HeidelTime GATE Wrapper
First Description

Jannik Strötgen

March 21, 2016

Abstract

As HeidelTime can be used for several different languages and domains, and as the HeidelTime GATE wrapper can be used with different settings additionally, there are some things that should be considered and that have to be set up when using HeidelTime within GATE.

1 Possible Usages

The HeidelTime GATE wrapper can be used in such a way that other GATE components (e.g., ANNIE) perform linguistic preprocessing. On the other hand, the wrapper can be used in such a way, that preprocessing is performed by the wrapper. Then, however, other tools (depending on the languages that are to be processed) have to be installed and configured.

A further important thing to know is that depending on the document type (domain of the documents), HeidelTime performs different strategies. In particular, it is important to know the difference between news-style and narrative-style documents. In news-style documents, the document creation time plays an important role, e.g., for expressions such as "On Monday" or "In March", the document creation time is typically the reference time. In contrast, in narrative-style documents, the reference time of such expressions has to be determined in the text itself and the document creation time is not important at all (e.g., in Wikipedia articles).

2 The HeidelTime GATE Wrapper: Files and Processing Resources

Currently, the HeidelTime GATE wrapper contains the following files. We'll probably have to think about if we want to have one "GATE-Time" plugin instead of single ones for HeidelTime and one for event detection.

(a) config.props
(b) creole.xml
(c) HeidelTime-Standalone-Manual.pdf
(d) Tagger_Heideltime.jar
(e) lib/de.unihd.dbs.heideltime.standalone.jar
(f) lib/uima-core.jar

2.1 Details about the Files

- **config.props** and **HeidelTime-Standalone-Manual.pdf**

For the HeidelTime GATE wrapper, the HeidelTime standalone version is required. If one wants that the wrapper performs linguistic preprocessing (sentence splitting, tokenization, part-of-speech tagging), then other tools have to be installed, e.g., the TreeTagger for several languages, Stanford POS Tagger for Arabic, and other tools for Vietnamese and Croatian. Details how to install all required components are contained in **HeidelTime-Standalone-Manual.pdf**. The paths to the tools have to be configured in **config.props**.

If the HeidelTime GATE wrapper shall not perform linguistic preprocessing, then these tools do not have to be installed and not configured in **config.props**

- **lib/de.unihd.dbs.heideltime.standalone.jar**

This file is the jar file of the HeidelTime standalone version. We have to think about whether it makes sense to include it directly in the GATE plugin that we distribute or if we want that users have to download it separately. I think it makes sense to include it (it is GPL licensed), however, it should be replaced once there is a new HeidelTime version available. In general the HeidelTime GATE wrapper is (should) be implemented in such a way that **lib/de.unihd.dbs.heideltime.standalone.jar** can just be replaced once a new HeidelTime version is available.

- **lib/uima-core.jar**

This is a UIMA jar file which is also required. Although it could be replaced once there is a new (UIMA) version available, this is less crucial than keeping the HeidelTime jar file up-to-date.

- **creole.xml**

This is the standard GATE creole file.

- **Tagger_Heideltime.jar**

The jar file required for processing in GATE.

2.2 Processing Resources

Tagger_HeidelTime contains two processing units:

- DCTParser
- HeidelTime

In the following, both will be described in detail.

3 DCTParser – providing DCT information

The DCTParser contains the following parameters (all runtime paramters):

- dctParsingFormat: [timeml, manualdate]
- inputASName: name of annotation set, where DCT is stored, e.g., "Original markups" for the TempEval-3 corpora. (cf. very last section)
- manuallySetDct: if format is set to "manualdate", the user can set a date manually and this date is stored as DCT by DCTParser
- outputASName: name of annotation set for output

I tried to use inputASName, outputASName in the same way as it is done in other GATE components.

If processing news (news-style and also colloquial) documents, it is important that HeidelTime knows the document creation time (DCT) of the documents. Note that it is not the time when the documents have been loaded into GATE. Instead, it is the time when the document was written, e.g., when a news document was published. To provide the DCT of a document / all documents in the corpus, the DCTParser can be used. It can be used in two ways:

- to parse the DCT out of TimeML-style xml documents, e.g., the corpora TempEval-3 TimeBank, TempEval-3 Aquaint, and TempEval-3 platinum contain DCT information in this format. (cf. very last section)
- to manually set the DCT for a document or a full corpus.

It might make sense to add further parsing formats to DCTParser, e.g., that the dct can be parsed out of a document's name (e.g., if the documents in a corpus are named like "NYT-20100910-article1.txt" and "NYT-20100911-article1.txt"). It is crucial to know that if a corpus contains many documents, then, the documents typically have differing DCTs. Currently, the DCT can only be parsed if it is available in TimeML-style format, or it can be manually provided for the document or the full corpus. If HeidelTime processes news documents with wrong DCT information, relative and underspecified expressions will, of course, be normalized incorrectly.

If the documents that are to be processed are narrative documents (e.g., Wikipedia documents), no document creation time is required. The HeidelTime GATE wrapper can handle this automatically if the domain of the HeidelTime component is set to "narratives" (see next section).

Note that HeidelTime (described next) requires that the DCTParser output and that the linguistic processing annotations (e.g., sentence, token...) are stored in the same annotation set if existing annotations shall be used (e.g., those of ANNIE).

4 HeidelTime

HeidelTime's parameters are: (the first three are not runtime parameters, as different models have to be loaded depending on language and domain).

- config.props
- documentType [narratives, news, colloquial, scientific]
- language [english, german, dutch, englishcoll, englishsci, *and many more*]

Runtime parameters are:

- creationDateAnnotationType: if DCTParser is used to set the DCT, then the value is "DCT"
- doPreprocessing: [true, false]
- inputASName: name of annotation set, where token, sentence, pos information are stored (if any)
- outputASName: name of annotation set for output
- posAnnotationNameAsTokenAttribute: name of the 'part-of-speech' attribute the 'Token' annotations (if using ANNIE, this is 'category')
- sentenceAnnotationType: name of the 'Sentence' annotation (if using ANNIE, this is 'Sentence')
- tokenAnnotationType: name of the 'Token' annotation (if using ANNIE, this is 'Token')

HeidelTime can be used for many languages and four domains (in particular news and narrative, but also colloquial and autonomic for English – see Heideltime standalone Manual). Note that HeidelTime can perform linguistic preprocessing for all the languages if respective tools are installed correctly and configured correctly in the **config.props** file. I don't know what languages can be processed by GATE...

If processing HeidelTime narrative-style documents, it is not important that DCT information is available for the documents. If news-style (and colloquial) documents are processed, then DCT information is crucial and processing fails, if no DCT information is available. For this, creationDateAnnotationType has to contain information about the DCT annotation (see above).

HeidelTime can be used in such a way that the linguistic preprocessing is performed internally. For this further tools have to be set-up (see above) and the parameter doPreprocessing has to be set to 'true'. In this case, some other parameters are ignored (about Sentence, Token, POS).

If other preprocessing annotations shall be used (e.g., those of ANNIE) then doPreprocessing has to be set to 'false' and the other parameters (about Sentence, Token, POS) have to be provided correctly.

5 Testing HeidelTime GATE Wrapper

In addition to your standard test, you can test HeidelTime with the corpora, I attached to the email.

For this, load the Tagger_HeidelTime plugin, then, add the processing resources:

- DCTParser
- HeidelTime (english, news)
- HeidelTime (english, narratives)
- HeidelTime (german, narratives)

Also set up the TreeTagger as described in the Standalone Manual and add the path to the TreeTagger in config.props.

5.1 Processing TempEval-3 platinum data

First, use TempEval-3 platinum and process it with the following two 'workflows', and use HeidelTime-english-news:

1st workflow:

1. DCTParser (dctParsingFormat 'timeml') 2. HeidelTime (doPreprocessing 'true', creationDateAnnotationType 'DCT' – the other paramters don't matter.

2nd workflow: 1. ANNIE English Tokenizer (with default settings) 2. ANNIE Sentence Splitter (with default settings) 3. ANNIE POS Tagger (with default settings) 4. DCTParser (dctParsingFormat 'timeml', inputASName 'Original markups' - if no other markups specified) 5. HeidelTime (doPreprocessing 'false', creationDateAnnotationType 'DCT', posAnnotationNameAsTokenAttribute 'category', sentenceAnnotationType 'Sentence', tokenAnnotationType 'Token')

5.2 Processing WikiWars data

Second, use the WikiWars documents and process it with the following two 'workflows', and use HeidelTime-english-narratives.

1st workflow:

1. HeidelTime (doPreprocessing 'true', the other paramters don't matter).

2nd workflow: 1. ANNIE English Tokenizer (with default settings) 2. ANNIE Sentence Splitter (with default settings) 3. ANNIE POS Tagger (with default settings) 4. HeidelTime (doPreprocessing 'false', posAnnotationNameAsTokenAttribute 'category', sentenceAnnotationType 'Sentence', tokenAnnotationType 'Token')

5.3 Processing WikiWarsDE data

Third, use the WikiWarsDE data and process it with the following workflow and use HeidelTime-german-narratives

1st workflow: 1. HeidelTime (doPreprocessing 'true', the other paramters don't matter).

If there are existing GATE components for preprocessing German, you can try to use them accordingly.