Install Dutch nlp modules on Lisa

Paul Huygen <paul.huygen@huygen.nl>

27th January 2015
15:34 h.

Abstract

This is a description and documentation of the installation of the current NLP modules on Lisa, so that they can be used in pipelines.

Contents

1 Introduction

This document describes the current set-up of pipeline that annotates dutch texts in order to extract knowledge. The pipeline has been set up by the Computational Lexicology an Terminology Lab (CLTL [1]) as part of the newsreader [2].

Apart from describing the pipeline set-up, the document actually constructs the pipeline. The described version has been made with an aim to run it on a specific supercomputer (Lisa, Surfsara, Amsterdam [3]), but it can probably be implemented on other unix-like systems without problems.

The installation has been parameterized. 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.

1. `http://wordpress.let.vupr.nl`
2. http://www.newsreader-project.eu
3. https://surfsara.nl/systems/lisa

1.1 List of the modules to be installed

Table 1 lists the modules in the pipeline. The column *source* indicates the origin of the module.

module	**directory**	**source**	**script**	**Details**
Tokenizer	`ixa-pipe-tok`	EHU	tok	
morphosyntactic parser	`morphosyntactic_parser_nl`	Github	`mor`	
alpinohack	`clean_hack`	This doc.	alpinohack	4
NERC	`../env/java/jars`	TAR	nerc	
WSD	`svm_wsd`	TAR	wsd	
Onto	`vua-ontotagger-v1.0`	TAR	onto	
Heidel	`NAF-HeidelTime`	Github	heideltime	
SRL	`vua-srl-nl`	Github	srl	
NED	`ixa-pipe-ned`	EHU	ned	
Nom. coref	`/dev/null`	None	nomcoref	
Ev. coref	`/dev/null`	None	evcoref	
Opinion miner	`/dev/null`	None	opinimin	
Framenet sem. role label.	`/dev/null`	None	fsrl	

Table 1: 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; **script:** Script to be included in a pipeline.

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 are available from the dedicated repository on `u017940.si.ehu.es`. A username and password are needed to access these modules. This is indicated as EHU.
3. Some modules have not been officially published in a repository or the repositrory is not yet known by the author. These modules have been packed in a tar-ball that can be obtained by the author. This is indicated as TAR.

The modules themselves use other utilities like dependency-taggers and POS taggers. These utilities are listed in table 2.

module	**directory**	**source**	**Details**
KafNafParserPy	`python/KafNafParserPy`	Github	
Alpino	`Alpino`	RUG	
Ticcutils	`ticcutils-0.7`	ILK	
Timbl	`timbl-6.4.6`	ILK	
Treetagger			

Table 2: 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.

Table 3 lists the source of the modules and utilities that can be installed from an open source.

module	**source**	**URL**
Tokenizer	Github	https://github.com/ixa-ehu/ixa-pipe-tok.git
Morphosynt. p.	Github	`https://github.com/cltl/morphosyntactic_parser_nl.git`
heideltime.	Github	`https://github.com/cltl/morphosyntactic_parser_nl.git`
Alpino	RUG	`Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz`
Ticcutils	ILK	ticcutils-0.7.tar.gz
Timble	ILK	timbl-6.4.6.tar.gz

Table 3: Sources of the modules

1.2 File-structure of the pipeline

The files that make up the pipeline are organised in set of directories:

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

modules: Contains the program code of each module in a subdirectory. Furthermore, it contains a subdirectory `python` for python software-modules, subdirectory `jars` for jar files and subdirectory /usrlocal/ for binaries and libs that are used by modules.

bin: Contains for each of the 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.

nuweb: Contains this document, the nuweb source that creates the documents and the sources and a Makefile to perform the actions.

⟨ *directories to create* 4a ⟩ ≡
```
        ../modules ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *directories to create* 4b ⟩ ≡
```
        ../bin ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *directories to create* 4c ⟩ ≡
```
        ../modules/usrlocal ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *directories to create* 4d ⟩ ≡
```
        ../modules/usrlocal/bin ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *directories to create* 4e ⟩ ≡
```
        ../modules/usrlocal/lib ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *directories to create* 4f ⟩ ≡
```
        ../modules/python ../env/java/jars ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

Make binaries findable:

⟨ *set local bin directory* 4g ⟩ ≡
```

        export PATH=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/usrlocal/bin:$PATH
        ◇
```
Fragment referenced in 23a.

2 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 file sets up the programming environment in scripts.

"../bin/progenv" 5a≡
```
PIPEROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
PIPEBIN=$PIPEROOT/bin
PIPEMODD=$PIPEROOT/modules
⟨ set up java environment in scripts 6b, ... ⟩
⟨ activate the python environment 8a, ... ⟩
◇
```

⟨ *set up programming environment* 5b ⟩ ≡
```
source /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/bin/progenv
◇
```
Fragment referenced in 13bh, 15e, 16c, 17g, 21a.

2.1 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 `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-`

⟨ *directories to create* 5c ⟩ ≡
```
../env/java ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *check this first* 5d ⟩ ≡
```
if
  [ ! -e /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/server-jre-7u72-linux-x64.tar.
then
  echo "Cannot find  /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/server-jre-7u72-li
  exit 4
fi
◇
```
Fragment referenced in 11a.

⟨ *set up java* 5e ⟩ ≡
```
⟨ unpack the java tarball 6a ⟩
◇
```
Fragment referenced in 11a.

⟨ *unpack the java tarball* 6a ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java
tar -xzf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/server-jre-7u72-linux-x64.tar.
◇
```
Fragment referenced in 5e.

⟨ *set up java environment in scripts* 6b ⟩ ≡
```

export JAVA_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java/jdk1.7.0_72
export PATH=$JAVA_HOME/bin:$PATH
◇
```
Fragment defined by 6bd.
Fragment referenced in 5a, 11a.
Defines: `JAVA_HOME` Never used.

Put jars in the jar subdirectory of the java directory:

⟨ *directories to create* 6c ⟩ ≡
```
../env/java/jars ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *set up java environment in scripts* 6d ⟩ ≡
```

export JARDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java/jars
◇
```
Fragment defined by 6bd.
Fragment referenced in 5a, 11a.

2.2 Maven

⟨ *directories to create* 6e ⟩ ≡
```

/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/apache-maven-3.0.5 ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

⟨ *install maven* 6f ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env
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 defined by 6f, 7a.
Fragment referenced in 11a.

⟨ *install maven* 7a ⟩ ≡

```
export MAVEN_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/apache-maven-3.0.
export PATH=${MAVEN_HOME}/bin:${PATH}
◇
```

Fragment defined by 6f, 7a.
Fragment referenced in 11a.

⟨ *remove maven* 7b ⟩ ≡

```
rm -rf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/apache-maven-3.0.5
◇
```

Fragment never referenced.

2.3 Python

⟨ *set up python* 7c ⟩ ≡
  ⟨ *create a virtual environment for Python* 7d ⟩
  ⟨ *activate the python environment* 8a, ... ⟩
  ⟨ *install kafnafparserpy* 8d ⟩
  ⟨ *install python packages* 9a ⟩
  ◇

Fragment referenced in 11a.

2.3.1 Virtual environment

Create a virtual environment.

⟨ *create a virtual environment for Python* 7d ⟩ ≡
  ⟨ *test whether virtualenv is present on the host* 7e ⟩

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env
virtualenv venv
◇
```

Fragment referenced in 7c.
Uses: `virtualenv` 7e.

⟨ *test whether virtualenv is present on the host* 7e ⟩ ≡

```
which virtualenv
if
  [ $? -ne 0 ]
then
  echo Please install virtualenv
  exit 1
fi
◇
```

Fragment referenced in 7d.
Defines: `virtualenv` 7d.

⟨ *activate the python environment* 8a ⟩ ≡

```
source /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/venv/bin/activate
◇
```

Fragment defined by 8ac.
Fragment referenced in 5a, 7c, 22b, 23a.
Defines: `activate` Never used.

Subdirectory `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python` will contain general Python packages like KafnafParserPy.

⟨ *directories to create* 8b ⟩ ≡

```
/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python ◇
```

Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

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

⟨ *activate the python environment* 8c ⟩ ≡

```
export PYTHONPATH=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python:$PYTHONPAT
◇
```

Fragment defined by 8ac.
Fragment referenced in 5a, 7c, 22b, 23a.
Defines: `PYTHONPATH` Never used.

2.3.2 KafNafParserPy

A cornerstone Pythonmodule for the pipeline is KafNafParserPy. It is a feature of this module that it cannot be installed with PIP, but that you can put it somewhere and then put the somewhere in your PYTHONPATH.

⟨ *install kafnafparserpy* 8d ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/python
DIRN=KafNafParserPy
⟨ move module (8e $DIRN ) 9b ⟩
git clone https://github.com/cltl/KafNafParserPy.git
if
  [ $? -gt 0 ]
then
  ⟨ logmess (8f Cannot install current $DIRN version ) 27e ⟩
  ⟨ re-instate old module (8g $DIRN ) 9d ⟩
else
  ⟨ remove old module (8h $DIRN ) 9c ⟩
fi
◇
```

Fragment referenced in 7c.

2.3.3 Python packages

Install python packages:

lxml:
pyyaml: for coreference-graph

⟨ *install python packages* 9a ⟩ ≡
```
pip install lxml
pip install pyyaml
◇
```
Fragment referenced in 7c.
Defines: `lxml` Never used, `pyyaml` Never used.

3 Installation

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

3.1 Installing vs. updating

When the install-script installs something that has already been installed, it moves the installed module to a temporary location and then tries to install the module from its source. If that is successfull it removes the vormer version of the module, otherwise it moves the old version back.

The following macro's can be used to move or remove modules, provided they are called when the modules directory is the default directory.

⟨ *move module* 9b ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
◇
```
Fragment referenced in 8d, 10a, 12a, 28a.

⟨ *remove old module* 9c ⟩ ≡
```
rm -rf old.@1
◇
```
Fragment referenced in 8d, 10a, 12a, 28a.

⟨ *re-instate old module* 9d ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
⟨ logmess (9e $MESS ) 27e ⟩

◇
```
Fragment referenced in 8d, 10a, 12a, 28a.

3.2 Installation from Github

The following macro can be used to install a module from github. It needs as parameters:

1. Name of the module.
2. Name of the root directory.
3. Github URL to clone from.

⟨ *install from github* 10a ⟩ ≡
```
MODNAM=@1
DIRN=@2
GITU=@3
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (10b $DIRN ) 9b ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (10c Cannot install current $MODNAM version ) 27e ⟩
  ⟨ re-instate old module (10d $DIRN ) 9d ⟩
else
  ⟨ remove old module (10e $DIRN ) 9c ⟩
fi

◇
```
Fragment referenced in 13d, 15a, 17a, 20a, 21c, 22d.

3.3 Installation from EHU

Some of the modules cannot be easily obtained available on Github, but there is a complete package on EHU.

3.4 Installation from the snapshot

For some modules a public repository is not available or not known. They must be installed from a tarball with snapshots that can be obtained from the author. Let us first check whether we have the snapshot and complain if we don't. We expect the file `/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-`

⟨ *unpack snapshots or die* 10f ⟩ ≡
```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
if
  [ -e nl-pipeline_snapshots_20150127.tgz ]
then
  tar -zxf nl-pipeline_snapshots_20150127.tgz
fi
if
  [ ! -e snapshots ]
then
  echo "No module snapshots"
  exit 1
fi
◇
```
Fragment referenced in 11a.

3.5 The installation script

The installation is performed by script `install-modules`

```
"../bin/install-modules" 11a≡
        #!/bin/bash
        echo Set up environment
        ⟨ variables of install-modules 27d ⟩
        ⟨ check this first 5d ⟩
        ⟨ unpack snapshots or die 10f ⟩
        echo ... Java
        ⟨ set up java 5e ⟩
        ⟨ set up java environment in scripts 6b, ... ⟩
        ⟨ install maven 6f, ... ⟩
        echo ... Python
        ⟨ set up python 7c ⟩
        echo ... Alpino
        ⟨ install Alpino 12a ⟩
        ⟨ install the spotlight server 19a, ... ⟩
        ⟨ install the treetagger utility 24c, ... ⟩
        ⟨ install the ticcutils utility 26b ⟩
        ⟨ install the timbl utility 26c ⟩
        echo Tokenizer
        ⟨ install the tokenizer 13a ⟩
        echo Morphosyntactic parser
        ⟨ install the morphosyntactic parser 13d ⟩
        ⟨ install the NERC module 15g ⟩
        ⟨ install coreference-base 15a ⟩
        ⟨ install the WSD module 17a ⟩
        ⟨ install the onto module 20g ⟩
        ⟨ install the heideltime module 21c ⟩
        ⟨ install the srl module 22d ⟩
        ⟨ install the NED module 20a ⟩

        ◇
```

```
⟨ make scripts executable 11b ⟩ ≡
        chmod 775  ../bin/install-modules
        ◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.6 Install utilities and resources

3.6.1 Alpino

Install Alpino from the website of Gertjan van Noort.

Module

⟨ *install Alpino* 12a ⟩ ≡
```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (12b Alpino ) 9b ⟩
wget http://www.let.rug.nl/vannoord/alp/Alpino/binary/versions/Alpino-x86_64-linux-glibc2.5-20548-sic
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz
  SUCCES=$?
  rm -rf Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (12c Installed Alpino ) 27e ⟩
  ⟨ remove old module (12d Alpino ) 9c ⟩
else
  ⟨ re-instate old module (12e Alpino ) 9d ⟩
fi
◇
```
Fragment referenced in 11a.

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* 12f ⟩ ≡
```

export ALPINO_HOME=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/Alpino
◇
```
Fragment referenced in 13h.
Defines: `ALPINO_HOME` Never used.

3.7 Install modules

3.7.1 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.

Not yet included in this script is the set-up of an environment to use the specified version of Java (Oracle 1.7) and Maven (3). For now, we assume that it is there. This is a todo item.

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* 13a ⟩ ≡
```
tempdir=‘mktemp -d -t tok.XXXXXX‘
cd $tempdir
git clone https://github.com/ixa-ehu/ixa-pipe-tok.git
cd ixa-pipe-tok
mvn clean package
mv target/ixa-pipe-tok-1.6.6.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/ja
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
◇
```
Fragment referenced in 11a.

Script The script runs the tokenizerscript.

"../bin/tok" 13b≡
```
#!/bin/bash
⟨ set up programming environment 5b ⟩
JARFILE=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java/jars/ixa-pipe-tok-1.6.
java -jar $JARFILE tok -l nl --inputkaf
◇
```

⟨ *make scripts executable* 13c ⟩ ≡
```
chmod 775  ../bin/tok
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.7.2 Morphosyntactic parser

Module

⟨ *install the morphosyntactic parser* 13d ⟩ ≡

```
⟨ install from github (13e morphsynparser,13f morphosyntactic_parser_nl,13g https://github.com/cltl/morphosyn
◇
```
Fragment referenced in 11a.

Script

"../bin/mor" 13h≡
```
#!/bin/bash
⟨ set up programming environment 5b ⟩
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
MODDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/morphosyntactic_parser_n
⟨ set alpinohome 12f ⟩
cat | python $MODDIR/core/morph_syn_parser.py
◇
```

⟨ *make scripts executable* 14a ⟩ ≡
```
chmod 775  ../bin/mor
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.7.3 Alpino hack

Install a hack that removes output from Alpino that cannot be interpreted by following modules. It is just a small python script. Actually, it may no longer be necessary.

Module

⟨ *directories to create* 14b ⟩ ≡
```
../modules/alpinohack ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.

"../modules/alpinohack/clean_hack.py" 14c≡
```
#!/usr/bin/python
import sys

input = sys.stdin

output = ''

for line in input:
    line = line.replace('"--','"#')
    line = line.replace('--"','#"')
    output += line

print output

◇
```
Uses: print 34a.

Script

"../bin/alpinohack" 14d≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
HACKDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/alpinohack
cat | python  $HACKDIR/clean_hack.py

◇
```

⟨ *make scripts executable* 14e ⟩ ≡
```
chmod 775  ../bin/alpinohack
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.7.4 Nominal coreference-base

Get this thing from Github (`https://github.com/opener-project/coreference-base/`) and use the instruction of `https://github.com/opener-project/coreference-base/blob/master/core/README.md`.

Module

⟨ *install coreference-base* 15a ⟩ ≡

```
        ⟨ install from github (15b coreference-base,15c coreference-base,15d https://github.com/opener-project/corefe
        pip install --upgrade  hg+https://bitbucket.org/Josu/pykaf#egg=pykaf
        pip install --upgrade  networkx
        ◇
```

Fragment referenced in 11a.

Script

"../bin/coreference-base" 15e≡

```
        #!/bin/bash
        ⟨ set up programming environment 5b ⟩
        cd $PIPEMODD/coreference-base/core
        cat | python -m corefgraph.process.file --language nl --singleton --sieves NO
        ◇
```

⟨ *make scripts executable* 15f ⟩ ≡

```
        chmod 775  ../bin/coreference-base
        ◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.7.5 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, the Nerc module of the standard English pipeline, that is not yet public available, has been put in the snapshot-tarball.

⟨ *install the NERC module* 15g ⟩ ≡

```
        ⟨ compile the nerc jar 16a ⟩
        ⟨ get the nerc models 16b ⟩

        cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/EHU-nerc  /home/paul/pr
        ◇
```

Fragment referenced in 11a.

⟨ *compile the nerc jar* 16a ⟩ ≡

```
TEMPDIR==‘mktemp -d -t nerc.XXXXXX‘
cd $TEMPDIR
git clone https://github.com/ixa-ehu/ixa-pipe-nerc
cd ixa-pipe-nerc/
mvn clean package
mv target/ixa-pipe-nerc-1.3.3.jar /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/j
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/nuweb
rm -rf $TEMPDIR
◇
```

Fragment referenced in 15g.
Uses: nuweb 36b.

⟨ *get the nerc models* 16b ⟩ ≡

```
mkdir -p ../modules/EHU-nerc
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/EHU-nerc/nerc-resources
◇
```

Fragment referenced in 15g.

Script

"../bin/nerc" 16c≡

```
#!/bin/bash
⟨ set up programming environment 5b ⟩
MODDIR=$PIPEMODD/EHU-nerc
JAR=$JARDIR/ixa-pipe-nerc-1.3.3.jar
MODEL=nl-local-conll02-testa.bin
cat | java -jar $JAR tag -m $MODDIR/nerc-resources/nl/$MODEL
◇
```

⟨ *make scripts executable* 16d ⟩ ≡

```
chmod 775  ../bin/nerc
◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.8 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.

3.8.1 Module

⟨ *install the WSD module* 17a ⟩ ≡
```
        ⟨ install from github (17b wsd,17c svm_wsd,17d https://github.com/cltl/svm_wsd.git ) 10a ⟩
        cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/svm_wsd
        ⟨ install svm lib 17e ⟩
        ⟨ download svm models 17f ⟩



        ◇
```
Fragment referenced in 11a.

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

⟨ *install svm lib* 17e ⟩ ≡
```
        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 17a.

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

⟨ *download svm models* 17f ⟩ ≡
```
        cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/svm_wsd
        echo 'Downloading models...(could take a while)'
        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 17a.

3.8.2 Script

"../bin/wsd" 17g≡
```
        #!/bin/bash
        # WSD -- wrapper for word-sense disambiguation
        # 8 Jan 2014 Ruben Izquierdo
        # 16 sep 2014 Paul Huygen
        ⟨ set up programming environment 5b ⟩
        WSDDIR=$PIPEMODD/svm_wsd
        WSDSCRIPT=dsc_wsd_tagger.py
        cat | python $WSDDIR/$WSDSCRIPT --naf
        ◇
```

⟨ *make scripts executable* 18a ⟩ ≡

```
chmod 775  ../bin/wsd
◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

2

3.9 Lexical-unit converter

3.9.1 Module

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

⟨ *install the lu2synset converter* 18b ⟩ ≡

```

cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/lexicalunitconvertor /h
◇
```

Fragment never referenced.

3.9.2 Script

"../bin/lu2synset" 18c≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
JAVALIBDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/lexicalunitconvertor
RESOURCESDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/lexicalunitconvert
JARFILE=WordnetTools-1.0-jar-with-dependencies.jar
java -Xmx812m -cp  $JAVALIBDIR/$JARFILE vu.wntools.util.NafLexicalUnitToSynsetReferences \
   --wn-lmf "$RESOURCESDIR/cornetto2.1.lmf.xml" --format naf
◇
```

3.10 Spotlight

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.ja`
> 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 dbPeadia server: Italian server:
>
> ```
> java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar it http://local
> ```

Dutch server:
`java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://local`

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

⟨ *install the spotlight server* 19a ⟩ ≡

```
mkdir -p /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
cp ../snapshots/spotlight/dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar .
◇
```

Fragment defined by 19ab.
Fragment referenced in 11a.

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

⟨ *install the spotlight server* 19b ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/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 19ab.
Fragment referenced in 11a.

⟨ *start the spotlight server* 19c ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/spotlight
java -jar -Xmx8g dbpedia-spotlight-0.7-jar-with-dependencies-candidates.jar nl http://localhost:2060/
◇
```

Fragment referenced in 19d.

⟨ *check/start the spotlight server* 19d ⟩ ≡

```
spottasks=‘netstat -an | grep :2060 | wc -l‘
if
  [ $spottasks -eq 0 ]
then
  ⟨ start the spotlight server 19c ⟩
  sleep 60
fi
◇
```

Fragment referenced in 20e.

3.11 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 NER shuld 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.

3.11.1 Module

⟨ *install the* NED *module* 20a ⟩ ≡

```
⟨ install from github (20b ned,20c ixa-pipe-ned,20d https://github.com/ixa-ehu/ixa-pipe-ned.git ) 10a ⟩
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/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 /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/ja
◇
```

Fragment referenced in 11a.

3.11.2 Script

"../bin/ned" 20e≡

```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
JARDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/env/java/jars
⟨ check/start the spotlight server 19d ⟩
cat | java -jar $JARDIR/ixa-pipe-ned-1.1.1.jar  -p 2060 -e candidates -i /home/paul/projecten/cltl/pi
◇
```

⟨ *make scripts executable* 20f ⟩ ≡

```
chmod 775  ../bin/ned
◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.12 Ontotagger

We do not yet have a source-repository of the Ontotagger module. Therefore, install from a snapshot (`vua-ontotagger-v1.0.tar.gz`).

3.12.1 Module

⟨ *install the onto module* 20g ⟩ ≡

```
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
tar -xzf /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/vua-ontotagger-v1.0.
cp -r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/snapshots/vua-ontotagger-v1.0 /ho
chmod -R o+r /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
◇
```

Fragment referenced in 11a.

3.12.2 Script

"../bin/onto" 21a≡

```
#!/bin/bash
⟨ set up programming environment 5b ⟩
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
ONTODIR=$PIPEMODD/vua-ontotagger-v1.0
JARDIR=$ONTODIR/lib
RESOURCESDIR=$ONTODIR/resources
PREDICATEMATRIX="$RESOURCESDIR/PredicateMatrix_nl_lu_withESO.v0.2.role.txt"
GRAMMATICALWORDS="$RESOURCESDIR/grammaticals/Grammatical-words.nl"
TMPFIL=‘mktemp -t stap6.XXXXXX‘
cat >$TMPFIL

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

JAVA_ARGS="--mappings \"fn;mcr;ili;eso\" "
JAVA_ARGS="$JAVA_ARGS  --key odwn-eq"
JAVA_ARGS="$JAVA_ARGS  --version 1.1"
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

◇
```

⟨ *make scripts executable* 21b ⟩ ≡

```
chmod 775  ../bin/onto
◇
```

Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.13 Heideltime

3.13.1 Module

⟨ *install the heideltime module* 21c ⟩ ≡

```

⟨ install from github (21d heideltime,21e NAF-HeidelTime,21f git@github.com:PaulHuygen/NAF-HeidelTime.git  ) 1
⟨ adapt heideltime's config.props 22a ⟩

◇
```

Fragment referenced in 11a.

⟨ *adapt heideltime's config.props* 22a ⟩ ≡
```
CONFIL=NAF-HeidelTime/config.props
tempfil=`mktemp -t heideltmp.XXXXXX`
mv $CONFIL $tempfil
MODDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
TREETAGDIR=treetagger
AWKCOMMAND='/^treeTaggerHome/ {$0="treeTaggerHome = /home/paul/projecten/cltl/pipelines/dutch-nlp-mod
gawk "$AWKCOMMAND" $tempfil >$CONFIL
◇
```
Fragment referenced in 21c.
Uses: `print` 34a.

3.13.2 Script

`"../bin/heideltime"` 22b≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
HEIDELDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/NAF-HeidelTime
TEMPDIR=`mktemp -t -d heideltmp.XXXXXX`
cd $HEIDELDIR
⟨ activate the python environment 8a, ... ⟩
iconv -t utf-8//IGNORE | python $HEIDELDIR/HeidelTime_NafKaf.py $HEIDELDIR/heideltime-standalone/ $TE
◇
```

⟨ *make scripts executable* 22c ⟩ ≡
```
chmod 775  ../bin/heideltime
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

3.14 Semantic Role labelling

3.14.1 Module

⟨ *install the srl module* 22d ⟩ ≡

⟨ *install from github* (22e `srl`,22f `vua-srl-nl`,22g `https://github.com/newsreader/vua-srl-nl.git`) 10a ⟩
◇

Fragment referenced in 11a.

3.14.2 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" 23a≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
SRLDIR=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/vua-srl-nl
TEMPDIR=‘mktemp -d -t SRLTMP.XXXXXX‘
cd $SRLDIR
⟨ set local bin directory 4g ⟩
⟨ activate the python environment 8a, ... ⟩
INPUTFILE=$TEMPDIR/inputfile
FEATUREVECTOR=$TEMPDIR/csvfile
TIMBLOUTPUTFILE=$TEMPDIR/timblpredictions
◇
```
File defined by 23abcde.

Create a feature-vector.

"../bin/srl" 23b≡
```
cat | tee  $INPUTFILE | python nafAlpinoToSRLFeatures.py > $FEATUREVECTOR
◇
```
File defined by 23abcde.

Run the trained model on the feature-vector.

"../bin/srl" 23c≡
```

timbl -mO:I1,2,3,4 -i e-mags_mags_press_newspapers.wgt -t $FEATUREVECTOR -o $TIMBLOUTPUTFILE >/dev/nu
◇
```
File defined by 23abcde.

Insert the SRL values into the NAF file.

"../bin/srl" 23d≡
```
python timblToAlpinoNAF.py $INPUTFILE $TIMBLOUTPUTFILE
◇
```
File defined by 23abcde.

Clean up.

"../bin/srl" 23e≡
```
rm -rf $TEMPDIR
◇
```
File defined by 23abcde.

⟨ *make scripts executable* 23f ⟩ ≡
```
chmod 775  ../bin/srl
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

4 Utilities

4.1 Test script

The following script pushes a single sentence through the modules of the pipeline.

"../bin/test" 24a≡
```
#!/bin/bash
ROOT=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
TESTDIR=$ROOT/test
BIND=$ROOT/bin
mkdir -p $TESTDIR
cd $TESTDIR
cat $ROOT/nuweb/testin.naf | $BIND/tok > $TESTDIR/test.tok.naf
cat test.tok.naf | $BIND/mor > $TESTDIR/test.mor.naf
cat test.mor.naf | $BIND/nerc > $TESTDIR/test.nerc.naf
cat $TESTDIR/test.nerc.naf | $BIND/wsd > $TESTDIR/test.wsd.naf
cat $TESTDIR/test.wsd.naf | $BIND/ned  > $TESTDIR/test.ned.naf
cat $TESTDIR/test.ned.naf | $BIND/onto > $TESTDIR/test.onto.naf
cat $TESTDIR/test.onto.naf | $BIND/heideltime > $TESTDIR/test.times.naf
cat $TESTDIR/test.times.naf | $BIND/srl  > $TESTDIR/test.srl.naf

◇
```
Uses: nuweb 36b.

⟨ *make scripts executable* 24b ⟩ ≡
```
chmod 775  ../bin/test
◇
```
Fragment defined by 11b, 13c, 14ae, 15f, 16d, 18a, 20f, 21b, 22c, 23f, 24b.
Fragment referenced in 40c.

4.2 Treetagger

4.2.1 Module

Installation 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 treetager 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 24c, 25abcde, 26a.
Fragment referenced in 11a.

The source tarball, scripts and the installation-script:

⟨ *install the treetagger utility* 25a ⟩ ≡
```
TREETAGSRC=tree-tagger-linux-3.2.tar.gz
TREETAGSCRIPTS=tagger-scripts.tar.gz
TREETAG_INSTALLSCRIPT=install-tagger.sh
◇
```
Fragment defined by 24c, 25abcde, 26a.
Fragment referenced in 11a.

Parametersets:

⟨ *install the treetagger utility* 25b ⟩ ≡
```
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 24c, 25abcde, 26a.
Fragment referenced in 11a.

Download everything in the target directory:

⟨ *install the treetagger utility* 25c ⟩ ≡
```

mkdir -p /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$TREETAGDIR
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$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 24c, 25abcde, 26a.
Fragment referenced in 11a.

Run the install-script:

⟨ *install the treetagger utility* 25d ⟩ ≡
```
chmod 775 $TREETAG_INSTALLSCRIPT
./$TREETAG_INSTALLSCRIPT
◇
```
Fragment defined by 24c, 25abcde, 26a.
Fragment referenced in 11a.

Make the treetagger utilities available for everbody.

⟨ *install the treetagger utility* 25e ⟩ ≡
```

chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$TREETAGDIR/bin
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$TREETAGDIR/cmd
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$TREETAGDIR/doc
chmod o+x /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/$TREETAGDIR/lib
./$TREETAG_INSTALLSCRIPT
◇
```
Fragment defined by 24c, 25abcde, 26a.
Fragment referenced in 11a.

Remove the tarballs:

⟨ *install the treetagger utility* 26a ⟩ ≡

```
rm $TREETAGSRC
rm $TREETAGSCRIPTS
rm $TREETAG_INSTALLSCRIPT
rm $DUTCHPARS_UTF_GZ
rm $DUTCH_TAGSET
rm $DUTCHPARS_2_GZ
◇
```

Fragment defined by 24c, 25abcde, 26a.
Fragment referenced in 11a.

4.3 Timbl and ticcutils

4.3.1 Module

Timbl and ticcutils are installed from their source-tarballs. The installation is not (yet?) completely reproducibe because it uses the currently available c-compiler. 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 `usrlocal` subdirectory of the modules directory.

⟨ *install the ticcutils utility* 26b ⟩ ≡

```
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 27a ⟩
◇
```

Fragment referenced in 11a.

⟨ *install the timbl utility* 26c ⟩ ≡

```
URL=http://software.ticc.uvt.nl/timbl-6.4.6.tar.gz
TARB=timbl-6.4.6.tar.gz
DIR=timbl-6.4.6
⟨ unpack ticcutils or timbl 27a ⟩
◇
```

Fragment referenced in 11a.

⟨ *unpack ticcutils or timbl* 27a ⟩ ≡
```
SUCCES=0
ticbeldir=`mktemp -t -d tickbel.XXXXXX`
cd $ticbeldir
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  cd $DIR
  ./configure --prefix=/home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules/usrlocal
  make
  make install
fi
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa
rm -rf $ticbeldir
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (27b Installed $DIR ) 27e ⟩
else
  ⟨ logmess (27c NOT installed $DIR ) 27e ⟩
fi
◇
```
Fragment referenced in 26bc.

4.4 Logging

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

⟨ *variables of install-modules* 27d ⟩ ≡
```
LOGLEVEL=1
◇
```
Fragment referenced in 11a.

⟨ *logmess* 27e ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
◇
```
Fragment referenced in 8d, 9d, 10a, 12a, 27a, 28a.

4.5 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* 28a ⟩ ≡
```
SUCCES=0
cd /home/paul/projecten/cltl/pipelines/dutch-nlp-modules-on-Lisa/modules
⟨ move module (28b $DIR ) 9b ⟩
wget $URL
SUCCES=$?
if
  [ $SUCCES -eq 0 ]
then
  tar -xzf $TARB
  SUCCES=$?
  rm -rf $TARB
fi
if
  [ $SUCCES -eq 0 ]
then
  ⟨ logmess (28c Installed $DIR ) 27e ⟩
  ⟨ remove old module (28d $DIR ) 9c ⟩
else
  ⟨ re-instate old module (28e $DIR ) 9d ⟩
fi
◇
```
Fragment never referenced.

5 Testing

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 macroÂ´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_dutch-nlp-modules-on-Lisa.w`. Figure 1 shows pathways to

Figure 1: 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.

translate it into printable/viewable documents and to extract the program sources. Table 4 lists

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

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

A.3 Translate and run

This chapter assembles the Makefile for this project.

`"Makefile"` 29≡

⟨ *default target* 30a ⟩

⟨ *parameters in Makefile* 30c, ... ⟩

⟨ *impliciete make regels* 33a, ... ⟩
⟨ *expliciete make regels* 30e, ... ⟩
⟨ *make targets* 34a, ... ⟩
◇

The default target of make is `all`.

⟨ *default target* 30a ⟩ ≡
```
all : ⟨ all targets 30b ⟩
.PHONY : all

◇
```
Fragment referenced in 29.
Defines: `all` Never used, `PHONY` 33b.

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

⟨ *all targets* 30b ⟩ ≡
```
dutch-nlp-modules-on-Lisa.pdf◇
```
Fragment referenced in 30a.
Uses: `pdf` 34a.

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* 30c ⟩ ≡
```
.SUFFIXES: .pdf .w .tex .html .aux .log .php

◇
```
Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 34a.

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.

⟨ *parameters in Makefile* 30d ⟩ ≡
```
NUWEB=../bin/nuweb
◇
```
Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Defines: `NUWEB` 30e, 35abc, 36b, 38c, 39c, 40c.
Uses: `nuweb` 36b.

⟨ *expliciete make regels* 30e ⟩ ≡
```
$(NUWEB): ../nuweb-1.58
        cd ../nuweb-1.58 && make nuweb
        cp ../nuweb-1.58/nuweb $(NUWEB)

◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: `NUWEB` 30d, `nuweb` 36b.

⟨ *expliciete make regels* 31a ⟩ ≡
```
../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 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: `nuweb` 36b.

A.5 Pre-processing

To make usable things from the raw input `a_dutch-nlp-modules-on-Lisa.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* 31b ⟩ ≡
```
m4_dutch-nlp-modules-on-Lisa.w : a_dutch-nlp-modules-on-Lisa.w
        gawk '{if(match($$0, "@%")) {printf("%s", substr($$0,1,RSTART-1))} else print}' a_dutch-nlp-m
          | gawk '{gsub(/[\\][\$$]/, "$$");print}'  > m4_dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: `print` 34a.

A.5.2 Run the M4 pre-processor

⟨ *expliciete make regels* 31c ⟩ ≡
```
dutch-nlp-modules-on-Lisa.w : m4_dutch-nlp-modules-on-Lisa.w inst.m4
        m4 -P m4_dutch-nlp-modules-on-Lisa.w > dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.

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* 32a ⟩ ≡

```
FIGFILES=fileschema
```

◇

Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Defines: `FIGFILES` 32b, 37b.

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* 32b ⟩ ≡

```
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 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 34b, `PDF_FIG_NAMES` 34b, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 32a.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 33a ⟩ ≡
```
%.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 33a, 34b, 37c.
Fragment referenced in 29.
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 `dutch-nlp-modules-on-Lisa.bib`. To create this file, copy the auxiliary file to another file `auxfil.aux`, but replace the argument of the command `\bibdata{dutch-nlp-modules-on-Lisa}` 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* 33b ⟩ ≡
```
bibfile : dutch-nlp-modules-on-Lisa.aux /home/paul/bin/mkportbib
        /home/paul/bin/mkportbib dutch-nlp-modules-on-Lisa litprog

.PHONY : bibfile
◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: `PHONY` 30a.

A.6.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 34a ⟩ ≡
```
pdf : dutch-nlp-modules-on-Lisa.pdf

print : dutch-nlp-modules-on-Lisa.pdf
        lpr dutch-nlp-modules-on-Lisa.pdf

view : dutch-nlp-modules-on-Lisa.pdf
        evince dutch-nlp-modules-on-Lisa.pdf

◇
```
Fragment defined by 34a, 37a, 40bc.
Fragment referenced in 29.
Defines: `pdf` 30bc, 34b, `print` 14c, 22a, 31b, `view` Never used.

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

The w2pdf script The three processors nuweb, LaTeX and bibTeX are intertwined. LaTeX and bibTeX 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 LaTeX 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`.

Note, that in the following `make` construct, the implicit rule `.w.pdf` is not used. It turned out, that make did not calculate the dependencies correctly when I did use this rule.

⟨ *impliciete make regels* 34b ⟩ ≡
```
%.pdf : %.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*

◇
```
Fragment defined by 33a, 34b, 37c.
Fragment referenced in 29.
Uses: `pdf` 34a, `PDFT_NAMES` 32b, `PDF_FIG_NAMES` 32b.

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* 34c ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 4abcdef, 5c, 6ce, 8b, 14b, 34c.
Fragment referenced in 40b.
Uses: `nuweb` 36b.

⟨ *parameters in Makefile* 34d ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Uses: `nuweb` 36b.

⟨ *expliciete make regels* 35a ⟩ ≡
```
$(W2PDF) : dutch-nlp-modules-on-Lisa.w $(NUWEB)
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: NUWEB 30d.

"../nuweb/bin/w2pdf" 35b≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20150127 at 1534h: Generated by nuweb from a_dutch-nlp-modules-on-Lisa.w
NUWEB=/usr/local/bin/nuweb

LATEXCOMPILER=pdflatex
⟨ filenames in nuweb compile script 35d ⟩
⟨ compile nuweb 35c ⟩

◇
```
Uses: NUWEB 30d, nuweb 36b.

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

⟨ *compile nuweb* 35c ⟩ ≡
```
NUWEB=/usr/local/bin/nuweb

⟨ run the processors until the aux file remains unchanged 36c ⟩
⟨ remove the copy of the aux file 36a ⟩
◇
```
Fragment referenced in 35b.
Uses: NUWEB 30d, nuweb 36b.

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 LaTeX 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* 35d ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 35b.
Defines: auxfil 36c, 39ab, indexfil 36c, 39a, nufil 36b, 39ac, oldaux 36ac, 39ab, oldindexfil 36c, 39a,
texfil 36b, 39ac, trunk 36b, 39acd.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 36a ⟩ ≡
```
rm $oldaux
◇
```
Fragment referenced in 35c, 38d.
Uses: `oldaux` 35d, 39a.

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

⟨ *run the three processors* 36b ⟩ ≡
```
$NUWEB $nufil
$LATEXCOMPILER $texfil
makeindex $trunk
bibtex $trunk
◇
```
Fragment referenced in 36c.
Defines: `bibtex` 39cd, `makeindex` 39cd, `nuweb` 16a, 24a, 30de, 31a, 34cd, 35bc, 37a, 38bc.
Uses: `nufil` 35d, 39a, `NUWEB` 30d, `texfil` 35d, 39a, `trunk` 35d, 39a.

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* 36c ⟩ ≡
```
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 36b ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
◇
```
Fragment referenced in 35c.
Uses: `auxfil` 35d, 39a, `indexfil` 35d, `oldaux` 35d, 39a, `oldindexfil` 35d.

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`.

Nuweb creates a LaTeX file that is suitable for `latex2html` if the source file has `.hw` as suffix instead of `.w`. However, this feature is not compatible with tex4ht.

Make html file:

⟨ *make targets* 37a ⟩ ≡
```
html : ../nuweb/html/dutch-nlp-modules-on-Lisa.html

◇
```
Fragment defined by 34a, 37a, 40bc.
Fragment referenced in 29.
Uses: `nuweb` 36b.

The HTML file depends on its source file and the graphics files.

Make lists of the graphics files and copy them.

⟨ *parameters in Makefile* 37b ⟩ ≡
```
HTML_PS_FIG_NAMES=$(foreach fil,$(FIGFILES), m4_htmldocdir/$(fil).pstex)
HTML_PST_NAMES=$(foreach fil,$(FIGFILES), m4_htmldocdir/$(fil).pstex_t)
◇
```
Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Uses: `FIGFILES` 32a.

⟨ *impliciete make regels* 37c ⟩ ≡
```
m4_htmldocdir/%.pstex : %.pstex
        cp  $< $@

m4_htmldocdir/%.pstex_t : %.pstex_t
        cp  $< $@

◇
```
Fragment defined by 33a, 34b, 37c.
Fragment referenced in 29.

Copy the nuweb file into the html directory.

⟨ *expliciete make regels* 37d ⟩ ≡
```
m4_htmlsource : dutch-nlp-modules-on-Lisa.w
        cp  dutch-nlp-modules-on-Lisa.w m4_htmlsource

◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.

We also need a file with the same name as the documentstyle and suffix `.4ht`. Just copy the file `report.4ht` from the tex4ht distribution. Currently this seems to work.

⟨ *expliciete make regels* 37e ⟩ ≡
```
m4_4htfildest : m4_4htfilsource
        cp m4_4htfilsource m4_4htfildest

◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.

Copy the bibliography.

⟨ *expliciete make regels* 38a ⟩ ≡
```
m4_htmlbibfil : m4_nuwebdir/dutch-nlp-modules-on-Lisa.bib
        cp m4_nuwebdir/dutch-nlp-modules-on-Lisa.bib m4_htmlbibfil


◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.

Make a dvi file with `w2html` and then run `htlatex`.

⟨ *expliciete make regels* 38b ⟩ ≡
```

../nuweb/html/dutch-nlp-modules-on-Lisa.html : m4_htmlsource m4_4htfildest $(HTML_PS_FIG_NAMES) $(HTM
        cp w2html ../bin
        cd ../bin && chmod 775 w2html
        cd m4_htmldocdir && ../bin/w2html dutch-nlp-modules-on-Lisa.w


◇
```
Fragment defined by 30e, 31abc, 33b, 35a, 37de, 38ab.
Fragment referenced in 29.
Uses: `nuweb` 36b.

Create a script that performs the translation.

`"w2html"` 38c≡
```
#!/bin/bash
# w2html -- make a html file from a nuweb file
# usage: w2html [filename]
#  [filename]: Name of the nuweb source file.
`#' m4_header
echo "translate " $1 >w2html.log
NUWEB=/usr/local/bin/nuweb

⟨ filenames in w2html 39a ⟩

⟨ perform the task of w2html 38d ⟩


◇
```
Uses: `NUWEB` 30d, `nuweb` 36b.

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* 38d ⟩ ≡
```
⟨ run the html processors until the aux file remains unchanged 39b ⟩
⟨ remove the copy of the aux file 36a ⟩
◇
```
Fragment referenced in 38c.

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 LaTeX 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* 39a ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 38c.
Defines: `auxfil` 35d, 36c, 39b, `nufil` 35d, 36b, 39c, `oldaux` 35d, 36ac, 39b, `texfil` 35d, 36b, 39c, `trunk` 35d, 36b, 39cd.
Uses: `indexfil` 35d, `oldindexfil` 35d.

⟨ *run the html processors until the aux file remains unchanged* 39b ⟩ ≡
```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 39c ⟩
done
⟨ run tex4ht 39d ⟩

◇
```
Fragment referenced in 38d.
Uses: `auxfil` 35d, 39a, `oldaux` 35d, 39a.

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

⟨ *run the html processors* 39c ⟩ ≡
```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 39b.
Uses: `bibtex` 36b, `makeindex` 36b, `nufil` 35d, 39a, `NUWEB` 30d, `texfil` 35d, 39a, `trunk` 35d, 39a.

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* 39d ⟩ ≡
```
tex '\def\filename{{dutch-nlp-modules-on-Lisa}{idx}{4dx}{ind}} \input idxmake.4ht'
makeindex -o $trunk.ind $trunk.4dx
bibtex $trunk
htlatex $trunk
◇
```
Fragment referenced in 39b.
Uses: `bibtex` 36b, `makeindex` 36b, `trunk` 35d, 39a.

create the program sources Run nuweb, but suppress the creation of the LaTeX 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* 40a ⟩ ≡
```
MKDIR = mkdir -p

◇
```
Fragment defined by 30cd, 32ab, 34d, 37b, 40a.
Fragment referenced in 29.
Defines: `MKDIR` 40b.

⟨ *make targets* 40b ⟩ ≡
```
DIRS = ⟨ directories to create 4a, ... ⟩

$(DIRS) :
        $(MKDIR) $@

◇
```
Fragment defined by 34a, 37a, 40bc.
Fragment referenced in 29.
Defines: `DIRS` 40c.
Uses: `MKDIR` 40a.

⟨ *make targets* 40c ⟩ ≡
```
sources : dutch-nlp-modules-on-Lisa.w $(DIRS) $(NUWEB)
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
        ⟨ make scripts executable 11b, ... ⟩

◇
```
Fragment defined by 34a, 37a, 40bc.
Fragment referenced in 29.
Uses: `DIRS` 40b, `NUWEB` 30d.

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.

B.2 URL’s

Nuweb: `nuweb.sourceforge.net`
Apache Velocity: `m4_velocityURL`
Velocitytools: `m4_velocitytoolsURL`
Parameterparser tool: `m4_parameterparserdocURL`
Cookietool: `m4_cookietooldocURL`
VelocityView: `m4_velocityviewURL`

VelocityLayoutServlet: m4_velocitylayoutservletURL
Jetty: m4_jettycodehausURL
UserBase javadoc: m4_userbasejavadocURL
VU corpus Management development site: http://code.google.com/p/vucom

C Indexes

C.1 Filenames

C.2 Macro's

C.3 Variables