

ClearTK: A UIMA Toolkit for Statistical Natural Language Processing

Philip V. Ogren, M.S.

Philipp G. Wetzler, M.S.

Steven J. Bethard, Ph.D.

**Computational Language and Education Research
(CLEAR)**

University of Colorado at Boulder

Overview

- Introduction
- Feature extraction
- Machine Learning
 - Training
 - Classifying

Introduction

- Software in CSLR
 - Perl, C++, Python, Java, TCL
 - Supports interesting research papers
 - Works only if you are the person who wrote it sitting at the computer that it was developed on.
- Who wants it?
 - Students, collaborators, industry
 - nobody
- Rewriting code that should exist as shared infrastructure

Introduction

- Changed name to Clear – need software that reflects name change!
 - Documented
 - Distributable / Easy to install
 - Reusable / Foundational
 - Tested
 - Useful

ClearTK

- Framework for Statistical NLP
 - Wrappers for ML libraries
 - OpenNLP Maxent, Mallet CRF, LIBSVM, SVM^{light}
 - Feature extraction library
 - Windowed features, syntactic path, ngrams, etc.
 - No dependence on a type system
- Components
 - Collection Readers (eg ACE, CoNLL, Genia, PTB, Propbank)
 - Chunker, SRL, POS tagging, dictionary matching
 - Wrappers
 - independent of type system in some cases (e.g. chunker and tokenizer)

ClearTK

- Unit Tests!
 - 195 test suites
 - 2400 assertions
 - 63% “code coverage” (EclEmma)
 - Infrastructure
 - How do you unit test methods of an annotator?
- Released
 - <http://clear.colorado.edu/ClearTK>
 - Version 0.9
 - 1.0 expected by end of the summer
 - Free for research-use license
 - With source code (linked from javadocs)

Feature Extraction

- Feature extractor
 - Anything that returns instances of Feature
 - Feature is a POJO
 - Name
 - Value
 - Contextual information specified by subclasses
- SimpleFeatureExtractor
 - `List<Feature> extract(JCas jCas, Annotation focusAnnotation)`

Simple Feature Extractor

- Spanned text extractor
- “type path” extractor
 - How to get a feature value from an annotation
 - “pos”, “concept/id”,
 - `de.julielab.jules.types.Token`
 - “depRel/head/posTag/value”
 - Lacks ability to query
 - Pos tag where tagsetId = “PTB”
 - Looking for advice about this
 - Workaround is write a feature extractor that extracts just exactly what you need (easy!)

Window Feature Extractor

- Initialize with
 - Type from which features are extracted
 - Scope/orientation
 - simple feature extractor
- Extract with
 - Focus annotation
 - Bounds annotation – defines “out-of-bounds”
- Example
 - Initialize with
 - Token.class
 - Start index = 0, End index = 3, Orientation = LEFT
 - Type path extractor (path = “pos”)
 - Extract with
 - Focus = NamedEntityMention
 - Bounds = Sentence.class
- Other Extractors
 - WindowNGramExtractor
 - BagExtractor
 - NGramExtractor

Feature Extractors

- Distance
- Syntactic path (type system dependent)
- Head word
- White space
- Relative position
- Proliferators
 - Capitalization
 - Character n-gram
 - Contains hyphen
 - Lower case
 - Numeric type

Machine Learning

- Instance
 - Features
 - Outcome
- Classifier
 - Wrapper around ML library
 - `T classify (Instance) , T classify(List<Instance>)`
- Training data writer
 - Writes instances out in format consumable by learner
 - `consume(Instance), consume(List<Instance>)`

Machine Learning

- Instance Consumer
 - T consume(Instance), T consume(List<Instance>)
 - ClassifierAnnotator – delegates to classify methods
 - Training data writer - produces data in format suitable for ML model building
- Annotation Handler
 - Takes an InstanceConsumer
 - Handles logic of
 - Iterating through annotations of interest
 - Performing feature extraction
 - Calling instance consumer

Example

```
for each sentence
  for each token
    extract features for token
    create Instance for token
  results = consumer.consume(instances)
if results not null
  do something with results
```

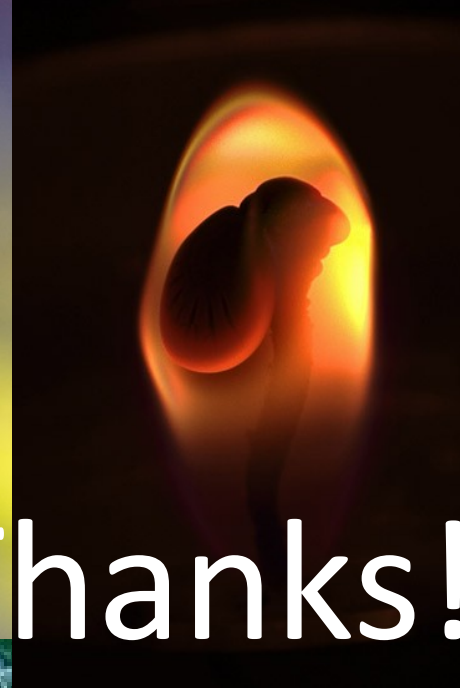
- InstanceConsumer / AnnotationHandler optional
 - Can use classifier wrappers and feature extraction libraries without these constructs
 - Without them, we found ourselves writing two versions of the annotation handling logic.

Components

- ChunkerHandler
 - Takes ChunkLabeler (chunks2Labels, labels2Chunks)
- Corpora
 - Ace 2005, CoNLL 2003, CoNLL 2005, Genia, PTB, Probank
- Dictionary term finder
 - Finds exact matches from term list
 - fast
 - 2M terms, 500K words, 8.25 seconds for tokenization, .25 seconds for term matching (8K terms matched).
 - Good for features...
- Tokenizer
 - Regex / space insertion, chunker
- Semantic Role Labeling

Future directions

- Version 1.0
 - Unit test, unit test, unit test
 - Documentation
 - A fair bit is currently deprecated (or should be)
 - Plan to be backwards compatible from 0.9
 - Release notes will detail any incompatibilities
- Beyond
 - Integrate dependency parsing
 - Improve semantic role labeling



Thanks!