HeidelTime Standalone Manual
Version 2.1

Julian Zell, Jannik Strötgen (Heidelberg University)
zell@informatik.uni-heidelberg.de, stroetgen@uni-hd.de

December 2015

Abstract

This document contains information on how to install and use the standalone version of HeidelTime. HeidelTime is a multilingual, cross-domain temporal tagger for the extraction and normalization of temporal expressions from documents, developed at the Heidelberg University by Strötgen and Gertz [8, 9, 10].
The original version of HeidelTime is designed to run within a proper UIMA-Pipeline [1]. With this standalone version the original version is wrapped in such a way that it can be run with fewer prerequisites and, in particular, without UIMA.
HeidelTime Standalone comes with resources for 13 languages: English, German, French, Spanish, Italian, Vietnamese, Arabic, Dutch, Chinese, Russian, Croatian, Portuguese and Estonian. In addition, resources called English-colloquial and English-scientific can be used to process colloquial English text (e.g., SMS and tweets) and scientific literature (e.g., clinical trials).
Dutch resources were developed and kindly provided by Matje van de Camp (Tilburg University)[13]. French resources were provided by VÃľronique Moriceau (LIMSI-CNRS)[5]. A preliminary version of Russian resources was kindly provided by Elena Klyachko[3]. Luka Skukan[7] kindly contributed resources for Croatian. Zunsik Lim[14] has kindly contributed his resources for Portuguese.
In addition to manually crafted resources, as of version 2.0, we support over 200 automatically-generated languages.
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 additionally supported.

Contents

1 Preface

This document contains information about how to install and use the standalone version of HeidelTime. HeidelTime itself is a multilingual temporal tagger for the extraction and normalization of temporal expressions from documents, developed at the University of Heidelberg from Strötgen and Gertz [8, 9, 10]
The original version of HeidelTime is designed to run within a proper UIMA-Pipeline [1]. With this standalone version the original version is wrapped such that it can be run with fewer prerequisites and especially without UIMA.

2 Quick Start

This section will briefly outline what is necessary in order to get HeidelTime Standalone going. See Section 3 for a more detailed description.

1. Install Java Runtime Environment [11] in order to execute Java programs.
2. Install TreeTagger [6] with the parameter files for English, German, Dutch, Spanish, Italian, French, Chinese and Russian.
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 `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 and domain, see Section 4.1.

3 Installation

This section explains the steps necessary to use 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 more information about possible command line arguments.

- `config.props`
  Configuration file; it has to be located in the same directory as the executable. See Section 3.3 for more information about the configuration of HeidelTime Standalone.

- **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:

1. The Java Runtime Environment [11]

2. A compatible pre-processing tool that is capable of identifying language tokens, part of speech and sentence boundaries in all languages supported by HeidelTime. We decided to use TreeTagger [6] for English, German, Dutch, Spanish, Italian, Russian, French, Chinese, Portuguese and Estonian. You will need to download and install so called "parameter files" for those languages as well (all that are available, e.g., for German, download the Latin1 and the UTF-8 variants), to provide TreeTagger with the necessary functionality (see the TreeTagger website `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/` for more information or Section 3.3 of `https://github.com/HeidelTime/heideltime/blob/master/doc/readme.txt` for wget commands).

3. To process Chinese documents, please grab a copy of the Chinese TreeTagger parameter file from Serge Sharoff's page `http://corpus.leeds.ac.uk/tools/zh/` as well as a copy of the Chinese Tokenizer `https://drive.google.com/uc?id=0B1ZoOwaeRsbva2F3NThLd3ptRWM`. Extract the parameter files into the TreeTagger home directory so the files from the `lib` and `cmd` folders land in the TreeTager folders. Extract the tokenizer into its own directory and remember the path for the configuration later (Section 3.3).

4. To process Russian documents, please grab a copy of the Russian parameter file by Serge Sharoff from `http://corpus.leeds.ac.uk/mocky/` and extract it into the TreeTagger's `lib` folder.

5. For Arabic documents, you will need to download a full package of the Stanford POS Tagger [12] from `http://nlp.stanford.edu/software/tagger.shtml`

6. In order to process documents in Croatian, you will need to download a copy of hunpos [4] from `https://code.google.com/p/hunpos/` as well as the Croatian tagger model file for it from `http://nlp.ffzg.hr/resources/models/tagging/`.

   Note: If you use HeidelTime Standalone on Windows, please see Appendix A.

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*
  This variable has to point to the root directory of TreeTagger that you will need to use for most languages. Example: `/opt/treetagger/`

For Chinese

- *chineseTokenizerPath*
  This variable has to point to the directory where the Chinese Tokenizer Script and files are. Example: `/opt/treetagger/chinese-tokenizer/`

For use with Vietnamese

- *sent_model_path*
  This variable needs to point to the *folder* where JVnTextPro's sentence segmentation model is stored.
  Example: `/opt/jvntextpro/models/jvnsensegmenter`

- *word_model_path*
  This variable needs to point to the *folder* where JVnTextPro's segmentation model is stored.
  Example: `/opt/jvntextpro/models/jvnsegmenter`

- *pos_model_path*
  This variable needs to point to the *folder* where JVnTextPro's part of speech model is stored.
  Example: `/opt/jvntextpro/models/jvnpostag/maxent`

For use with Arabic

- *model_path*
  This variable needs to point to the path where StanfordPOSTagger's tagger model *file* is stored.
  Example: `/opt/stanfordpostagger/models/arabic.tagger`

- *config_path*
  This variable can be set to point to the path where StanfordPOSTagger's config model *file* is stored. This setting is optional and can be left empty.
  Example: `/opt/stanfordpostagger/tagger.config`

For use with Croatian

- *hunpos_path*
  This variable must point to the **folder** where the hunpos executable is located.
  Example: `/opt/hunpos/`

- *hunpos_model_name*
  This variable needs to represent the **name** of the hunpos model file used, residing in the `hunpos_path` set above.
  Example: `model.hunpos.mte5.defnpout`

General options

- *considerDate*
  Indicates whether HeidelTime should consider Timex3 expressions of type DATE.

- *considerDuration*
  Indicates whether HeidelTime should consider Timex3 expressions of type DURATION.

- *considerSet*
  Indicates whether HeidelTime should consider Timex3 expressions of type SET.

- *considerTime*
  Indicates whether HeidelTime should consider Timex3 expressions of type TIME.

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.

Extra steps for Arabic tagging

To tag Arabic documents, you will need to utilize a different command line scheme. First, you will 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>:<$2>:$CLASSPATH""

or under Windows:
"set HT_CP=<$1>;<$2>;%CLASSPATH%"

where
<$1> is the path to StanfordPOSTagger's `.jar` file, e.g. `/opt/stanfordpostagger/stanford-postagger.jar` and
<$2> 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.

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 used only if "-t" is set to NEWS or COLLOQUIAL. It is used to resolve relative temporal expression such as "today". The default value is the current date on the local machine.
-l	Language	Language of the document. Possible values are: ENGLISH, GERMAN, DUTCH, ENGLISHCOLL (for -t COLLOQUIAL), ENGLISHSCI (for -t SCIENTIFIC), SPANISH, ITALIAN, ARABIC, VIETNAMESE, FRENCH, CHINESE, RUSSIAN, CROATIAN, PORTUGUESE, ESTONIAN. The default is ENGLISH.
-t	Type	Type of the document specified by <*file*>. Possible values are: NARRATIVES, NEWS, COLLOQUIAL and SCIENTIFIC. The default value is NARRATIVES. The major difference between these types is the consideration of "-dct" if type is set to NEWS or COLLOQUIAL.
-o	Output Type	Type of the result. Possible values are: XMI and TIMEML. The default value is TIMEML.
-e	Encoding	Encoding of the document that is to be processed, e.g., UTF-8, ISO-8859-1, . . . Default value is UTF-8.
-c	Configuration file	Relative or absolute path to the configuration file. Default file is 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"

Option	Name	Description
-pos	POS Tagger	Lets you choose a specific part of speech tagger; either STANFORDPOSTAGGER, TREETAGGER or NO. Note that for Arabic or Vietnamese documents, we allow only StanfordPOSTagger and JVnTextPro respectively. Please take note of the prerequisites in Section 4.1. NO will not do any language-sensitive preprocessing and produce worse results.
-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. You may save the output to a file by using the following command:

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

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

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.

4.2 Creating new resources

A brief step-by-step introduction into developing resources for HeidelTime can be found on our project's wiki here:
`https://github.com/HeidelTime/heideltime/wiki/Developing-Resources`

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., `german`) 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. `german-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 german-improved file.txt
```

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:

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

in the TreeTagger's `lib/` folder.

4.3 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.
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.

4.4 Maven

To use HeidelTime Standalone in your project as a Maven dependency, most of the configuration that is described in this document is still necessary. In order to use HeidelTime Standalone, simply input these dependencies in your `pom.xml`:

```
<dependency>
        <groupId>org.apache.uima</groupId>
        <artifactId>uimaj-core</artifactId>
        <version>2.8.1</version>
</dependency>
<dependency>
        <groupId>com.github.heideltime</groupId>
        <artifactId>heideltime</artifactId>
        <version>2.1</version>
</dependency>
```

This will enable you to use HeidelTime Standalone's basic functionality. If you want to make use of the Stanford POS Tagger Wrapper included in HeidelTime Standalone, you will also need to add this dependency:

```
<dependency>
        <groupId>edu.stanford.nlp</groupId>
        <artifactId>stanford-corenlp</artifactId>
        <version>3.3.1</version>
</dependency>
```

To use the JVnTextPro preprocessor (e.g., for Vietnamese documents), you need to add these dependencies:

```
<dependency>
        <groupId>args4j</groupId>
        <artifactId>args4j</artifactId>
        <version>2.32</version>
</dependency>
<dependency>
        <groupId>com.dbtsai.lbfgs</groupId>
        <artifactId>lbfgs</artifactId>
        <version>0.1</version>
</dependency>
```

5 License

The initial version of HeidelTime Standalone was created by Andreas Fay.

If you use HeidelTime, please cite one of the papers describing HeidelTime: [8, 10]. Thank you.

For details, see `http://dbs.ifi.uni-heidelberg.de/heideltime/` or `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] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] 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.

[13] 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.

[14] Zunsik Lim. Portuguese resources, 2015.

A Information for Windows Users

If you are using HeidelTime standalone on Windows, you have to download and install a Perl interpreter, e.g. ActivePerl from `http://www.activestate.com/activeperl`, as well as the Windows version of the TreeTagger [6], 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:
  `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger-windows-3.2.zip`
- The tagging scripts:
  `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tagger-scripts.tar.gz`

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.