HeidelTime Standalone Manual – Version 2.2

Jannik Strötgen (Max Planck Institute for Informatics)
Julian Zell (Heidelberg University)
jannik.stroetgen@mpi-inf.mpg.de

October 2016

Abstract

This manual contains information on how to install and use the standalone version of the multilingual, cross-domain temporal tagger HeidelTime.

Contents

1 Preface

HeidelTime is a multilingual, cross-domain temporal tagger for the extraction and normalization of temporal expressions from documents. Its development started at Heidelberg University [9, 10, 12, 13], and the original version of HeidelTime is designed to run within a UIMA pipeline [1]. With the standalone version, HeidelTime is wrapped in such a way that it can be run with fewer prerequisites and, in particular, outside of UIMA. This manual contains information on how to install and use the standalone version of HeidelTime, and how to call HeidelTime from within a Java program.

HeidelTime comes with **manually developed resources for 13 languages**: English, German, French, Spanish, Italian, Vietnamese, Arabic, Dutch, Chinese, Russian, Croatian, Portuguese and Estonian. Resources called englishcoll and englishsci can be used to process colloquial English text (e.g., SMS and tweets) and scientific literature (e.g., clinical trials) [11]. In addition to the manually developed language resources, HeidelTime contains **automatically created resources for more than 200 languages** and can be used as a baseline temporal tagger for basically every language in the world [13].

HeidelTime can process documents of different domains. In all languages, the news and the narratives domains are supported. For news documents, the document creation time is crucial, for narratives (e.g., Wikipedia articles) it is not. For English, colloquial and scientific are supported with specific language resources (englishcoll, englishsci).

Resources for **several languages were developed at other institutes**:

- Dutch (Matje van de Camp, Tilburg University, [16])
- French (Véronique Moriceau, LIMSI-CNRS, [6])
- Croatian (Luka Skukan, University of Zagreb, [8])
- Russian (preliminary resources, Elena Klyachko, [3])
- Portuguese (Zunsik Lim, [17])

Information about using HeidelTime with its automatically created resources for any of the 200+ supported languages can be found in Section 5.

2 Quick Start

This section briefly outlines what is necessary to setup and run HeidelTime Standalone. See Section 3 for a more detailed description, in particular if you experience any problems.

1. Install Java Runtime Environment [14] in order to execute Java programs.
2. Install TreeTagger [7] with the parameter files for English, German, Dutch, Spanish, Italian, French, Chinese, Russian, Portuguese and Estonian (or only those of the languages you want to process).
3. Ensure the path to your local TreeTagger installation is set correctly. Therefore, check the variable *treeTaggerHome* in `config.props`. It has to point to the root directory of your TreeTagger installation.
4. Change to the directory containing HeidelTime's executable jar file: `de.unihd.dbs.heideltime.standalone.jar`
5. Run HeidelTime Standalone using
   "java -jar de.unihd.dbs.heideltime.standalone.jar <file>"
   where *<file>* is the path to a text document.

To find out how to set additional parameters, e.g., how to specify the language, domain, and document creation time, see Section 4.1.

3 Installation

This section explains in more details how to setup HeidelTime Standalone.

3.1 Files

HeidelTime Standalone comes with three files and two folders:

- `de.unihd.dbs.heideltime.standalone.jar` Executable java file; see Section 4 for information about command line arguments.
- `config.props` Configuration file; it has to be located in the same directory as the executable. See Section 3.3 for configuration options.
- **src**/ Folder containing the source files that were used to generate the executable jar file `de.unihd.dbs.heideltime.standalone.jar`.
- **doc**/ Folder containing the Javadoc files.
- `Manual.pdf` This file.

3.2 Prerequisites

HeidelTime Standalone requires the following components to be installed. Note: If you use HeidelTime Standalone on Windows, see Appendix B.

1. Java Runtime Environment [14]

2. TreeTagger and its parameter files. For preprocessing (tokenization, sentence splitting, part-of-speech tagging), we use the TreeTagger [7] as default tool for several languages (English, German, Dutch, Spanish, Italian, French, Chinese, Russian, Portuguese and Estonian).

   Download from the TreeTagger website[1] into one directory the TreeTagger, its tagging scripts, installation scripts, and all parameter files (all available files for all languages mentioned above or the ones you want to process, e.g., for German, the Latin1 and the UTF-8 files). See the TreeTagger website for details how to install the TreeTagger. The HeidelTime UIMA kit readme (Section 3.3) file[2] contains wget-commands to conveniently download all required files. Please check the TreeTagger license though.

3. To process Chinese, download the Chinese TreeTagger parameter file from Serge Sharoff's page[3] as well as a copy of the Chinese Tokenizer[4]. Extract the parameter files into the TreeTagger home directory so the files from the `lib` and `cmd` folders land in the respective TreeTager folders. Extract the tokenizer into its own directory and remember the path for the correct configuration (Section 3.3).

4. To process Russian, download the Russian parameter file by Serge Sharoff[5] and extract it into the TreeTagger's `lib` folder.

5. To process Vietnamese, download JVnTextPro [2].[6]

6. To process Arabic, download the full package of the Stanford POS Tagger [15].[7]

7. To process Croatian, download the hunpos tagger [4][8] and the Croatian model file[9].

[1] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`
[2] `https://github.com/HeidelTime/heideltime/blob/master/doc/readme.txt`
[3] `http://corpus.leeds.ac.uk/tools/zh/`
[4] `https://drive.google.com/uc?id=0B1ZoOwaeRsbva2F3NThLd3ptRWM`
[5] `http://corpus.leeds.ac.uk/mocky/`
[6] `http://sourceforge.net/projects/jvntextpro/`
[7] `http://nlp.stanford.edu/software/tagger.shtml`
[8] `https://code.google.com/p/hunpos/`
[9] `http://nlp.ffzg.hr/resources/models/tagging/`

3.3 Configuration

After the installation of the prerequisites mentioned in Section 3.2, there are a few parameters to set up in the configuration file `config.props`:

For most languages

- *treeTaggerHome*: TreeTagger root directory.
  Example: `/opt/treetagger/`

For Chinese

- *chineseTokenizerPath*: folder of Chinese tokenizer script and files.
  Example: `/opt/treetagger/chinese-tokenizer/`

For use with Vietnamese

- *sent_model_path*: folder of JVnTextPro's sentence segmentation model.
  Example: `/opt/jvntextpro/models/jvnsensegmenter`
- *word_model_path*: folder of JVnTextPro's segmentation model.
  Example: `/opt/jvntextpro/models/jvnsegmenter`
- *pos_model_path*: folder of JVnTextPro's part of speech model.
  Example: `/opt/jvntextpro/models/jvnpostag/maxent`

For use with Arabic

- *model_path* path to StanfordPOSTagger's tagger model.
  Example: `/opt/stanfordpostagger/models/arabic.tagger`
- *config_path* path to StanfordPOSTagger's config model. This setting is optional and can be left empty.
  Example: `/opt/stanfordpostagger/tagger.config`

For use with Croatian

- *hunpos_path*: folder of the hunpos executable.
  Example: `/opt/hunpos/`
- *hunpos_model_name*: **name** of the hunpos model, located in `hunpos_path`.
  Example: `model.hunpos.mte5.defnpout`

General options

- *considerDate*: shall Timex3 expressions of type DATE be considered?
- *considerDuration*: shall Timex3 expressions of type DURATION be considered?
- *considerSet*: shall Timex3 expressions of type SET be considered?
- *considerTime*: shall Timex3 expressions of type TIME be considered?

Note that HeidelTime 2.2 supports temponym tagging. To perform temponym tagging, set *considerTemponym* to "true". Temponym tagging resources are only available for English yet. Attention: initialization will take some time, i.e., set this variable to "false" if you don't need temponyms.

All other options are not meant to be changed and therefore skipped in this section.

4 Usage

This section explains how to use HeidelTime Standalone both as a command line tool and as a component in other Java projects.

4.1 Command Line Usage

To use HeidelTime Standalone, open a command line terminal and switch to the directory containing `de.unihd.dbs.heideltime.standalone.jar`. You then are able to run it using the following command:

"java -jar de.unihd.dbs.heideltime.standalone.jar <file> [options]"

where *<file>* is the path to a text document on your hard disk and *[options]* are possible options explained in Table 1.

Note: some extra steps are required to process Arabic and Vietnamese as described below.

Table 1: Command line arguments of HeidelTime Standalone.

Option	Name	Description
-dct	Document Creation Time	Date of the format YYYY-MM-DD when the document specified by *<file>* was created. This information is crucial and used if "-t" is set to NEWS or COLLOQUIAL. It is used to resolve relative temporal expressions such as "today". Default: current date on local machine.

Option	Name	Description
-l	Language	Language of the document. Possible values: ENGLISH, GERMAN, DUTCH, ENGLISHCOLL (for -t COLLOQUIAL), ENGLISHSCI (for -t SCIENTIFIC), SPANISH, ITALIAN, ARABIC, VIETNAMESE, FRENCH, CHINESE, RUSSIAN, CROATIAN, PORTUGUESE, ESTONIAN or any AUTO-language contained in HeidelTime's resource folder. Default: ENGLISH.
-t	Type	Type of the document specified by *<file>*. Possible values: NARRATIVES, NEWS, COLLOQUIAL and SCIENTIFIC. Default: NARRATIVES. **This parameter is important!** Set "NEWS" if creation time of the document is crucial and NARRATIVES otherwise, e.g., when processing Wikipedia.
-pos	POS Tagger	Lets you choose a specific part of speech tagger; either STANFORDPOSTAGGER or TREETAGGER. Note that for Arabic or Vietnamese documents, we allow only StanfordPOSTagger and JVnTextPro respectively. Please take note of the prerequisites in Section 4.1. To process languages for which no pos tagger is available (i.e., most AUTO-languages), set NO. (For details, see Section 5).
-o	Output Type	Type of the result. Possible values: XMI and TIMEML. Default: TIMEML.
-e	Encoding	Encoding of the document that is to be processed, e.g., UTF-8, ISO-8859-1, ... Default: UTF-8.
-c	Configuration file	Relative/absolute path to configuration file. Default: config.props
-v/-vv	Verbosity	Turns on verbose or very verbose logging.
-it	IntervalTagger	Enables the IntervalTagger and outputs recognized intervals.
-locale	Locale	Lets you set a custom locale to run HeidelTime under. Format is: X_Y, where X is from ISO 639 and Y is from ISO 3166, e.g.: "en_GB"
-h	Help	Shows you a list of commands and usage information

You may omit any of the options since they are optional. HeidelTime Standalone will however force you to enter a valid document path. It will output an XMI- or TimeML-document to the standard output stream containing all annotations made by HeidelTime. To save the output use:

```
"java -jar de.unihd.dbs.heideltime.standalone.jar <file> [options]
> <outputfile>"
```

where <*outputfile*> is the path to the file the output will be saved to.

Encoding settings: HeidelTime Standalone can process files of different encodings. However, independent of the input encoding, the output is always encoded as UTF-8. If the default encoding of your Java Virtual Machine is not UTF-8, **you have to set the encoding to UTF-8** using the -Dfile.encoding option:

```
"java -Dfile.encoding=UTF-8 -jar de.unihd.dbs.heideltime.standalone.jar
<file> [options]"
```

If the encoding of the document that is to be processed is not UTF-8, you can specify the encoding with parameter "-e" as described in Table 1.

Extra steps for Arabic and Vietnamese tagging

To tag Arabic and Vietnamese documents, you need to utilize a different command line scheme. First, you have to set the `HT_CP` variable to include HeidelTime Standalone's class files as well as those of the languages' respective taggers:

Under Unix/Linux/Mac OS X:
"export HT_CP="<$1>:<$2>:<$3>:$CLASSPATH""
or under Windows:
"set HT_CP=<$1>;<$2>;<$3>;%CLASSPATH%"
where
<$1> is the path to JVnTextPro's `bin` folder, e.g., `/opt/jvntextpro/bin/`,
<$2> is the path to StanfordPOSTagger's `.jar` file, e.g.,
`/opt/stanfordpostagger/stanford-postagger.jar` and
<$3> is `de.unihd.dbs.heideltime.standalone.jar`

Once you have this variable set, you can use the following command line:

```
"java -cp $HT_CP de.unihd.dbs.heideltime.standalone.HeidelTimeStandalone
<file> [options]"
```

where <*file*> is the path to a text document on your hard disk and *[options]* are possible options explained in Table 1.

4.2 Component in other Projects

To use HeidelTime Standalone as a component in other projects, you have to prepare the executable jar file `de.unihd.dbs.heideltime.standalone.jar`: Add the configuration file `config.props` to the main directory of the executable using a proper archive tool. Once this is done you can copy the executable wherever you want and use it like a library. Alternatively, the location of the `config.props` can be specified programmatically as shown below.
To run HeidelTime Standalone, instantiate an object of *HeidelTimeStandalone*. To do so, you simply have to provide the desired language and type that is to be processed (see Table 1 for further information). To actually run HeidelTime, you have to call *process* on the recently instantiated object of type *HeidelTimeStandalone* with the text to be processed. If this text is of type NEWS (remember your decision when instantiating a *HeidelTimeStandalone* object), you have to provide the document creation time as well. As a result you will get a string containing the TimeML document with all annotations made by HeidelTime for further treatment.

In Appendix A, we provide example code for processing English news and English narratives (assuming that the `config.props` is located in the lib folder of the java project).

4.3 Temponym Tagging

HeidelTime Standalone can be used to perform temponym tagging in addition to standard temporal tagging. Temponyms are non-standard temporal expressions (i.e., regular text phrases) which carry time information, e.g., "John F. Kennedy's death" will be normalized to "1963-11-22". Note that we use TIMEX3 tags for temponyms, although they are not defined as TIMEX3 temporal expressions in TimeML. If a temponym lasts longer than a day, we assign interval information to it, e.g., the Costa Rican Civil War lasted from 1948-03-12 to 1948-04-24. In such cases, we add TIMEX3INTERVAL tags and tag the phrase as:

<TIMEX3INTERVAL earliestBegin="1948-03-12" latestBegin="1948-03-12" earliestEnd="1948-04-24" latestEnd="1948-04-24"><TIMEX3 tid="t100015" type="TEMPONYM" value="1948-04-24">Costa Rican Civil War</TIMEX3></TIMEX3INTERVAL>

To perform temponym tagging, set "considerTemponym" to "true" in the config.props file.

Thus, if you want proper TimeML output, temponym tagging should not be done. For details on temponym tagging see: [5].

5 HeidelTime for 200+ Languages

Starting with version 2.0, HeidelTime contains automatically created resources for more than 200 languages. Since temporal tagging of most of these languages has never been addressed before, HeidelTime is the first temporal tagger for those languages. Thus, although the quality between the languages differs significantly, HeidelTime can be used as a baseline for temporal tagging all these languages.
While part-of-speech constraints in HeidelTime's rules are useful and typically result in better temporal tagging quality, not for every language, a part-of-speech tagger exists. Thus all language resources that are automatically created, i.e., called "auto-x" in the resource folder do not require any part-of-speech information. To be able to run these languages with HeidelTime's standalone version, set the "-pos" parameter to "NO".

Finally, note that the automatically created resources can be used as starting point for further developments.

For details, please see: [13]

6 Creating or improving resources

A brief step-by-step introduction into developing resources for HeidelTime can be found on our project's wiki.[10]

It is noteworthy that as of HeidelTime 1.9, it is also possible to create resources using only the Standalone edition as opposed to the full UIMA kit.

To create resources, you should copy one of the existing resource folders (e.g., `auto-polish`) from inside the program's archive into the folder where this file resides. To obtain this folder, you can either copy it out of the `resources/` folder of the HeidelTime UIMA kit, or extract it from the HeidelTime Standalone `.jar` file using a program like WinZip or 7zip.

Once you have copied out the folder, you should rename it to something that is unique, e.g., `polish-improved`. By default, HeidelTime Standalone will prefer external versions of the resources over the ones built into the `.jar` file, and to evade possibly confusing situations, unique names should be used.

To check whether the folder you have changed is recognized as a resource folder, you should run HeidelTime Standalone with the `-vv` switch; this will list all recognized language resource folders, including your new one.

In order to run HeidelTime Standalone with the new language resources you have created, you will need to supply the language's name like so:

`java -jar de....jar -l polish-improved file.txt`

[10] `https://github.com/HeidelTime/heideltime/wiki/Developing-Resources`

Please note that unless a specific preprocessing engine is specified, this name is passed on to the TreeTaggerWrapper which then expects a parameter file under that name to exist. If you extend an existing language, you can just copy the existing parameter files in the TreeTagger's `lib/` folder, e.g., for German:

- `german-utf8.par` ⇒ `german-improved.par`
- `german-abbreviations-utf8` ⇒ `german-improved-abbreviations`

For the auto-language, no pos tagger is used so this step is not necessary.

7 Questions and Issues

If you run into any trouble, if you have any questions, or if you have any suggestions for improving HeidelTime, please check the FAQ wiki site on `https://github.com/HeidelTime/heideltime`, contact us either via email, or open an issue on HeidelTime's github page.

8 License

 If you use HeidelTime, please cite one of the respective papers describing HeidelTime, e.g., [9, 12]. Thank you.

The initial HeidelTime Standalone version was created by Andreas Fay. For further details, see HeidelTime's github page.[11]

[11] `https://github.com/HeidelTime/heideltime`

References

[1] Apache Software Foundation. Apache UIMA, June 2011. URL `http://uima.apache.org/`.

[2] Thu-Trang Nguyen Cam-Tu Nguyen, Xuan-Hieu Phan. JVnTextPro, April 2013. URL `http://sourceforge.net/projects/jvntextpro/`.

[3] Elena Klyachko. Russian resources, 2014.

[4] Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos: An Open Source Trigram Tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 209–212. ACL, 2007.

[5] Erdal Kuzey, Jannik Strötgen, Vinay Setty, and Gerhard Weikum. Temponym Tagging: Temporal Scopes for Textual Phrases. In *Proceedings of the 6th Temporal Web Analytics Workshop (TempWeb'16)*, pages 841–842, 2016.

[6] Véronique Moriceau and Xavier Tannier. French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC '14)*, pages 3239–3243. ELRA, 2014.

[7] Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, 1994.

[8] Luka Skuka, Goran Glavaš, and Jan Šnajder. HeidelTime.HR: Extracting and Normalizing Temporal Expressions in Croatian. In *Proceedings of the 9th Language Technologies Conference*, pages 99–103, 2014.

[9] Jannik Strötgen and Michael Gertz. HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 321–324. ACL, 2010.

[10] Jannik Strötgen and Michael Gertz. HeidelTime, May 2012. URL `http://dbs.ifi.uni-heidelberg.de/heideltime/`.

[11] Jannik Strötgen and Michael Gertz. Temporal Tagging on Different Domains: Challenges, Strategies, and Gold Standards. In *Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753. ELRA, 2012.

[12] Jannik Strötgen and Michael Gertz. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[13] Jannik Strötgen and Michael Gertz. A Baseline Temporal Tagger for all Languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547. ACL, 2015.

[14] Sun Microsystems. Java, March 2011. URL `http://www.java.com`.

[15] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '03)*, pages 173–180. ACL, 2003.

[16] Matje van de Camp and Henning Christiansen. Resolving Relative Time Expressions in Dutch Text with Constraint Handling Rules. In *Proceedings of the 7th International Workshop on Constraint Solving and Language Processing (CSLP '12)*, pages 74–85. Springer, 2012.

[17] Zunsik Lim. Portuguese resources, 2015.

A Example Java Code

```
package de.mpii.heideltime;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import de.unihd.dbs.heideltime.standalone.*;
import de.unihd.dbs.heideltime.standalone.exceptions.*;
import de.unihd.dbs.uima.annotator.heideltime.resources.Language;

public class RunHeideltimeInJava {

  public static void main(String[] args) throws
          DocumentCreationTimeMissingException, ParseException {
    // some parameters
    OutputType outtype = OutputType.XMI;
    POSTagger postagger = POSTagger.TREETAGGER;
    // or: faster, but worse results; no TreeTagger required
    // POSTagger postagger = POSTagger.NO;
    String conffile = "lib/config.props";

    // initialize HeidelTime for English news
    HeidelTimeStandalone hsNews = new HeidelTimeStandalone(Language.ENGLISH,
        DocumentType.NEWS, outtype, conffile, postagger);

    // initialize HeidelTime for English narrative
    HeidelTimeStandalone hsNarratives = new HeidelTimeStandalone(Language.ENGLISH,
        DocumentType.NARRATIVES, outtype, conffile, postagger);

    // process English narratives
    String narrativeText = "This is a text with a date in English: "
        + "January 24, 2009 and also two weeks later.";
    String xmiNarrativeOutput = hsNarratives.process(narrativeText);
    System.err.println("NARRATIVE*****" + xmiNarrativeOutput);

    // process English news (after handling DCT)
    String dctString = "2016-04-29";
    String newsText = "Today, I write a text with a date in English: "
        + "January 24, 2009 and also two weeks later. But what was two weeks ago?";

    DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
    Date dct = df.parse(dctString);

    String xmiNewsOutput = hsNews.process(newsText, dct);
    System.err.println("NEWS*******" + xmiNewsOutput);
  }
}
```

B Information for Windows Users

If you are using HeidelTime standalone on Windows, you have to download and install a Perl interpreter, e.g., ActivePerl[12], as well as the Windows version of the TreeTagger [7], including parameter files for the languages you want to process. A set of initial files to download and extract to the same folder are the following (newer versions may be available):

- The Windows Version of the TreeTagger[13]
- The tagging scripts[14]

As for the parameter files for the respective languages, you will need to put any `.par` files in the `lib/` folder, any *`language`*`-abbreviations` in `lib/` and any `tree-tagger-`*`language`* script file in `cmd/`.

Once this is set up, you will need to specify the *treeTaggerHome*-Variable in `config.props` as described in Section 3.3. After that, you should be able to run HeidelTime Standalone as described in Section 4.1.

[12] `http://www.activestate.com/activeperl`

[13] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger-windows-3.2.zip`

[14] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tagger-scripts.tar.gz`