

Bilingual NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

30th March 2016
12:47 h.

Abstract

This is a description and documentation of the installation of an instrument to annotate Dutch or English documents with NLP tags.

Contents

1	Introduction	3
1.1	List of the modules to be installed	3
1.2	The things that are not open-source yet	3
1.3	Multi-linguality	5
1.4	File-structure of the pipeline	5
2	How to obtain modules and other material	7
2.1	Location-dependency	7
2.2	Reversible update	7
2.3	Installation from Github	7
2.4	Installation from the snapshot	8
3	Java and Python environment	9
3.1	Java	9
3.2	Maven	10
3.3	Java 1.6	11
3.4	Python	11
3.4.1	Virtual environment	12
3.4.2	Transplant the virtual environment	13
3.4.3	KafNafParserPy	14
3.4.4	Python packages	14
4	Installation of the modules	15
4.1	Conditional installation of the modules	15
4.2	The installation script	16
4.3	Check availability of resources	21
4.4	Parameters in module-scripts	21
4.5	Install utilities and resources	22
4.5.1	Process synchronisation	22
4.5.2	Prefix of scripts that run modules	23
4.5.3	Language detection	23
4.5.4	Alpino	24
4.5.5	Treetagger	25
4.5.6	Timbl and Ticcutils	27

4.5.7	The Boost library	28
4.5.8	Spotlight	28
4.5.9	VUA-pylib	34
4.5.10	SVMLight	34
4.5.11	CRFsuite	34
4.6	Install modules	35
4.6.1	Install tokenizer	35
4.6.2	Topic analyser	36
4.6.3	Morphosyntactic parser	36
4.6.4	Pos tagger	37
4.6.5	Constituent parser	37
4.6.6	NED-reranker	38
4.6.7	Wikify module	38
4.6.8	UKB	38
4.6.9	IMS-WSD	39
4.6.10	SRL server	40
4.6.11	SRL Dutch nominals	41
4.6.12	FBK-time module	42
4.6.13	FBK-temprel module	44
4.6.14	FBK-causalrel module	45
4.6.15	Factuality module	46
4.6.16	Nominal coreference-base	46
4.6.17	Named entity recognition (NERC)	47
4.6.18	Wordsense-disambiguation	48
4.6.19	Lexical-unit converter	50
4.6.20	NED	50
4.6.21	Ontotagger	51
4.6.22	Framenet SRL	52
4.6.23	Heideltime	52
4.6.24	Semantic Role labelling	54
4.6.25	SRL postprocessing	55
4.6.26	Event coreference	56
4.6.27	Dbpedia-ner	57
4.6.28	Nominal events	58
4.6.29	Opinion miner	58
5	Utilities	59
5.1	Test script	59
5.2	Logging	62
5.3	Misc	63
A	How to read and translate this document	63
A.1	Read this document	64
A.2	Process the document	64
A.3	The Makefile for this project.	65
A.4	Get Nuweb	66
A.5	Pre-processing	67
A.5.1	Process ‘dollar’ characters	67
A.5.2	Run the M4 pre-processor	67
A.6	Typeset this document	67
A.6.1	Figures	68
A.6.2	Bibliography	69
A.6.3	Create a printable/viewable document	69
A.6.4	Create HTML files	72

A.7 Create the program sources	75
A.8 Restore paths after transplantation	76
B References	77
B.1 Literature	77
C Indexes	77
C.1 Filenames	77
C.2 Macro's	78
C.3 Variables	80

1 Introduction

This document describes the current set-up of a pipeline that annotates texts in order to extract knowledge. The pipeline has been set up by the Computational Lexicology and Terminology Lab (CLTL¹) as part of the newsreader² project. It accepts and produces texts in the NAF (Newsreader Annotation Format) format.

Apart from describing the pipeline set-up, the document actually constructs the pipeline. The pipeline has been installed on a (Ubuntu) Linux computer.

The installation has been parameterised. The locations and names that you read (and that will be used to build the pipeline) have been read from variables in file `inst.m4` in the `nuweb` directory.

The pipeline is bi-lingual. It is capable to annotate Dutch and English texts. It recognizes the language from the “lang” attribute of the NAF element of the document.

The aim is, to install the pipeline from open-source modules that can e.g. be obtained from Github. However, that aim is only partially fulfilled. Some of the modules still contain elements that are not open-source or data that are not freely available. Because of lack of time, the current version of the installer installs the English pipeline from a frozen repository of the Newsreader Project.

1.1 List of the modules to be installed

Table 1 lists the modules that are installed. Some of the modules are used for both languages (Dutch and English), some for only one of them.

Table 2 lists the modules in the pipeline. The column *source* indicates the origin of the module. The modules are obtained in one of the following ways:

1. If possible, the module is directly obtained from an open-source repository like Github.
2. Some modules have not been officially published in a repository. These modules have been packed in a tar-ball that can be obtained by the author. In table 2 this has been indicated as SNAPSHOT.

The modules themselves use other utilities like dependency-taggers and POS taggers. These utilities are listed in table 3.

1.2 The things that are not open-source yet

The aim is, that the pipeline-system is completely open-sourced, so that anybody can install it from sources like Github. However, a lot of elements are not yet open-sourced, but need private kludges. The following is a list of not-yet open things.

1. <http://wordpress.let.vupr.nl>
2. <http://www.newsreader-project.eu>

Module	NL	EN	EN component
Tokenizer	ixa-pipe-tok	ixa-pipe-tok	
Topic detection		ixa-pipe-topic	EHU-topic.v30
POS/MOR	morphosyntactic_parser_nl	EHU-pos.v30	EHU-pos.v30
Constit. parser		ixa-pipe-parse	EHU-parse.v30
NERC	ixa-pipe-nerc	ixa-pipe-nerc	
UKB		UKB	EHU-ukb.v30
WSD	svm_wsd	ims-wsd	VUA-ims-wsd.v30
NED	ixa-pipe-ned	ixa-pipe-ned	
Heideltime	ixa-pipe-time		
FBK-time		FBK-time.v30	FBK-time.v30
FBK-temprel		FBK-temprel.v30	FBK-temprel.v30
FBK-causalrel		FBK-causalrel.v30	FBK-causalrel.v30
Onto-tagger	onto-tagger		
SRL	vua-srl-nl	EHU-srl-server	\verbEHU-srl-server
Nominal event det.	nominal-event-detection		
NED-reranker		domain_model	VUA-popen-nedreranker.v30
Wikify		ixa-pipe-wikify	EHU-wikify.v30
factuality			VUA-factuality.v30
Opinion-miner	opinion-miner	opinion-miner	
Eventcoref	vua-eventcoreference_v2	vua-eventcoreference_v2	

Table 1: List of modules to be installed. **Module**: functional name of the module; **NL**: modules used in Dutch pipeline

; **EN**: modules used in English pipeline; **EN components** Name of the module in the EHU repository.

Module	Section	Source	Commit	Script
Tokenizer	4.6.1	Github	56f83ce4b61680346f15e5d4e6de6293764f7383	tok
morphosyntactic parser	4.6.3	Github	807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184	mor
NERC	4.6.17	Gith./snap	ca02c931bc0b200ccdb8b5795a7552e4cc0d4802	nerc
WSD	4.6.18	Gith./snap	030043903b42f77cd20a9b2443de137e2efe8513	wsd
Onto-tagger	4.6.21	snapshot		onto
Heideltime	4.6.23	Gith./snap.	da4604a7b33975e977017440cbc10f7d59917ddf	heidelttime
SRL	4.6.24	Github	675d22d361289ede23df11dcd17195f008c54bf	srl
SRL-POST	4.6.25	snapshot		postsrl
NED	4.6.20	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	4.6.16	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	4.6.26	snapshot		evcoref
Opinion miner	4.6.29	Github		opininin
Framenet SRL	4.6.22	snapshot		fsrl
Dbpedia_ner	4.6.27	Github	ab1dcdbd860f0ff29bc979f646dc382122a101fc2	dbpner

Table 2: List of the modules to be installed. Column description: **directory**: Name of the subdirectory below subdirectory *modules* in which it is installed; **source**: From where the module has been obtained; **commit**: Commit-name or version-tag **script**: Script to be included in a pipeline. **Note**: The tokenizer module has been temporarily obtained from the snapshot, because the commit that we used has disappeared from the Github repository.

Module	Version	Section	Source
KafNafParserPy	Feb 1, 2015	3.4.3	Github
Alpino	20706	4.5.4	RUG
Ticcutils	0.7	4.5.6	ILK
Timbl	6.4.6	4.5.6	ILK
Treetagger	3.2	4.5.5	Uni. München
Spotlight server	0.7	4.5.8	Spotlight

Table 3: List of the modules to be installed. Column description: **directory**: Name of the subdirectory below *mod* in which it is installed; **Source**: From where the module has been obtained; **script**: Script to be included in a pipeline.

1.3 Multi-linguality

This version of the pipeline is multi-lingual, i.e. it can annotate Dutch as well as English documents. It finds the language of the document in the `language` attribute of the `NAF` element. Actually, the current version is bi-lingual, because it is only able to process Dutch or English documents.

1.4 File-structure of the pipeline

The files that make up the pipeline are organised in set of directories as shown in figure 1. The

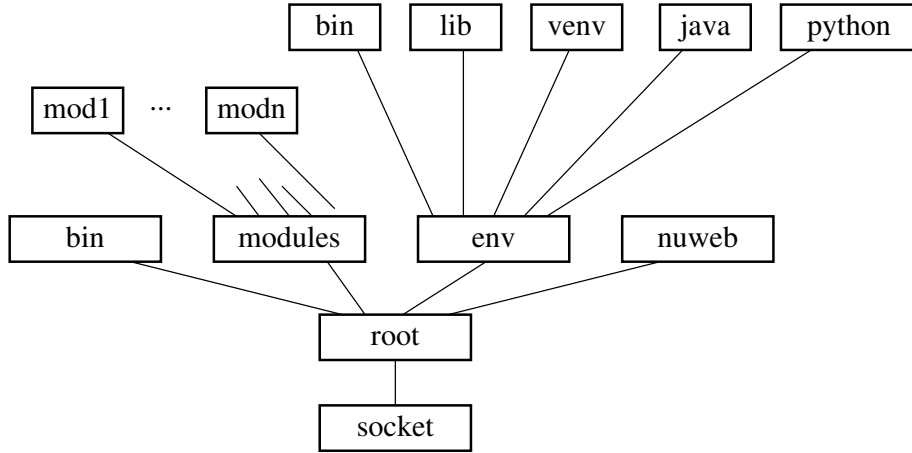


Figure 1: *Directory-structure of the pipeline (see text).*

directories have the following functions.

socket: The directory in the host where the pipeline is to be implemented.

root: The root of the pipeline directory-structure.

nuweb: This directory contains this document and everything to create the pipeline from the open sources of the modules.

modules: Contains subdirectories with the NLP modules that can be applied in the pipeline.

bin: Contains for each of the applicable modules a script that reads NAF input, passes it to the module in the `modules` directory and produces the output on standard out. Furthermore, the subdirectory contains the script `install-modules` that performs the installation, and a script `test` that shows that the pipeline works in a trivial case.

env: The programming environment. It contains a.o. the Java development kit, Python, the Python virtual environment (`venv`), libraries and binaries.

< directories to create 5a > ≡
`../modules` ◇

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.
 Fragment referenced in 76a.

< directories to create 5b > ≡
`../bin ../env/bin` ◇

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.
 Fragment referenced in 76a.

< directories to create 6a > ≡
../env/lib ◇

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.
 Fragment referenced in 76a.

< directories to create 6b > ≡
../env/etc ◇

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.
 Fragment referenced in 76a.

The following macro defines variable `piperoot` and makes it to point to the root directory in figure 1. Next it defines variables that point to other directories in the figure. The value-setting of `piperoot` can be overruled by defining the variable before running any of the script. In this way the directory tree can be moved to another location, even to another computer, after successful installation.

< set variables that point to the directory-structure 6c > ≡
 if
 ["\$piperoot" == ""]
 then
 export piperoot=/mnt/sdb1/nlpp
 fi
 export pipesocket=\${piperoot%/nlpp}
 export nuwebdir=\$piperoot/nuweb
 export envdir=\$piperoot/env
 export envbindir=\$envdir/bin
 export envlibdir=\$envdir/lib
 export modulesdir=\$piperoot/modules
 export pipebin=\$piperoot/bin
 export javadir=\$envdir/java
 export jarsdir=\$javadir/jars
 ◇

Fragment defined by 6cd, 8f, 10f.
 Fragment referenced in 6e, 17a, 77.
 Uses: nuweb 72b.

Add the environment `bin` directory to `PATH`:

< set variables that point to the directory-structure 6d > ≡
 export PATH=\$envbindir:\$PATH
 ◇

Fragment defined by 6cd, 8f, 10f.
 Fragment referenced in 6e, 17a, 77.
 Defines: PATH 10bf, 11d, 46d.

Put the macro to set variables in a script that can later be sourced by the scripts of the pipeline modules.

"../env/bin/progenv" 6e≡
 #!/bin/bash
< set variables that point to the directory-structure 6c, ... >
 export progenvset=0
 ◇

File defined by 6e, 9c.

2 How to obtain modules and other material

As illustrated in tables 2 and 3, most of the modules are obtained as source-code from Github, some of the modules or parts of some modules are downloaded from a snapshot, and some of the utilities are obtained in binary form from the supplier.

This section builds standardised methods to obtain modules and utilities from Github or from the snapshot.

2.1 Location-dependency

The basic way of installation is, to clone this repository from Github on the intended location in the file-system of the target computer and then run the install-scripts. However, it may be advantageous to be able to transplant a complete installation to another location in another computer. This could be done by making all path-descriptions in all scripts relative to anchorpoints within the installation, while it may be hard to find such anchorpoints in advance. Therefore, we take another approach in which we supply a script that repairs paths-descriptions after the transplantation (section A.8).

2.2 Reversible update

This script might be used to update an existing installation. To minimize the risk that the “update” actually ruins an existing installation, move existing modules away before installing the latest version. When the new modules has been installed succesfully, the moved module will be removed. The following macro’s help to achieve this:

```

< move module 7a > ≡
  if
    [ -e @1 ]
  then
    mv @1 old.@1
  fi
  ◇

```

Fragment referenced in 8a, 14b, 63b.

```

< remove old module 7b > ≡
  rm -rf old.@1
  ◇

```

Fragment referenced in 8a, 14b, 63b.

```

< re-instate old module 7c > ≡
  mv old.@1 @1
  MESS="Replaced previous version of @1"
  < logmess (7d $MESS ) 63a >
  ◇

```

Fragment referenced in 8a, 14b, 63b.

2.3 Installation from Github

The following macro can be used to install a module from Github. Before issuing this macro, the following four variables must be set:

MODNAM: Name of the module.

DIRN: Name of the root directory of the module.

GITU: Github URL to clone from.

GITC: Github commit-name or version tag.

```

< install from github 8a > ≡
    cd $modulesdir
    < move module (8b $DIRN ) 7a >
    git clone $GITU
    if
        [ $? -gt 0 ]
    then
        < logmess (8c Cannot install current $MODNAM version ) 63a >
        < re-instate old module (8d $DIRN ) 7c >
    else
        < remove old module (8e $DIRN ) 7b >
        cd $modulesdir/$DIRN
        git checkout $GITC
    fi

```

◇

Fragment referenced in 36d, 41c, 49a, 50d, 53b, 54e, 57c, 58c.

2.4 Installation from the snapshot

The sources for the non-open parts of the pipeline are collected in directory `t_nlpp_resources`. They can be accessed via SSH from url `m4_snapshotURL`. Before installing the pipeline download the snapshot on top of directory `snapshotsocket`.

```

< set variables that point to the directory-structure 8f > ≡
    if
        [ ! $snapshotsocket ]
    then
        export snapshotsocket=/mnt/sdb1
    fi

```

◇

Fragment defined by 6cd, 8f, 10f.

Fragment referenced in 6e, 17a, 77.

The snapshot can be accessed over `scp` on URL `newsreader@kyoto.let.vu.nl`. Access is protected by a public/private key system. So, a private key is needed and this program expects to find the key as `$pipesocket/nrkey`. The key can be obtained from the author. Let us check whether we indeed do have the key:

```

< check this first 8g > ≡
    if
        [ ! -e $pipesocket/nrkey ]
    then
        echo "No key to connect to snapshot!"
        exit 1
    fi

```

◇

Fragment defined by 8g, 21e.

Fragment referenced in 17a.

Update the local snapshot repository.

```

< get the snapshot 9a > ≡
    cd $snapshotsocket
    rsync -e "ssh -i $HOME/nrkey" -rlt newsreader@kyoto.let.vu.nl:t_nlpp_resources .
    ◇

```

Fragment referenced in 17a.

3 Java and Python environment

To be independent from the software environment of the host computer and to perform reproducible processing, the pipeline features its own Java and Python environment. The costs of this feature are that the pipeline takes more disk-space by reproducing infra-structure that is already present in the system and that installation takes more time.

The following macro generates a script that specifies the programming environment. Initially it is empty, because we have to create the programming environment first.

```

< create javapython script 9b > ≡
    echo '#!/bin/bash' > /mnt/sdb1/nlpp/env/bin/javapython
    ◇

```

Fragment referenced in 17a.

Cause the module scripts to read the javapython script.

```

"../env/bin/progenv" 9c ≡
    source $envbindir/javapython
    ◇

```

File defined by 6e, 9c.

3.1 Java

To install Java, download `server-jre-7u72-linux-x64.tar.gz` from <http://www.oracle.com/technetwork/java/javase/downloads/server-jre7-downloads-1931105.html>. Find it in the root directory and unpack it in a subdirectory of `envdir`.

```

< directories to create 9d > ≡
    ../env/java ◇

```

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.

Fragment referenced in 76a.

```

< set up java 9e > ≡
    < begin conditional install (9f java_installed ) 15c >
        cd $envdir/java
        tar -xzf $snapshotsocket/t_nlpp_resources/server-jre-7u72-linux-x64.tar.gz
    < end conditional install (9g java_installed ) 16a >
    ◇

```

Fragment defined by 9e, 10b.

Fragment referenced in 17a.

Remove the java-ball when cleaning up:

```
< clean up 10a > ≡
    rm -rf $pipesocket/server-jre-7u72-linux-x64.tar.gz
    ◇
```

Fragment defined by 10a, 11a, 25c, 66d.

Fragment referenced in 65c.

Set variables for Java.

```
< set up java 10b > ≡

    echo 'export JAVA_HOME=$envdir/java/jdk1.7.0_72' >> /mnt/sdb1/nlpp/env/bin/javapython
    echo 'export PATH=$JAVA_HOME/bin:$PATH' >> /mnt/sdb1/nlpp/env/bin/javapython
    export JAVA_HOME=$envdir/java/jdk1.7.0_72
    export PATH=$JAVA_HOME/bin:$PATH
    ◇
```

Fragment defined by 9e, 10b.

Fragment referenced in 17a.

Uses: PATH 6d.

Put jars in the jar subdirectory of the java directory:

```
< directories to create 10c > ≡
    ../env/java/jars ◇
```

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.

Fragment referenced in 76a.

3.2 Maven

Some Java-based modules can best be compiled with [Maven](#).

```
< directories to create 10d > ≡
    ../env/apache-maven-3.0.5 ◇
```

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.

Fragment referenced in 76a.

```
< install maven 10e > ≡
    cd $envdir
    wget http://apache.rediris.es/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-
    bin.tar.gz
    tar -xzf apache-maven-3.0.5-bin.tar.gz
    rm apache-maven-3.0.5-bin.tar.gz
    ◇
```

Fragment referenced in 17a.

```
< set variables that point to the directory-structure 10f > ≡
    export MAVEN_HOME=$envdir/apache-maven-3.0.5
    export PATH=${MAVEN_HOME}/bin:${PATH}
    ◇
```

Fragment defined by 6cd, 8f, 10f.

Fragment referenced in 6e, 17a, 77.

Uses: PATH 6d.

When the installation has been done, remove maven, because it is no longer needed.

```

< clean up 11a > ≡
  rm -rf ../env/apache-maven-3.0.5
  < remove installed-variable (11b maven_installed ) 16b >
  ◇

```

Fragment defined by 10a, 11a, 25c, 66d.

Fragment referenced in 65c.

3.3 Java 1.6

Java 1.7 is able to run nearly all the modules of the pipeline that are based on Java. However, there is one exception, i.e. the `ims-wsd` module, that needs Java version 1.6. So, we have to install that version of Java as well.

```

< install Java 1.6 11c > ≡
  cd $envdir/java
  $snapshotsocket/t_nlpp_resources/jre-6u45-linux-x64.bin
  ◇

```

Fragment referenced in 17a.

Insert the following macro in scripts that need to run Java 1.6.

```

< set up Java 1.6 11d > ≡
  export JAVA_HOME=$envdir/java/jre1.6.0_45
  export PATH=$JAVA_HOME/bin:$PATH
  ◇

```

Fragment referenced in 39e.

Uses: PATH 6d.

3.4 Python

Set up the environment for Python (version 2.7). I could not find an easy way to set up Python from scratch. Therefore we will use Python 2.7 if it has been installed on the host. Otherwise, we will use a binary distribution obtained from [ActiveState](#). A tarball of ActivePython can be obtained from the snapshot.

In order to be independent of the software on the host, we generate a virtual Python environment. In the virtual environment we will install `KafNafParserPy` and other Python packages that are needed.

```

< set up python 11e > ≡
  < check/install the correct version of python 12a >
  < create a virtual environment for Python 12c >
  < activate the python environment 13a, ... >
  < update pip 13d >
  < install kafnafparserpy 14b >
  < install python packages 15a, ... >
  ◇

```

Fragment referenced in 17a.

```

< check/install the correct version of python 12a > ≡
pythonok='python --
version 2>&1 | gawk '{if(match($2, "2.7")) print "yes" ; else print "no" }'
if
[ "$pythonok" == "no" ]
then
  < install ActivePython 12b >
fi
◇

```

Fragment referenced in 11e.
 Defines: pythonok Never used.
 Uses: print 70a.

Unpack the tarball in a temporary directory and install active python in the `env` subdirectory of `nlpp`. It turns out that you must upgrade `pip`, `virtualenv` and `setuptools` after the installation (see <https://github.com/ActiveState/activepython-docker/commit/10fff72069e51dbd36330cb8a7c2f0845bcd7b3> and <https://github.com/ActiveState/activepython-docker/issues/1>).

```

< install ActivePython 12b > ≡
pytinsdir='mktemp -d -t activepyt.XXXXXX'
cd $pytinsdir
tar -xzf $snapshotsocket/t_nlpp_resources/ActivePython-2.7.8.10-linux-x86_64.tar.gz
acdir='ls -1'
cd $acdir
./install.sh -I $envdir
cd $piperoot
rm -rf $pytinsdir
pip install -U pip virtualenv setuptools
◇

```

Fragment referenced in 12a.
 Uses: virtualenv 12d.

3.4.1 Virtual environment

Create a virtual environment. To begin this, we need the Python module `virtualenv` on the host.

```

< create a virtual environment for Python 12c > ≡
< test whether virtualenv is present on the host 12d >
cd $envdir
virtualenv venv
◇

```

Fragment referenced in 11e.
 Uses: virtualenv 12d.

```

< test whether virtualenv is present on the host 12d > ≡
which virtualenv
if
[ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
◇

```

Fragment referenced in 12c.
 Defines: virtualenv 12bc.

```

< activate the python environment 13a > ≡
    source $envdir/venv/bin/activate
    echo 'source $envdir/venv/bin/activate' >> /mnt/sdb1/nlpp/env/bin/javapython
    ◇

```

Fragment defined by 13ac.
 Fragment referenced in 11e, 17a.
 Defines: activate 14a.

Subdirectory `$envdir/python` will contain general Python packages like `KafnafParserPy`.

```

< directories to create 13b > ≡
    ../env/python ◇

```

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.
 Fragment referenced in 76a.

Activation of Python include pointing to the place where Python packages are:

```

< activate the python environment 13c > ≡
    echo ex-
    port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /mnt/sdb1/nlpp/env/bin/javapython
    export PYTHONPATH=$envdir/python:$PYTHONPATH
    ◇

```

Fragment defined by 13ac.
 Fragment referenced in 11e, 17a.
 Defines: PYTHONPATH Never used.

Update pip in the virtual environment, because otherwise it keeps complaining about outdated versions

```

< update pip 13d > ≡
    pip install --upgrade pip
    ◇

```

Fragment referenced in 11e.

3.4.2 Transplant the virtual environment

It turns out that the script “activate” to engage the virtual environment contains an absolute path, in the definition of `VIRTUAL_ENV`

```

< set paths after transplantation 14a > ≡
transdir='mktemp -d -t trans.XXXXXX'
cd $transdir
cat <<EOF >redef.awk
#!/usr/bin/gawk -f
BEGIN { envd="$envdir/venv"}

/^VIRTUAL_ENV=/ { print "VIRTUAL_ENV=\"\" envd \"\"\"
                  next
                  }

{print}
EOF

mv $envdir/venv/bin/activate .
gawk -f redef.awk ./activate > $envdir/venv/bin/activate
cd $projroot
rm -rf $transdir
◇

```

Fragment referenced in 77.
 Uses: activate 13a, print 70a.

3.4.3 KafNafParserPy

A cornerstone Pythonmodule for the pipeline is [KafNafParserPy](#). It is a feature of this module that you cannot install it with PIP, but that you can add it to your PYTHONPATH.

```

< install kafnafparserpy 14b > ≡
cd $envdir/python
DIRN=KafNafParserPy
< move module (14c $DIRN ) 7a >
git clone https://github.com/cltl/KafNafParserPy.git
if
[ $? -gt 0 ]
then
  < logmess (14d Cannot install current $DIRN version ) 63a >
  < re-instate old module (14e $DIRN ) 7c >
else
  < remove old module (14f $DIRN ) 7b >
fi
◇

```

Fragment referenced in 11e.

3.4.4 Python packages

Install python packages:

lxml:

pyyaml: for coreference-graph

pynaf:

requests: for networkx

networkx: for corefbase.

```

< install python packages 15a > ≡
    pip install lxml
    pip install pyyaml
    pip install --upgrade git+https://github.com/ixa-ehu/pynaf.git
    pip install --upgrade requests
    pip install --upgrade networkx
    ◇

```

Fragment defined by 15a, 55g.

Fragment referenced in 11e.

Defines: `lxml` Never used, `networkx` Never used, `pyyaml` Never used.

4 Installation of the modules

This section describes how the modules are obtained from their (open-)source and installed.

4.1 Conditional installation of the modules

Next section generates a script that installs everything.

Installation is very time-intensive. To prevent that everything is re-installed every time that the module-installer is run, there is a list of variables, the *modulelist*, that are set when a module has been installed. To re-install that module, remove the variable from the list and then re-run the installer. It maintains a list of the modules and utilities that it has installed and installs only modules and utilities that are not on the list. So in order to re-install a module that has already been installed, remove it from the list and then re-run the module-installer.

The modulelist is in fact a script named `/mnt/sdb1/nlpp/installed_modules` that sets Bash variables. It ought to be sourced if it is present.

Initially the list is not present. When a module or a utility has been installed, an instruction to set a variable is written in or appended to the list.

```

< read the list of installed modules 15b > ≡
    if
        [ -e /mnt/sdb1/nlpp/installed_modules ]
    then
        source /mnt/sdb1/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 17a.

```

< begin conditional install 15c > ≡
    if
        [ ! $01 ]
    then
        ◇

```

Fragment referenced in 9e, 17a, 18aj, 19aj, 20ahq, 21a, 51d, 52a.

```

< else conditional install 15d > ≡
    else
        ◇

```

Fragment never referenced.

```

< end conditional install 16a > ≡
    echo "export @1=0" >> /mnt/sdb1/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 9e, 17a, 18aj, 19aj, 20ahq, 21a, 51d, 52a.

Remove a variable from the list of installed modules, e.g. after a clean-up.

```

< remove installed-variable 16b > ≡
    cd $piperoot
    mv /mnt/sdb1/nlpp/installed_modules old.modulelist
    cat old.modulelist | gawk '/@1/ {next}; {print}' >/mnt/sdb1/nlpp/installed_modules
    ◇

```

Fragment referenced in 11a.

Uses: **print** 70a.

4.2 The installation script

The installation is performed by script `install-modules`.

The first part of the script installs the utilities:


```

"../bin/install-modules" 17a≡
    #!/bin/bash
    echo Set up environment
    < set variables that point to the directory-structure 6c, ... >
    < read the list of installed modules 15b >
    < check this first 8g, ... >
    < begin conditional install (17b repo_installed) 15c >
        < get the snapshot 9a >
    < end conditional install (17c repo_installed) 16a >
    < variables of install-modules 62c >
    < create javapython script 9b >
    echo ... Java
    < set up java 9e, ... >
    < begin conditional install (17d maven_installed) 15c >
        < install maven 10e >
    < end conditional install (17e maven_installed) 16a >
    < begin conditional install (17f java16_installed) 15c >
        < install Java 1.6 11c >
    < end conditional install (17g java16_installed) 16a >

    echo ... Python
    if
    [ $python_installed ]
    then
        < activate the python environment 13a, ... >
    fi
    < begin conditional install (17h python_installed) 15c >
        < set up python 11e >
    < end conditional install (17i python_installed) 16a >
    < begin conditional install (17j sematree_installed) 15c >
        < install sematree 22b >
    < end conditional install (17k sematree_installed) 16a >
    echo ... Alpino
    < begin conditional install (17l alpino_installed) 15c >
        < install Alpino 25a >
    < end conditional install (17m alpino_installed) 16a >
    echo ... Spotlight
    < begin conditional install (17n spotlight_installed) 15c >
        < install the Spotlight server 29a, ... >
    < end conditional install (17o spotlight_installed) 16a >
    echo ... Treetagger
    < begin conditional install (17p treetagger_installed) 15c >
        < install the treetagger utility 25d, ... >
    < end conditional install (17q treetagger_installed) 16a >
    echo ... Ticcutils and Timbl
    < begin conditional install (17r ticctimbl_installed) 15c >
        < install the ticcutils utility 27b >
        < install the timbl utility 27c >
    < end conditional install (17s ticctimbl_installed) 16a >
    echo ... Boost
    < begin conditional install (17t boost_installed) 15c >
        < install boost 28b >
    < end conditional install (17u boost_installed) 16a >

    echo ... VUA-pylib, SVMlight, CRFsuite
    < begin conditional install (17v miscutils_installed) 15c >
        < install VUA-pylib 34a >
        < install SVMlight 34b >
        < install CRFsuite 35a >
    < end conditional install (17w miscutils_installed) 16a >

```

◇

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

Next, install the modules:

```
"../bin/install-modules" 18a≡
echo Install modules
  <begin conditional install (18b tokenizer_installed) 15c>
    echo ... Tokenizer
    <install the tokenizer 35b>
  <end conditional install (18c tokenizer_installed) 16a>
  <begin conditional install (18d topic_installed) 15c>
    echo ... Topic detector
    <install the topic analyser 36a>
  <end conditional install (18e topic_installed) 16a>
  <begin conditional install (18f morpar_installed) 15c>
    echo ... Morphosyntactic parser
    <install the morphosyntactic parser 36d>
  <end conditional install (18g morpar_installed) 16a>
  <begin conditional install (18h pos_installed) 15c>
    echo "... Pos tagger (for english docs)"
    <install the pos tagger 37b>
  <end conditional install (18i pos_installed) 16a>
◇
```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```
"../bin/install-modules" 18j≡
  <begin conditional install (18k constparse_installed) 15c>
    echo "... Constituent parser (for english docs)"
    <install the constituents parser 37e>
  <end conditional install (18l constparse_installed) 16a>
  <begin conditional install (18m nerc_installed) 15c>
    echo ... NERC
    <install the NERC module 47d>
  <end conditional install (18n nerc_installed) 16a>
  <begin conditional install (18o ned_installed) 15c>
    echo ... NED
    <install the NED module 50d>
  <end conditional install (18p ned_installed) 16a>
  <begin conditional install (18q nedrer_installed) 15c>
    echo ...NED reranker
    <install the NED-reranker module 38c>
  <end conditional install (18r nedrer_installed) 16a>
  <begin conditional install (18s wikify_installed) 15c>
    echo ...WIKIfy module
    <install the wikify module 38f>
  <end conditional install (18t wikify_installed) 16a>
◇
```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 19a≡
  < begin conditional install (19b UKB_installed ) 15c >
    echo ... UKB module
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-ukb.v30.tgz
  < end conditional install (19c UKB_installed ) 16a >
  < begin conditional install (19d ims_wsd_installed ) 15c >
    echo ...ims-wsd module
    < install the ims-wsd module 39d >
  < end conditional install (19e ims_wsd_installed ) 16a >
  < begin conditional install (19f srl_server_installed ) 15c >
    echo ...srl-server module
    < install the srl-server module 40a >
  < end conditional install (19g srl_server_installed ) 16a >
  < begin conditional install (19h srl_dutch_nominals_installed ) 15c >
    echo ...srl-dutch-nominal module
    < install the srl-dutch-nominals module 41c >
  < end conditional install (19i srl_dutch_nominals_installed ) 16a >
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 19j≡
  < begin conditional install (19k FBK_time_installed ) 15c >
    echo ... FBK-time module
    < install the FBK-time module 42 >
  < end conditional install (19l FBK_time_installed ) 16a >
  < begin conditional install (19m FBK_templrel_installed ) 15c >
    echo ... FBK-templrel module
    < install the FBK-templrel module 44b >
  < end conditional install (19n FBK_templrel_installed ) 16a >
  < begin conditional install (19o FBK_causalrel_installed ) 15c >
    echo ... FBK-causalrel module
    < install the FBK-causalrel module 45c >
  < end conditional install (19p FBK_causalrel_installed ) 16a >
  < begin conditional install (19q factuality_installed ) 15c >
    echo ... factuality module
    < install the factuality module 46c >
  < end conditional install (19r factuality_installed ) 16a >
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 20a≡
  < begin conditional install (20b corefb_installed ) 15c >
    echo ... Coreference base
    < install coreference-base 47a >
  < end conditional install (20c corefb_installed ) 16a >
  < begin conditional install (20d wsd_installed ) 15c >
    echo ... WSD
    < install the WSD module 49a >
  < end conditional install (20e wsd_installed ) 16a >
  < begin conditional install (20f onto_installed ) 15c >
    echo ... Ontotagger
    < install the onto module 52d >
  < end conditional install (20g onto_installed ) 16a >
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 20h≡
  < begin conditional install (20i heidel_installed ) 15c >
    echo ... Heideltime
    < install the heideltime module 53a >
  < end conditional install (20j heidel_installed ) 16a >
  < begin conditional install (20k SRL_installed ) 15c >
    echo ... SRL
    < install the srl module 54e >
  < end conditional install (20l SRL_installed ) 16a >
  < begin conditional install (20m eventcoref_installed ) 15c >
    echo ... Event-coreference
    < install the event-coreference module 56d >
  < end conditional install (20n eventcoref_installed ) 16a >
  < begin conditional install (20o lu2synset_installed ) 15c >
    echo ... lu2synset
    < install the lu2synset converter 50a >
  < end conditional install (20p lu2synset_installed ) 16a >
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 20q≡
  < begin conditional install (20r dbpner_installed ) 15c >
    echo ... dbpedia-ner
    < install the dbpedia-ner module 57c >
  < end conditional install (20s dbpner_installed ) 16a >
  < begin conditional install (20t nomevent_installed ) 15c >
    echo ... nominal event
    < install the nomevent module ? >
  < end conditional install (20u nomevent_installed ) 16a >
  < begin conditional install (20v post_SRL_installed ) 15c >
    echo ... post-SRL
    < install the post-SRL module 56a >
  < end conditional install (20w post_SRL_installed ) 16a >
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

"../bin/install-modules" 21a≡
  < begin conditional install (21b opimin_installed ) 15c >
    echo ... opinion-miner
    < install the opinion-miner 58c, ... >
  < end conditional install (21c opimin_installed ) 16a >

  echo Final
  ◇

```

File defined by 17a, 18aj, 19aj, 20ahq, 21a.

```

< make scripts executable 21d > ≡
  chmod 775 ../bin/install-modules
  ◇

```

Fragment defined by 21d, 30g, 76b.

Fragment referenced in 76c.

4.3 Check availability of resources

Test for some resources that we need and that may not be available on this host.

```

< check this first 21e > ≡
  < check whether mercurial is present 21f >
  ◇

```

Fragment defined by 8g, 21e.

Fragment referenced in 17a.

```

< check whether mercurial is present 21f > ≡
  which hg
  if
    [ $? -ne 0 ]
  then
    echo Please install Mercurial.
    exit 1
  fi
  ◇

```

Fragment referenced in 21e.

Defines: hg Never used.

4.4 Parameters in module-scripts

Some modules need parameters. All modules need a language specification. The language can be passed as exported variable `naflang`, but it can also be passed as argument `-l`. Furthermore, some modules need contact with a Spotlight server. With the arguments `-h` and `-b` the host and port of a running Spotlight-server can be passed.

The code to obtain command-line arguments in Bash has been obtained from [Stackoverflow](#). The following fragment reads the arguments `-l language`, `-h spotlighthost` and `-p spotlightport`:

```

< get cmdline-arguments 22a > ≡
    while [[ $# > 1 ]]
    do
        key="$1"

        case $key in
            -l|--language)
                naflang="$2"
                shift # past argument
                ;;
            -h|--spothost)
                spotlighthost="$2"
                shift # past argument
                ;;
            -p|--spotport)
                spotlightport="$2"
                shift # past argument
                ;;
            *)
                # unknown option
                ;;
        esac
        shift # past argument or value
    done
    ◇

```

Fragment referenced in 29f.

Uses: `naflang` 24a.

4.5 Install utilities and resources

4.5.1 Process synchronisation

We will see that we sometimes have to install server-applications. However, it is possible that multiple processes are running pipeline modules in parallel, and then it may occur that two instances of a module try to install the same server-application. Therefore, we must make sure that only one application at a time is able to start the server.

The program `sematree`, found at <http://www.pixelbeat.org/scripts/sematree/> enables to do this. When invoked with argument “acquire”, the name of a “lockfile” and a time to wait (-1 means “wait an indefinite time”), it checks whether the lockfile exists. If that is the case, it either waits or fails. When the lockfile is not (or no longer) present, `sematree` creates the lockfile.

When installing `Sematree`, set the default directory for lock-files. We set this as a subdirectory of the `env` tree. However, in some cases, notably when running in a node in Lisa, we need a directory on the filesystem of the node itself.

```

< install sematree 22b > ≡
    cat $snapshotsocket/t_nlpp_resources/sematree | \
        sed "s|/var/run|mnt/sdb1/nlpp/env/etc/sematree|g" \
        > $envbindir/sematree
    ◇

```

Fragment referenced in 17a.

4.5.2 Prefix of scripts that run modules

Each module will be run by a Bash script located in subdirectory `bin`. The start of these scripts will have similar content. Insert the following macro to include this similar content, with the name of the module-directory as argument:

```

< start of module-script 23a > ≡
    #!/bin/bash
    < get the path to the module-script 23b >
    source /mnt/sdb1/nlpp/env/bin/progenv
    export LC_ALL=en_US.UTF-8
    export LANG=en_US.UTF-8
    export LANGUAGE=en_US.UTF-8
    ROOT=$piperoot
    MODDIR=$modulesdir/@1
    < run in subshell when naflang is not known 24b >
    < run only if language is English or Dutch 24c >
    ◇

```

Fragment referenced in 35c, 36be, 37c, 38adg, 39be, 40bdf, 41d, 43a, 45a, 46ad, 47b, 48ce, 49d, 50b, 51b, 52eg, 54c, 55a, 56b, 57ad, 58a, 59b.

Set variable `scriptpath` to the full path of the script that is running, order to be able to re-run it.

```

< get the path to the module-script 23b > ≡
    scriptdir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
    scriptname=${0##*/}
    scriptpath=$scriptdir/$scriptname
    ◇

```

Fragment referenced in 23a.

Defines: `scriptpath` 24b.

4.5.3 Language detection

The following script `../env/bin/langdetect.py` discerns the language of a NAF document. If it cannot find that attribute it prints `unknown`. The macro `set the language variable` uses this script to set variable `naflang`. All pipeline modules expect that this variable has been set.

```

"../env/bin/langdetect.py" 23c≡
    #!/usr/bin/env python
    # langdetect -- Detect the language of a NAF document.
    #
    import xml.etree.ElementTree as ET
    import sys
    import re
    xmldoc = sys.stdin.read()
    #print xmldoc
    root = ET.fromstring(xmldoc)
    # print root.attrib['lang']
    lang = "unknown"
    for k in root.attrib:
        if re.match(".*lang$", k):
            language = root.attrib[k]
    print language
    ◇

```

Uses: `print` 70a.

```

< set the language variable 24a > ≡
    naflang='cat @1 | /mnt/sdb1/nlpp/env/bin/langdetect.py'
    export naflang
    ◇

```

Fragment referenced in 60a.

Defines: **naflang** 22a, 24bc, 30a, 31ab, 33a, 35c, 48ae, 57a, 60a.

The module-scripts depend on the existence of variable **naflang**. In most cases this is not a problem because the scripts run in a surrounding script that sets **naflang**. However, a users may occasionally run a module-script stand-alone e.g. to debug. In that case, we can read the language from the NAF, set variable **naflang**, and then run the module-script in a subshell. We assume that variable **scriptpath** contains the path of the script itself.

The macro does the following if **naflang** has not been set:

1. Save the content of standard in to a temporary file.
2. Run **langdetect** with the temporary file as input and set the **naflang** variable.
3. Run the script **\$scriptpath** (i.e. itself) with the temporary file as input.
4. Remove the temporary file.
5. Exit itself with the errorcode of the sub-script that it has run.

```

< run in subshell when naflang is not known 24b > ≡
    if
        [ "$naflang" == "" ]
    then
        naffile='mktemp -t naf.XXXXXX'
        cat >$naffile
        naflang='cat $naffile | python $envbindir/langdetect.py'
        export naflang
        cat $naffile | $scriptpath
        result=$?
        rm $naffile
        exit $result
    fi
    ◇

```

Fragment referenced in 23a.

Uses: **naflang** 24a, **scriptpath** 23b.

```

< run only if language is English or Dutch 24c > ≡
    if
        [ ! "$naflang" == "nl" ] && [ ! "$naflang" == "en" ]
    then
        exit 6
    fi
    ◇

```

Fragment referenced in 23a.

Uses: **naflang** 24a.

4.5.4 Alpino

Binary versions of Alpino can be obtained from the [official Alpino website](#) of Gertjan van Noort. However, it seems that older versions are not always retained there, or the location of older versions change. Therefore we have a copy in the snapshot.

Module

```

< install Alpino 25a > ≡
  if
    [ ! $alpino_installed ]
  then
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
    sicstus.tar.gz
    echo "export alpino_installed=0" >> /mnt/sdb1/nlpp/installed_modules
  fi
  ◇

```

Fragment referenced in 17a.

Currently, alpino is not used as a pipeline-module on its own, but it is included in other pipeline-modules. Modules that use Alpino should set the following variables:

```

< set alpinohome 25b > ≡
  export ALPINO_HOME=$modulesdir/Alpino
  ◇

```

Fragment referenced in 36e.

Defines: ALPINO_HOME Never used.

Remove the tarball when cleaning up:

```

< clean up 25c > ≡
  rm -rf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
  sicstus.tar.gz
  ◇

```

Fragment defined by 10a, 11a, 25c, 66d.

Fragment referenced in 65c.

4.5.5 Treetagger

Installation of Treetagger goes as follows (See [Treetagger's homepage](#)):

1. Download and unpack the Treetagger tarball. This generates the subdirectories `bin`, `cmd` and `doc`
2. Download and unpack the tagger-scripts tarball

The location where Treetagger comes from and the location where it is going to reside:

```

< install the treetagger utility 25d > ≡
  TREETAGDIR=treetagger
  TREETAG_BASIS_URL=http://www.cis.uni-muenchen.de/%7Eeschmid/tools/TreeTagger/data/
  TREETAGURL=http://www.cis.uni-muenchen.de/%7Eeschmid/tools/TreeTagger/data/
  ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

The source tarball, scripts and the installation-script:

```

< install the treetagger utility 26a > ≡
    TREETAGSRC=tree-tagger-linux-3.2.tar.gz
    TREETAGSCRIPTS=tagger-scripts.tar.gz
    TREETAG_INSTALLSCRIPT=install-tagger.sh
    ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

Parametersets:

```

< install the treetagger utility 26b > ≡
    DUTCHPARS_UTF_GZ=dutch-par-linux-3.2-utf8.bin.gz
    DUTCH_TAGSET=dutch-tagset.txt
    DUTCHPARS_2_GZ=dutch2-par-linux-3.2-utf8.bin.gz
    ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

Download everything in the target directory:

```

< install the treetagger utility 26c > ≡
    mkdir -p $modulesdir/$TREETAGDIR
    cd $modulesdir/$TREETAGDIR
    wget $TREETAGURL/$TREETAGSRC
    wget $TREETAGURL/$TREETAGSCRIPTS
    wget $TREETAGURL/$TREETAG_INSTALLSCRIPT
    wget $TREETAGURL/$DUTCHPARS_UTF_GZ
    wget $TREETAGURL/$DUTCH_TAGSET
    wget $TREETAGURL/$DUTCHPARS_2_GZ
    ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

Run the install-script:

```

< install the treetagger utility 26d > ≡
    chmod 775 $TREETAG_INSTALLSCRIPT
    ./ $TREETAG_INSTALLSCRIPT
    ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

Make the treetagger utilities available for everybody.

```

< install the treetagger utility 26e > ≡
    chmod -R o+rx $modulesdir/$TREETAGDIR/bin
    chmod -R o+rx $modulesdir/$TREETAGDIR/cmd
    chmod -R o+r $modulesdir/$TREETAGDIR/doc
    chmod -R o+rx $modulesdir/$TREETAGDIR/lib
    ◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

Remove the tarballs:

```

< install the treetagger utility 27a > ≡
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇

```

Fragment defined by 25d, 26abcde, 27a.

Fragment referenced in 17a.

4.5.6 Timbl and Ticcutils

Timbl and Ticcutils are installed from their source-tarballs. The installation is not (yet?) completely reproducibe because it uses the C-compiler that happens to be available on the host. Installation involves:

1. Download the tarball in a temporary directory.
2. Unpack the tarball.
3. cd to the unpacked directory and perform `./configure`, `make` and `make install`. Note the argument that causes the files to be installed in the `lib` and the `bin` sub-directories of the `env` directory.

```

< install the ticcutils utility 27b > ≡
URL=http://software.ticc.uvt.nl/ticcutils-0.7.tar.gz
TARB=ticcutils-0.7.tar.gz
DIR=ticcutils-0.7
< unpack ticcutils or timbl 27d >
◇

```

Fragment referenced in 17a, 28a.

```

< install the timbl utility 27c > ≡
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
< unpack ticcutils or timbl 27d >
◇

```

Fragment referenced in 17a, 28a.

```

< unpack ticcutils or timbl 27d > ≡
SUCCES=0
ticbeldir='mktemp -t -d tickbel.XXXXXX'
cd $ticbeldir
tar -xzf $snapshotsocket/t_nlpp_resources/$TARB
cd $DIR
sh ./bootstrap.sh
./configure --prefix=$envdir
make
make install
cd $piperoot
rm -rf $ticbeldir
◇

```

Fragment referenced in 27bc.

When the installation has been transplanted, Timbl and Ticcutils have to be re-installed.

```
⟨ re-install modules after the transplantation 28a ⟩ ≡
  ⟨ install the ticcutils utility 27b ⟩
  ⟨ install the timbl utility 27c ⟩
  ◇
```

Fragment referenced in 77.

4.5.7 The Boost library

Theoretically, it is possible to download a tarball with boost from [it's repository](#) and then install it. However, I did not succeed in doing this. Therefore, I ripped the installed boost from Surfsara's Hadoop installation and put it in the `env` dir.

```
⟨ install boost 28b ⟩ ≡
  cd $envdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20160103_boost_1_54_bin.tgz
  ◇
```

Fragment referenced in 17a.

4.5.8 Spotlight

A Spotlight server occupies a lot of memory and we need two of them, one for each language. We may be lucky and have a spotlight server running somewhere. Otherwise we have to install the server ourselves.

Install Spotlight in the way that Itziar Aldabe (<mailto:itziar.aldabe@ehu.es>) described:

The NED module works for English, Spanish, Dutch and Italian. The module returns multiple candidates and correspondences for all the languages. If you want to integrate it in your Dutch or Italian pipeline, you will need:

1. The jar file with the dbpedia-spotlight server. You need the version that Aitor developed in order to correctly use the "candidates" option. You can copy it from the English VM. The jar file name is `dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar`
2. The Dutch/Italian model for the dbpedia-spotlight. You can download them from: <http://spotlight.sztaki.hu/downloads/>
3. The jar file with the NED module: `ixa-pipe-ned-1.0.jar`. You can copy it from the English VM too.
4. The file: `wikipedia-db.v1.tar.gz`. You can download it from: <http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz>. This file contains the required information to do the mappings between the wikipedia-entries. The zip file contains three files: `wikipedia-db`, `wikipedia-db.p` and `wikipedia-db.t`

To start the dbpedia server: Italian server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar \
  it http://localhost:2050/rest
```

Dutch server:

```
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://localhost:2
```

We set 8Gb for the English server, but the Italian and Dutch Spotlight will require less memory.

So, let us do that:

```

< install the Spotlight server 29a > ≡
  cd $envdir
  tar -xzf $snapshotsocket/t_nlpp_resources/spotlightnl.tgz
  cd $envdir/spotlight
  < get spotlight model ball (29b nl.tar.gz ) 29d >
  < get spotlight model ball (29c en_2+2.tar.gz ) 29d >
  ◇

```

Fragment defined by 29ae.

Fragment referenced in 17a.

```

< get spotlight model ball 29d > ≡
  wget http://spotlight.sztaki.hu/downloads/archive/2014/@1
  tar -xzf @1
  rm @1
  ◇

```

Fragment referenced in 29a.

We choose to put the Wikipedia database in the spotlight directory.

```

< install the Spotlight server 29e > ≡
  cd $envdir/spotlight
  wget http://ixa2.si.ehu.es/ixa-pipes/models/wikipedia-db.v1.tar.gz
  tar -xzf wikipedia-db.v1.tar.gz
  rm wikipedia-db.v1.tar.gz
  ◇

```

Fragment defined by 29ae.

Fragment referenced in 17a.

The macro `check/start spotlight` does the following:

1. Check whether spotlight runs on the default spotlighthost.
2. If that is not the case, and the defaulthost is not `localhost`, check whether Spotlight runs on `localhost`.
3. If a running spotlightserver is still not found, start a spotlightserver on `localhost`.

Start Spotlight if it doesn't run already. Spotlight ought to run on `localhost` unless variable `spotlighthost` exists. In that case, check whether a Spotlight server can be contacted on that host. Otherwise, change `spotlighthost` to `localhost` and check whether a Spotlight server runs there. If that is not the case, start up a Spotlight server on `localhost`.

The following script, `check_start_spotlight`, has the following three optional arguments:

language: Default is exported variable `naflang` if it exists, or `en`.

spotlighthost: Name of a host that probably runs a Spotlightserver. Default: exported variable `spotlighthost` if it exists, or `localhost`.

spotlightport: Default: exported variable `spotlightport` if it exists or either 2020 or 2060 for English resp. Dutch.

```

"../bin/check_start_spotlight" 29f≡
  #!/bin/bash
  source /mnt/sdb1/nlpp/env/bin/progenv
  < get commandline-arguments 22a >
  < set default arguments for Spotlight 30a >
  ◇

```

File defined by 29f, 30b.

Fill in default values when they cannot be found in exported variables nor in command-line arguments.

```

< set default arguments for Spotlight 30a > ≡
    if
        [ "$spotlighthost" == "" ]
    then
        spotlighthost=130.37.53.11
    fi
    if
        [ "$spotlightport" == "" ]
    then
        if
            [ "$naflang" == "nl" ]
        then
            spotlightport=2060
        else
            spotlightport=2020
        fi
    fi
    fi
    ◇

```

Fragment referenced in 29f.

Uses: `naflang` 24a.

```

"../bin/check_start_spotlight" 30b≡
    < check listener on host, port (30c $spotlighthost,30d $spotlightport ) 31c >
    if
        [ $spotlightrunning -ne 0 ]
    then
        if
            [ ! "$spotlighthost" == "localhost" ]
        then
            export spotlighthost="localhost"
            < check listener on host, port (30e $spotlighthost,30f $spotlightport ) 31c >
        fi
    fi
    if
        [ $spotlightrunning -ne 0 ]
    then
        < start the Spotlight server on localhost 33a, ... >
    fi
    echo $spotlighthost:$spotlightport
    ◇

```

File defined by 29f, 30b.

```

< make scripts executable 30g > ≡
    chmod 775 ../bin/check_start_spotlight
    ◇

```

Fragment defined by 21d, 30g, 76b.

Fragment referenced in 76c.

Use function `check_start_spotlight` to find and exploit a running Spotlight-server or to die (with exit code 5) if no server can be found or created. The macro uses implicitly the exported variables `spotlighthost` and `spotlightport` if they exist.

```

⟨find a spotlightserver or exit 31a⟩ ≡
    spothostport='/mnt/sdb1/nlpp/bin/check_start_spotlight -l $naflang'
    export spotlighthost='echo $spothisport | gawk -F ":" '{print $1}''
    export spotlightport='echo $spothisport | gawk -F ":" '{print $2}''
    echo "Spotlight server found on $spothisport." >&2
    if
        [ "$spotlighthost" == "none" ]
    then
        echo "No Spotlight-server found."
        exit 5
    fi
    ◇

```

Fragment referenced in 38g, 51b, 60a.

Uses: `naflang` 24a, `print` 70a.

Set the port-number and the language resource for Spotlight, dependent of the language that the user gave as argument.

```

⟨get spotlight language parameters 31b⟩ ≡
    if
        [ "$naflang" == "nl" ]
    then
        spotlightport=2060
    else
        spotlightport=2020
    fi
    ◇

```

Fragment never referenced.

Uses: `naflang` 24a.

The following macro has a hostname and a port-number as arguments. It checks whether something in the host listens on the port and sets variable `success` accordingly:

```

⟨check listener on host, port 31c⟩ ≡
    exec 6<>/dev/tcp/01/02 2>/dev/null
    spotlightrunning=$?
    exec 6<&-
    exec 6>&-
    ◇

```

Fragment referenced in 30b, 33c.

If variable `spotlighthost` does not exist, set it to localhost. Test whether a Spotlightserver runs on `spotlighthost`. If that fails and `spotlighthost` did not point to localhost, try localhost.

If the previous attempts were not succesfull, start the spotlightserver on localhost.

If some spotlightserver has been contacted, set variable `spotlightrunning`. Otherwise exit. At the end variable `spotlighthost` ought to contain the address of the Spotlight-host.

```

< try to obtain a running spotlightserver 32a > ≡
  < test whether spotlighthost runs (32b $spotlighthost ) 32e >
  if
    [ ! $spotlightrunning ]
  then
    if
      [ "$spotlighthost" != "localhost" ]
    then
      export spotlighthost=localhost
      < test whether spotlighthost runs (32c $spotlighthost ) 32e >
    fi
  fi
  if
    [ ! $spotlightrunning ]
  then
    < start the Spotlight server on localhost 33a, ... >
    < test whether spotlighthost runs (32d $spotlighthost ) 32e >
  fi
  if
    [ ! $spotlightrunning ]
  then
    echo "Cannot start spotlight"
    exit 4
  fi
  ◇

```

Fragment never referenced.

Test whether the Spotlightserver runs on a given host. The “spotlight-test” does not really test Spotlight, but it tests whether something is listening on the port and host where we expect Spotlight. I found the test-construction that is used here on [Stackoverflow](#). If the test is positive, set variable `spotlightrunning` to 0. Otherwise, unset that variable.

```

< test whether spotlighthost runs 32e > ≡
  exec 6<>/dev/tcp/@1/2060
  if
    [ $? -eq 0 ]
  then
    export spotlightrunning=0
  else
    spotlightrunning=
  fi
  exec 6<&-
  exec 6>&-
  ◇

```

Fragment referenced in [32a](#).

When trying to start the Spotlight-server on localhost, take care that only one process does this. So we do this:

1. Try to acquire a lock without waiting for it.
2. If we got the lock, run the Spotlight java program in background.
3. If we got the lock, release it.
4. If we did not get the lock, wait for the lock to be released by the process that started the spotlight-server.

But first, we specify the resources for the Spotlight-server.


```

< start the Spotlight server on localhost 33a > ≡
  if
    [ "$naflang" == "nl" ]
  then
    spotresource="nl"
  else
    spotresource="en_2+2"
  fi
  spotlightjar=dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar
  ◇

```

Fragment defined by 33ab.

Fragment referenced in 30b, 32a.

Uses: naflang 24a.

```

< start the Spotlight server on localhost 33b > ≡
  local oldd='pwd'
  cd /mnt/sdb1/nlpp/env/spotlight
  $envbindir/sematree acquire spotlock 0
  gotit=$?
  if
    [ $gotit == 0 ]
  then
    java -jar -Xmx8g $spotlightjar $spotresource \
      http://localhost:$spotlightport/rest &
    < wait until the spotlight server is up or faulty 33c >
    $envbindir/sematree release spotlock
  else
    < wait until the spotlight server is up or faulty 33c >
  fi
  cd $oldd
  ◇

```

Fragment defined by 33ab.

Fragment referenced in 30b, 32a.

When the Sportlight server has been started, it takes up to a minute until it really listens on its port. When there is something wrong, it will never listen, of course. Therefore, we give it three minutes. If after that time still nothing listens, we set `spotlighthost` to `none`, indicating that something has gone wrong.

```

< wait until the spotlight server is up or faulty 33c > ≡
  trial=0
  maxtrials=12
  while
    trial=$((trial+1))
    < check listener on host, port (33d $spotlighthost,33e $spotlightport ) 31c >
    [ $spotlightrunning -ne 0 ] && [ $trial -le $maxtrials ]
  do
    sleep 10
  done
  if
    [ spotlightrunning -ne 0 ]
  then
    export spotlighthost="none"
  fi
  ◇

```

Fragment referenced in 33b.

Start the Spotlight if it is not already running. First find out what the host is on which we may expect to find a listening Spotlight.

Variable `spotlighthost` contains the address of the host where we expect to find Spotlight. If the expectation does not come true, and the Spotlighthost was not localhost, test whether Spotlight can be found on localhost. If the spotlight-server cannot be found, start it up on localhost.

4.5.9 VUA-pylib

Module VUA-pylib is needed for the opinion-miner. Install it in the Python library

```
< install VUA-pylib 34a > ≡
    cd $envdir/python
    git clone https://github.com/cltl/VUA_pylib.git
    ◇
```

Fragment referenced in 17a.

4.5.10 SVMlight

SVMlight supplies a Support Vector Machine. It is used by the opinion-miner. SVMlight can be obtained from [the site](#) where it is documented.

Installation goes like this:

```
< install SVMlight 34b > ≡
    tempdir='mktemp -d -t SVMlight.XXXXXX'
    cd $tempdir
    wget http://download.joachims.org/svm_light/current/svm_light.tar.gz
    tar -xzf svm_light.tar.gz
    make all
    cp svm_classify /mnt/sdb1/nlpp/env/bin/
    cp svm_learn /mnt/sdb1/nlpp/env/bin/
    cd /mnt/sdb1/nlpp
    rm -rf $tempdir
    ◇
```

Fragment referenced in 17a.

Uses: all 65b.

4.5.11 CRFSuite

[CRFSuite](#) is an implementation of Conditional Random Fields (CRF). Module [opinion-miner-de-luxe](#) needs it. It can be installed from it's sources, but I did not manage to this. Therefore, currently we use a pre-compiled ball.

```

<install CRFSuite 35a> ≡
    tempdir='mktemp -d -t crfsuite.XXXXXX'
    cd $tempdir
    tar -xzf $snapshotsocket/t_nlpp_resources/crfsuite-0.12-x86_64.tar.gz
    cd crfsuite-0.12
    cp -r bin/crfsuite $envbindir/
    mkdir -p $envdir/include/
    cp -r include/* $envdir/include/
    mkdir -p $envdir/lib/
    cp -r lib/* $envdir/lib/
    cd /mnt/sdb1/nlpp
    rm -rf $tempdir
    ◇

```

Fragment referenced in 17a.

4.6 Install modules

4.6.1 Install tokenizer

Module The tokenizer is just a jar that has to be run in Java. Although the jar is directly available from <http://ixa2.si.ehu.es/ixa-pipes/download.html>, we prefer to compile the package in order to make this thing ready for reproducible set-ups.

To install the tokenizer, we proceed as follows:

1. Clone the source from github into a temporary directory.
2. Compile to produce the jar file with the tokenizer.
3. move the jar file into the jar directory.
4. remove the tempdir with the sourcecode.

```

<install the tokenizer 35b> ≡
    tempdir='mktemp -d -t tok.XXXXXX'
    cd $tempdir
    git clone https://github.com/ixa-ehu/ixa-pipe-tok.git
    cd ixa-pipe-tok
    git checkout 56f83ce4b61680346f15e5d4e6de6293764f7383
    mvn clean package
    mv target/ixa-pipe-tok-1.8.0.jar $jarsdir
    cd $piperoot
    rm -rf $tempdir
    ◇

```

Fragment referenced in 18a.

Script The script runs the tokenizerscript.

```

"../bin/tok" 35c≡
    <start of module-script (35d $jarsdir ) 23a>
    JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
    java -Xmx1000m -jar $JARFILE tok -l $naflang --inputkaf
    ◇

```

4.6.2 Topic analyser

The English pipeline contains a topic analyser that seems not yet fit for Dutch. Get it from the Newsreader repo and update the config file.

```
< install the topic analyser 36a > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-topic.v30.tgz
  cd $modulesdir/EHU-topic.v30
  mv conf.prop old.conf.prop
  gawk '{gsub("/home/newsreader/components", subs); print}' subs=$modulesdir old.conf.prop >conf.prop
  ◇
```

Fragment referenced in 18a.

Uses: print 70a.

Script:

```
"../bin/topic" 36b≡
  < start of module-script (36c EHU-topic.v30 ) 23a >
  java -Xmx1000m -jar $MODDIR/ixa-pipe-topic-1.0.1.jar -p $MODDIR/conf.prop
  ◇
```

4.6.3 Morphosyntactic parser

Module

```
< install the morphosyntactic parser 36d > ≡
  MODNAM=morphsynparser
  DIRN=morphosyntactic_parser_nl
  GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
  GITH=807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184
  < install from github 8a >
  cd $modulesdir/morphosyntactic_parser_nl
  git checkout 807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184
  ◇
```

Fragment referenced in 18a.

Script The morpho-syntactic module parses the sentences with Alpino. Alpino takes a lot of time to handle long sentences. Therefore the morpho-syntactic module has an option `-t` to set a time-out (in minutes) for sentence parsing.

```
"../bin/mor" 36e≡
  < start of module-script (36f morphosyntactic_parser_nl ) 23a >
  < get the mor time-out parameter 37a >
  < set alpinohome 25b >
  cat | python $MODDIR/core/morph_syn_parser.py $timeoutarg
  ◇
```

Use `getopts` to read the `-t` option.

```

⟨ get the mor time-out parameter 37a ⟩ ≡
    OPTIND=1
    stimeout=
    timeoutarg=
    while getopts "t:" opt; do
        case "$opt" in
            t) stimeout=$OPTARG
                ;;
            esac
        done
        shift $((OPTIND-1))
        if
            [ $stimeout ]
        then
            timeoutarg="-t $stimeout"
        fi
    fi
    ◇

```

Fragment referenced in 36e.

4.6.4 Pos tagger

In the Dutch pipeline the morpho-syntactic parser fulfills the role of Pos tagger. In the English pipeline we use the pos-tagger from EHU.

Module

```

⟨ install the pos tagger 37b ⟩ ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-pos.v30.tgz
    cd $modulesdir/EHU-topic.v30
    ◇

```

Fragment referenced in 18a.

Script

```

"../bin/pos" 37c ≡
    ⟨ start of module-script (37d EHU-pos.v30 ) 23a ⟩
    java -Xmx1000m -jar ${MODDIR}/ixa-pipe-pos-1.4.3.jar tag -m ${MODDIR}/en-maxent-
    100-c5-baseline-dict-penn.bin
    ◇

```

4.6.5 Constituent parser

Module

```

⟨ install the constituents parser 37e ⟩ ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-parse.v30.tgz
    cd $modulesdir/conspardir
    chmod 775 *.jar
    chmod 775 *.bin
    ◇

```

Fragment referenced in 18j.

Script

```
"../bin/constpars" 38a≡
  ⟨ start of module-script (38b EHU-parse.v30 ) 23a ⟩
  java -Xmx1000m -jar ${MODDIR}/ixa-pipe-parse-1.1.1.jar parse -g sem -
  m ${MODDIR}/en-parser-chunking.bin
  ◇
```

4.6.6 NED-reranker

Module

```
⟨ install the NED-reranker module 38c ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-popen-nedreranker.v30.tgz
  ◇
```

Fragment referenced in 18j.

Script

```
"../bin/nedrer" 38d≡
  ⟨ start of module-script (38e VUA-popen-nedreranker.v30 ) 23a ⟩
  cd $MODDIR
  python $MODDIR/domain_model.py
  ◇
```

4.6.7 Wikify module

Module

```
⟨ install the wikify module 38f ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-wikify.v30.tgz
  ◇
```

Fragment referenced in 18j.

Script The Wikify module needs DBpedia to generate “markables”.

```
"../bin/wikify" 38g≡
  ⟨ start of module-script (38h EHU-wikify.v30 ) 23a ⟩
  ⟨ find a spotlightserver or exit 31a ⟩
  cd $MODDIR
  java -Xmx1000m -jar ${MODDIR}/ixa-pipe-wikify-1.2.1.jar -s http://$spotlighthost -
  p $spotlightport
  ◇
```

4.6.8 UKB

UKB needs boost libraries and Perl version 5. For now, we consider them installed.

Module

< install the UKB module 39a > ≡

◇

Fragment never referenced.

Script Put the path to the boost libraries in the LD_LIBRARY_PATH variable and then run UKB.

```
"../bin/ukb" 39b≡
  < start of module-script (39c EHU-ukb.v30 ) 23a >
  cd $MODDIR
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$envdir/lib:$envdir/boost_1_54_0/stage/lib
  ${MODDIR}/naf_ukb/naf_ukb.pl -x ${MODDIR}/ukb/bin/ukb_wsd -K ${MODDIR}/wn30-
  ili_lkb/wn30g.bin64 -D ${MODDIR}/wn30-ili_lkb/wn30.lex - -- --dict_weight --
  dgraph_dfs --dgraph_rank ppr
```

◇

4.6.9 IMS-WSD

Module The package itself supplies an installation script that seems usable. However, today I am in a hurry and just install the module as it comes from the EHU repository.

Although the Hadoop implementation runs this module with Java 1.7, I could only run `ims+wsd` Java 1.6. Using Java 1.7 causes run-time errors “Platform not recognised” and the resulting NAF’s do not contain WordNet references. So, we had to install Java 1.6.

The scripts contain explicit paths that must be corrected:

`ims/testPlain`: Explicit path to Java binary.

`path_to_ims.py`: Set variable `PATH_TO_IMS`.

```
< install the ims-wsd module 39d > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-ims-wsd.v30.tgz
  cd VUA-ims-wsd.v30
  thisDir='pwd'
  echo PATH_TO_IMS = ""$thisDir/ims"" > path_to_ims.py
  cd ims
  cp testPlain.bash old.testPlain.bash
  sedcommand='s|/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0.x86_64/jre/bin/java|java|g'
  cat old.testPlain.bash | sed $sedcommand >testPlain.bash
```

◇

Fragment referenced in 19a.

Script

```
"../bin/ewsd" 39e≡
  < start of module-script (39f VUA-ims-wsd.v30 ) 23a >
  < set up Java 1.6 11d >
  #Setting the output to be ili-wn30 synsets instead of sensekeys
  $MODDIR/call_ims.py -ili30
```

◇

4.6.10 SRL server

The EHU SRL-module, that we use for English documents, has been set up as a server/client system. Hence, we have to start the server before we can process something.

We don't know in advance whether we run the pipeline for a single text or from a whole bunch of text and hence we do not know whether it is advisable that the server keeps running, occupying precious memory. Therefore, currently we just start and stop the server every time that we use it.

Module

```

< install the srl-server module 40a > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-srl-server.tgz
  cd EHU-srl-server
  mkdir -p /mnt/sdb1/nlpp/env/etc/pid
  ◇

```

Fragment referenced in 19a.

Scripts Generate three scripts: **start_eSRL**, **stop_esrl** and **eSRL**, resp. to start the SRL server, to stop it and to process a NAF file.

```

"../bin/start_eSRL" 40b ≡
  < start of module-script (40c EHU-srl-server ) 23a >
  < start EHU SRL server if it isn't running 41a >
  ◇

```

```

"../bin/stop_eSRL" 40d ≡
  < start of module-script (40e EHU-srl-server ) 23a >
  < stop EHU SRL server 41b >
  ◇

```

```

"../bin/eSRL" 40f ≡
  < start of module-script (40g EHU-srl-server ) 23a >
  /mnt/sdb1/nlpp/bin/start_eSRL
  java -Xmx1000m -cp $MODDIR/IXA-EHU-srl-3.0.jar ixa.srl.SRLClient en
  ◇

```



```

< start EHU SRL server if it isn't running 41a > ≡
pidFile=/mnt/sdb1/nlpp/env/etc/pid/SRLServer.pid
portInfo=$(nmap -p 5005 localhost | grep open)
if [ -z "$portInfo" ]; then
    >&2 echo "Starting srl-server as it is not running"
    java -Xms2500m -cp $MODDIR/IXA-EHU-srl-
3.0.jar ixa.srl.SRLServer en &> /dev/null &
    pid=$!
    echo $pid > $pidFile
    sleep 60
    >&2 echo "Server running: ${pid}"
else
    >&2 echo "Server already running.."
fi
◇

```

Fragment referenced in 40b.

```

< stop EHU SRL server 41b > ≡
pidFile=/mnt/sdb1/nlpp/env/etc/pid/SRLServer.pid
if
[ -e "$pidFile" ]
then
    kill 'echo $pidFile'
    rm $pidFile
fi
◇

```

Fragment referenced in 40d.

4.6.11 SRL Dutch nominals

Module

```

< install the srl-dutch-nominals module 41c > ≡
MODNAM=srl-dutch-nominals
DIRN=vua-srl-dutch-nominal-events
GITU=https://github.com/newsreader/vua-srl-dutch-nominal-events
GITC=6115b3168978acf809916cd2da512295d109d8fb
< install from github 8a >
cd $modulesdir/vua-srl-dutch-nominal-events
chmod 775 vua-srl-dutch-additional-roles.py
◇

```

Fragment referenced in 19a.

Script

```

"../bin/srl-dutch-nominals" 41d ≡
< start of module-script (41e vua-srl-dutch-nominal-events ) 23a >
cd $MODDIR
cat | $MODDIR/vua-srl-dutch-additional-roles.py
◇

```

4.6.12 FBK-time module

Module

```
< install the FBK-time module 42 > ≡  
    cd $modulesdir  
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-time.v30.tgz  
    ◇
```

Fragment referenced in [19j](#).

Script The script is rather complicated. I just copied it from the original makers, with one exception: Originally at the end of the script there was a pipe consisting of two Java programs. However, that didn't seem to work in one of the computers that we use, therefore we have split the pipe using `mytemp` as temporary storage.

```

"../bin/FBK-time" 43a≡
  < start of module-script (43b FBK-time.v30 ) 23a >
  BEGINTIME='date +%Y-%m-%dT%H:%M:%S%Z'
  YAMCHA=$MODDIR/tools
  timdir='mktemp -d -t time.XXXXXX'
  FILETXP=$timdir/TimePro.txp
  CHUNKIN=$timdir/TimePro.naf
  FILEOUT=$timdir/TimeProOUT.txp
  TIMEPRONORMIN=$timdir/TimeProNormIN.txp
  JAVAMAXHEAP=2g
  mytemp=$timdir/mytemp
  result=0
  cd $MODDIR
  cat > $CHUNKIN

  JAVACLASSPATH="lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar:lib/NAFtoTXP_v11.jar"
  JAVAMODULE=eu.fbk.newsreader.naf.NAFtoTXP_v11
  cat $CHUNKIN | \
    java -Xmx$JAVAMAXHEAP -cp $JAVACLASSPATH $JAVAMODULE $FILETXP chunk+entity timex
  < stop on error (43c Java: $JAVACLASSPATH:$JAVAMODULE ) 44a >
  #echo "Saving... $FILETXP"
  tail -n +4 $FILETXP | awk -f resources/english-rules > $FILEOUT
  head -n +4 $FILETXP > $TIMEPRONORMIN

  cat $FILEOUT | \
    $YAMCHA/yamcha-0.33/usr/local/bin/yamcha \
      -m models/tempeval3_silver-data.model \
    >> $TIMEPRONORMIN
  < stop on error (43d yamcha ) 44a >
  JAVACLASSPATH="lib/scala-library.jar:lib/timenorm-0.9.1-SNAPSHOT.jar"
  JAVACLASSPATH=$JAVACLASSPATH:"lib/threetenbp-0.8.1.jar:lib/TimeProNorm_v2.5.jar"
  JAVAMODULE=eu.fbk.timePro.TimeProNormApply
  cat $TIMEPRONORMIN | \
    java -Xmx$JAVAMAXHEAP -cp $JAVACLASSPATH $JAVAMODULE $FILETXP
  < stop on error (43e Java: $JAVACLASSPATH:$JAVAMODULE ) 44a >
  rm $FILEOUT
  rm $TIMEPRONORMIN

  JAVACLASSPATH="lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar:lib/NAFtoTXP_v11.jar"
  JAVAMODULE=eu.fbk.newsreader.naf.NAFtoTXP_v11
  cat $CHUNKIN | java -Xmx$JAVAMAXHEAP -
  cp $JAVACLASSPATH $JAVAMODULE $FILEOUT chunk+morpho+timex+event eval
  < stop on error (43f Java: $JAVACLASSPATH:$JAVAMODULE ) 44a >
  JAVACP1="lib/TXptoNAF_v5.jar:lib/jdom-2.0.5.jar:lib/kaflib-naf-1.1.9.jar"
  JAVAMOD1=eu.fbk.newsreader.naf.TXptoNAF_v4
  JAVACP2="lib/kaflib-naf-1.1.9.jar:lib/jdom-2.0.5.jar:lib/TimeProEmptyTimex_v2.jar"
  JAVAMOD2=eu.fbk.timepro.TimeProEmptyTimex
  java -Xmx$JAVAMAXHEAP -Dfile.encoding=UTF8 -
  cp $JAVACP1 $JAVAMOD1 $CHUNKIN $FILETXP "$BEGINTIME" TIMEX3 > $mytemp
  cat $mytemp | java -Xmx$JAVAMAXHEAP -Dfile.encoding=UTF8 -
  cp $JAVACP2 $JAVAMOD2 $FILEOUT
  < stop on error (43g Java: $JAVACLASSPATH:$JAVAMODULE ) 44a >
  rm $FILETXP
  rm $CHUNKIN
  rm -rf $timdir
  ◇

```

When one of the programs in the script fail, stop processing. Pass the error-code and write a message to locate the failing program. Remove the temporary directory. However, there is a problem. One of the java programs always results with result-code 1.

```

< stop on error 44a > ≡
    result=$?
    if
        [ $result -ne 0 ]
    then
        cd $MODDIR
        echo Error: @1 >&2
        rm -rf $timdir
        exit $result
    fi
    ◇

```

Fragment referenced in 43a.

4.6.13 FBK-temprel module

Module

```

< install the FBK-temprel module 44b > ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-temprel.v30.tgz
    < repair FBK-*rel's run.sh.hadoop (44c FBK-temprel.v30 ) 44d >
    ◇

```

Fragment referenced in 19j.

Script run.sh.hadoop seems to be obsolete in the original tarball:

1. The class-path argument in one of the Java statement refers to an obsolete jar (kaflib-naf-1.1.8 instead of kaflib-naf-1.1.9)
2. Another class-path argument refers to PredicateTimeAnchor_tlink.jar instead of PredicateTimeAnchor.jar
3. A “sh” statement is used. The problem is, that in Ubuntu /bin/sh points to bin/dash and the script (temprel-pipeline-per-file-NWR.sh) does not seem to be compatible with dash.

Therefore, we need to repair the script. We will need to repair the script in the FBK-causalrel module in a similar way, and therefore provide the module-directory as argument.

```

< repair FBK-*rel's run.sh.hadoop 44d > ≡
    cd $modulesdir/@1
    mv run.sh.hadoop old.run.sh.hadoop
    cat old.run.sh.hadoop | \
        sed s/kaflib-naf-1.1.8/kaflib-naf-1.1.9/g | \
        sed s/TimeAnchor_tlink.jar/TimeAnchor.jar/g | \
        sed "s/sh temprel/bash temprel/g" | \
        sed "s/java /java -Xmx2g /g" \
    >run.sh.hadoop
    chmod 775 run.sh.hadoop
    ◇

```

Fragment referenced in 44b, 45c.

Script The original run script seems to not only read the input naf from standard in, but also to obtain the input naf as a file that an argument points to. This constructions makes the pipeline complicated, therefore, we generate the naf file within the script.

The original script generates temporary files in the `temp` directory of the host-computer, and prefixes the names of the temporary files with a random number to prevent confusion between tempfiles of different instances of this module. We generate a temp-directory per instance.

```
"../bin/FBK-temprel" 45a≡
  ⟨ start of module-script (45b FBK-temprel.v30 ) 23a ⟩
  cd $MODDIR
  scratchDir='mkttemp -d -t temprel.XXXXXX'
  cat >$scratchDir/in.naf
  ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
  rm -rf $scratchDir

  ◇
```

4.6.14 FBK-causalrel module

Module

```
⟨ install the FBK-causalrel module 45c ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-causalrel.v30.tgz
  ⟨ repair FBK-*rel's run.sh.hadoop (45d FBK-causalrel.v30 ) 44d ⟩
  ◇
```

Fragment referenced in 19j.

Like in FBK-temprel, script run.sh.hadoop seems not to work out of the box:

1. The class-path argument in one of the Java statement refers to an obsolete jar (`kaflib-naf-1.1.8` instead of `kaflib-naf-1.1.9`)
2. A “sh” statement is used. The problem is, that in Ubuntu `/bin/sh` points to `bin/dash` and the script (`temprel-pipeline-per-file-NWR.sh`) does not seem to be compatible with `dash`.

Therefore, we need to repair that script like we did in FBK-temprel.

```
⟨ repair causalrel's run.sh.hadoop 45e ⟩ ≡
  cd $modulesdir/FBK-causalrel.v30
  mv run.sh.hadoop old.run.sh.hadoop
  cat old.run.sh.hadoop | \
    sed s/kaflib-naf-1.1.8/kaflib-naf-1.1.9/g | \
    sed s/TimeAnchor_tlink.jar/TimeAnchor.jar/g | \
    sed s/sh temprel/bash temprel/g | \
  >run.sh.hadoop
  chmod 775 run.sh.hadoop
  ◇
```

Fragment never referenced.

Script

```
"../bin/FBK-causalrel" 46a≡
  ⟨ start of module-script (46b FBK-causalrel.v30 ) 23a ⟩
  cd $MODDIR
  scratchDir='mktemp -d -t causalrel.XXXXXX'
  cat >$scratchDir/in.naf
  ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
  rm -rf $scratchDir
  ◇
```

4.6.15 Factuality module

Module

```
⟨ install the factuality module 46c ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-factuality.v30.tgz
  ◇
```

Fragment referenced in 19j.

Script

```
"../bin/factuality" 46d≡
  ⟨ start of module-script (46e VUA-factuality.v30 ) 23a ⟩
  cd $MODDIR
  #local settings to prevent perl from complaining
  export LANGUAGE=en_US.UTF-8
  export LANG=en_US.UTF-8
  export LC_ALL=en_US.UTF-8

  rootDir=${MODDIR}
  tmpDir=$(mktemp -d -t factuality.XXXXXX)

  export PATH=$PATH:${rootDir}:.
  # ex-
  port LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${rootDir}/../opt/lib/${rootDir}/../opt/boost_1_54_0/stage/lib
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mnt/sdb1/nlpp/env/lib/

  #mkdir -p ${scratchDir}/test

  python ${rootDir}/vua_factuality_naf_wrapper.py -t /mnt/sdb1/nlpp/env/bin/timbl -
  p ${rootDir} ${tmpDir}/
  ◇
```

4.6.16 Nominal coreference-base

The source of this module in Github (<https://github.com/opener-project/coreference-base.git>) does not seem to work well with NAF. Therefore, we use the version from the official English pipeline, that we find in the snapshot.

Module

```

< install coreference-base 47a > ≡
    cd $modulesdir
    tar -xzf /mnt/sdb1/t_nlp_resources/20151220_EHU-corefgraph.v30.tgz
    ◇

```

Fragment referenced in 20a.

Script

```

"../bin/coreference-base" 47b ≡
    < start of module-script (47c EHU-corefgraph.v30 ) 23a >
    cd $MODDIR/corefgraph
    cat | python -m corefgraph.process.file --reader NAF --writer NAF
    ◇

```

4.6.17 Named entity recognition (NERC)

Module The Nerc program can be installed from Github (<https://github.com/ixa-ehu/ixa-pipe-nerc>). However, the model that is needed is not publicly available. Therefore, models have been put in the snapshot-tarball.

```

< install the NERC module 47d > ≡
    < compile the nerc jar 47e >
    < get the nerc models 48b >
    ◇

```

Fragment referenced in 18j.

The nerc module is a Java program that is contained in a jar. Put the source from Github in a temporary directory, compile the jar with java and move the jar to the jars directory.

```

< compile the nerc jar 47e > ≡
    TEMPDIR='mktemp -d -t nerc.XXXXXX'
    cd $TEMPDIR
    git clone https://github.com/ixa-ehu/ixa-pipe-nerc
    cd ixa-pipe-nerc/
    git checkout ca02c931bc0b200ccdb8b5795a7552e4cc0d4802
    mvn clean package
    mv target/ixa-pipe-nerc-1.5.4.jar $jarsdir/
    cd $nuwebdir
    rm -rf $TEMPDIR
    ◇

```

Fragment referenced in 47d.

The current version of the pipeline uses the following models, that have been made available by Rodrigo Agerri on december 15, 2015.

The tarball `dutch-nerc-models.tar.gz` contains the models `nl-clusters-conll102.bin` and `nl-clusters-sonar.bin`. Both models have been placed in subdirectory `/m4_nerc_nl_dir/nerc_models/nl` of the snapshot.

The model for English can be found in the newsreader-repository.

Choose a model dependent of the language.

```

< select language-dependent features 48a > ≡
    if
        [ "$naflang" == "nl" ]
    then
        export nercmodel=nl/nl-clusters-conll02.bin
    else
        export nercmodel=en/en-newsreader-clusters-3-class-muc7-conll03-ontonotes-4.0.bin
    fi
◇

```

Fragment referenced in 60a.

Uses: **naflang** 24a.

The tarball `20160301_nerc_models.tgz` contains in subdirectories `nl` and `en` a dutch resp. an english nerc-model. They have been randomly selected from a number of models that are available in <http://ixa2.si.ehu.es/ixa-pipes/models/nerc-models-1.5.4.tgz>.

```

< get the nerc models 48b > ≡
    cd $modulesdir
    tar -p -xzf /mnt/sdb1/t_nlpp_resources/20160301_nerc_models.tgz
◇

```

Fragment referenced in 47d.

Script Make a script that uses the conll02 model and a script that uses the Sonar model

```

"../bin/nerc_conll02" 48c≡
    < start of module-script (48d m4_nerc_nl_dir ) 23a >
    JAR=$jarsdir/ixa-pipe-nerc-1.5.4.jar
    MODEL=nl-clusters-conll02.bin
    cat | java -Xmx1000m -jar $JAR tag -m $MODDIR/nl/$MODEL
◇

"../bin/nerc" 48e≡
    < start of module-script (48f nerc-models ) 23a >
    JAR=$jarsdir/ixa-pipe-nerc-1.5.4.jar
    if
        [ "$naflang" == "nl" ]
    then
        nercmodel=$modulesdir/nerc_models/nl/nl-6-class-clusters-sonar.bin
    else
        nercmodel=$modulesdir/nerc_models/en/en-best-clusters-conll03.bin
    fi
    java -Xmx1500m -jar $JAR tag -m $nercmodel
◇

```

4.6.18 Wordsense-disambiguation

Install WSD from its Github source (https://github.com/cltl/svm_wsd.git). According to the `readme` of that module, the next thing to do is, to execute install-script `install.sh` or `install_naf.sh`. The latter script installs a “Support-Vector-Machine” (SVM) module, “Dutch-SemCor” (DSC) models and `KafNafParserPy`.

Module

```

< install the WSD module 49a > ≡
MODNAM=wsd
DIRN=svm_wsd
GITU=https://github.com/cltl/svm_wsd.git
GITC=030043903b42f77cd20a9b2443de137e2efe8513
< install from github 8a >
cd $modulesdir/svm_wsd
< install svm lib 49b >
< download svm models 49c >

```

◇

Fragment referenced in 20a.

This part has been copied from `install_naf.sh` in the WSD module.

```

< install svm lib 49b > ≡
mkdir lib
cd lib
wget --no-check-
certificate https://github.com/cjlin1/libsvm/archive/master.zip 2>/dev/null
zip_name='ls -1 | head -1'
unzip $zip_name > /dev/null
rm $zip_name
folder_name='ls -1 | head -1'
mv $folder_name libsvm
cd libsvm/python
make > /dev/null 2> /dev/null
echo LIBSVM installed correctly lib/libsvm

```

◇

Fragment referenced in 49a.

This part has also been copied from `install_naf.sh` in the WSD module.

```

< download svm models 49c > ≡
cd $modulesdir/svm_wsd
#tar -xzf $pipesocket/m4_wsd_snapball
wget --user=cltl --
password='.cltl.' kyoto.let.vu.nl/~izquierdo/models_wsd_svm_dsc.tgz 2> /dev/null
echo 'Unzipping models...'
tar xzf models_wsd_svm_dsc.tgz
rm models_wsd_svm_dsc.tgz
echo 'Models installed in folder models'

```

◇

Fragment referenced in 49a.

Script

```

"../bin/wsd" 49d≡
< start of module-script (49e svm_wsd ) 23a >
WSDSCRIPT=dsc_wsd_tagger.py
cat | python $MODDIR/$WSDSCRIPT --naf -ref odwnSY

```

◇

4.6.19 Lexical-unit converter

Module There is not an official repository for this module yet, so copy the module from the tarball.

```
< install the lu2synset converter 50a > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/lu2synset.tgz
  ◇
```

Fragment referenced in 20h.

Script

```
"../bin/lu2synset" 50b ≡
  < start of module-script (50c lexicalunitconvertor ) 23a >
  JAVA_LIBDIR=$MODDIR/lib
  RESOURCESDIR=$MODDIR/resources
  JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
  java -Xmx812m -
  cp $JAVA_LIBDIR/$JARFILE vu.wntools.util.NafLexicalUnitToSynsetReferences \
    --wn-lmf "$RESOURCESDIR/cornetto2.1.lmf.xml" --format naf
  ◇
```

4.6.20 NED

The NED module is rather picky about the structure of the NAF file. In any case, it does not accept a file that has been produced by the ontotagger. Hence, in a pipeline NED should be executed before the ontotagger.

The NED module wants to consult the Dbpedia Spotlight server, so that one has to be installed somewhere. For this moment, let us suppose that it has been installed on localhost.

Module

```
< install the NED module 50d > ≡
  < put spotlight jar in the Maven repository 51a >
  MODNAM=ned
  DIRN=ixa-pipe-ned
  GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
  GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
  < install from github 8a >
  cd $modulesdir/ixa-pipe-ned
  mvn -Dmaven.compiler.target=1.7 -Dmaven.compiler.source=1.7 clean package
  mv target/ixa-pipe-ned-1.1.1.jar $jarsdir/
  ◇
```

Fragment referenced in 18j.

NED needs to have dbpedia-spotlight-0.7.jar in the local Maven repository. That is a different jar than the jar that we use to start Spotlight.

```

< put spotlight jar in the Maven repository 51a > ≡
    echo Put Spotlight jar in the Maven repository.
    tempdir='mktemp -d -t simplespot.XXXXXX'
    cd $tempdir
    wget http://spotlight.sztaki.hu/downloads/archive/2014/dbpedia-spotlight-0.7.jar
    wget http://spotlight.sztaki.hu/downloads/archive/2014/nl.tar.gz
    tar -xzf nl.tar.gz
    MVN_SPOTLIGHT_OPTIONS="-Dfile=dbpedia-spotlight-0.7.jar"
    MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DgroupId=ixa"
    MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DartifactId=dbpedia-spotlight"
    MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -Dversion=0.7"
    MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -Dpackaging=jar"
    MVN_SPOTLIGHT_OPTIONS="$MVN_SPOTLIGHT_OPTIONS -DgeneratePom=true"
    mvn install:install-file $MVN_SPOTLIGHT_OPTIONS

    cd $PROJROOT
    rm -rf $tempdir
    ◇

```

Fragment referenced in 50d.

Script NED needs to contact a Spotlight-server.

```

"../bin/ned" 51b ≡
    < start of module-script (51c ) 23a >
    ROOT=$piperoot
    JARDIR=$jarsdir
    < find a spotlightserver or exit 31a >
    cat | java -Xmx1000m -jar $jarsdir/ixa-pipe-ned-1.1.1.jar -
    H http://$spotlighthost -p $spotlightport -e candidates -
    i $envdir/spotlight/wikipedia-db -n nlEn
    ◇

```

4.6.21 Ontotagger

Module To install the onto-module, do the following:

1. Create a jar with the Java-code of the program.
2. Create a module-directory that contains the resources for the onto-tagger.
3. Create a script that executes the ontotagger on a naf.

Create the ontotagger jar if that hasn't been done already:

```

< install the onto-jar 51d > ≡
    < begin conditional install (51e ontojar_installed ) 15c >
    obdir='mktemp -d -t ontobuild.XXXXXX'
    cd $obdir
    git clone https://github.com/cltl/vua-ontotagger-v1.0.git
    cd vua-ontotagger-v1.0
    git checkout a251dbc2c30bfc77f258fdd5e9c58eb89c198ae7
    mvn install
    mv target/ontotagger-1.0-jar-with-dependencies.jar $jarsdir/
    cd $piperoot
    rm -rf $obdir
    < end conditional install (51f ontojar_installed ) 16a >
    ◇

```

Fragment referenced in 52d.

```

< install the vua-resources 52a > ≡
  < begin conditional install (52b vua_resources_installed ) 15c >
  cd $modulesdir
  git clone https://github.com/cltl/vua-resources.git
  cd $piperoot
  < end conditional install (52c vua_resources_installed ) 16a >
  ◇

```

Fragment referenced in 52d.

Install the onto-module from its Github source.

```

< install the onto module 52d > ≡
  < install the onto-jar 51d >
  < install the vua-resources 52a >
  ◇

```

Fragment referenced in 20a.

Script

```

"../bin/onto" 52e≡
  < start of module-script (52f vua-ontotagger-v1.0 ) 23a >
  cd $MODDIR/scripts
  cat | $MODDIR/scripts/predicate-matrixtagger.sh
  ◇

```

4.6.22 Framenet SRL

The framenet SRL is part of the package that contains the ontotagger. We only need a different script.

Script The script contains a hack, because the framesrl script produces spurious lines containing “frameMap.size()=...”. A GAWK script removes these lines.

```

"../bin/framesrl" 52g≡
  < start of module-script (52h vua-ontotagger-v1.0 ) 23a >
  cd $MODDIR/scripts
  cat | $MODDIR/scripts/srl-framenet-
  tagger | gawk '/^frameMap.size()/ {next}; {print}'
  ◇

```

4.6.23 Heideltime

Module The code for Heideltime can be found in [Github](#). However, we use a compiled Heideltime Jar, compiled by Antske Fokkens, because some bugs have been repaired in that version.

Use Heideltime via a wrapper, `ixa-pipe-time`, obtained from [Github](#).

Heideltime uses `treetagger`. It expects to find the location of `treetagger` in a variable `TreetaggerHome` in config-file `config.props`.

```

< install the heideltime module 53a > ≡
    moduledir=/mnt/sdb1/nlpp/modules/ixa-pipe-time
    < clone the heideltime wrapper 53b >
    < put Antske's material in the heideltime wrapper 53d >
    < compile the heideltime wrapper 53e >
    ◇

```

Fragment referenced in 20h.

```

< clone the heideltime wrapper 53b > ≡
    MODNAM=heideltime
    DIRN=ixa-pipe-time
    GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
    GITC=da4604a7b33975e977017440cbc10f7d59917ddf
    < install from github (53c ixa-pipe-time ) 8a >
    mkdir $moduledir/lib
    ◇

```

Fragment referenced in 53a.

In the wrapper we need the following extra material:

- A debugged version of the Heidelberg jar.
- A configuration file `config.props`, although it does not seem to be actually used.
- Another configuration file: `alpino-to-treetagger.csv`

The extra material has been provided by Antske Fokkens.

```

< put Antske's material in the heideltime wrapper 53d > ≡
    cd $modulesdir/$DIRN
    tar -xzf $snapshotsocket/t_nlpp_resources/20151123_antske_heideltime_stuff.tgz
    mv antske_heideltime_stuff/de.unihd.dbs.heideltime.standalone.jar lib/
    mv antske_heideltime_stuff/config.props .
    mv antske_heideltime_stuff/alpino-to-treetagger.csv .
    rm -rf antske_heideltime_stuff
    ◇

```

Fragment referenced in 53a.

Compile the Heideltime wrapper according to the [instruction](#) on Github.

```

< compile the heideltime wrapper 53e > ≡
    < get jvntextpro-2.0.jar 53f >
    < activate the install-to-project-repo utility 54a >
    cd /mnt/sdb1/nlpp/modules/$DIRN
    mvn clean install
    ◇

```

Fragment referenced in 53a.

```

< get jvntextpro-2.0.jar 53f > ≡
    cd /mnt/sdb1/nlpp/modules/$DIRN/lib
    wget http://ixa2.si.ehu.es/%7Ejibalar/jvntextpro-2.0.jar
    ◇

```

Fragment referenced in 53e.

Script `install-to-project-repo.py` generates a library in subdirectory `repo` and copies the jars that it finds in the `lib` subdirectory in this repo in such a way that Maven finds it there. Somewhere in the `install-to-project.py ... mvn` process the jars are copied in your local repository (`~/.m2`) too. As a result, only a Maven Guru understands precisely where Maven obtains its jar from and the best thing to do is to empty the `repo` subdirectory and the local repository before (re-) applying `install-to-project-repo.py`.

```
< activate the install-to-project-repo utility 54a > ≡
  < remove outdated heideltime jars 54b >
  cd /mnt/sdb1/nlpp/modules/$DIRN/
  git clone git@github.com:carchrae/install-to-project-repo.git
  mv install-to-project-repo/install-to-project-repo.py .
  rm -rf install-to-project-repo
  python ./install-to-project-repo.py
  ◇
```

Fragment referenced in 53e.

```
< remove outdated heideltime jars 54b > ≡
  rm -rf /mnt/sdb1/nlpp/modules/$DIRN/repo
  mkdir -p /mnt/sdb1/nlpp/modules/$DIRN/repo/local
  rm -rf $HOME/.m2/repository/local/de.unihd.dbs.heideltime.standalone
  rm -rf $HOME/.m2/repository/local/jvntextpro-2.0
  ◇
```

Fragment referenced in 54a.

Script

```
"../bin/heideltime" 54c≡
  < start of module-script (54d ixa-pipe-time ) 23a >
  MODDIR=$modulesdir/ixa-pipe-time
  cd $MODDIR
  iconv -t utf-8//IGNORE | java -Xmx1000m -jar target/ixa.pipe.time.jar -m alpino-to-
  treetagger.csv -c config.props
  ◇
```

4.6.24 Semantic Role labelling

Module

```
< install the srl module 54e > ≡
  MODNAM=srl
  DIRN=vua-srl-nl
  GITU=https://github.com/newsreader/vua-srl-nl.git
  GITC=675d22d361289ede23df11dcdb17195f008c54bf
  < install from github 8a >
  ◇
```

Fragment referenced in 20h.

Script First:

1. set the correct environment. The module needs python and timble.

- create a tempdir and in that dir a file to store the input and a (scv) file with the feature-vector.

```
"../bin/srl" 55a≡
  < start of module-script (55b vua-srl-nl ) 23a >
  MODDIR=$modulesdir/vua-srl-nl
  TEMPDIR='mktemp -d -t SRLTMP.XXXXXX'
  cd $MODDIR
  INPUTFILE=$TEMPDIR/inputfile
  FEATUREVECTOR=$TEMPDIR/csvfile
  TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
  ◇
```

File defined by 55acdef.

Create a feature-vector.

```
"../bin/srl" 55c≡
  cat | tee $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
  ◇
```

File defined by 55acdef.

Run the trained model on the feature-vector.

```
"../bin/srl" 55d≡
  timbl -m0:I1,2,3,4 -i 25Feb2015_e-mags_mags_press_newspapers.wgt -
  t $FEATUREVECTOR -o $TIMBLOUTPUTFILE >/dev/null 2>/dev/null
  ◇
```

File defined by 55acdef.

Insert the SRL values into the NAF file.

```
"../bin/srl" 55e≡
  python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
  ◇
```

File defined by 55acdef.

Clean up.

```
"../bin/srl" 55f≡
  rm -rf $TEMPDIR
  ◇
```

File defined by 55acdef.

4.6.25 SRL postprocessing

In addition to the Semantic Role Labeling there is hack that finds additional semantic roles.

Module Get the module from Github. Note that this module needs rdflib

```
< install python packages 55g > ≡
  pip install rdflib
  ◇
```

Fragment defined by 15a, 55g.

Fragment referenced in 11e.

Defines: `rdflib` Never used.

```

< install the post-SRL module 56a > ≡
  cd $modulesdir
  if
    [ -d vua-srl-postprocess ]
  then
    cd vua-srl-postprocess
    git pull
  else
    git clone https://github.com/newsreader/vua-srl-postprocess.git
    cd vua-srl-postprocess
  fi
  ◇

```

Fragment referenced in 20q.

Script

```

"../bin/postsr1" 56b≡
  < start of module-script (56c vua-srl-postprocess ) 23a >
  cd $MODDIR
  tmpdir='mktemp -d -t postsrl.XXXXX'
  cat >$tmpdir/infile
  python $MODDIR/main.py -i $tmpdir/infile -o $tmpdir/outfile
  cat $tmpdir/outfile
  rm -rf $tmpdir
  ◇

```

4.6.26 Event coreference

The event-coreference module is language-independent. Although the version in the EHU-repo is 3.0, the version 2.0 used in this pipeline seems to be more recent, so we will use that.

Module Install the module from the snapshot.

```

< install the event-coreference module 56d > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151217_vua-eventcoreference_v2.tgz
  cd vua-eventcoreference_v2
  cp lib/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar $jarsdir
  ◇

```

Fragment referenced in 20h.

Script


```

"../bin/evcoref" 57a≡
  < start of module-script (57b vua-eventcoreference_v2 ) 23a >
  RESOURCEDIR=$MODDIR/resources
  JARFILE=$jarsdir/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar

  if
    [ "$naflang" == 'nl' ]
  then
    lang_resource="odwn_orbn_gwg-LMF_1.3.xml"
  else
    lang_resource="wneng-30.lmf.xml.xpos"
  fi

  JAVAMODULE=eu.newsreader.eventcoreference.naf.EventCorefWordnetSim
  JAVAOPTIONS="--method leacock-chodorow"
  JAVAOPTIONS="$JAVAOPTIONS --wn-lmf $RESOURCEDIR/$lang_resource"
  JAVAOPTIONS="$JAVAOPTIONS --sim 2.0"
  JAVAOPTIONS="$JAVAOPTIONS --wsd 0.8"
  JAVAOPTIONS="$JAVAOPTIONS --
relations XPOS_NEAR_SYNONYM#HAS_HYPERONYM#HAS_XPOS_HYPERONYM#event"

  java -Xmx812m -cp $JARFILE $JAVAMODULE $JAVAOPTIONS

  ◇

```

4.6.27 Dbpedia-ner

Dbpedia-ner finds more named entities than NER, because it checks DBpedia for the candidate NE-'s.

Module

```

< install the dbpedia-ner module 57c > ≡
  MODNAM=dbpedia_ner
  DIRN=dbpedia_ner
  GITU=https://github.com/PaulHuygen/dbpedia_ner.git
  GITC=ab1dcdb860f0ff29bc979f646dc382122a101fc2
  < install from github 8a >
  ◇

```

Fragment referenced in 20q.

Script The main part of the module is a Python script. The README.md file of the Github repo lists the options that can be applied. One of the options is about the URL of the Spotlight server.

```

"../bin/dbpner" 57d≡
  < start of module-script (57e dbpedia_ner ) 23a >
  cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
  url http://$spotlighthost:2060/rest/candidates
  ◇

```

4.6.28 Nominal events

The module “postprocessing-nl” adds nominal events to the srl annotations. It has been obtained directly from the author (Piek Vossen). It is not yet available in a public repo. Probably in future versions the jar from the ontotagger module can be used for this module.

Module The nomevent-module uses the ontotagger jar en the resources from `vua-resources` that both have been installed already.

Script

```
"../bin/nomevent" 58a≡
  < start of module-script (58b vua-ontotagger-v1.0 ) 23a >
  cd $MODDIR/scripts
  cat | $MODDIR/scripts/m4_nomevent_subscript
  ◇
```

4.6.29 Opinion miner

Get `opinion-miner_deluxePP` from Github.

Module Install the module from Github.

```
< install the opinion-miner 58c > ≡
  MODNAM=opinion_miner_deluxePP
  DIRN=opinion_miner_deluxePP
  GITU=https://github.com/rubenIzquierdo/opinion_miner_deluxePP
  GITC=5f46af89f139080ae030abe70a540f693ac4676b
  < install from github 8a >
  ◇
```

Fragment defined by 58cd, 59a.

Fragment referenced in 21a.

The module contains a script `install_me.sh` that we will follow here. First install the CRF module that comes with the opinion-miner:

```
< install the opinion-miner 58d > ≡
  moduledir=$modulesdir/opinion_miner_deluxePP
  #Install CRF++
  crfdir='mktemp -d -t crf.XXXXXX'
  cd $crfdir
  tar xzf $moduledir/crf_lib/CRF++-0.58.tar.gz
  cd CRF++-0.58
  ./configure --prefix=$envdir
  make
  make install
  echo "PATH_TO_CRF_TEST='$envbindir/crf_test'" > $moduledir/path_crf.py
  cd $moduledir
  rm -rf $crfdir
  ◇
```

Fragment defined by 58cd, 59a.

Fragment referenced in 21a.

Next, download the trained models.

```

< install the opinion-miner 59a > ≡
    ##Download the models
    echo Downloading the trained models.
    tar -xzf $snapshotsocket/t_nlpp_resources/models_opinion_miner_deluxePP.tgz
    ◇

```

Fragment defined by 58cd, 59a.

Fragment referenced in 21a.

Script

```

"../bin/opinimin" 59b ≡
    < start of module-script (59c opinion_miner_deluxePP ) 23a >
    cd $MODDIR
    python tag_file.py -d hotel
    ◇

```

5 Utilities

5.1 Test script

The following script, `test`, pushes a test-document through the modules of the pipeline. if provided with an argument “nl” or “en”, it obtains a standard test-document in the appropriate language and puts it in `test/test.in.naf` (Let us call this file `TESTIN`). Otherwise, it expects to find file `test/test.in.naf`.

```

< get a testfile or die 59d > ≡
    if
        [ "$1" == "en" ]
    then
        cp $ROOT/nuweb/test.en.in.naf $TESTIN
    else
        if
            [ "$1" == "nl" ]
        then
            cp $ROOT/nuweb/test.nl.in.naf $TESTIN
        fi
    fi
    if
        [ ! -e $TESTIN ]
    then
        echo "Please supply test-file $TESTIN or specify language"
        exit 4
    fi
    ◇

```

Fragment referenced in 60a.

Uses: nuweb 72b.

```

"../bin/test" 60a≡
  #!/bin/bash
  ROOT=/mnt/sdb1/nlpp
  BIND=$ROOT/bin
  TESTDIR=$ROOT/test
  TESTIN=$ROOT/test/test.in.naf
  mkdir -p $TESTDIR
  <function to run a module in the test 61a, ... >
  <get a testfile or die 59d>
  cd $TESTDIR
  <set the language variable (60b $TESTIN ) 24a>
  <select language-dependent features 48a>
  <find a spotlightserver or exit 31a>
  if
    [ "$naflang" == "nl" ]
  then
    runmodule $TESTIN      tok                tok.naf
    runmodule tok.naf      mor                mor.naf
    runmodule mor.naf      nerc               nerc.naf
    runmodule nerc.naf     wsd                wsd.naf
    runmodule wsd.naf      ned                ned.naf
    runmodule ned.naf      heideltime         times.naf
    runmodule times.naf    onto               onto.naf
    runmodule onto.naf     srl                srl.naf
    runmodule srl.naf      nomevent           nomev.naf
    runmodule nomev.naf    srl-dutch-nominals psrl.naf
    runmodule psrl.naf     framesrl           fsrl.naf
    runmodule fsrl.naf     opinimin           opin.naf
    runmodule opin.naf     evcoref           out.naf
  else
    <annotate english document 62b>
  fi
  ◇

```

The following function, `runmodule`, applies a module on a naf file. When the module results in error, the function exits the script with the error code of the failing module.

```

⟨function to run a module in the test 61a⟩ ≡
function runmodule {
    local infile=$1
    local modulecommand=$BIND/$2
    local outfile=$3
    if
        [ $moduleresult -eq 0 ]
    then
        cat $infile | $modulecommand > $outfile
        moduleresult=$?
        if
            [ $moduleresult -gt 0 ]
        then
            failmodule=$modulecommand
            echo "Failed: module $modulecommand; result $moduleresult" >&2
            exit $moduleresult
        else
            echo "Completed: module $modulecommand; result $moduleresult" >&2
        fi
    fi
}
◇

```

Fragment defined by 61ab.
 Fragment referenced in 60a.
 Uses: moduleresult 61b.

Initialize the variable moduleresult:

```

⟨function to run a module in the test 61b⟩ ≡
    export moduleresult=0
◇

```

Fragment defined by 61ab.
 Fragment referenced in 60a.
 Defines: moduleresult 61a.

Correct sequence of the modules in the Dutch pipeline:

- tok
- mor
- nerc
- wsd
- ned
- heidel
- onto (predicate-matrix-tagger.sh uit vua-ontotagger-v1.0)
- srl
- Nominal event detectie
- vua-srl-extra
- framesrl (srl-framenet-tagger.sh uit vua-ontotagger-v1.0)
- opinion mining
- ecrf

```

⟨ annotate dutch document 62a ⟩ ≡
    runmodule $TESTIN      tok                tok.naf
    runmodule tok.naf      mor                mor.naf
    runmodule mor.naf      nerc               nerc.naf
    runmodule nerc.naf     wsd                wsd.naf
    runmodule wsd.naf      ned                ned.naf
    runmodule ned.naf      heideltime         times.naf
    runmodule times.naf    onto               onto.naf
    runmodule onto.naf     srl                srl.naf
    runmodule srl.naf      nomevent           nomev.naf
    runmodule nomev.naf    srl-dutch-nominals psrl.naf
    runmodule psrl.naf     framesrl           fsrl.naf
    runmodule fsrl.naf     opinimin           opin.naf
    runmodule opin.naf     evcoref            out.naf
    ◇

```

Fragment never referenced.

```

⟨ annotate english document 62b ⟩ ≡
    runmodule $TESTIN      tok                tok.naf
    runmodule tok.naf      topic              top.naf
    runmodule top.naf      pos                pos.naf
    runmodule pos.naf      constpars          consp.naf
    runmodule consp.naf    nerc               nerc.naf
    runmodule nerc.naf     ned                ned.naf
    runmodule nerc.naf     nedrer             nedr.naf
    runmodule nedr.naf     wikify             wikif.naf
    runmodule wikif.naf    ukb               ukb.naf
    runmodule ukb.naf      ewsd              ewsd.naf
    runmodule ewsd.naf     coreference-base   coref.naf
    runmodule coref.naf    eSRL              esrl.naf
    runmodule esrl.naf     FBK-time           time.naf
    runmodule time.naf     FBK-temprel        trel.naf
    runmodule trel.naf     FBK-causalrel       crel.naf
    runmodule crel.naf     evcoref            ecrf.naf
    runmodule ecrf.naf     factuality         fact.naf
    runmodule fact.naf     opinimin           out.naf
    ◇

```

Fragment referenced in 60a.

5.2 Logging

Write log messages to standard out if variable LOGLEVEL is equal to 1.

```

⟨ variables of install-modules 62c ⟩ ≡
    LOGLEVEL=1
    ◇

```

Fragment referenced in 17a.

```

<logmess 63a> ≡
  if
    [ $LOGLEVEL -gt 0 ]
  then
    echo @1
  fi
  ◇

```

Fragment referenced in 7c, 8a, 14b, 63b.

5.3 Misc

Install a module from a tarball: The macro expects the following three variables to be present:

URL: The URL tfrom where the taball can be downloaded.

TARB: The name of the tarball.

DIR; Name of the directory for the module.

Arg 1: URL; Arg 2: tarball; Arg 3: directory.

```

<install from tarball 63b> ≡
  SUCCES=0
  cd $modulesdir
  <move module (63c $DIR ) 7a>
  wget $URL
  SUCCES=$?
  if
    [ $SUCCES -eq 0 ]
  then
    tar -xzf $TARB
    SUCCES=$?
    rm -rf $TARB
  fi
  if
    [ $SUCCES -eq 0 ]
  then
    <logmess (63d Installed $DIR ) 63a>
    <remove old module (63e $DIR ) 7b>
  else
    <re-instate old module (63f $DIR ) 7c>
  fi
  ◇

```

Fragment never referenced.

A How to read and translate this document

This document is an example of *literate programming* [1]. It contains the code of all sorts of scripts and programs, combined with explaining texts. In this document the literate programming tool **nuweb** is used, that is currently available from Sourceforge (URL:nuweb.sourceforge.net). The advantages of Nuweb are, that it can be used for every programming language and scripting language, that it can contain multiple program sources and that it is very simple.

A.1 Read this document

The document contains *code scraps* that are collected into output files. An output file (e.g. `output.fil`) shows up in the text as follows:

```
"output.fil" 4a ≡
  # output.fil
  < a macro 4b >
  < another macro 4c >
  ◇
```

The above construction contains text for the file. It is labelled with a code (in this case 4a) The constructions between the < and > brackets are macro's, placeholders for texts that can be found in other places of the document. The test for a macro is found in constructions that look like:

```
< a macro 4b > ≡
  This is a scrap of code inside the macro.
  It is concatenated with other scraps inside the
  macro. The concatenated scraps replace
  the invocation of the macro.
```

Macro defined by 4b, 87e

Macro referenced in 4a

Macro's can be defined on different places. They can contain other macro's.

```
< a scrap 87e > ≡
  This is another scrap in the macro. It is
  concatenated to the text of scrap 4b.
  This scrap contains another macro:
  < another macro 45b >
```

Macro defined by 4b, 87e

Macro referenced in 4a

A.2 Process the document

The raw document is named `a_nlpp.w`. Figure 2 shows pathways to translate it into printable/viewable documents and to extract the program sources. Table 4 lists the tools that are

Tool	Source	Description
gawk	www.gnu.org/software/gawk/	text-processing scripting language
M4	www.gnu.org/software/m4/	Gnu macro processor
nuweb	nuweb.sourceforge.net	Literate programming tool
tex	www.ctan.org	Typesetting system
tex4ht	www.ctan.org	Convert T _E X documents into xml/html

Table 4: Tools to translate this document into readable code and to extract the program sources

needed for a translation. Most of the tools (except Nuweb) are available on a well-equipped Linux system.

```
< parameters in Makefile 64 > ≡
  NUWEB=../env/bin/nuweb
  ◇
```

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Uses: nuweb 72b.

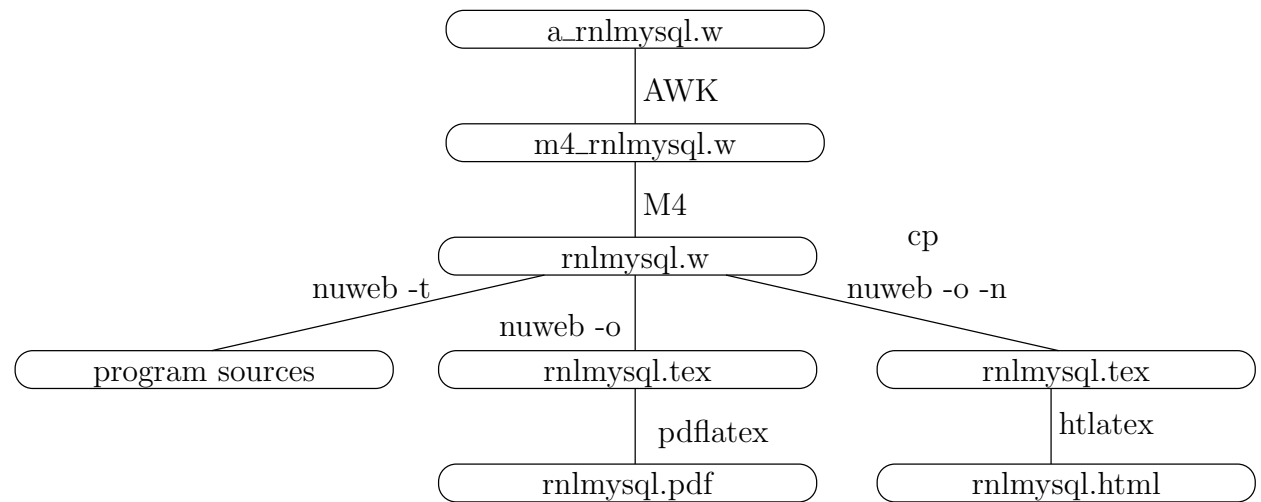


Figure 2: Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

```

"Makefile" 65a≡
    < default target 65b >

    < parameters in Makefile 64, ... >

    < impliciete make regels 69a, ... >
    < expliciete make regels 66c, ... >
    < make targets 65c, ... >
    ◇

```

The default target of make is `all`.

```

< default target 65b > ≡
    all : < all targets 66a >
    .PHONY : all

```

◇

Fragment referenced in 65a.
Defines: `all` 34b, `PHONY` 69b.

```

< make targets 65c > ≡
    clean:
        < clean up 10a, ... >

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.
Fragment referenced in 65a.

One of the targets is certainly the PDF version of this document.

\langle *all targets* 66a $\rangle \equiv$
`nlpp.pdf` \diamond

Fragment referenced in 65b.

Uses: pdf 70a.

We use many suffixes that were not known by the C-programmers who constructed the `make` utility. Add these suffixes to the list.

\langle *parameters in Makefile* 66b $\rangle \equiv$
`.SUFFIXES: .pdf .w .tex .html .aux .log .php`

\diamond

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Defines: SUFFIXES Never used.

Uses: pdf 70a.

A.4 Get Nuweb

An annoying problem is, that this program uses nuweb, a utility that is seldom installed on a computer. Therefore, we are going to install that first if it is not present. Unfortunately, nuweb is hosted on sourceforge and it is difficult to achieve automatic downloading from that repository. Therefore I copied one of the versions on a location from where it can be downloaded with a script.

Put the nuweb binary in the nuweb subdirectory, so that it can be used before the directory-structure has been generated.

\langle *expliciete make regels* 66c $\rangle \equiv$

```
nuweb: $(NUWEB)

$(NUWEB): ../nuweb-1.58
    mkdir -p ../env/bin
    cd ../nuweb-1.58 && make nuweb
    cp ../nuweb-1.58/nuweb $(NUWEB)
```

\diamond

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

Uses: nuweb 72b.

\langle *clean up* 66d $\rangle \equiv$
`rm -rf ../nuweb-1.58`

\diamond

Fragment defined by 10a, 11a, 25c, 66d.

Fragment referenced in 65c.

Uses: nuweb 72b.

```

⟨ expliciete make regels 67a ⟩ ≡
  ../nuweb-1.58:
    cd .. && wget http://kyoto.let.vu.nl/~huygen/nuweb-1.58.tgz
    cd .. && tar -xzf nuweb-1.58.tgz

```

◇

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

Uses: nuweb 72b.

A.5 Pre-processing

To make usable things from the raw input `a_nlpp.w`, do the following:

1. Process `$` characters.
2. Run the m4 pre-processor.
3. Run nuweb.

This results in a \LaTeX file, that can be converted into a PDF or a HTML document, and in the program sources and scripts.

A.5.1 Process ‘dollar’ characters

Many “intelligent” \TeX editors (e.g. the `auctex` utility of Emacs) handle `$` characters as special, to switch into mathematics mode. This is irritating in program texts, that often contain `$` characters as well. Therefore, we make a stub, that translates the two-character sequence `\$` into the single `$` character.

```

⟨ expliciete make regels 67b ⟩ ≡
  m4_nlpp.w : a_nlpp.w
    gawk '{if(match($$0, "@%")) {printf("%s", substr($$0,1,RSTART-
1))} else print}' a_nlpp.w \
    | gawk '{gsub(/[\$]/, "$$");print}' > m4_nlpp.w

```

◇

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

Uses: `print` 70a.

A.5.2 Run the M4 pre-processor

```

⟨ expliciete make regels 67c ⟩ ≡
  nlpp.w : m4_nlpp.w inst.m4
    m4 -P m4_nlpp.w > nlpp.w

```

◇

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

A.6 Typeset this document

Enable the following:

1. Create a PDF document.

2. Print the typeset document.
3. View the typeset document with a viewer.
4. Create a HTMLdocument.

In the three items, a typeset PDF document is required or it is the requirement itself.

A.6.1 Figures

This document contains figures that have been made by `xfig`. Post-process the figures to enable inclusion in this document.

The list of figures to be included:

```
<parameters in Makefile 68a> ≡
    FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Defines: FIGFILES 68b.

We use the package `figlatex` to include the pictures. This package expects two files with extensions `.pdftex` and `.pdftex_t` for `pdflatex` and two files with extensions `.pstex` and `.pstex_t` for the `latex/dvips` combination. Probably `tex4ht` uses the latter two formats too.

Make lists of the graphical files that have to be present for `latex/pdflatex`:

```
<parameters in Makefile 68b> ≡
    FIGFILENAMES=$(foreach fil,$(FIGFILES), $(fil).fig)
    PDFT_NAMES=$(foreach fil,$(FIGFILES), $(fil).pdftex_t)
    PDF_FIG_NAMES=$(foreach fil,$(FIGFILES), $(fil).pdftex)
    PST_NAMES=$(foreach fil,$(FIGFILES), $(fil).pstex_t)
    PS_FIG_NAMES=$(foreach fil,$(FIGFILES), $(fil).pstex)
```

◇

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Defines: FIGFILENAMES Never used, PDFT_NAMES 70b, PDF_FIG_NAMES 70b, PST_NAMES Never used,
PS_FIG_NAMES Never used.

Uses: FIGFILES 68a.

Create the graph files with program `fig2dev`:

```

< implicate make regels 69a > ≡
    %.eps: %.fig
        fig2dev -L eps $< > $@

    %.pstex: %.fig
        fig2dev -L pstex $< > $@

    .PRECIOUS : %.pstex
    %.pstex_t: %.fig %.pstex
        fig2dev -L pstex_t -p $*.pstex $< > $@

    %.pdftex: %.fig
        fig2dev -L pdftex $< > $@

    .PRECIOUS : %.pdftex
    %.pdftex_t: %.fig %.pstex
        fig2dev -L pdftex_t -p $*.pdftex $< > $@

```

◇

Fragment defined by 69a, 73c.

Fragment referenced in 65a.

Defines: fig2dev Never used.

A.6.2 Bibliography

To keep this document portable, create a portable bibliography file. It works as follows: This document refers in the |bibliography| statement to the local bib-file **nlpp.bib**. To create this file, copy the auxiliary file to another file **auxfil.aux**, but replace the argument of the command **\bibdata{nlpp}** to the names of the bibliography files that contain the actual references (they should exist on the computer on which you try this). This procedure should only be performed on the computer of the author. Therefore, it is dependent of a binary file on his computer.

```

< expliciete make regels 69b > ≡
    bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

    .PHONY : bibfile

```

◇

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

Uses: PHONY 65b.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

```

⟨ make targets 70a ⟩ ≡
    pdf : nlpp.pdf

    print : nlpp.pdf
           lpr nlpp.pdf

    view : nlpp.pdf
          evince nlpp.pdf

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

Defines: pdf 66ab, 70b, print 12a, 14a, 16b, 23c, 31a, 36a, 52g, 67b, view Never used.

Create the PDF document. This may involve multiple runs of nuweb, the L^AT_EX processor and the bibT_EX processor, and depends on the state of the aux file that the L^AT_EX processor creates as a by-product. Therefore, this is performed in a separate script, w2pdf.

The w2pdf script The three processors nuweb, L^AT_EX and bibT_EX are intertwined. L^AT_EX and bibT_EX create parameters or change the value of parameters, and write them in an auxiliary file. The other processors may need those values to produce the correct output. The L^AT_EX processor may even need the parameters in a second run. Therefore, consider the creation of the (PDF) document finished when none of the processors causes the auxiliary file to change. This is performed by a shell script w2pdf.

```

⟨ make targets 70b ⟩ ≡
    nlpp.pdf : nlpp.w $(W2PDF) $(PDF_FIG_NAMES) $(PDFT_NAMES)
              chmod 775 $(W2PDF)
              $(W2PDF) $*

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

Uses: pdf 70a, PDFT_NAMES 68b, PDF_FIG_NAMES 68b.

The following is an ugly fix of an unsolved problem. Currently I develop this thing, while it resides on a remote computer that is connected via the sshfs filesystem. On my home computer I cannot run executables on this system, but on my work-computer I can. Therefore, place the following script on a local directory.

```

⟨ directories to create 70c ⟩ ≡
    ../nuweb/bin ◇

```

Fragment defined by 5ab, 6ab, 9d, 10cd, 13b, 70c.

Fragment referenced in 76a.

Uses: nuweb 72b.

```

⟨ parameters in Makefile 70d ⟩ ≡
    W2PDF=../nuweb/bin/w2pdf

```

◇

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Uses: nuweb 72b.

```

< explicit make regels 71a > ≡
    $(W2PDF) : nlpp.w $(NUWEB)
              $(NUWEB) nlpp.w

```

◇

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.
 Fragment referenced in 65a.

```

"../nuweb/bin/w2pdf" 71b≡
    #!/bin/bash
    # w2pdf -- compile a nuweb file
    # usage: w2pdf [filename]
    # 20160330 at 1247h: Generated by nuweb from a_nlpp.w
    NUWEB=../env/bin/nuweb
    LATEXCOMPILER=pdflatex
    < filenames in nuweb compile script 71d >
    < compile nuweb 71c >

```

◇

Uses: nuweb 72b.

The script retains a copy of the latest version of the auxiliary file. Then it runs the four processors nuweb, L^AT_EX, MakeIndex and bibT_EX, until they do not change the auxiliary file or the index.

```

< compile nuweb 71c > ≡
    NUWEB=/mnt/sdb1/nlpp/env/bin/nuweb
    < run the processors until the aux file remains unchanged 72c >
    < remove the copy of the aux file 72a >

```

◇

Fragment referenced in 71b.
 Uses: nuweb 72b.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. .w) from the filename and create the names of the L^AT_EX file (ends with .tex), the auxiliary file (ends with .aux) and the copy of the auxiliary file (add old. as a prefix to the auxiliary filename).

```

< filenames in nuweb compile script 71d > ≡
    nufil=$1
    trunk=${1%.*}
    texfil=${trunk}.tex
    auxfil=${trunk}.aux
    oldaux=old.${trunk}.aux
    indexfil=${trunk}.idx
    oldindexfil=old.${trunk}.idx

```

◇

Fragment referenced in 71b.
 Defines: auxfil 72c, 74c, 75a, indexfil 72c, 74c, nufil 72b, 74c, 75b, oldaux 72ac, 74c, 75a, oldindexfil 72c, 74c, texfil 72b, 74c, 75b, trunk 72b, 74c, 75bc.

Remove the old copy if it is no longer needed.

```

⟨ remove the copy of the aux file 72a ⟩ ≡
    rm $oldaux
    ◇

```

Fragment referenced in 71c, 74b.
 Uses: oldaux 71d, 74c.

Run the three processors. Do not use the option `-o` (to suppress generation of program sources) for nuweb, because `w2pdf` must be kept up to date as well.

```

⟨ run the three processors 72b ⟩ ≡
    $NUWEB $nufil
    $LATEXCOMPILER $texfil
    makeindex $trunk
    bibtex $trunk
    ◇

```

Fragment referenced in 72c.
 Defines: bibtex 75bc, makeindex 75bc, nuweb 6c, 59d, 64, 66cd, 67a, 70cd, 71bc, 73a, 74a.
 Uses: nufil 71d, 74c, texfil 71d, 74c, trunk 71d, 74c.

Repeat to copy the auxiliary file and the index file and run the processors until the auxiliary file and the index file are equal to their copies. However, since I have not yet been able to test the `aux` file and the `idx` in the same test statement, currently only the `aux` file is tested.

It turns out, that sometimes a strange loop occurs in which the `aux` file will keep to change. Therefore, with a counter we prevent the loop to occur more than 10 times.

```

⟨ run the processors until the aux file remains unchanged 72c ⟩ ≡
    LOOPCOUNTER=0
    while
        ! cmp -s $auxfil $oldaux
    do
        if [ -e $auxfil ]
        then
            cp $auxfil $oldaux
        fi
        if [ -e $indexfil ]
        then
            cp $indexfil $oldindexfil
        fi
        ⟨ run the three processors 72b ⟩
        if [ $LOOPCOUNTER -ge 10 ]
        then
            cp $auxfil $oldaux
        fi;
    done
    ◇

```

Fragment referenced in 71c.
 Uses: auxfil 71d, 74c, indexfil 71d, oldaux 71d, 74c, oldindexfil 71d.

A.6.4 Create HTML files

HTML is easier to read on-line than a PDF document that was made for printing. We use `tex4ht` to generate HTML code. An advantage of this system is, that we can include figures in the same way as we do for `pdflatex`.

To create a HTML doc, we do the following:

1. Create a directory `../nuweb/html` for the HTML document.
2. Put the nuweb source in it, together with style-files that are needed (see variable `HTMLSOURCE`).
3. Put the script `w2html` in it and make it executable.
4. Execute the script `w2html`.

Make a list of the entities that we mentioned above:

```
<parameters in Makefile 73a> ≡
    htmldir=../nuweb/html
    htmlsource=nlpp.w nlpp.bib html.sty artikel3.4ht w2html
    htmlmaterial=$(foreach fil, $(htmlsource), $(htmldir)/$(fil))
    htmltarget=$(htmldir)/nlpp.html
◇
```

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Uses: nuweb 72b.

Make the directory:

```
<expliciete make regels 73b> ≡
    $(htmldir) :
        mkdir -p $(htmldir)
◇
```

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

The rule to copy files in it:

```
<impliciete make regels 73c> ≡
    $(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
◇
```

Fragment defined by 69a, 73c.

Fragment referenced in 65a.

Do the work:

```
<expliciete make regels 73d> ≡
    $(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
◇
```

Fragment defined by 66c, 67abc, 69b, 71a, 73bd.

Fragment referenced in 65a.

Invoke:

```
<make targets 73e> ≡
    htm : $(htmldir) $(htmltarget)
◇
```

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

Create a script that performs the translation.

```
"w2html" 74a≡
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
# [filename]: Name of the nuweb source file.
# 20160330 at 1247h: Generated by nuweb from a_nlpp.w
echo "translate " $1 >w2html.log
NUWEB=/mnt/sdb1/nlpp/env/bin/nuweb
⟨filenames in w2html 74c⟩

⟨perform the task of w2html 74b⟩
```

◇

Uses: **nuweb** 72b.

The script is very much like the **w2pdf** script, but at this moment I have still difficulties to compile the source smoothly into HTML and that is why I make a separate file and do not recycle parts from the other file. However, the file works similar.

```
⟨perform the task of w2html 74b⟩ ≡
  ⟨run the html processors until the aux file remains unchanged 75a⟩
  ⟨remove the copy of the aux file 72a⟩
◇
```

Fragment referenced in 74a.

The user provides the name of the nuweb file as argument. Strip the extension (e.g. **.w**) from the filename and create the names of the L^AT_EX file (ends with **.tex**), the auxiliary file (ends with **.aux**) and the copy of the auxiliary file (add **old.** as a prefix to the auxiliary filename).

```
⟨filenames in w2html 74c⟩ ≡
nufil=$1
trunk=${1%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```

Fragment referenced in 74a.

Defines: **auxfil** 71d, 72c, 75a, **nufil** 71d, 72b, 75b, **oldaux** 71d, 72ac, 75a, **texfil** 71d, 72b, 75b, **trunk** 71d, 72b, 75bc.

Uses: **indexfil** 71d, **oldindexfil** 71d.

```

⟨run the html processors until the aux file remains unchanged 75a⟩ ≡
    while
        ! cmp -s $auxfil $oldaux
    do
        if [ -e $auxfil ]
        then
            cp $auxfil $oldaux
        fi
        ⟨run the html processors 75b⟩
    done
    ⟨run tex4ht 75c⟩

```

◇

Fragment referenced in 74b.

Uses: auxfil 71d, 74c, oldaux 71d, 74c.

To work for HTML, nuweb *must* be run with the `-n` option, because there are no page numbers.

```

⟨run the html processors 75b⟩ ≡
    $NUWEB -o -n $nufil
    latex $texfil
    makeindex $trunk
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 75a.

Uses: bibtex 72b, makeindex 72b, nufil 71d, 74c, texfil 71d, 74c, trunk 71d, 74c.

When the compilation has been satisfied, run makeindex in a special way, run bibtex again (I don't know why this is necessary) and then run htlatex another time.

```

⟨run tex4ht 75c⟩ ≡
    tex '\def\filename{{\nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
    makeindex -o $trunk.ind $trunk.4dx
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 75a.

Uses: bibtex 72b, makeindex 72b, trunk 71d, 74c.

A.7 Create the program sources

Run nuweb, but suppress the creation of the L^AT_EX documentation. Nuweb creates only sources that do not yet exist or that have been modified. Therefore make does not have to check this. However, “make” has to create the directories for the sources if they do not yet exist. So, let's create the directories first.

```

⟨parameters in Makefile 75d⟩ ≡
    MKDIR = mkdir -p

```

◇

Fragment defined by 64, 66b, 68ab, 70d, 73a, 75d.

Fragment referenced in 65a.

Defines: MKDIR 76a.

```

< make targets 76a > ≡
    DIRS = < directories to create 5a, ... >

```

```

    $(DIRS) :
        $(MKDIR) $@

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

Defines: DIRS 76c.

Uses: MKDIR 75d.

```

< make scripts executable 76b > ≡
    chmod -R 775 ../bin/*
    chmod -R 775 ../env/bin/*

```

◇

Fragment defined by 21d, 30g, 76b.

Fragment referenced in 76c.

```

< make targets 76c > ≡
    sources : nlpp.w $(DIRS) $(NUWEB)
              $(NUWEB) nlpp.w
              < make scripts executable 21d, ... >

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

Uses: DIRS 76a.

A.8 Restore paths after transplantation

When an existing installation has been transplanted to another location, many path indications have to be adapted to the new situation. The scripts that are generated by nuweb can be repaired by re-running nuweb. After that, configuration files of some modules must be modified.

```

< make targets 76d > ≡
    transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant

```

◇

Fragment defined by 65c, 70ab, 73e, 76acd.

Fragment referenced in 65a.

In order to work as expected, the following script must be re-made after a transplantation.

```

"../env/bin/transplant" 77≡
    #!/bin/bash
    LOGLEVEL=1
    < set variables that point to the directory-structure 6c, ... >
    < set paths after transplantation 14a >
    < re-install modules after the transplantation 28a >

    ◇

```

B References

B.1 Literature

References

- [1] Donald E. Knuth. Literate programming. Technical report STAN-CS-83-981, Stanford University, Department of Computer Science, 1983.

C Indexes

C.1 Filenames

```

"../bin/check_start_spotlight" Defined by 29f, 30b.
"../bin/constpars" Defined by 38a.
"../bin/coreference-base" Defined by 47b.
"../bin/dbpner" Defined by 57d.
"../bin/eSRL" Defined by 40f.
"../bin/evcoref" Defined by 57a.
"../bin/ewsd" Defined by 39e.
"../bin/factuality" Defined by 46d.
"../bin/FBK-causalrel" Defined by 46a.
"../bin/FBK-temprel" Defined by 45a.
"../bin/FBK-time" Defined by 43a.
"../bin/framesrl" Defined by 52g.
"../bin/heideltime" Defined by 54c.
"../bin/install-modules" Defined by 17a, 18aj, 19aj, 20ahq, 21a.
"../bin/lu2synset" Defined by 50b.
"../bin/mor" Defined by 36e.
"../bin/ned" Defined by 51b.
"../bin/nedrer" Defined by 38d.
"../bin/nerc" Defined by 48e.
"../bin/nerc_conll02" Defined by 48c.
"../bin/nomevent" Defined by 58a.
"../bin/onto" Defined by 52e.
"../bin/opinimin" Defined by 59b.
"../bin/pos" Defined by 37c.
"../bin/postersrl" Defined by 56b.
"../bin/srl" Defined by 55acdef.
"../bin/srl-dutch-nominals" Defined by 41d.
"../bin/start_eSRL" Defined by 40b.
"../bin/stop_eSRL" Defined by 40d.
"../bin/test" Defined by 60a.
"../bin/tok" Defined by 35c.
"../bin/topic" Defined by 36b.
"../bin/ukb" Defined by 39b.

```

"../bin/wikify" Defined by 38g.
 "../bin/wsd" Defined by 49d.
 "../env/bin/langdetect.py" Defined by 23c.
 "../env/bin/progenv" Defined by 6e, 9c.
 "../env/bin/transplant" Defined by 77.
 "../nuweb/bin/w2pdf" Defined by 71b.
 "Makefile" Defined by 65a.
 "w2html" Defined by 74a.

C.2 Macro's

<activate the install-to-project-repo utility 54a> Referenced in 53e.
 <activate the python environment 13ac> Referenced in 11e, 17a.
 <all targets 66a> Referenced in 65b.
 <annotate dutch document 62a> Not referenced.
 <annotate english document 62b> Referenced in 60a.
 <begin conditional install 15c> Referenced in 9e, 17a, 18aj, 19aj, 20ahq, 21a, 51d, 52a.
 <check listener on host, port 31c> Referenced in 30b, 33c.
 <check this first 8g, 21e> Referenced in 17a.
 <check whether mercurial is present 21f> Referenced in 21e.
 <check/install the correct version of python 12a> Referenced in 11e.
 <clean up 10a, 11a, 25c, 66d> Referenced in 65c.
 <clone the heideltime wrapper 53b> Referenced in 53a.
 <compile nuweb 71c> Referenced in 71b.
 <compile the heideltime wrapper 53e> Referenced in 53a.
 <compile the nerc jar 47e> Referenced in 47d.
 <create a virtual environment for Python 12c> Referenced in 11e.
 <create javapython script 9b> Referenced in 17a.
 <default target 65b> Referenced in 65a.
 <directories to create 5ab, 6ab, 9d, 10cd, 13b, 70c> Referenced in 76a.
 <download svm models 49c> Referenced in 49a.
 <else conditional install 15d> Not referenced.
 <end conditional install 16a> Referenced in 9e, 17a, 18aj, 19aj, 20ahq, 21a, 51d, 52a.
 <expliciete make regels 66c, 67abc, 69b, 71a, 73bd> Referenced in 65a.
 <filenames in nuweb compile script 71d> Referenced in 71b.
 <filenames in w2html 74c> Referenced in 74a.
 <find a spotlightserver or exit 31a> Referenced in 38g, 51b, 60a.
 <function to run a module in the test 61ab> Referenced in 60a.
 <get a testfile or die 59d> Referenced in 60a.
 <get commandline-arguments 22a> Referenced in 29f.
 <get jvntextpro-2.0.jar 53f> Referenced in 53e.
 <get spotlight language parameters 31b> Not referenced.
 <get spotlight model ball 29d> Referenced in 29a.
 <get the mor time-out parameter 37a> Referenced in 36e.
 <get the nerc models 48b> Referenced in 47d.
 <get the path to the module-script 23b> Referenced in 23a.
 <get the snapshot 9a> Referenced in 17a.
 <impliciete make regels 69a, 73c> Referenced in 65a.
 <install ActivePython 12b> Referenced in 12a.
 <install Alpino 25a> Referenced in 17a.
 <install boost 28b> Referenced in 17a.
 <install coreference-base 47a> Referenced in 20a.
 <install CRFsuite 35a> Referenced in 17a.
 <install from github 8a> Referenced in 36d, 41c, 49a, 50d, 53b, 54e, 57c, 58c.
 <install from tarball 63b> Not referenced.
 <install Java 1.6 11c> Referenced in 17a.
 <install kafnafparserpy 14b> Referenced in 11e.
 <install maven 10e> Referenced in 17a.

<install python packages 15a, 55g> Referenced in 11e.
 <install sematree 22b> Referenced in 17a.
 <install svm lib 49b> Referenced in 49a.
 <install SVMLight 34b> Referenced in 17a.
 <install the constituents parser 37e> Referenced in 18j.
 <install the dbpedia-ner module 57c> Referenced in 20q.
 <install the event-coreference module 56d> Referenced in 20h.
 <install the factuality module 46c> Referenced in 19j.
 <install the FBK-causalrel module 45c> Referenced in 19j.
 <install the FBK-temprel module 44b> Referenced in 19j.
 <install the FBK-time module 42> Referenced in 19j.
 <install the heideltime module 53a> Referenced in 20h.
 <install the ims-wsd module 39d> Referenced in 19a.
 <install the lu2synset converter 50a> Referenced in 20h.
 <install the morphosyntactic parser 36d> Referenced in 18a.
 <install the NERC module 47d> Referenced in 18j.
 <install the nomevent module ?> Referenced in 20q.
 <install the onto module 52d> Referenced in 20a.
 <install the onto-jar 51d> Referenced in 52d.
 <install the opinion-miner 58cd, 59a> Referenced in 21a.
 <install the pos tagger 37b> Referenced in 18a.
 <install the post-SRL module 56a> Referenced in 20q.
 <install the Spotlight server 29ae> Referenced in 17a.
 <install the srl module 54e> Referenced in 20h.
 <install the srl-dutch-nominals module 41c> Referenced in 19a.
 <install the srl-server module 40a> Referenced in 19a.
 <install the ticcutils utility 27b> Referenced in 17a, 28a.
 <install the timbl utility 27c> Referenced in 17a, 28a.
 <install the tokenizer 35b> Referenced in 18a.
 <install the topic analyser 36a> Referenced in 18a.
 <install the treetagger utility 25d, 26abcde, 27a> Referenced in 17a.
 <install the UKB module 39a> Not referenced.
 <install the vua-resources 52a> Referenced in 52d.
 <install the wikify module 38f> Referenced in 18j.
 <install the WSD module 49a> Referenced in 20a.
 <install the NED-reranker module 38c> Referenced in 18j.
 <install the NED module 50d> Referenced in 18j.
 <install VUA-pylib 34a> Referenced in 17a.
 <logmess 63a> Referenced in 7c, 8a, 14b, 63b.
 <make scripts executable 21d, 30g, 76b> Referenced in 76c.
 <make targets 65c, 70ab, 73e, 76acd> Referenced in 65a.
 <move module 7a> Referenced in 8a, 14b, 63b.
 <parameters in Makefile 64, 66b, 68ab, 70d, 73a, 75d> Referenced in 65a.
 <perform the task of w2html 74b> Referenced in 74a.
 <put Antske's material in the heideltime wrapper 53d> Referenced in 53a.
 <put spotlight jar in the Maven repository 51a> Referenced in 50d.
 <re-install modules after the transplantation 28a> Referenced in 77.
 <re-instate old module 7c> Referenced in 8a, 14b, 63b.
 <read the list of installed modules 15b> Referenced in 17a.
 <remove installed-variable 16b> Referenced in 11a.
 <remove old module 7b> Referenced in 8a, 14b, 63b.
 <remove outdated heideltime jars 54b> Referenced in 54a.
 <remove the copy of the aux file 72a> Referenced in 71c, 74b.
 <repair causalrel's run.sh.hadoop 45e> Not referenced.
 <repair FBK-*rel's run.sh.hadoop 44d> Referenced in 44b, 45c.
 <run in subshell when naflang is not known 24b> Referenced in 23a.
 <run only if language is English or Dutch 24c> Referenced in 23a.
 <run tex4ht 75c> Referenced in 75a.

<run the html processors [75b](#)> Referenced in [75a](#).
 <run the html processors until the aux file remains unchanged [75a](#)> Referenced in [74b](#).
 <run the processors until the aux file remains unchanged [72c](#)> Referenced in [71c](#).
 <run the three processors [72b](#)> Referenced in [72c](#).
 <select language-dependent features [48a](#)> Referenced in [60a](#).
 <set alpinohome [25b](#)> Referenced in [36e](#).
 <set default arguments for Spotlight [30a](#)> Referenced in [29f](#).
 <set paths after transplantation [14a](#)> Referenced in [77](#).
 <set the language variable [24a](#)> Referenced in [60a](#).
 <set up java [9e](#), [10b](#)> Referenced in [17a](#).
 <set up Java 1.6 [11d](#)> Referenced in [39e](#).
 <set up python [11e](#)> Referenced in [17a](#).
 <set variables that point to the directory-structure [6cd](#), [8f](#), [10f](#)> Referenced in [6e](#), [17a](#), [77](#).
 <start EHU SRL server if it isn't running [41a](#)> Referenced in [40b](#).
 <start of module-script [23a](#)> Referenced in [35c](#), [36be](#), [37c](#), [38adg](#), [39be](#), [40bdf](#), [41d](#), [43a](#), [45a](#), [46ad](#), [47b](#), [48ce](#),
[49d](#), [50b](#), [51b](#), [52eg](#), [54c](#), [55a](#), [56b](#), [57ad](#), [58a](#), [59b](#).
 <start the Spotlight server on localhost [33ab](#)> Referenced in [30b](#), [32a](#).
 <stop EHU SRL server [41b](#)> Referenced in [40d](#).
 <stop on error [44a](#)> Referenced in [43a](#).
 <test whether spotlightserver runs [32e](#)> Referenced in [32a](#).
 <test whether virtualenv is present on the host [12d](#)> Referenced in [12c](#).
 <try to obtain a running spotlightserver [32a](#)> Not referenced.
 <unpack ticcutils or timbl [27d](#)> Referenced in [27bc](#).
 <update pip [13d](#)> Referenced in [11e](#).
 <variables of install-modules [62c](#)> Referenced in [17a](#).
 <wait until the spotlight server is up or faulty [33c](#)> Referenced in [33b](#).

C.3 Variables

activate: [13a](#), [14a](#).
 all: [34b](#), [65b](#).
 ALPINO_HOME: [25b](#).
 auxfil: [71d](#), [72c](#), [74c](#), [75a](#).
 bibtex: [72b](#), [75bc](#).
 DIRS: [76a](#), [76c](#).
 fig2dev: [69a](#).
 FIGFILENAMES: [68b](#).
 FIGFILES: [68a](#), [68b](#).
 hg: [21f](#).
 indexfil: [71d](#), [72c](#), [74c](#).
 lxml: [15a](#).
 makeindex: [72b](#), [75bc](#).
 MKDIR: [75d](#), [76a](#).
 moduleresult: [61a](#), [61b](#).
 naflang: [22a](#), [24a](#), [24bc](#), [30a](#), [31ab](#), [33a](#), [35c](#), [48ae](#), [57a](#), [60a](#).
 networkx: [15a](#).
 nufil: [71d](#), [72b](#), [74c](#), [75b](#).
 nuweb: [6c](#), [59d](#), [64](#), [66cd](#), [67a](#), [70cd](#), [71bc](#), [72b](#), [73a](#), [74a](#).
 oldaux: [71d](#), [72ac](#), [74c](#), [75a](#).
 oldindexfil: [71d](#), [72c](#), [74c](#).
 PATH: [6d](#), [10bf](#), [11d](#), [46d](#).
 pdf: [66ab](#), [70a](#), [70b](#).
 PDFT_NAMES: [68b](#), [70b](#).
 PDF_FIG_NAMES: [68b](#), [70b](#).
 PHONY: [65b](#), [69b](#).
 print: [12a](#), [14a](#), [16b](#), [23c](#), [31a](#), [36a](#), [52g](#), [67b](#), [70a](#).
 PST_NAMES: [68b](#).
 PS_FIG_NAMES: [68b](#).

pythonok: [12a](#).
PYTHONPATH: [13c](#).
pyyaml: [15a](#).
rdflib: [55g](#).
scriptpath: [23b](#), [24b](#).
SUFFIXES: [66b](#).
texfil: [71d](#), [72b](#), [74c](#), [75b](#).
trunk: [71d](#), [72b](#), [74c](#), [75bc](#).
view: [70a](#).
virtualenv: [12bc](#), [12d](#).