):660–7.
5. Dligach D, Bethard S, Becker L, Miller T, Savova GK. Discovering body site and severity modifiers in clinical texts. *J Am Med Inform Assoc JAMIA*. 2014 Jun;21(3):448–54.
6. Miller T, Dligach D, Bethard S, Pradhan S, Lin C, Savova G. Discovering Time Expressions in Clinical Text. In: Annual symposium of the American Medical Informatics Association [Internet]. Washington, DC, USA; 2013 [cited 2015 Mar 27]. Available from: <http://knowledge.amia.org/amia-55142-a2013e-1.580047/t-03-1.584514/f-003-1.584515/a-342-1.584603/a-355-1.584598?qr=1>
7. Chen L, Dligach D, Miller T, Bethard S, Savova G. Layered temporal modeling for the clinical domain. *J Am Med Inf Assoc JAMIA*. 2015;