cTAKES 3.0 - Dependency Parser and Semantic Role Labeler

{scrollbar}

65%Contents of this Page2Menu cTAKES 3.0 to Include

Overview of Dependency Parser

Dependency parsers provide syntactic information about sentences. Unlike deep parsers, they do not explicitly find phrases (e.g., NP or VP); rather, they find the dependencies between words. For example, "hormone replacement therapy" would have deep structure:

(NP (NML (NN hormone) (NN replacement)) (NN therapy))

but its dependency structure would show that "hormone" depends on "replacement" and "replacement" in turn depends on "therapyl" Below, the first column of numbers indicates the ID of the word, and the second number indicates what it is dependent on.

23 hormone hormone NN 24 NMOD 24 replacement replacement NN 25 NMOD 25 therapy therapy NN 22 PMOD

Dependency parses can be labeled as well, e.g., we could specify that "hormone" is in a noun-modifier (i.e., NMOD) relationship with "therapy" in the example above (the last column).

This project provides a UIMA wrapper and some utilities for ClearParser, a transition-based dependency parser that achieves state-of-the-art accuracy and speed.

The implementation in cTAKES v1.1 is based on revision 75.

ClearParser is described in:

"K-best, Locally Pruned, Transition-based Dependency Parsing Using Robust Risk Minimization." Jinho D. Choi, Nicolas Nicolov, *Collections of Recent Advances in Natural Language Processing V*, 205-216, John Benjamins, Amsterdam & Philadelphia, 2009.

Dependency parses often assume lemmas (normalized word forms) and POS tags as input. This cTAKES component infers lemmas and POS tags from upstream LVG and POS tagger components.

Overview of Semantic Role Labeler

The semantic role labeler assigns the predicate-argument structure of the sentence. (Who did what to whom when and where.)

Analysis Engines and other Descriptors

analysis_engine/ClearParserAE.xml

This analysis engine wraps the dependency parser's prediction function (i.e., finding dependency trees from text). It takes lemmas and POS tags from the normalizedForm and partOfSpeech attributes of BaseTokens that have been found in cTAKES (i.e., are in the CAS). This is the analysis engine that should be dropped into any new pipelines to get dependency parses.

Parameters

DependencyModelFile
the file that contains the ClearParser transition model
the file that contains the features (leave this as feature.xml unless
you want to modify the internals of the dependency parser)

LexiconDirectory
the directory that contains ClearParser tagsets, etc.

MorphDictionaryDirectory
leave empty to use POS tags from cTAKES. Enter en_dict to use the Clear Morphological Analyzer.

analysis_engine/ClearParserPlaintextAggregate.xml

An aggregate engine appropriate for use with CVD or other tools that act directly on plain text.

analysis_engine/ClearParserTokenizedAggregate.xml and ClearParserTokenizedInfPosAggregate.xml

Aggregate engines appropriate for use in CPEs. The first of these assumes that POS tags have been given in an upstream component or directly from data. The second infers POS tags using the cTAKES POS tagger.

analysis_engine/ClearTrainerAE.xml and ClearTrainerAggregate.xml

These analysis engines train models for use in ClearParserAE. See "Training a model Training data" Section below for further details.

analysis_engine/LemAssigner.xml, LvgBaseTokenAnnotator.xml, and PosAssigner.xml

These analysis engines are upstream components that complement ClearTrainerAE. See "Training a model Training data" Section below for further details.

collection_reader/DependencyFileCollectionReader.xml

Reads in a single file with dependency data in the formats described in "Data Format" section below. The file is treated as a single document with many sentences that are separated by blank lines.

cas_consumer/DependencyNodeWriter.xml

Writes ConllDependencyNode objects (the internal form used for dependency parses) to the .dep format. Refer to the "Data Format" section below

Resources and Models

clinques.mod

The main ClearParser model packaged with cTAKES v1.1. This is trained on a corpus of 1600 clinical questions.

lexicon*/

A directory of additional files for a ClearParser model. For example, "deprel.txt" contains the set of dependency labels; "pos.txt" contains the set of POS tags.

When doing training within this project, the specified lexicon directory must first be created separately.

en_dict/

A directory used by the Clear Morphological Analyzer to create lemmas. This analyzer is an alternative to using LVG output from cTAKES. Descriptor files in the project are set up to use the Clear Morphological Analyzer if this valid location is passed as the "Morph Dictionary Directory" parameter to Analysis Engines.

feature.xml

Tells the dependency parser what features to base its dependency decisions on.

config_en.xml

This file is not used when cTAKES is running ClearParser. You can follow the manual on the ClearParser website to run tests or training from the command line, which would make use of this file.

en_clinques.headrules

In order to convert from standard phrase structure trees, head rules tell you which child in a tree is the head (loosely, the most important). These are used by clear.engine.PhraseToDep.

Data Format

The format of training data into DependencyFileCollectionReader and of output data from DependencyNodeWriter is the same. Files should have one word per line alongside several other tab-delimited attributes. Sentences are separated by a blank line.

An example snippet from data/sample.dep of dependency data is shown in the "Data Format" section repeated here.

Example: Dependency parser data: .dep format (the first line is for reference).

ID FORM LEMMA POS HEAD DEPREL 1 The the DT 3 NMOD 2 study study NN 3 NMOD 3 physician physician NN 4 SBJ 4 called call VBD 0 ROOT 5 a a DT 6 NMOD 6 hematologist hematologist NN 4 OBJ

Description of .dep format fields:
ID
The word number within a sentence. There is an implied ROOT word that has and ID of 0.

FORM
The word itself.

LEMMA
A normalized form of the word, stripping suffixes, etc.

POS
The part-of-speech associated with the word.

HEAD
The ID of the word that the word on the current line is dependent on.

DEPREL
The relationship between the current word and its head. This may be syntactic or semantic.

The popular CONLL format is also supported for input into cTAKES; this requires several extra columns. However, not all of those columns will be used for ClearParser parsing.

Derivative Formats

The parser will use formats derived from this .dep format, as well.

Alternative formats

.mlem ID, FORM, LEMMA, HEAD, DEPREL .mpos ID, FORM, POS, HEAD, DEPREL .min ID, FORM, HEAD, DEPREL .tok ID, FORM

Of these, .tok is typically used for testing the actual parses, and the rest are used for training new models.

Conversion between formats

Data from resources such as WSJ or Genia typically come as trees that were originally used for deep parsers (as in the example above). The dependency parser project comes with a tool to convert between those trees and the .dep format.

java -cp lib/args4j-2.0.12.jar:bin clear.engine.PhraseToDep *-h resources/en_clinques.headrules* -i *<input-file>* -m resources/en_dict *<output-file>*
Where
<input-file>- A file in bracketed tree format, e.g, data/sample.tree
<output-file>- A file to be written out in .dep format

To down-convert between .dep and other formats, use cut in **Linux**:

cut -f 1,2,3,5,6 data/sample.dep > data/sample.mlem cut -f 1,2,4,5,6 data/sample.dep > data/sample.mpos cut -f 1,2,5,6 data/sample.dep > data/sample.min cut -f 1,2 data/sample.dep > data/sample.tok In **Windows**, this operation can be done in spreadsheet applications like Excel by:

- using Data > Import > From Text File,
- selecting the .dep file,
- identifying tabs as the delimiter,
- deleting unwanted columns
- File > Export in a tab-delimited format.

Training a model Training data

The packaged model is one trained on Clinical Questions, in part because it is small enough to package with cTAKES. If this is not your domain, you may want to train other models. At the time of cTAKES release 1.1, however, there are no clinical document Treebanks to train on.

The other options are:

- **GENIA**

GENIA is a literature mining project in molecular biology from the University of Tokyo. Its corpus, a collection of biomedical literature, has been annotated with phrase-structure trees. You can download a copy atGENIA from the University of Tokyo

- **Penn Treebank**

The Penn Treebank project annotates naturally-occurring text for linguistic structure. To obtain a copy of Release 2 is non-trivial, Please read Penn Treebank on the University of Pennsylvania website.

Training a model in Eclipse

There are many types of models that can be trained, based on how much you want to rely on cTAKES components and how much you want to rely on other components.

1. Download and install the C++ version of liblinear from National Taiwan University; this requires much less memory than the default Java version.
2.Train a model

- To create a model using cTAKES POS tags and lemmas with Eclipse:

1. Create a <your-data>.min file from <your-data>.dep (see the section called "Conversion between formats")
2. Use the UIMA_CPE_GUI---dependency parser launch.
3. Load desc/collection_processing_engine/ClearTrainerPosLemTestCPE.xml
4. Put your filename under "Dependency File"
5. Make sure "Training Mode" is checked
6. Rename the "Dependency Model File" and "Lexicon Directory" according to what you want.
7. Make sure "Trainer Path" is a valid relative path from >cTAKES_HOME>/dependency parser to a vaid liblinear binary train file.

- To create a model using gold standard POS tags and Clear Morphological Analyzer lemmas:

java -cp lib/hppc-0.3.1.jar:bin:resources clear.engine.DepTrain "'*-t<training_data> -c <configuration_file>*
Where
**-t<training_data>*an OpenNLP training data file.
**-c <configuration_file>*the file name of the resulting model. The name should end with either
.txt (for a plain text model) or .bin.gz (for a compressed binary model)