

Bilingual NLP pipeline

Paul Huygen <paul.huygen@huygen.nl>

7th January 2016
14:42 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	8
2.4	Installation from the snapshot	8
3	Java and Python environment	9
3.1	Java	10
3.2	Maven	11
3.3	Java 1.6	11
3.4	Python	12
3.4.1	Virtual environment	13
3.4.2	Transplant the virtual environment	14
3.4.3	KafNafParserPy	14
3.4.4	Python packages	15
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	Install utilities and resources	21
4.4.1	Process synchronisation	21
4.4.2	Prefix of scripts that run modules	22
4.4.3	Language detection	22
4.4.4	Alpino	23
4.4.5	Treetagger	24
4.4.6	Timbl and Ticcutils	25
4.4.7	The Boost library	26

4.4.8	Spotlight	27
4.4.9	VUA-pylib	31
4.4.10	SVMLight	31
4.4.11	CRFsuite	32
4.5	Install modules	32
4.5.1	Install tokenizer	32
4.5.2	Topic analyser	33
4.5.3	Morphosyntactic parser	33
4.5.4	Pos tagger	34
4.5.5	Constituent parser	35
4.5.6	NED-reranker	35
4.5.7	Wikify module	36
4.5.8	UKB	36
4.5.9	IMS-WSD	37
4.5.10	SRL server	37
4.5.11	SRL Dutch nominals	39
4.5.12	FBK-time module	39
4.5.13	FBK-temprel module	41
4.5.14	FBK-causalrel module	42
4.5.15	Factuality module	42
4.5.16	Nominal coreference-base	43
4.5.17	Named entity recognition (NERC)	44
4.5.18	Wordsense-disambiguation	45
4.5.19	Lexical-unit converter	46
4.5.20	NED	47
4.5.21	Ontotagger	48
4.5.22	Framenet SRL	49
4.5.23	Heideltime	50
4.5.24	Semantic Role labelling	52
4.5.25	SRL postprocessing	53
4.5.26	Event coreference	54
4.5.27	Dbpedia-ner	54
4.5.28	Nominal events	55
4.5.29	Opinion miner	56
5	Utilities	57
5.1	Test script	57
5.2	Logging	59
5.3	Misc	59
A	How to read and translate this document	60
A.1	Read this document	60
A.2	Process the document	60
A.3	The Makefile for this project.	61
A.4	Get Nuweb	62
A.5	Pre-processing	63
A.5.1	Process ‘dollar’ characters	63
A.5.2	Run the M4 pre-processor	63
A.6	Typeset this document	64
A.6.1	Figures	64
A.6.2	Bibliography	65
A.6.3	Create a printable/viewable document	65
A.6.4	Create HTML files	68
A.7	Create the program sources	71

A.8	Restore paths after transplantation	72
B	References	73
B.1	Literature	73
C	Indexes	73
C.1	Filenames	73
C.2	Macro's	74
C.3	Variables	76

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.vu.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
Corefgraph			EHU-corefgraph.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	4.5.1	Github	56f83ce4b61680346f15e5d4e6de6293764f7383	tok
morphosyntactic parser	4.5.3	Github	807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184	mor
NERC	4.5.17	Gith./snap	ca02c931bc0b200ccdb8b5795a7552e4cc0d4802	nerc
WSD	4.5.18	Gith./snap	030043903b42f77cd20a9b2443de137e2efe8513	wsd
Onto-tagger	4.5.21	snapshot		onto
Heideltime	4.5.23	Gith./snap.	da4604a7b33975e977017440cbc10f7d59917ddf	heidelttime
SRL	4.5.24	Github	675d22d361289ede23df11dcdb17195f008c54bf	srl
SRL-POST	4.5.25	snapshot		postsrl
NED	4.5.20	Github	d35d4df5cb71940bf642bb1a83e2b5b7584010df	ned
Nom. coref	4.5.16	Github	bfa5aec0fa498e57fe14dd4d2c51365dd09a0757	nomcoref
Ev. coref	4.5.26	snapshot		evcoref
Opinion miner	4.5.29	Github		opinimin
Framenet SRL	4.5.22	snapshot		fsrl
Dbpedia_ner	4.5.27	Github	ab1dcdb860f0ff29bc979f646dc382122a101fc2	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	4.4.4	RUG
Ticcutils	0.7	4.4.6	ILK
Timbl	6.4.6	4.4.6	ILK
Treetagger	3.2	4.4.5	Uni. München
Spotlight server	0.7	4.4.8	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

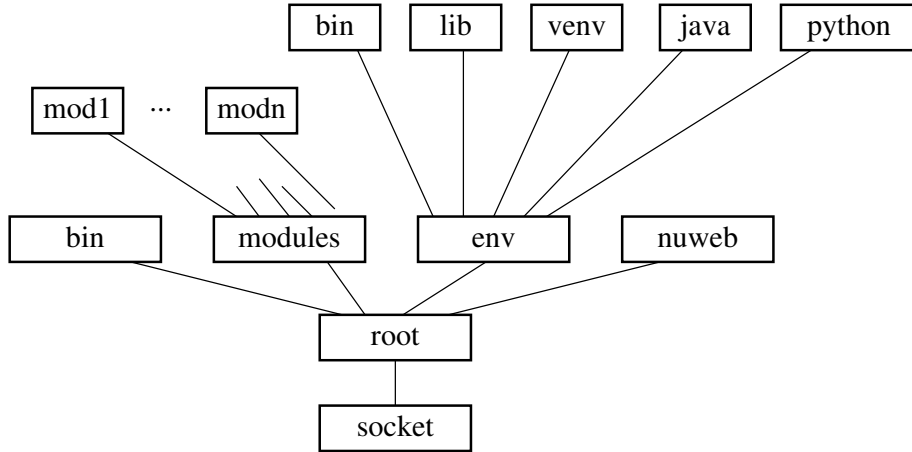


Figure 1: Directory-structure of the pipeline (see text).

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 6a > ≡
../modules ◇

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
 Fragment referenced in 72a.

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

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
 Fragment referenced in 72a.

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

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
 Fragment referenced in 72a.

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

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
 Fragment referenced in 72a.

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 6e > ≡
 if
 ["\$piperoot" == ""]
 then
 export piperoot=/home/phuijgen/nlp/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 6e, 7a, 9a, 11c.
 Fragment referenced in 7b, 17a, 73.
 Uses: nuweb 68b.

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

```

< set variables that point to the directory-structure 7a > ≡
    export PATH=$envbindir:$PATH
    ◇

```

Fragment defined by 6e, 7a, 9a, 11c.

Fragment referenced in 7b, 17a, 73.

Defines: PATH 10g, 11c, 12a, 43a, 57a.

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

```

"../env/bin/progenv" 7b≡
    #!/bin/bash
    < set variables that point to the directory-structure 6e, ... >
    export progenvset=0
    ◇

```

File defined by 7b, 10a.

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 7c > ≡
    if
    [ -e @1 ]
    then
        mv @1 old.@1
    fi
    ◇

```

Fragment referenced in 8d, 15a, 59c.

```

< remove old module 8a > ≡
    rm -rf old.@1
    ◇

```

Fragment referenced in 8d, 15a, 59c.

```

< re-instate old module 8b > ≡
    mv old.@1 @1
    MESS="Replaced previous version of @1"
    < logmess (8c $MESS ) 59b >
    ◇

```

Fragment referenced in 8d, 15a, 59c.

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 8d > ≡
    cd $modulesdir
    < move module (8e $DIRN ) 7c >
    git clone $GITU
    if
        [ $? -gt 0 ]
    then
        < logmess (8f Cannot install current $MODNAM version ) 59b >
        < re-instate old module (8g $DIRN ) 8b >
    else
        < remove old module (8h $DIRN ) 8a >
        cd $modulesdir/$DIRN
        git checkout $GITC
    fi
    ◇

```

Fragment referenced in 34a, 39b, 43c, 45f, 47c, 50d, 52d, 55a.

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 9a > ≡
  if
    [ ! $snapshotsocket ]
  then
    export snapshotsocket=/home/phuijgen/nlp
  fi
  ◇

```

Fragment defined by 6e, 7a, 9a, 11c.

Fragment referenced in 7b, 17a, 73.

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 9b > ≡
  if
    [ ! -e $pipesocket/nrkey ]
  then
    echo "No key to connect to snapshot!"
    exit 1
  fi
  ◇

```

Fragment defined by 9b, 21e.

Fragment referenced in 17a.

Update the local snapshot repository.

```

< get the snapshot 9c > ≡
  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 9d > ≡
  echo '#!/bin/bash' > /home/phuijgen/nlp/nlpp/env/bin/javapython
  ◇

```

Fragment referenced in 17a.

Cause the module scripts to read the javapython script.

```
"../env/bin/progenv" 10a≡
    source $envbindir/javapython
    ◇
```

File defined by 7b, 10a.

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 10b > ≡
    ../env/java ◇
```

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
Fragment referenced in 72a.

```
< set up java 10c > ≡
    < begin conditional install (10d java_installed ) 16b >
        cd $envdir/java
        tar -xzf $snapshotsocket/t_nlpp_resources/server-jre-7u72-linux-x64.tar.gz
    < end conditional install (10e java_installed ) 16d >
    ◇
```

Fragment defined by 10cg.
Fragment referenced in 17a.

Remove the java-ball when cleaning up:

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

Fragment defined by 10f, 11d, 24b, 63a.
Fragment referenced in 62a.

Set variables for Java.

```
< set up java 10g > ≡

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

Fragment defined by 10cg.
Fragment referenced in 17a.
Uses: PATH 7a.

Put jars in the jar subdirectory of the java directory:

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

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.
Fragment referenced in 72a.

3.2 Maven

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

```
<directories to create 11a> ≡
  ../env/apache-maven-3.0.5 ◇
```

Fragment defined by [6abcd](#), [10bh](#), [11a](#), [13e](#), [66c](#).

Fragment referenced in [72a](#).

```
<install maven 11b> ≡
  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 11c> ≡
  export MAVEN_HOME=$envdir/apache-maven-3.0.5
  export PATH=${MAVEN_HOME}/bin:${PATH}
  ◇
```

Fragment defined by [6e](#), [7a](#), [9a](#), [11c](#).

Fragment referenced in [7b](#), [17a](#), [73](#).

Uses: [PATH 7a](#).

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

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

Fragment defined by [10f](#), [11d](#), [24b](#), [63a](#).

Fragment referenced in [62a](#).

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 11f> ≡
  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 12a > ≡
    export JAVA_HOME=$envdir/java/jre1.6.0_45
    export PATH=$JAVA_HOME/bin:$PATH
    ◇

```

Fragment referenced in 37d.

Uses: PATH 7a.

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 12b > ≡
    < check/install the correct version of python 12c >
    < create a virtual environment for Python 13b >
    < activate the python environment 13d, ... >
    < install kafnafparserpy 15a >
    < install python packages 15f, ... >
    ◇

```

Fragment referenced in 17a.

```

< check/install the correct version of python 12c > ≡
pythonok='python --
version 2>&1 | gawk '{if(match($2, "2.7")) print "yes" ; else print "no" }'
if
[ "$pythonok" == "no" ]
then
    < install ActivePython 13a >
fi
◇

```

Fragment referenced in 12b.

Defines: pythonok Never used.

Uses: print 66a.

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 13a > ≡
    pytinsdir='mktemp -d -t activepyt.XXXXXX'
    cd $pytinsdir
    tar -xzf $snapshotsocket/t_nlpp_resources/ActivePython-2.7.8.10-linux-x86_64.tar.gz
    accdir='ls -1'
    cd $accdir
    ./install.sh -I $envdir
    cd $piperoot
    rm -rf $pytinsdir
    pip install -U pip virtualenv setuptools
    ◇

```

Fragment referenced in 12c.

Uses: virtualenv 13c.

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 13b > ≡
    < test whether virtualenv is present on the host 13c >
    cd $envdir
    virtualenv venv
    ◇

```

Fragment referenced in 12b.

Uses: virtualenv 13c.

```

< test whether virtualenv is present on the host 13c > ≡
    which virtualenv
    if
        [ $? -ne 0 ]
    then
        echo Please install virtualenv
        exit 1
    fi
    ◇

```

Fragment referenced in 13b.

Defines: virtualenv 13ab.

```

< activate the python environment 13d > ≡
    source $envdir/venv/bin/activate
    echo 'source $en-
vdir/venv/bin/activate' >> /home/phuijgen/nlp/nlpp/env/bin/javapython
    ◇

```

Fragment defined by 13d, 14a.

Fragment referenced in 12b, 17a.

Defines: activate 14b.

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

```

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

```

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.

Fragment referenced in 72a.

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

```
< activate the python environment 14a > ≡
echo ex-
port 'PYTHONPATH=$envdir/python:$PYTHONPATH' >> /home/phuijgen/nlp/nlpp/env/bin/javapython
export PYTHONPATH=$envdir/python:$PYTHONPATH
◇
```

Fragment defined by 13d, 14a.
 Fragment referenced in 12b, 17a.
 Defines: PYTHONPATH Never used.

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 14b > ≡
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 73.
 Uses: activate 13d, print 66a.

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 15a > ≡
  cd $envdir/python
  DIRN=KafNafParserPy
  < move module (15b $DIRN ) 7c >
  git clone https://github.com/cltl/KafNafParserPy.git
  if
    [ $? -gt 0 ]
  then
    < logmess (15c Cannot install current $DIRN version ) 59b >
    < re-instate old module (15d $DIRN ) 8b >
  else
    < remove old module (15e $DIRN ) 8a >
  fi
  ◇

```

Fragment referenced in 12b.

3.4.4 Python packages

Install python packages:

lxml:

pyyaml: for coreference-graph

```

< install python packages 15f > ≡
  pip install lxml
  pip install pyyaml
  ◇

```

Fragment defined by 15f, 53e.

Fragment referenced in 12b.

Defines: **lxml** 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 `/home/phuijgen/nlp/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 16a > ≡
    if
        [ -e /home/phuijgen/nlp/nlpp/installed_modules ]
    then
        source /home/phuijgen/nlp/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 17a.

```

< begin conditional install 16b > ≡
    if
        [ ! $@1 ]
    then
        ◇

```

Fragment referenced in 10c, 17a, 18aj, 19aj, 20ahq, 21a.

```

< else conditional install 16c > ≡
    else
        ◇

```

Fragment never referenced.

```

< end conditional install 16d > ≡
    echo "export @1=0" >> /home/phuijgen/nlp/nlpp/installed_modules
    fi
    ◇

```

Fragment referenced in 10c, 17a, 18aj, 19aj, 20ahq, 21a.

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

```

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

```

Fragment referenced in 11d.

Uses: `print` 66a.

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 6e, ... >
    < read the list of installed modules 16a >
    < check this first 9b, ... >
    < begin conditional install (17b repo_installed) 16b >
        < get the snapshot 9c >
    < end conditional install (17c repo_installed) 16d >
    < variables of install-modules 59a >
    < create javapython script 9d >
    echo ... Java
    < set up java 10c, ... >
    < begin conditional install (17d maven_installed) 16b >
        < install maven 11b >
    < end conditional install (17e maven_installed) 16d >
    < begin conditional install (17f java16_installed) 16b >
        < install Java 1.6 11f >
    < end conditional install (17g java16_installed) 16d >

    echo ... Python
    if
    [ $python_installed ]
    then
        < activate the python environment 13d, ... >
    fi
    < begin conditional install (17h python_installed) 16b >
        < set up python 12b >
    < end conditional install (17i python_installed) 16d >
    < begin conditional install (17j sematree_installed) 16b >
        < install sematree 22a >
    < end conditional install (17k sematree_installed) 16d >
    echo ... Alpino
    < begin conditional install (17l alpino_installed) 16b >
        < install Alpino 23f >
    < end conditional install (17m alpino_installed) 16d >
    echo ... Spotlight
    < begin conditional install (17n spotlight_installed) 16b >
        < install the Spotlight server 27a, ... >
    < end conditional install (17o spotlight_installed) 16d >
    echo ... Treetagger
    < begin conditional install (17p treetagger_installed) 16b >
        < install the treetagger utility 24c, ... >
    < end conditional install (17q treetagger_installed) 16d >
    echo ... Ticcutils and Timbl
    < begin conditional install (17r ticctimbl_installed) 16b >
        < install the ticcutils utility 26a >
        < install the timbl utility 26b >
    < end conditional install (17s ticctimbl_installed) 16d >
    echo ... Boost
    < begin conditional install (17t boost_installed) 16b >
        < install boost 26e >
    < end conditional install (17u boost_installed) 16d >

    echo ... VUA-pylib, SVMlight, CRFsuite
    < begin conditional install (17v miscutils_installed) 16b >
        < install VUA-pylib 31g >
        < install SVMlight 32a >
        < install CRFsuite 32b >
    < end conditional install (17w miscutils_installed) 16d >

```

◇

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) 16b>
    echo ... Tokenizer
    <install the tokenizer 33a>
  <end conditional install (18c tokenizer_installed) 16d>
  <begin conditional install (18d topic_installed) 16b>
    echo ... Topic detector
    <install the topic analyser 33d>
  <end conditional install (18e topic_installed) 16d>
  <begin conditional install (18f morpar_installed) 16b>
    echo ... Morphosyntactic parser
    <install the morphosyntactic parser 34a>
  <end conditional install (18g morpar_installed) 16d>
  <begin conditional install (18h pos_installed) 16b>
    echo "... Pos tagger (for english docs)"
    <install the pos tagger 35a>
  <end conditional install (18i pos_installed) 16d>
◇
```

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

```
"../bin/install-modules" 18j≡
  <begin conditional install (18k constparse_installed) 16b>
    echo "... Constituent parser (for english docs)"
    <install the constituents parser 35d>
  <end conditional install (18l constparse_installed) 16d>
  <begin conditional install (18m nerc_installed) 16b>
    echo ... NERC
    <install the NERC module 44a>
  <end conditional install (18n nerc_installed) 16d>
  <begin conditional install (18o ned_installed) 16b>
    echo ... NED
    <install the NED module 47c>
  <end conditional install (18p ned_installed) 16d>
  <begin conditional install (18q nedrer_installed) 16b>
    echo ...NED reranker
    <install the NED-reranker module 35g>
  <end conditional install (18r nedrer_installed) 16d>
  <begin conditional install (18s wikify_installed) 16b>
    echo ...WIKIfy module
    <install the wikify module 36c>
  <end conditional install (18t wikify_installed) 16d>
◇
```

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

```

"../bin/install-modules" 19a≡
  < begin conditional install (19b UKB_installed ) 16b>
    echo ... UKB module
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-ukb.v30.tgz
  < end conditional install (19c UKB_installed ) 16d>
  < begin conditional install (19d ims_wsd_installed ) 16b>
    echo ...ims-wsd module
    < install the ims-wsd module 37c>
  < end conditional install (19e ims_wsd_installed ) 16d>
  < begin conditional install (19f srl_server_installed ) 16b>
    echo ...srl-server module
    < install the srl-server module 38a>
  < end conditional install (19g srl_server_installed ) 16d>
  < begin conditional install (19h srl_dutch_nominals_installed ) 16b>
    echo ...srl-dutch-nominal module
    < install the srl-dutch-nominals module 39b>
  < end conditional install (19i srl_dutch_nominals_installed ) 16d>
  ◇

```

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

```

"../bin/install-modules" 19j≡
  < begin conditional install (19k FBK_time_installed ) 16b>
    echo ... FBK-time module
    < install the FBK-time module 39e>
  < end conditional install (19l FBK_time_installed ) 16d>
  < begin conditional install (19m FBK_temprel_installed ) 16b>
    echo ... FBK-temprel module
    < install the FBK-temprel module 41a>
  < end conditional install (19n FBK_temprel_installed ) 16d>
  < begin conditional install (19o FBK_causalrel_installed ) 16b>
    echo ... FBK-causalrel module
    < install the FBK-causalrel module 42a>
  < end conditional install (19p FBK_causalrel_installed ) 16d>
  < begin conditional install (19q factuality_installed ) 16b>
    echo ... factuality module
    < install the factuality module 42f>
  < end conditional install (19r factuality_installed ) 16d>
  ◇

```

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

```

"../bin/install-modules" 20a≡
  < begin conditional install (20b corefb_installed ) 16b >
    echo ... Coreference base
    < install coreference-base 43c >
  < end conditional install (20c corefb_installed ) 16d >
  < begin conditional install (20d wsd_installed ) 16b >
    echo ... WSD
    < install the WSD module 45f >
  < end conditional install (20e wsd_installed ) 16d >
  < begin conditional install (20f onto_installed ) 16b >
    echo ... Ontotagger
    < install the onto module 48d >
  < end conditional install (20g onto_installed ) 16d >
  ◇

```

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

```

"../bin/install-modules" 20h≡
  < begin conditional install (20i heidel_installed ) 16b >
    echo ... Heideltime
    < install the heideltime module 50c >
  < end conditional install (20j heidel_installed ) 16d >
  < begin conditional install (20k srl_installed ) 16b >
    echo ... SRL
    < install the srl module 52d >
  < end conditional install (20l srl_installed ) 16d >
  < begin conditional install (20m eventcoref_installed ) 16b >
    echo ... Event-coreference
    < install the event-coreference module 54c >
  < end conditional install (20n eventcoref_installed ) 16d >
  < begin conditional install (20o lu2synset_installed ) 16b >
    echo ... lu2synset
    < install the lu2synset converter 46e >
  < end conditional install (20p lu2synset_installed ) 16d >
  ◇

```

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

```

"../bin/install-modules" 20q≡
  < begin conditional install (20r dbpner_installed ) 16b >
    echo ... dbpedia-ner
    < install the dbpedia-ner module 55a >
  < end conditional install (20s dbpner_installed ) 16d >
  < begin conditional install (20t nomevent_installed ) 16b >
    echo ... nominal event
    < install the nomevent module 55d >
  < end conditional install (20u nomevent_installed ) 16d >
  < begin conditional install (20v post_SRL_installed ) 16b >
    echo ... post-SRL
    < install the post-SRL module 53f >
  < end conditional install (20w post_SRL_installed ) 16d >
  ◇

```

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

```

"../bin/install-modules" 21a≡
  < begin conditional install (21b opimin_installed ) 16b>
    echo ... opinion-miner
    < install the opinion-miner 56a, ... >
  < end conditional install (21c opimin_installed ) 16d>

  echo Final
  ◇

```

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

```

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

```

Fragment defined by 21d, 23c, 72b.

Fragment referenced in 72c.

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 9b, 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 43c.

4.4 Install utilities and resources

4.4.1 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 22a > ≡
    cat $snapshotsocket/t_nlpp_resources/sematree | \
        sed "s|/var/run|/home/phuijgen/nlp/nlpp/env/etc/sematree|g" \
    > $envbindir/sematree
    ◇
```

Fragment referenced in 17a.

4.4.2 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 22b > ≡
    #!/bin/bash
    source /home/phuijgen/nlp/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
    ◇
```

Fragment referenced in 23a, 33be, 34b, 35be, 36ad, 37ad, 38bdf, 39c, 40a, 41d, 42d, 43ad, 45bd, 46c, 47a, 48b, 49a, 50a, 52be, 54ad, 55be, 57a.

4.4.3 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 `lang`. All pipeline modules expect that this variable has been set.

```
"../env/bin/langdetect.py" 22c≡
    #!/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: `lang` 23d, `print` 66a.

```

"../bin/langdetect" 23a≡
  < start of module-script (23b ) 22b >
  echo 'cat | python $envbindir/langdetect.py'
  ◇

< make scripts executable 23c > ≡
  chmod 775 /home/phuijgen/nlp/nlpp/bin/langdetect
  ◇

```

Fragment defined by 21d, 23c, 72b.

Fragment referenced in 72c.

```

< set the language variable 23d > ≡
  naflang='cat @1 | /home/phuijgen/nlp/nlpp/bin/langdetect'
  export naflang
  ◇

```

Fragment referenced in 57c.

Defines: lang 22c.

Currently, the pipeline understands only English and Dutch. The following macro aborts pipeline processing when the language is not English or Dutch.

```

< abort when the language is not English or Dutch 23e > ≡
  if
    [ ! "$naflang" == 'nl' ] && [ ! "$naflang" == "en" ]
  then
    echo Language of NAF document not set. >&2
    echo Set variable "naflang" to "en" of "nl" and try again. >&2
    echo Aborting ':-(' >&2
    exit 4
  fi
  ◇

```

Fragment referenced in 33be.

4.4.4 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 23f > ≡
  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" >> /home/phuijgen/nlp/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 24a > ≡
    export ALPINO_HOME=$modulesdir/Alpino
    ◇
```

Fragment referenced in [34b](#).

Defines: `ALPINO_HOME` Never used.

Remove the tarball when cleaning up:

```
< clean up 24b > ≡
    rm -rf $snapshotsocket/t_nlpp_resources/Alpino-x86_64-linux-glibc2.5-20706-
    sicstus.tar.gz
    ◇
```

Fragment defined by [10f](#), [11d](#), [24b](#), [63a](#).

Fragment referenced in [62a](#).

4.4.5 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 24c > ≡
    TREETAGDIR=treetagger
    TREETAG_BASIS_URL=http://www.cis.uni-muenchen.de/%7Eschmid/tools/TreeTagger/data/
    TREETAGURL=http://www.cis.uni-muenchen.de/%7Eschmid/tools/TreeTagger/data/
    ◇
```

Fragment defined by [24cde](#), [25abcd](#).

Fragment referenced in [17a](#).

The source tarball, scripts and the installation-script:

```
< install the treetagger utility 24d > ≡
    TREETAGSRC=tree-tagger-linux-3.2.tar.gz
    TREETAGSCRIPTS=tagger-scripts.tar.gz
    TREETAG_INSTALLSCRIPT=install-tagger.sh
    ◇
```

Fragment defined by [24cde](#), [25abcd](#).

Fragment referenced in [17a](#).

Parametersets:

```
< install the treetagger utility 24e > ≡
    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 [24cde](#), [25abcd](#).

Fragment referenced in [17a](#).

Download everything in the target directory:

```
< install the treetagger utility 25a > ≡
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 24cde, 25abcd.

Fragment referenced in 17a.

Run the install-script:

```
< install the treetagger utility 25b > ≡
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```

Fragment defined by 24cde, 25abcd.

Fragment referenced in 17a.

Make the treetagger utilities available for everybody.

```
< install the treetagger utility 25c > ≡
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 24cde, 25abcd.

Fragment referenced in 17a.

Remove the tarballs:

```
< install the treetagger utility 25d > ≡
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```

Fragment defined by 24cde, 25abcd.

Fragment referenced in 17a.

4.4.6 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 26a > ≡
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 26c >
◇

```

Fragment referenced in 17a, 26d.

```

< install the timbl utility 26b > ≡
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
< unpack ticcutils or timbl 26c >
◇

```

Fragment referenced in 17a, 26d.

```

< unpack ticcutils or timbl 26c > ≡
SUCCE=0
ticbeldir='mktemp -t -d tickbel.XXXXXX'
cd $ticbeldir
tar -xzf $snapshotsocket/t_nlpp_resources/$TARB
cd $DIR
./configure --prefix=$envdir
make
make install
cd $piperoot
rm -rf $ticbeldir
◇

```

Fragment referenced in 26ab.

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

```

< re-install modules after the transplantation 26d > ≡
< install the ticcutils utility 26a >
< install the timbl utility 26b >
◇

```

Fragment referenced in 73.

4.4.7 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 26e > ≡
cd $envdir
tar -xzf $snapshotsocket/t_nlpp_resources/20160103_boost_1_54_bin.tgz
◇

```

Fragment referenced in 17a.

4.4.8 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 27a > ≡
  cd $envdir
  tar -xzf $snapshotsocket/t_nlpp_resources/spotlightnl.tgz
  cd $envdir/spotlight
  < get spotlight model ball (27b nl.tar.gz ) 27d >
  < get spotlight model ball (27c en_2+2.tar.gz ) 27d >
  ◇
```

Fragment defined by 27a, 28a.

Fragment referenced in 17a.

```
< get spotlight model ball 27d > ≡
  wget http://spotlight.sztaki.hu/downloads/@1
  tar -xzf @1
  rm @1
  ◇
```

Fragment referenced in 27a.

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

```

< install the Spotlight server 28a > ≡
  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 27a, 28a.

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 default host is not `localhost`, check whether Spotlight runs on `localhost`.
3. If a running spotlightserver is still not found, start a spotlightserver on `localhost`.

```

< function to check/start spotlight 28b > ≡
function check_start_spotlight {
  language=$1
  < get spotlight language parameters 29a >
  spotlighthost=130.37.53.38
  < check listener on host, port (28c $spotlighthost,28d $spotlightport ) 29b >
  if
    [ $spotlightrunning -ne 0 ]
  then
    if
      [ ! "$spotlighthost" == "localhost" ]
    then
      export spotlighthost="localhost"
      < check listener on host, port (28e $spotlighthost,28f $spotlightport ) 29b >
    fi
  fi
  if
    [ $spotlightrunning -ne 0 ]
  then
    < start the Spotlight server on localhost 31a >
  fi
  export spotlighthost
  export spotlightrunning
}
  ◇

```

Fragment referenced in 57c.

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

```

< get spotlight language parameters 29a > ≡
    if
        [ "$language" == "nl" ]
    then
        spotlightport=2060
        spotlightresource="nl"
    else
        spotlightport=2020
        spotlightresource="en_2+2"
    fi
◇

```

Fragment referenced in 28b.

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 29b > ≡
    exec 6<>/dev/tcp/@1/@2
    spotlightrunning=$?
    exec 6<&-
    exec 6>&-
◇

```

Fragment referenced in 28b, 31ad.

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 30a > ≡
  < test whether spotlighthost runs (30b $spotlighthost ) 30e >
  if
    [ ! $spotlightrunning ]
  then
    if
      [ "$spotlighthost" != "localhost" ]
    then
      export spotlighthost=localhost
      < test whether spotlighthost runs (30c $spotlighthost ) 30e >
    fi
  fi
  if
    [ ! $spotlightrunning ]
  then
    < start the Spotlight server on localhost 31a >
    < test whether spotlighthost runs (30d $spotlighthost ) 30e >
  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 30e > ≡
  exec 6<>/dev/tcp/@1/2060
  if
    [ $? -eq 0 ]
  then
    export spotlightrunning=0
  else
    spotlightrunning=
  fi
  exec 6<&-
  exec 6>&-
  ◇

```

Fragment referenced in [30a](#).

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

1. Acquire a lock.
2. Check that in the mean time Spotlight has not been started by another process.
3. Run the Spotlight java program if Spotlight does still not run.
4. release the lock

```

< start the Spotlight server on localhost 31a > ≡
    if
        [ "$naflang" == "nl" ]
    then
        spotresource="nl"
    else
        spotresource="en_2+2"
    fi
    cd /home/phuijgen/nlp/nlpp/env/spotlight
    sematree acquire spotlock -1
    < check listener on host, port (31b $spotlighthost,31c $spotlightport ) 29b >
    if
        [ ! $spotlightrunning -eq 0 ]
    then
        java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-
        candidates.jar $spotresource http://localhost:$spotlightport/rest &
        < wait until the spotlight server is up 31d >
    fi
    sematree release spotlock
◇

```

Fragment referenced in 28b, 30a.

```

< wait until the spotlight server is up 31d > ≡
    spotlightrunning=1
    while
        [ $spotlightrunning -ne 0 ]
    do
        sleep 10
        < check listener on host, port (31e $spotlighthost,31f $spotlightport ) 29b >
    done
◇

```

Fragment referenced in 31a.

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.

4.4.9 VUA-pylib

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

```

< install VUA-pylib 31g > ≡
    cd $envdir/python
    git clone https://github.com/clt1/VUA_pylib.git
◇

```

Fragment referenced in 17a.

4.4.10 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 32a> ≡
  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 /home/phuijgen/nlp/nlpp/env/bin/
  cp svm_learn /home/phuijgen/nlp/nlpp/env/bin/
  cd /home/phuijgen/nlp/nlpp
  rm -rf $tempdir
◇
```

Fragment referenced in 17a.

Uses: all 61c.

4.4.11 CRFsuite

CRFsuite is an implementation of Conditional Random Fields (CRF). Module **opinion-miner-deluxe** 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 32b> ≡
  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 /home/phuijgen/nlp/nlpp
  rm -rf $tempdir
◇
```

Fragment referenced in 17a.

4.5 Install modules

4.5.1 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 33a > ≡
    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" 33b≡
    < start of module-script (33c $jarsdir ) 22b >
    < abort when the language is not English or Dutch 23e >
    JARFILE=$jarsdir/ixa-pipe-tok-1.8.0.jar
    java -Xmx1000m -jar $JARFILE tok -l $naflang --inputkaf
    ◇

```

4.5.2 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 33d > ≡
    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 66a.

Script:

```

"../bin/topic" 33e≡
    < start of module-script (33f EHU-topic.v30 ) 22b >
    < abort when the language is not English or Dutch 23e >
    java -jar $MODDIR/ixa-pipe-topic-1.0.1.jar -p $MODDIR/conf.prop
    ◇

```

4.5.3 Morphosyntactic parser

Module

```

< install the morphosyntactic parser 34a > ≡
MODNAM=morphsynparser
DIRN=morphosyntactic_parser_nl
GITU=https://github.com/cltl/morphosyntactic_parser_nl.git
GITC=807e938ce4ebb71afd9d7c7f42d9d9ac5f98a184
< install from github 8d >
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" 34b≡
< start of module-script (34c morphosyntactic_parser_nl ) 22b >
< get the mor time-out parameter 34d >
< set alpinohome 24a >
cat | python $MODDIR/core/morph_syn_parser.py $timeoutarg
◇

```

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

```

< get the mor time-out parameter 34d > ≡
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
◇

```

Fragment referenced in 34b.

4.5.4 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 35a > ≡
  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" 35b≡
  < start of module-script (35c EHU-pos.v30 ) 22b >
  java -jar ${MODDIR}/ixa-pipe-pos-1.4.3.jar tag -m ${MODDIR}/en-maxent-100-c5-
  baseline-dict-penn.bin
  ◇

```

4.5.5 Constituent parser

Module

```

< install the constituents parser 35d > ≡
  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" 35e≡
  < start of module-script (35f EHU-parse.v30 ) 22b >
  java -jar ${MODDIR}/ixa-pipe-parse-1.1.1.jar parse -g sem -m ${MODDIR}/en-parser-
  chunking.bin
  ◇

```

4.5.6 NED-reranker

Module

```

< install the NED-reranker module 35g > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-popen-nedreranker.v30.tgz
  ◇

```

Fragment referenced in [18j](#).

Script

```
"../bin/nedrer" 36a≡
  ⟨ start of module-script (36b VUA-popen-nedreranker.v30 ) 22b ⟩
  cd $MODDIR
  python $MODDIR/domain_model.py
  ◇
```

4.5.7 Wikify module

Module

```
⟨ install the wikify module 36c ⟩ ≡
  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" 36d≡
  ⟨ start of module-script (36e EHU-wikify.v30 ) 22b ⟩
  if
    [ "$naflang" == "nl" ]
  then
    spotlightport=2060
  else
    spotlightport=2020
  fi
  [ $spotlightrunning ] || source /home/phuijgen/nlp/nlpp/bin/start-spotlight

  cd $MODDIR
  java -jar ${MODDIR}/ixa-pipe-wikify-1.2.1.jar -s http://$spotlighthost -
  p $spotlightport
  ◇
```

4.5.8 UKB

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

Module

```
⟨ install the UKB module 36f ⟩ ≡
  ◇
```

Fragment never referenced.

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

```
"../bin/ukb" 37a≡
  < start of module-script (37b EHU-ukb.v30 ) 22b >
  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
  ◇
```

4.5.9 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 37c > ≡
  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" 37d≡
  < start of module-script (37e VUA-ims-wsd.v30 ) 22b >
  < set up Java 1.6 12a >
  #Setting the output to be ili-wn30 synsets instead of sensekeys
  $MODDIR/call_ims.py -ili30
  ◇
```

4.5.10 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 38a > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_EHU-srl-server.tgz
  cd EHU-srl-server
  mkdir -p /home/phuijgen/nlp/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" 38b≡
  < start of module-script (38c EHU-srl-server ) 22b >
  < start EHU SRL server if it isn't running 38h >
  ◇

```

```

"../bin/stop_eSRL" 38d≡
  < start of module-script (38e EHU-srl-server ) 22b >
  < stop EHU SRL server 39a >
  ◇

```

```

"../bin/eSRL" 38f≡
  < start of module-script (38g EHU-srl-server ) 22b >
  /home/phuijgen/nlp/nlpp/bin/start_eSRL
  java -cp $MODDIR/IXA-EHU-srl-3.0.jar ixa.srl.SRLClient en
  ◇

```

```

< start EHU SRL server if it isn't running 38h > ≡
  pidFile=/home/phuijgen/nlp/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 38b.

```

< stop EHU SRL server 39a > ≡
    pidFile=/home/phuijgen/nlp/nlpp/env/etc/pid/SRLServer.pid
    if
        [ -e "$pidFile" ]
    then
        kill 'echo $pidFile'
        rm $pidFile
    fi
    ◇

```

Fragment referenced in 38d.

4.5.11 SRL Dutch nominals

Module

```

< install the srl-dutch-nominals module 39b > ≡
    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 8d >
    cd $modulesdir/vua-srl-dutch-nominal-events
    chmod 775 vua-srl-dutch-additional-roles.py
    ◇

```

Fragment referenced in 19a.

Script

```

"../bin/srl-dutch-nominals" 39c ≡
    < start of module-script (39d vua-srl-dutch-nominal-events ) 22b >
    cd $MODDIR
    cat | $MODDIR/vua-srl-dutch-additional-roles.py
    ◇

```

4.5.12 FBK-time module

Module

```

< install the FBK-time module 39e > ≡
    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" 40a≡
  < start of module-script (40b FBK-time.v30 ) 22b >
  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=$timdir/TimeProNormIN.txp
  mytemp=$timdir/mytemp
  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 -cp $JAVACLASSPATH $JAVAMODULE $FILETXP chunk+entity timex

  #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

  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 -cp $JAVACLASSPATH $JAVAMODULE $FILETXP

  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 -
  cp $JAVACLASSPATH $JAVAMODULE $FILEOUT chunk+morpho+timex+event eval

  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 -Dfile.encoding=UTF8 -
  cp $JAVACP1 $JAVAMOD1 $CHUNKIN $FILETXP "$BEGINTIME" TIMEX3 > $mytemp
  cat $mytemp | java -Dfile.encoding=UTF8 -cp $JAVACP2 $JAVAMOD2 $FILEOUT

  rm $FILETXP
  rm $CHUNKIN
  rm -rf $timdir
  ◇

```


4.5.13 FBK-temprel module

Module

```

⟨ install the FBK-temprel module 41a ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-temprel.v30.tgz
  ⟨ repair FBK-*rel's run.sh.hadoop (41b FBK-temprel.v30 ) 41c ⟩
  ◇

```

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 41c ⟩ ≡
  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 41a, 42a.

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" 41d≡
  ⟨ start of module-script (41e FBK-temprel.v30 ) 22b ⟩
  cd $MODDIR
  scratchDir='mktemp -d -t temprel.XXXXXX'
  cat >$scratchDir/in.naf
  cat $scratchDir/in.naf | ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
  rm -rf $scratchDir
  ◇

```

4.5.14 FBK-causalrel module

Module

```

⟨ install the FBK-causalrel module 42a ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_FBK-causalrel.v30.tgz
  ⟨ repair FBK-*rel's run.sh.hadoop (42b FBK-causalrel.v30 ) 41c ⟩
  ◇

```

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 42c ⟩ ≡
  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" 42d≡
  ⟨ start of module-script (42e FBK-causalrel.v30 ) 22b ⟩
  cd $MODDIR
  scratchDir='mktemp -d -t causalrel.XXXXXX'
  cat >$scratchDir/in.naf
  cat $scratchDir/in.naf | ./run.sh.hadoop $MODDIR $scratchDir $scratchDir/in.naf
  rm -rf $scratchDir
  ◇

```

4.5.15 Factuality module

Module

```

⟨ install the factuality module 42f ⟩ ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/20151220_VUA-factuality.v30.tgz
  ◇

```

Fragment referenced in 19j.

Script

```
"../bin/factuality" 43a≡
  ⟨ start of module-script (43b VUA-factuality.v30 ) 22b ⟩
  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:/home/phuijgen/nlp/nlpp/env/lib/

  #mkdir -p ${scratchDir}/test

  python ${rootDir}/vua_factuality_naf_wrapper.py -
  t /home/phuijgen/nlp/nlpp/env/bin/timbl -p ${rootDir} ${tmpDir}/

  ◇
```

4.5.16 Nominal coreference-base

Get this thing from Github (<https://github.com/opener-project/coreference-base/>) and apply the instruction of <https://github.com/opener-project/coreference-base/blob/master/core/README.md>. We implement it, but it does not work yet, because it is too picky on the structure of the NAF format.

Module

```
⟨ install coreference-base 43c ⟩ ≡
  MODNAM=coreference-base
  DIRN=coreference-base
  GITU=https://github.com/opener-project/coreference-base.git
  GITC=bfa5aec0fa498e57fe14dd4d2c51365dd09a0757
  ⟨ install from github 8d ⟩
  pip install --upgrade hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
  pip install --upgrade networkx

  ◇
```

Fragment referenced in 20a.

Uses: hg 21f.

Script

```
"../bin/coreference-base" 43d≡
  ⟨ start of module-script (43e coreference-base ) 22b ⟩
  cd $MODDIR/core
  cat | python -m corefgraph.process.file --language nl --singleton --sieves NO

  ◇
```

4.5.17 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 44a > ≡
  < compile the nerc jar 44b >
  < get the nerc models 45a >
```

◇

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 44b > ≡
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 44a.

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 44c > ≡
if
  [ "$naflang" == "nl" ]
then
  export nercmodel=nl/nl-clusters-conll102.bin
else
  export nercmodel=en/en-newsreader-clusters-3-class-muc7-conll103-ontonotes-4.0.bin
fi
```

◇

Fragment referenced in 57c.

The tarball `20160105_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 45a > ≡
    cd $modulesdir
    tar -xzf /home/phuijgen/nlp/t_nlpp_resources/20160105_nerc_models.tgz
    ◇

```

Fragment referenced in 44a.

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

```

"../bin/nerc_conll02" 45b≡
    < start of module-script (45c m4_nerc_nl_dir ) 22b >
    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" 45d≡
    < start of module-script (45e nerc-models ) 22b >
    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 -jar $JAR tag -m $nercmodel
    ◇

```

4.5.18 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 45f > ≡
    MODNAM=wsd
    DIRN=svm_wsd
    GITU=https://github.com/cltl/svm_wsd.git
    GITC=030043903b42f77cd20a9b2443de137e2efe8513
    < install from github 8d >
    cd $modulesdir/svm_wsd
    < install svm lib 46a >
    < download svm models 46b >

    ◇

```

Fragment referenced in 20a.

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

```
< install svm lib 46a > ≡
  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 45f.

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

```
< download svm models 46b > ≡
  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 45f.

Script

```
"../bin/wsd" 46c≡
  < start of module-script (46d svm_wsd ) 22b >
  WSDSCRIPT=dsc_wsd_tagger.py
  cat | python $MODDIR/$WSDSCRIPT --naf -ref odwnSY
  ◇
```

4.5.19 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 46e > ≡
  cd $modulesdir
  tar -xzf $snapshotsocket/t_nlpp_resources/lu2synset.tgz
  ◇
```

Fragment referenced in 20h.

Script

```
"../bin/lu2synset" 47a≡
  ⟨ start of module-script (47b lexicalunitconvertor ) 22b⟩
  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
  ◇
```

4.5.20 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 47c ⟩ ≡
  ⟨ put spotlight jar in the Maven repository 48a ⟩
  MODNAM=ned
  DIRN=ixa-pipe-ned
  GITU=https://github.com/ixa-ehu/ixa-pipe-ned.git
  GITC=d35d4df5cb71940bf642bb1a83e2b5b7584010df
  ⟨ install from github 8d ⟩
  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 48a> ≡
    echo Put Spotlight jar in the Maven repository.
    tempdir='mktemp -d -t simplespot.XXXXXX'
    cd $tempdir
    wget http://spotlight.sztaki.hu/downloads/dbpedia-spotlight-0.7.jar
    wget http://spotlight.sztaki.hu/downloads/nl.tar.gz
    tar -xzf nl.tar.gz
    MVN_SPOTLIGHT_OPTIONS="-Dfile=dbpedia-spotlight-0.7.jar"
    MVN_SPOTLIGHT_OPTIONS="$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 47c.

Script NED needs to contact a Spotlight-server.

```

"../bin/ned" 48b≡
    <start of module-script (48c ) 22b>
    #!/bin/bash
    source /home/phuijgen/nlp/nlpp/env/bin/progenv
    ROOT=$piperoot
    JARDIR=$jarsdir
    if
        [ "$naflang" == "nl" ]
    then
        spotlightport=2060
    else
        spotlightport=2020
    fi
    [ $spotlightrunning ] || source /home/phuijgen/nlp/nlpp/bin/start-spotlight
    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
    ◇

```

4.5.21 Ontotagger

We do not yet have a source-repository of the Ontotagger module. Therefore, install from a snapshot (20151217_vua-ontotagger-v1.0.tgz).

Module

```

<install the onto module 48d> ≡
    cd $modulesdir
    tar -xzf $snapshotsocket/t_nlpp_resources/20151217_vua-ontotagger-v1.0.tgz
    chmod -R o+r $modulesdir/vua-ontotagger-v1.0
    ◇

```

Fragment referenced in 20a.

Script

```
"../bin/onto" 49a≡
  ( start of module-script (49b vua-ontotagger-v1.0 ) 22b )
  JARDIR=$MODDIR/lib
  RESOURCEDIR=$MODDIR/resources
  PREDICATEMATRIX="$RESOURCEDIR/PredicateMatrix.v1.3.txt.role.odwn"
  GRAMMATICALWORDS="$RESOURCEDIR/grammaticals/Grammatical-words.nl"
  TMPFIL='mktemp -t stap6.XXXXXX'
  cat >$TMPFIL

  CLASSPATH=$JARDIR/ontotagger-1.0-jar-with-dependencies.jar
  JAVASCRIPT=eu.kyotoproject.main.KafPredicateMatrixTagger
  MAPPINGS="fn;mcr;ili;eso"
  JAVA_ARGS="--mappings $MAPPINGS"
  JAVA_ARGS="$JAVA_ARGS --key odwn-eq"
  JAVA_ARGS="$JAVA_ARGS --version 1.2"
  JAVA_ARGS="$JAVA_ARGS --predicate-matrix $PREDICATEMATRIX"
  JAVA_ARGS="$JAVA_ARGS --grammatical-words $GRAMMATICALWORDS"
  JAVA_ARGS="$JAVA_ARGS --naf-file $TMPFIL"
  java -Xmx1812m -cp $CLASSPATH $JAVASCRIPT $JAVA_ARGS
  rm -rf $TMPFIL

  ◇
```

4.5.22 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" 50a≡
  < start of module-script (50b vua-ontotagger-v1.0 ) 22b>
  ONTODIR=$modulesdir/vua-ontotagger-v1.0
  JARDIR=$MODDIR/lib
  RESOURCEDIR=$MODDIR/resources
  PREDICATEMATRIX="$RESOURCEDIR/PredicateMatrix_nl_lu_withES0.v0.2.role.txt"
  GRAMMATICALWORDS="$RESOURCEDIR/grammaticals/Grammatical-words.nl"
  TMPFIL='mktemp -t framesrl.XXXXXX'
  cat >$TMPFIL

  CLASSPATH=$JARDIR/ontotagger-1.0-jar-with-dependencies.jar
  JAVASCRIPT=eu.kyotoproject.main.SrlFrameNetTagger

  JAVA_ARGS="--naf-file $TMPFIL"
  JAVA_ARGS="$JAVA_ARGS --format naf"
  JAVA_ARGS="$JAVA_ARGS --frame-ns fn:"
  JAVA_ARGS="$JAVA_ARGS --role-ns fn-role;;pb-role;;fn-pb-role;;eso-role:"
  JAVA_ARGS="$JAVA_ARGS --ili-ns mcr:ili"
  JAVA_ARGS="$JAVA_ARGS --sense-conf 0.25"
  JAVA_ARGS="$JAVA_ARGS --frame-conf 70"

  java -Xmx1812m -
  cp $CLASSPATH $JAVASCRIPT $JAVA_ARGS | gawk '/^frameMap.size()/ {next}; {print}'
  rm -rf $TMPFIL

◇

```

4.5.23 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 50c> ≡
  moduledir=/home/phuijgen/nlp/nlpp/modules/ixa-pipe-time
  < clone the heideltime wrapper 50d>
  < put Antske's material in the heideltime wrapper 51a>
  < compile the heideltime wrapper 51b>

◇

```

Fragment referenced in [20h](#).

```

< clone the heideltime wrapper 50d> ≡
  MODNAM=heideltime
  DIRN=ixa-pipe-time
  GITU=https://github.com/ixa-ehu/ixa-pipe-time.git
  GITC=da4604a7b33975e977017440cbc10f7d59917ddf
  < install from github (50e ixa-pipe-time ) 8d>
  mkdir $moduledir/lib

◇

```

Fragment referenced in [50c](#).

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 51a > ≡
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 50c.

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

```
< compile the heideltime wrapper 51b > ≡
< get jvntextpro-2.0.jar 51c >
< activate the install-to-project-repo utility 51d >
cd /home/phuijgen/nlp/nlpp/modules/$DIRN
mvn clean install
◇
```

Fragment referenced in 50c.

```
< get jvntextpro-2.0.jar 51c > ≡
cd /home/phuijgen/nlp/nlpp/modules/$DIRN/lib
wget http://ixa2.si.ehu.es/%7Ejibalar/jvntextpro-2.0.jar
◇
```

Fragment referenced in 51b.

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 51d > ≡
< remove outdated heideltime jars 52a >
cd /home/phuijgen/nlp/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 51b.

```

< remove outdated heideltime jars 52a > ≡
rm -rf /home/phuijgen/nlp/nlpp/modules/$DIRN/repo
mkdir -p /home/phuijgen/nlp/nlpp/modules/$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 51d.

Script

```

"../bin/heideltime" 52b≡
< start of module-script (52c ixa-pipe-time ) 22b >
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
◇

```

4.5.24 Semantic Role labelling

Module

```

< install the srl module 52d > ≡
MODNAM=srl
DIRN=vua-srl-nl
GITU=https://github.com/newsreader/vua-srl-nl.git
GITC=675d22d361289ede23df11dcdb17195f008c54bf
< install from github 8d >
◇

```

Fragment referenced in 20h.

Script First:

1. set the correct environment. The module needs python and timble.
2. create a tempdir and in that dir a file to store the input and a (scv) file with the feature-vector.

```

"../bin/srl" 52e≡
< start of module-script (52f vua-srl-nl ) 22b >
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 52e, 53abcd.

Create a feature-vector.

```

"../bin/srl" 53a≡
    cat | tee $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
    ◇

```

File defined by 52e, 53abcd.

Run the trained model on the feature-vector.

```

"../bin/srl" 53b≡
    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 52e, 53abcd.

Insert the SRL values into the NAF file.

```

"../bin/srl" 53c≡
    python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
    ◇

```

File defined by 52e, 53abcd.

Clean up.

```

"../bin/srl" 53d≡
    rm -rf $TEMPDIR
    ◇

```

File defined by 52e, 53abcd.

4.5.25 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 53e⟩ ≡
    pip install rdflib
    ◇

```

Fragment defined by 15f, 53e.

Fragment referenced in 12b.

Defines: rdflib Never used.

```

⟨install the post-SRL module 53f⟩ ≡
    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" 54a≡
  ⟨ start of module-script (54b vua-srl-postprocess ) 22b ⟩
  cd $MODDIR
  tempdir='mktemp -d -t postsr1.XXXXX'
  cat >$tempdir/infile
  python $MODDIR/main.py -i $tempdir/infile -o $tempdir/outfile
  cat $tempdir/outfile
  rm -rf $tempdir
  ◇
```

4.5.26 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 54c ⟩ ≡
  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" 54d≡
  ⟨ start of module-script (54e vua-eventcoreference_v2 ) 22b ⟩
  RESOURCESDIR=$MODDIR/resources
  JARFILE=$jarsdir/EventCoreference-1.0-SNAPSHOT-jar-with-dependencies.jar

  JAVAMODULE=eu.newsreader.eventcoreference.naf.EventCorefWordnetSim
  JAVAOPTIONS="--method leacock-chodorow"
  JAVAOPTIONS="$JAVAOPTIONS --wn-lmf $RESOURCESDIR/wneng-30.lmf.xml.xpos"
  JAVAOPTIONS="$JAVAOPTIONS --sim 2.0"
  JAVAOPTIONS="$JAVAOPTIONS --relations has_hyperonym#event#has_hypernym"

  java -Xmx812m -cp $JARFILE $JAVAMODULE $JAVAOPTIONS
  ◇
```

4.5.27 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 55a > ≡
MODNAM=dbpedia_ner
DIRN=dbpedia_ner
GITU=https://github.com/PaulHuygen/dbpedia_ner.git
GITC=ab1dcbd860f0ff29bc979f646dc382122a101fc2
< install from github 8d >
◇

```

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" 55b ≡
< start of module-script (55c dbpedia_ner ) 22b >
cat | iconv -f ISO8859-1 -t UTF-8 | $MODDIR/dbpedia_ner.py -
url http://$spotlighthost:2060/rest/candidates
◇

```

4.5.28 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

```

< install the nomevent module 55d > ≡
cd $modulesdir
tar -xzf $snapshotssocket/t_nlpp_resources/20151231_vua-nominal-event-detection-
nl.tgz
◇

```

Fragment referenced in 20q.

Script

```

"../bin/nomevent" 55e ≡
< start of module-script (55f vua-nominal-event-detection-nl ) 22b >
LIBDIR=$MODDIR/lib
RESOURCESDIR=$MODDIR/resources

JAR=$LIBDIR/ontotagger-1.0-jar-with-dependencies.jar
JAVAMODULE=eu.kyotoproject.main.NominalEventCoreference
cat | iconv -f ISO8859-1 -t UTF-8 | java -Xmx812m -cp $JAR $JAVAMODULE --framenet-
lu $RESOURCESDIR/nl-luIndex.xml
◇

```

4.5.29 Opinion miner

To run the opinion-miner, the following things are needed:

- SVMlight
- crfsuite
- vua-pylib

Module The module can be cloned from Github. However, currently there are problems with the Github installation. Therefore we borrow the opinion miner from the English NWR pipeline.

```
<install the opinion-miner 56a> ≡
  cd /home/phuijgen/nlp/nlpp/modules
  tar -xzf /home/phuijgen/nlp/t_nlpp_resources/20151012VUA-opinion-miner.tgz
  ◇
```

Fragment defined by 56ac.

Fragment referenced in 21a.

The opinion-miner needs a configuration file that is located in the directory where the model-data resides. In this pipeline we will use model-data derived from news-articles. An alternative model, derived from hotel evaluations can also be used. Put the configuration file in the `etc` subdir and copy it to its proper location during the installation of the opinion-miner.

```
"../env/etc/opini.cfg" 56b≡
[general]
output_folder = /home/phuijgen/nlp/nlpp/modules/VUA-opinion-
miner/final_models/ennl/news_cfg1

[crfsuite]
path_to_binary = /home/phuijgen/nlp/nlpp/env/bin/crfsuite

[svmlight]
path_to_binary_learn = /home/phuijgen/nlp/nlpp/env/bin/svm_learn
path_to_binary_classify = /home/phuijgen/nlp/nlpp/env/bin/svm_classify
  ◇
```

```
<install the opinion-miner 56c> ≡
  cd VUA-opinion-miner
  cat /home/phuijgen/nlp/nlpp/env/etc/opini.cfg | \
    sed s/ennl/nl/g > $modulesdir/VUA-opinion-
miner/final_models/nl/news_cfg1/config.cfg
  cat /home/phuijgen/nlp/nlpp/env/etc/opini.cfg | \
    sed s/ennl/en/g > $modulesdir/VUA-opinion-
miner/final_models/en/news_cfg1/config.cfg
  ◇
```

Fragment defined by 56ac.

Fragment referenced in 21a.

Script


```

"../bin/opinimin" 57a≡
  < start of module-script (57b VUA-opinion-miner ) 22b >
  cd $MODDIR
  export PATH=$PATH:.
  if
    [ "$naflang" == "nl" ]
  then
    modelconf=$MODDIR/final_models/nl/news_cfg1
  else
    modelconf=$MODDIR/final_models/en/news_cfg1
  fi
  python classify_kaf_naf_file.py -m $modelconf
  ◇

```

5 Utilities

5.1 Test script

The following script pushes a test-document through the modules of the pipeline.

```

"../bin/test" 57c≡
  #!/bin/bash
  ROOT=/home/phuijgen/nlp/nlpp
  TESTDIR=$ROOT/test
  < function to check/start spotlight 28b >
  TESTIN=$ROOT/nuweb/test.nl.in.naf
  if
    [ "$1" == "en" ]
  then
    TESTIN=$ROOT/nuweb/test.en.in.naf
  fi
  BIND=$ROOT/bin
  mkdir -p $TESTDIR
  cd $TESTDIR
  < set the language variable (57d $TESTIN ) 23d >
  < select language-dependent features 44c >
  check_start_spotlight $naflang
  if
    [ "$naflang" == "nl" ]
  then
    cat $TESTIN      | $BIND/tok                > tok.naf
    cat tok.naf      | $BIND/mor                > mor.naf
    cat mor.naf      | $BIND/nerc               > nerc.naf
    cat nerc.naf     | $BIND/wsd                > wsd.naf
    cat wsd.naf      | $BIND/ned                > ned.naf
    cat ned.naf      | $BIND/heideltime        > times.naf
    cat times.naf    | $BIND/onto               > onto.naf
    cat onto.naf     | $BIND/srl                > srl.naf
    cat srl.naf      | $BIND/nomevent           > nomev.naf
    cat nomev.naf    | $BIND/srl-dutch-nominals > psrl.naf
    cat psrl.naf     | $BIND/framesrl          > fsrl.naf
    cat fsrl.naf     | $BIND/opinimin           > opin.naf
    cat opin.naf     | $BIND/evcoref           > out.naf
  else
    < annotate english document 58b >
  fi
  ◇

```

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 58a > ≡
cat $TESTIN      | $BIND/tok                > tok.naf
cat tok.naf      | $BIND/mor                > mor.naf
cat mor.naf      | $BIND/nerc               > nerc.naf
cat nerc.naf     | $BIND/wsd                > wsd.naf
cat wsd.naf      | $BIND/ned                > ned.naf
cat ned.naf      | $BIND/heideltime         > times.naf
cat times.naf    | $BIND/onto               > onto.naf
cat onto.naf     | $BIND/srl                > srl.naf
cat srl.naf      | $BIND/nomevent           > nomev.naf
cat nomev.naf    | $BIND/postsrsl           > psrl.naf
cat psrl.naf     | $BIND/framesrl           > fsrl.naf
cat fsrl.naf     | $BIND/opinimin           > opin.naf
cat opin.naf     | $BIND/evcoref            > out.naf
◇

```

Fragment never referenced.

```

< annotate english document 58b > ≡
cat $TESTIN      | $BIND/tok                > tok.naf
cat tok.naf      | $BIND/topic              > top.naf
cat top.naf      | $BIND/pos                > pos.naf
cat pos.naf      | $BIND/constpars          > consp.naf
cat consp.naf    | $BIND/nerc               > nerc.naf
cat nerc.naf     | $BIND/nedrer             > nedr.naf
cat nedr.naf     | $BIND/wikify             > wikif.naf
cat wikif.naf    | $BIND/ukb                > ukb.naf
cat ukb.naf      | $BIND/ewsd               > ewsd.naf
cat ewsd.naf     | $BIND/esrl               > esrl.naf
cat esrl.naf     | $BIND/FBK-time           > time.naf
cat time.naf     | $BIND/FBK-temprel        > trel.naf
cat trel.naf     | $BIND/FBK-causalrel      > crel.naf
cat crel.naf     | $BIND/evcoref            > ecrf.naf
cat ecrf.naf     | $BIND/factuality         > fact.naf
cat fact.naf     | $BIND/opinimin           > out.naf
◇

```

Fragment referenced in 57c.

5.2 Logging

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

```
< variables of install-modules 59a > ≡
    LOGLEVEL=1
◇
```

Fragment referenced in [17a](#).

```
< logmess 59b > ≡
    if
    [ $LOGLEVEL -gt 0 ]
    then
    echo @1
    fi
◇
```

Fragment referenced in [8bd](#), [15a](#), [59c](#).

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 59c > ≡
    SUCCES=0
    cd $modulesdir
    < move module (59d $DIR ) 7c >
    wget $URL
    SUCCES=$?
    if
    [ $SUCCES -eq 0 ]
    then
    tar -xzf $TARB
    SUCCES=$?
    rm -rf $TARB
    fi
    if
    [ $SUCCES -eq 0 ]
    then
    < logmess (59e Installed $DIR ) 59b >
    < remove old module (59f $DIR ) 8a >
    else
    < re-instate old module (59g $DIR ) 8b >
    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 \TeX 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.

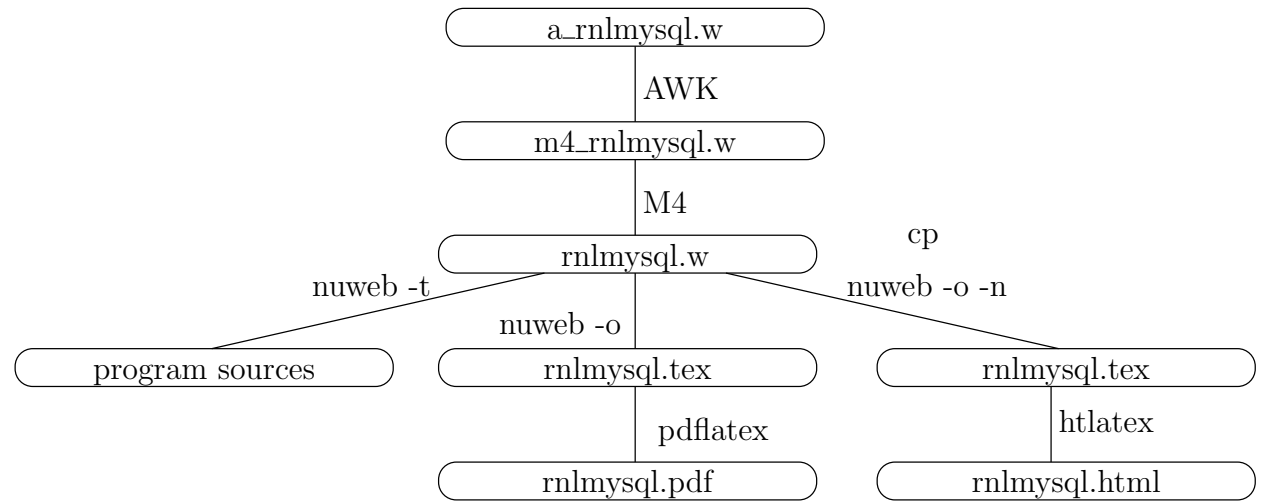


Figure 2: Translation of the raw code of this document into printable/viewable documents and into program sources. The figure shows the pathways and the main files involved.

< parameters in Makefile 61a > \equiv
 NUWEB=../env/bin/nuweb
 \diamond

Fragment defined by 61a, 62c, 64ab, 66d, 69a, 71d.
 Fragment referenced in 61b.
 Uses: nuweb 68b.

A.3 The Makefile for this project.

This chapter assembles the Makefile for this project.

```

"Makefile" 61b  $\equiv$ 
  < default target 61c >

  < parameters in Makefile 61a, ... >

  < impliciete make regels 65a, ... >
  < expliciete make regels 62d, ... >
  < make targets 62a, ... >
 $\diamond$ 

```

The default target of make is `all`.

```

< default target 61c >  $\equiv$ 
  all : < all targets 62b >
  .PHONY : all
 $\diamond$ 

```

Fragment referenced in 61b.
 Defines: `all` 32a, `PHONY` 65b.

```

< make targets 62a > ≡
    clean:
        < clean up 10f, ... >

```

◇

Fragment defined by 62a, 66ab, 69e, 72acd.
 Fragment referenced in 61b.

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

```

< all targets 62b > ≡
    nlpp.pdf◇

```

Fragment referenced in 61c.
 Uses: pdf 66a.

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

```

< parameters in Makefile 62c > ≡
    .SUFFIXES: .pdf .w .tex .html .aux .log .php

```

◇

Fragment defined by 61a, 62c, 64ab, 66d, 69a, 71d.
 Fragment referenced in 61b.
 Defines: SUFFIXES Never used.
 Uses: pdf 66a.

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.

```

< expliciete make regels 62d > ≡

    nuweb: $(NUWEB)

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

```

◇

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.
 Fragment referenced in 61b.
 Uses: nuweb 68b.

```

⟨ clean up 63a ⟩ ≡
    rm -rf ../nuweb-1.58
    ◇

```

Fragment defined by 10f, 11d, 24b, 63a.
 Fragment referenced in 62a.
 Uses: nuweb 68b.

```

⟨ expliciete make regels 63b ⟩ ≡
    ../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 62d, 63bcd, 65b, 67a, 69bd.
 Fragment referenced in 61b.
 Uses: nuweb 68b.

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 63c ⟩ ≡
    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 62d, 63bcd, 65b, 67a, 69bd.
 Fragment referenced in 61b.
 Uses: print 66a.

A.5.2 Run the M4 pre-processor

```

⟨ expliciete make regels 63d ⟩ ≡
    nlpp.w : m4_nlpp.w inst.m4
        m4 -P m4_nlpp.w > nlpp.w
    ◇

```

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.
 Fragment referenced in 61b.

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 64a> ≡
    FIGFILES=fileschema directorystructure
```

◇

Fragment defined by 61a, 62c, 64ab, 66d, 69a, 71d.

Fragment referenced in 61b.

Defines: FIGFILES 64b.

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 64b> ≡
    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 61a, 62c, 64ab, 66d, 69a, 71d.

Fragment referenced in 61b.

Defines: FIGFILENAMES Never used, PDFT_NAMES 66b, PDF_FIG_NAMES 66b, PST_NAMES Never used,
PS_FIG_NAMES Never used.

Uses: FIGFILES 64a.

Create the graph files with program `fig2dev`:


```

< implicate make regels 65a > ≡
    %.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 65a, 69c.

Fragment referenced in 61b.

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 65b > ≡
    bibfile : nlpp.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib nlpp litprog

    .PHONY : bibfile

```

◇

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.

Fragment referenced in 61b.

Uses: PHONY 61c.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

```

⟨ make targets 66a ⟩ ≡
    pdf : nlpp.pdf

    print : nlpp.pdf
           lpr nlpp.pdf

    view : nlpp.pdf
           evince nlpp.pdf

    ◇

```

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

Defines: pdf 62bc, 66b, print 12c, 14b, 16e, 22c, 33d, 50a, 63c, 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 66b ⟩ ≡
    nlpp.pdf : nlpp.w $(W2PDF) $(PDF_FIG_NAMES) $(PDFT_NAMES)
              chmod 775 $(W2PDF)
              $(W2PDF) $*

    ◇

```

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

Uses: pdf 66a, PDFT_NAMES 64b, PDF_FIG_NAMES 64b.

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 66c ⟩ ≡
    ../nuweb/bin ◇

```

Fragment defined by 6abcd, 10bh, 11a, 13e, 66c.

Fragment referenced in 72a.

Uses: nuweb 68b.

```

⟨ parameters in Makefile 66d ⟩ ≡
    W2PDF=../nuweb/bin/w2pdf

    ◇

```

Fragment defined by 61a, 62c, 64ab, 66d, 69a, 71d.

Fragment referenced in 61b.

Uses: nuweb 68b.

```

< expliciete make regels 67a > ≡
    $(W2PDF) : nlpp.w $(NUWEB)
              $(NUWEB) nlpp.w

```

◇

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.
 Fragment referenced in 61b.

```

"../nuweb/bin/w2pdf" 67b≡
    #!/bin/bash
    # w2pdf -- compile a nuweb file
    # usage: w2pdf [filename]
    # 20160107 at 1442h: Generated by nuweb from a_nlpp.w
    NUWEB=../env/bin/nuweb
    LATEXCOMPILER=pdflatex
    < filenames in nuweb compile script 67d >
    < compile nuweb 67c >

```

◇

Uses: nuweb 68b.

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 67c > ≡
    NUWEB=/home/phuijgen/nlp/nlpp/env/bin/nuweb
    < run the processors until the aux file remains unchanged 68c >
    < remove the copy of the aux file 68a >

```

◇

Fragment referenced in 67b.
 Uses: nuweb 68b.

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 67d > ≡
    nufil=$1
    trunk=${1%.*}
    texfil=${trunk}.tex
    auxfil=${trunk}.aux
    oldaux=old.${trunk}.aux
    indexfil=${trunk}.idx
    oldindexfil=old.${trunk}.idx

```

◇

Fragment referenced in 67b.
 Defines: auxfil 68c, 70c, 71a, indexfil 68c, 70c, nufil 68b, 70c, 71b, oldaux 68ac, 70c, 71a, oldindexfil 68c, 70c, texfil 68b, 70c, 71b, trunk 68b, 70c, 71bc.

Remove the old copy if it is no longer needed.

```

⟨ remove the copy of the aux file 68a ⟩ ≡
    rm $oldaux
    ◇

```

Fragment referenced in 67c, 70b.

Uses: oldaux 67d, 70c.

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 68b ⟩ ≡
    $NUWEB $nufil
    $LATEXCOMPILER $texfil
    makeindex $trunk
    bibtex $trunk
    ◇

```

Fragment referenced in 68c.

Defines: bibtex 71bc, makeindex 71bc, nuweb 6e, 57c, 61a, 62d, 63ab, 66cd, 67bc, 69a, 70a.

Uses: nufil 67d, 70c, texfil 67d, 70c, trunk 67d, 70c.

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 68c ⟩ ≡
    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 68b ⟩
        if [ $LOOPCOUNTER -ge 10 ]
        then
            cp $auxfil $oldaux
        fi;
    done
    ◇

```

Fragment referenced in 67c.

Uses: auxfil 67d, 70c, indexfil 67d, oldaux 67d, 70c, oldindexfil 67d.

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 69a> ≡
    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 61a, 62c, 64ab, 66d, 69a, 71d.

Fragment referenced in 61b.

Uses: nuweb 68b.

Make the directory:

```
<expliciete make regels 69b> ≡
    $(htmldir) :
        mkdir -p $(htmldir)
◇
```

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.

Fragment referenced in 61b.

The rule to copy files in it:

```
<impliciete make regels 69c> ≡
    $(htmldir)/% : % $(htmldir)
        cp $< $(htmldir)/
◇
```

Fragment defined by 65a, 69c.

Fragment referenced in 61b.

Do the work:

```
<expliciete make regels 69d> ≡
    $(htmltarget) : $(htmlmaterial) $(htmldir)
        cd $(htmldir) && chmod 775 w2html
        cd $(htmldir) && ./w2html nlpp.w
◇
```

Fragment defined by 62d, 63bcd, 65b, 67a, 69bd.

Fragment referenced in 61b.

Invoke:

```
<make targets 69e> ≡
    htm : $(htmldir) $(htmltarget)
◇
```

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

Create a script that performs the translation.

```
"w2html" 70a≡
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
# [filename]: Name of the nuweb source file.
# 20160107 at 1442h: Generated by nuweb from a_nlpp.w
echo "translate " $1 >w2html.log
NUWEB=/home/phuijgen/nlp/nlpp/env/bin/nuweb
⟨filenames in w2html 70c⟩

⟨perform the task of w2html 70b⟩
```

◇

Uses: **nuweb** 68b.

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 70b⟩ ≡
  ⟨run the html processors until the aux file remains unchanged 71a⟩
  ⟨remove the copy of the aux file 68a⟩
```

◇

Fragment referenced in 70a.

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 70c⟩ ≡
nufil=$1
trunk=${1%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
```

◇

Fragment referenced in 70a.

Defines: **auxfil** 67d, 68c, 71a, **nufil** 67d, 68b, 71b, **oldaux** 67d, 68ac, 71a, **texfil** 67d, 68b, 71b, **trunk** 67d, 68b, 71bc.

Uses: **indexfil** 67d, **oldindexfil** 67d.

```

⟨run the html processors until the aux file remains unchanged 71a⟩ ≡
    while
        ! cmp -s $auxfil $oldaux
    do
        if [ -e $auxfil ]
        then
            cp $auxfil $oldaux
        fi
        ⟨run the html processors 71b⟩
    done
    ⟨run tex4ht 71c⟩

```

◇

Fragment referenced in 70b.

Uses: auxfil 67d, 70c, oldaux 67d, 70c.

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

```

⟨run the html processors 71b⟩ ≡
    $NUWEB -o -n $nufil
    latex $texfil
    makeindex $trunk
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 71a.

Uses: bibtex 68b, makeindex 68b, nufil 67d, 70c, texfil 67d, 70c, trunk 67d, 70c.

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 71c⟩ ≡
    tex '\def\filename{{nlpp}{idx}{4dx}{ind}} \input idxmake.4ht'
    makeindex -o $trunk.ind $trunk.4dx
    bibtex $trunk
    htlatex $trunk

```

◇

Fragment referenced in 71a.

Uses: bibtex 68b, makeindex 68b, trunk 67d, 70c.

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 71d⟩ ≡
    MKDIR = mkdir -p

```

◇

Fragment defined by 61a, 62c, 64ab, 66d, 69a, 71d.

Fragment referenced in 61b.

Defines: MKDIR 72a.

```

< make targets 72a > ≡
    DIRS = < directories to create 6a, ... >

```

```

    $(DIRS) :
        $(MKDIR) $@

```

◇

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

Defines: DIRS 72c.

Uses: MKDIR 71d.

```

< make scripts executable 72b > ≡
    chmod -R 775 ../bin/*
    chmod -R 775 ../env/bin/*

```

◇

Fragment defined by 21d, 23c, 72b.

Fragment referenced in 72c.

```

< make targets 72c > ≡
    sources : nlpp.w $(DIRS) $(NUWEB)
              $(NUWEB) nlpp.w
              < make scripts executable 21d, ... >

```

◇

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

Uses: DIRS 72a.

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 72d > ≡
    transplant :
        touch a_nlpp.w
        $(MAKE) sources
        ../env/bin/transplant

```

◇

Fragment defined by 62a, 66ab, 69e, 72acd.

Fragment referenced in 61b.

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


```

"../env/bin/transplant" 73≡
    #!/bin/bash
    LOGLEVEL=1
    < set variables that point to the directory-structure 6e, ... >
    < set paths after transplantation 14b >
    < re-install modules after the transplantation 26d >

    ◇

```

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/constpars" Defined by 35e.
"../bin/coreference-base" Defined by 43d.
"../bin/dbpner" Defined by 55b.
"../bin/eSRL" Defined by 38f.
"../bin/evcoref" Defined by 54d.
"../bin/ewsd" Defined by 37d.
"../bin/factuality" Defined by 43a.
"../bin/FBK-causalrel" Defined by 42d.
"../bin/FBK-temprel" Defined by 41d.
"../bin/FBK-time" Defined by 40a.
"../bin/framesrl" Defined by 50a.
"../bin/heideltime" Defined by 52b.
"../bin/install-modules" Defined by 17a, 18aj, 19aj, 20ahq, 21a.
"../bin/langdetect" Defined by 23a.
"../bin/lu2synset" Defined by 47a.
"../bin/mor" Defined by 34b.
"../bin/ned" Defined by 48b.
"../bin/nedrer" Defined by 36a.
"../bin/nerc" Defined by 45d.
"../bin/nerc_conll02" Defined by 45b.
"../bin/nomevent" Defined by 55e.
"../bin/onto" Defined by 49a.
"../bin/opinimin" Defined by 57a.
"../bin/pos" Defined by 35b.
"../bin/postsrl" Defined by 54a.
"../bin/srl" Defined by 52e, 53abcd.
"../bin/srl-dutch-nominals" Defined by 39c.
"../bin/start_eSRL" Defined by 38b.
"../bin/stop_eSRL" Defined by 38d.
"../bin/test" Defined by 57c.
"../bin/tok" Defined by 33b.
"../bin/topic" Defined by 33e.
"../bin/ukb" Defined by 37a.

```

"../bin/wikify" Defined by 36d.
 "../bin/wsd" Defined by 46c.
 "../env/bin/langdetect.py" Defined by 22c.
 "../env/bin/progenv" Defined by 7b, 10a.
 "../env/bin/transplant" Defined by 73.
 "../env/etc/opini.cfg" Defined by 56b.
 "../nuweb/bin/w2pdf" Defined by 67b.
 "Makefile" Defined by 61b.
 "w2html" Defined by 70a.

C.2 Macro's

<abort when the language is not English or Dutch 23e> Referenced in 33be.
 <activate the install-to-project-repo utility 51d> Referenced in 51b.
 <activate the python environment 13d, 14a> Referenced in 12b, 17a.
 <all targets 62b> Referenced in 61c.
 <annotate dutch document 58a> Not referenced.
 <annotate english document 58b> Referenced in 57c.
 <begin conditional install 16b> Referenced in 10c, 17a, 18aj, 19aj, 20ahq, 21a.
 <check listener on host, port 29b> Referenced in 28b, 31ad.
 <check this first 9b, 21e> Referenced in 17a.
 <check whether mercurial is present 21f> Referenced in 21e.
 <check/install the correct version of python 12c> Referenced in 12b.
 <clean up 10f, 11d, 24b, 63a> Referenced in 62a.
 <clone the heideltime wrapper 50d> Referenced in 50c.
 <compile nuweb 67c> Referenced in 67b.
 <compile the heideltime wrapper 51b> Referenced in 50c.
 <compile the nerc jar 44b> Referenced in 44a.
 <create a virtual environment for Python 13b> Referenced in 12b.
 <create javapython script 9d> Referenced in 17a.
 <default target 61c> Referenced in 61b.
 <directories to create 6abcd, 10bh, 11a, 13e, 66c> Referenced in 72a.
 <download svm models 46b> Referenced in 45f.
 <else conditional install 16c> Not referenced.
 <end conditional install 16d> Referenced in 10c, 17a, 18aj, 19aj, 20ahq, 21a.
 <expliciete make regels 62d, 63bcd, 65b, 67a, 69bd> Referenced in 61b.
 <filenames in nuweb compile script 67d> Referenced in 67b.
 <filenames in w2html 70c> Referenced in 70a.
 <function to check/start spotlight 28b> Referenced in 57c.
 <get jvntextpro-2.0.jar 51c> Referenced in 51b.
 <get spotlight language parameters 29a> Referenced in 28b.
 <get spotlight model ball 27d> Referenced in 27a.
 <get the mor time-out parameter 34d> Referenced in 34b.
 <get the nerc models 45a> Referenced in 44a.
 <get the snapshot 9c> Referenced in 17a.
 <impliciete make regels 65a, 69c> Referenced in 61b.
 <install ActivePython 13a> Referenced in 12c.
 <install Alpino 23f> Referenced in 17a.
 <install boost 26e> Referenced in 17a.
 <install coreference-base 43c> Referenced in 20a.
 <install CRFsuite 32b> Referenced in 17a.
 <install from github 8d> Referenced in 34a, 39b, 43c, 45f, 47c, 50d, 52d, 55a.
 <install from tarball 59c> Not referenced.
 <install Java 1.6 11f> Referenced in 17a.
 <install kafnafparserpy 15a> Referenced in 12b.
 <install maven 11b> Referenced in 17a.
 <install python packages 15f, 53e> Referenced in 12b.
 <install sematree 22a> Referenced in 17a.

<install svm lib 46a> Referenced in 45f.
 <install SVMLight 32a> Referenced in 17a.
 <install the constituents parser 35d> Referenced in 18j.
 <install the dbpedia-ner module 55a> Referenced in 20q.
 <install the event-coreference module 54c> Referenced in 20h.
 <install the factuality module 42f> Referenced in 19j.
 <install the FBK-causalrel module 42a> Referenced in 19j.
 <install the FBK-temprel module 41a> Referenced in 19j.
 <install the FBK-time module 39e> Referenced in 19j.
 <install the heideltime module 50c> Referenced in 20h.
 <install the ims-wsd module 37c> Referenced in 19a.
 <install the lu2synset converter 46e> Referenced in 20h.
 <install the morphosyntactic parser 34a> Referenced in 18a.
 <install the NERC module 44a> Referenced in 18j.
 <install the nomevent module 55d> Referenced in 20q.
 <install the onto module 48d> Referenced in 20a.
 <install the opinion-miner 56ac> Referenced in 21a.
 <install the pos tagger 35a> Referenced in 18a.
 <install the post-SRL module 53f> Referenced in 20q.
 <install the Spotlight server 27a, 28a> Referenced in 17a.
 <install the srl module 52d> Referenced in 20h.
 <install the srl-dutch-nominals module 39b> Referenced in 19a.
 <install the srl-server module 38a> Referenced in 19a.
 <install the ticcutils utility 26a> Referenced in 17a, 26d.
 <install the timbl utility 26b> Referenced in 17a, 26d.
 <install the tokenizer 33a> Referenced in 18a.
 <install the topic analyser 33d> Referenced in 18a.
 <install the treetagger utility 24cde, 25abcd> Referenced in 17a.
 <install the UKB module 36f> Not referenced.
 <install the wikify module 36c> Referenced in 18j.
 <install the WSD module 45f> Referenced in 20a.
 <install the NED-reranker module 35g> Referenced in 18j.
 <install the NED module 47c> Referenced in 18j.
 <install VUA-pylib 31g> Referenced in 17a.
 <logmess 59b> Referenced in 8bd, 15a, 59c.
 <make scripts executable 21d, 23c, 72b> Referenced in 72c.
 <make targets 62a, 66ab, 69e, 72acd> Referenced in 61b.
 <move module 7c> Referenced in 8d, 15a, 59c.
 <parameters in Makefile 61a, 62c, 64ab, 66d, 69a, 71d> Referenced in 61b.
 <perform the task of w2html 70b> Referenced in 70a.
 <put Antske's material in the heideltime wrapper 51a> Referenced in 50c.
 <put spotlight jar in the Maven repository 48a> Referenced in 47c.
 <re-install modules after the transplantation 26d> Referenced in 73.
 <re-instate old module 8b> Referenced in 8d, 15a, 59c.
 <read the list of installed modules 16a> Referenced in 17a.
 <remove installed-variable 16e> Referenced in 11d.
 <remove old module 8a> Referenced in 8d, 15a, 59c.
 <remove outdated heideltime jars 52a> Referenced in 51d.
 <remove the copy of the aux file 68a> Referenced in 67c, 70b.
 <repair causalrel's run.sh.hadoop 42c> Not referenced.
 <repair FBK-*rel's run.sh.hadoop 41c> Referenced in 41a, 42a.
 <run tex4ht 71c> Referenced in 71a.
 <run the html processors 71b> Referenced in 71a.
 <run the html processors until the aux file remains unchanged 71a> Referenced in 70b.
 <run the processors until the aux file remains unchanged 68c> Referenced in 67c.
 <run the three processors 68b> Referenced in 68c.
 <select language-dependent features 44c> Referenced in 57c.
 <set alpinohome 24a> Referenced in 34b.

< set paths after transplantation [14b](#) > Referenced in [73](#).
 < set the language variable [23d](#) > Referenced in [57c](#).
 < set up java [10cg](#) > Referenced in [17a](#).
 < set up Java 1.6 [12a](#) > Referenced in [37d](#).
 < set up python [12b](#) > Referenced in [17a](#).
 < set variables that point to the directory-structure [6e](#), [7a](#), [9a](#), [11c](#) > Referenced in [7b](#), [17a](#), [73](#).
 < start EHU SRL server if it isn't running [38h](#) > Referenced in [38b](#).
 < start of module-script [22b](#) > Referenced in [23a](#), [33be](#), [34b](#), [35be](#), [36ad](#), [37ad](#), [38bdf](#), [39c](#), [40a](#), [41d](#), [42d](#), [43ad](#),
 [45bd](#), [46c](#), [47a](#), [48b](#), [49a](#), [50a](#), [52be](#), [54ad](#), [55be](#), [57a](#).
 < start the Spotlight server on localhost [31a](#) > Referenced in [28b](#), [30a](#).
 < stop EHU SRL server [39a](#) > Referenced in [38d](#).
 < test whether spotlightserver runs [30e](#) > Referenced in [30a](#).
 < test whether virtualenv is present on the host [13c](#) > Referenced in [13b](#).
 < try to obtain a running spotlightserver [30a](#) > Not referenced.
 < unpack ticcutils or timbl [26c](#) > Referenced in [26ab](#).
 < variables of install-modules [59a](#) > Referenced in [17a](#).
 < wait until the spotlight server is up [31d](#) > Referenced in [31a](#).

C.3 Variables

activate: [13d](#), [14b](#).
 all: [32a](#), [61c](#).
 ALPINO_HOME: [24a](#).
 auxfil: [67d](#), [68c](#), [70c](#), [71a](#).
 bibtex: [68b](#), [71bc](#).
 DIRS: [72a](#), [72c](#).
 fig2dev: [65a](#).
 FIGFILENAMES: [64b](#).
 FIGFILES: [64a](#), [64b](#).
 hg: [21f](#), [43c](#).
 indexfil: [67d](#), [68c](#), [70c](#).
 lang: [22c](#), [23d](#).
 lxml: [15f](#).
 makeindex: [68b](#), [71bc](#).
 MKDIR: [71d](#), [72a](#).
 nufil: [67d](#), [68b](#), [70c](#), [71b](#).
 nuweb: [6e](#), [57c](#), [61a](#), [62d](#), [63ab](#), [66cd](#), [67bc](#), [68b](#), [69a](#), [70a](#).
 oldaux: [67d](#), [68ac](#), [70c](#), [71a](#).
 oldindexfil: [67d](#), [68c](#), [70c](#).
 PATH: [7a](#), [10g](#), [11c](#), [12a](#), [43a](#), [57a](#).
 pdf: [62bc](#), [66a](#), [66b](#).
 PDFT_NAMES: [64b](#), [66b](#).
 PDF_FIG_NAMES: [64b](#), [66b](#).
 PHONY: [61c](#), [65b](#).
 print: [12c](#), [14b](#), [16e](#), [22c](#), [33d](#), [50a](#), [63c](#), [66a](#).
 PST_NAMES: [64b](#).
 PS_FIG_NAMES: [64b](#).
 pythonok: [12c](#).
 PYTHONPATH: [14a](#).
 pyyaml: [15f](#).
 rdflib: [53e](#).
 SUFFIXES: [62c](#).
 texfil: [67d](#), [68b](#), [70c](#), [71b](#).
 trunk: [67d](#), [68b](#), [70c](#), [71bc](#).
 view: [66a](#).
 virtualenv: [13ab](#), [13c](#).