Install Dutch nlp modules on Lisa

Paul Huygen <paul.huygen@huygen.nl>

29th September 2014
16:18 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

1.1 List of the modules to be installed

Table 3 lists the modules in the pipeline. The column *source* indicated the origin of the module.

module	**directory**	**source**	**script**	**Details**
Tokenizer	`tokenizer-base`	Github	tok	
Morphosynt. p.	`morphosyntactic_parser_nl`	Github	`mor`	Needs Alpino
Alpinohack	`clean_hack`	This doc.	alpinohack	
NER	`../modules/jars`	Lisa	ner	Open source?
WSD	`ukb`	Lisa	wsd	
Onto	`ontotagger`	Lisa	onto	
Heidel	`HeidelTimeModule`	Lisa	heideltime	
SRL	`HeidelTimeModule`	Lisa	srl	

Table 1: 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.

Ideally, modules are directly obtained from a public repository, e.g. Github or a website of the organisation where the module has been built. However, some of the modules are not yet available in this way and only a snapshot has been installed by hand in Lisa. Table /reftab:modulesources provides the URL's of the sources that have been obtained from a public repository.

module	**source**	**URL**
Tokenizer	Github	https://github.com/opener-project/tokenizer-base.git
Morphosynt. p.	Github	`https://github.com/cltl/morphosyntactic_parser_nl.git`
Alpino	RUG	`Alpino-x86_64-linux-glibc2.5-20548-sicstus.tar.gz`
Ticcutils	ILK	http://software.ticc.uvt.nl/ticcutils-0.7.tar.gz
Timble	ILK	http://ilk.uvt.nl/timbl/download-timbl.php

Table 2: Sources of the modules

Table **??**

2 Installation

The modules are placed in subdirectory mods and scripts to apply the modules in a pipeline are placed in subdirectory `bin`. The modules directory has a subdirectories `python` for for python utilities resp java modules.

module	directory	source	Details
KafNafParserPy	`python/KafNafParserPy`	Github	
Alpino	`Alpino`	RUG	
Ticcutils	`ticcutils-0.7`	ILK	
Timbl	`timbl-6.4.5`	ILK	
Treetagger			

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.

⟨ *directories to create* 3a ⟩ ≡
```
/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.

⟨ *directories to create* 3b ⟩ ≡
```
/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.

⟨ *directories to create* 3c ⟩ ≡
```
/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/python ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.

⟨ *directories to create* 3d ⟩ ≡
```
/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/jars ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.

Make the Python utilities findable with the following macro:

⟨ *set pythonpath* 3e ⟩ ≡
```
export PYTHONPATH=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/python:$PYTHONPATH
◇
```
Fragment referenced in 7a.

"../bin/install_modules" 3f≡
```
#!/bin/bash
⟨ variables of install_modules 10b ⟩
⟨ install the tokenizer 5h ⟩
⟨ install kafnafparserpy 9b ⟩
⟨ install the morphosyntactic parser 6h ⟩
⟨ install the NER module 8c ⟩

◇
```

⟨ *make scripts executable* 4a ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/install_modules
```
◇

Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

Installation goes as follows:

1. If the module exists already, move it to a temporary place.
2. Try to install the module from the source.
3. If that is successful, remove the old version. Otherwise, move the old version back to its original place.

⟨ *move module* 4b ⟩ ≡
```
if
 [ -e @1 ]
then
   mv @1 old.@1
fi
```
◇

Fragment referenced in 5a, 6b.

⟨ *remove old module* 4c ⟩ ≡
```
rm -rf old.@1
```
◇

Fragment referenced in 5a, 6b.

⟨ *re-instate old module* 4d ⟩ ≡
```
mv old.@1 @1
MESS="Replaced previous version of @1"
```
⟨ *logmess* (4e `$MESS`) 10c ⟩

◇

Fragment referenced in 5a, 6b.

⟨ *install from github* 5a ⟩ ≡

```
MODNAM=@1
DIRN=@2
GITU=@3
⟨ find leave and tree 5g ⟩
⟨ logmess (5b "TREE: $TREE; LEAVE: $LEAVE" ) 10c ⟩
cd $TREE
⟨ move module (5c $LEAVE ) 4b ⟩
git clone $GITU
if
  [ $? -gt 0 ]
then
  ⟨ logmess (5d Cannot install current $MODNAM version ) 10c ⟩
  ⟨ re-instate old module (5e $LEAVE ) 4d ⟩
else
  ⟨ remove old module (5f $LEAVE ) 4c ⟩
fi

◇
```

Fragment referenced in 5h, 6h, 9b.

Note: Par. 1: Directory; par 2: path to directory; par 3: directory name.

⟨ *find leave and tree* 5g ⟩ ≡

```
FULLDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/$DIRN
LEAVE=${FULLDIR##*/}
TREE=${FULLDIR%%$LEAVE}
◇
```

Fragment referenced in 5a.

2.1 Install tokenizer

2.1.1 Module

⟨ *install the tokenizer* 5h ⟩ ≡

```
⟨ install from github (5i tokenizer,5j tokenizer-base,5k https://github.com/opener-project/tokenizer-base.git
◇
```

Fragment referenced in 3f.

2.1.2 Script

The script just runs the tokenizerscript in Perl.

"../bin/tok" 5l≡

```
#!/bin/bash
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
TOKBINDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/tokenizer-base/core
cat | perl $TOKBINDIR/tokenizer-cli.pl -l nl t

◇
```

⟨ *make scripts executable* 6a ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/tok
```
◇

Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

2.2 Install Alpino

Install Alpino from the website of Gertjan van Noort.

2.2.1 Module

⟨ *install Alpino* 6b ⟩ ≡
```
SUCCES=0
cd /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules
⟨ move module (6c Alpino ) 4b ⟩
wget http://www.let.rug.nl/vannoord/alp/Alpino/binary/versions/Alpino-x86_64-linux-glibc2.5-20548-sics
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 (6d Installed Alpino ) 10c ⟩
  ⟨ remove old module (6e Alpino ) 4c ⟩
else
  ⟨ re-instate old module (6f Alpino ) 4d ⟩
fi
```
◇

Fragment never referenced.

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* 6g ⟩ ≡
```
export ALPINO_HOME=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/Alpino
```
◇

Fragment referenced in 7a.
Defines: `ALPINO_HOME` Never used.

2.3 Morphosyntactic parser

2.3.1 Module

⟨ *install the morphosyntactic parser* 6h ⟩ ≡

⟨ *install from github* (6i morphsynparser,6j morphosyntactic_parser_nl,6k https://github.com/cltl/morphosyntact
◇

Fragment referenced in 3f.

2.3.2 Script

"../bin/mor" 7a≡
```
#!/bin/bash
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
MODDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/morphosyntactic_parser_nl
⟨ set alpinohome 6g ⟩
⟨ set pythonpath 3e ⟩
cat | python $MODDIR/core/morph_syn_parser.py
◇
```

⟨ make scripts executable 7b ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/mor
◇
```
Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

2.4 Alpino hack

Install a hack that removes output from Alpino that cannot be interpreted by following modules. It is just a small python script.

2.4.1 Module

⟨ directories to create 7c ⟩ ≡
```
/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/alpinohack ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.

"../modules/alpinohack/clean_hack.py" 7d≡
```
#!/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 15b.

2.4.2 Script

"../bin/alpinohack" 8a≡
```
#!/bin/bash
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
HACKDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/alpinohack
cat | python  $HACKDIR/clean_hack.py

```
◇

⟨ *make scripts executable* 8b ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/alpinohack
```
◇

Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

2.5 Named entity recognition

2.5.1 Module

We do not (yet have the source code of the NER module. A snapshot is comprised in a jar library.

⟨ *install the NER module* 8c ⟩ ≡
```
cp /home/phuijgen/nlp/jars/ixa-pipe-nerc-1.1.0.jar ../modules/jars/
```
◇

Fragment referenced in 3f.

2.5.2 Script

"../bin/ner" 8d≡
```
#!/bin/bash
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
JARDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/jars
cat | java -jar $JARDIR/ixa-pipe-nerc-1.1.0.jar tag
```
◇

⟨ *make scripts executable* 8e ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/ner
```
◇

Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

2.6 Wordsense-disambiguation

2.6.1 Module

2.6.2 Script

"../modules/wsd" 9a≡

```
#!/bin/bash
# WSD -- wrapper for word-sense disambiguation
# 8 Jan 2014 Ruben Izquierdo
# 16 sep 2014 Paul Huygen
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
WSDDIR=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/modules/ukb
WSDSCRIPT=kaf_annotate_senses.pl
UKB=$ROOT/ukb_wsd_2.0
POSMAP=$ROOT/posmap.NGV.txt

if [ "$1" = "nl" ]
then
  GRAPH=$ROOT/cdb2.0-nld-all.infv.0.0.no-allwords.bin
  DICT=$ROOT/dictionary
else
  GRAPH=$ROOT/wn30g_eng.v20.bin
  DICT=$ROOT/wn30_eng_dict.txt
fi

iconv -t utf-8//IGNORE | $WSDDIR/$WSDSCRIPT -x $UKB -M $GRAPH -W $DICT -m $POSMAP
◇
```

Uses: all 12b.

2.7 KafNafParserPy

Several modules use KafNafParserpy to read and write NAF files.

2.7.1 Module

⟨ *install kafnafparserpy* 9b ⟩ ≡

⟨ *install from github* (9c kafnafparserpy,9d python/KafNafParserPy,9e https://github.com/cltl/KafNafParserPy.g
◇

Fragment referenced in 3f.

3 Utilities

3.1 Test script

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

"../bin/test" 9f≡

```
#!/bin/bash
ROOT=/home/phuijgen/nlp/dutch-nlp-modules-on-Lisa
BIND=$ROOT/bin
echo "De hond eet jus." | $BIND/tok | $BIND/mor | $BIND/alpinohack | $BIND/ner > $ROOT/test.out
◇
```

⟨ *make scripts executable* 10a ⟩ ≡
```
chmod 775  /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/test
```
◇

Fragment defined by 4a, 6a, 7b, 8be, 10a.
Fragment referenced in 22b.

3.2 Logging

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

⟨ *variables of install_modules* 10b ⟩ ≡
```
LOGLEVEL=1
```
◇

Fragment referenced in 3f.

⟨ *logmess* 10c ⟩ ≡
```
if
 [ $LOGLEVEL -gt 0 ]
then
 echo @1
fi
```
◇

Fragment referenced in 4d, 5a, 6b.

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.

⟨ *parameters in Makefile* 11 ⟩ ≡

```
NUWEB=/home/phuijgen/usrlocal/bin/nuweb
```

◇

Fragment defined by 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Uses: `nuweb` 17c.

A.3 Translate and run

This chapter assembles the Makefile for this project.

```
"Makefile" 12a≡
    ⟨ default target 12b ⟩

    ⟨ parameters in Makefile 11, ... ⟩

    ⟨ impliciete make regels 14b, ... ⟩
    ⟨ expliciete make regels 13a, ... ⟩
    ⟨ make targets 15b, ... ⟩
    ◇
```

The default target of make is `all`.

```
⟨ default target 12b ⟩ ≡
    all : ⟨ all targets 12c ⟩
    .PHONY : all

    ◇
```

Fragment referenced in 12a.
Defines: `all` 9a, `PHONY` 15a.

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

```
⟨ all targets 12c ⟩ ≡
    dutch-nlp-modules-on-Lisa.pdf◇
```

Fragment referenced in 12b.
Uses: `pdf` 15b.

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 12d ⟩ ≡
    .SUFFIXES: .pdf .w .tex .html .aux .log .php

    ◇
```

Fragment defined by 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Defines: `SUFFIXES` Never used.
Uses: `pdf` 15b.

A.4 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.4.1 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* 13a ⟩ ≡

```
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-mo
          | gawk '{gsub(/[\\][\$$]/, "$$");print}'  > m4_dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.
Uses: `print` 15b.

A.4.2 Run the M4 pre-processor

⟨ *expliciete make regels* 13b ⟩ ≡

```
dutch-nlp-modules-on-Lisa.w : m4_dutch-nlp-modules-on-Lisa.w
        m4 -P m4_dutch-nlp-modules-on-Lisa.w > dutch-nlp-modules-on-Lisa.w
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

A.5 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.5.1 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* 13c ⟩ ≡

```
FIGFILES=fileschema
```

◇

Fragment defined by 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Defines: `FIGFILES` 14a, 18c.

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* 14a ⟩ ≡

```
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 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Defines: `FIGFILENAMES` Never used, `PDFT_NAMES` 15c, `PDF_FIG_NAMES` 15c, `PST_NAMES` Never used, `PS_FIG_NAMES` Never used.
Uses: `FIGFILES` 13c.

Create the graph files with program `fig2dev`:

⟨ *impliciete make regels* 14b ⟩ ≡

```
%.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 14b, 15c, 19a.
Fragment referenced in 12a.
Defines: `fig2dev` Never used.

A.5.2 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* 15a ⟩ ≡
```
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 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.
Uses: PHONY 12b.

A.5.3 Create a printable/viewable document

Make a PDF document for printing and viewing.

⟨ *make targets* 15b ⟩ ≡
```
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 15b, 18b, 22ab.
Fragment referenced in 12a.
Defines: pdf 12cd, 15c, print 7d, 13a, 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* 15c ⟩ ≡
```
%.pdf : %.w $(W2PDF)  $(PDF_FIG_NAMES) $(PDFT_NAMES)
        chmod 775 $(W2PDF)
        $(W2PDF) $*
```
◇
Fragment defined by 14b, 15c, 19a.
Fragment referenced in 12a.
Uses: pdf 15b, PDFT_NAMES 14a, PDF_FIG_NAMES 14a.

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* 16a ⟩ ≡
```
../nuweb/bin ◇
```
Fragment defined by 3abcd, 7c, 16a.
Fragment referenced in 22a.
Uses: nuweb 17c.

⟨ *parameters in Makefile* 16b ⟩ ≡
```
W2PDF=../nuweb/bin/w2pdf
◇
```
Fragment defined by 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Uses: nuweb 17c.

⟨ *expliciete make regels* 16c ⟩ ≡
```
$(W2PDF) : dutch-nlp-modules-on-Lisa.w
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
◇
```
Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

"../nuweb/bin/w2pdf" 16d≡
```
#!/bin/bash
# w2pdf -- compile a nuweb file
# usage: w2pdf [filename]
# 20140929 at 1618h: Generated by nuweb from a_dutch-nlp-modules-on-Lisa.w
NUWEB=/home/phuijgen/usrlocal/bin/nuweb

LATEXCOMPILER=pdflatex
⟨ filenames in nuweb compile script 17a ⟩
⟨ compile nuweb 16e ⟩

◇
```
Uses: nuweb 17c.

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* 16e ⟩ ≡
```
NUWEB=m4_nuweb
⟨ run the processors until the aux file remains unchanged 18a ⟩
⟨ remove the copy of the aux file 17b ⟩
◇
```
Fragment referenced in 16d.

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* 17a ⟩ ≡

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

Fragment referenced in 16d.
Defines: `auxfil` 18a, 20c, 21a, `indexfil` 18a, 20c, `nufil` 17c, 20c, 21b, `oldaux` 17b, 18a, 20c, 21a, `oldindexfil` 18a, 20c, `texfil` 17c, 20c, 21b, `trunk` 17c, 20c, 21bc.

Remove the old copy if it is no longer needed.

⟨ *remove the copy of the aux file* 17b ⟩ ≡

```
rm $oldaux
◇
```

Fragment referenced in 16e, 20b.
Uses: `oldaux` 17a, 20c.

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* 17c ⟩ ≡

```
$NUWEB $nufil
$LATEXCOMPILER $texfil
makeindex $trunk
bibtex $trunk
◇
```

Fragment referenced in 18a.
Defines: `bibtex` 21bc, `makeindex` 21bc, `nuweb` 11, 16abd, 20a.
Uses: `nufil` 17a, 20c, `texfil` 17a, 20c, `trunk` 17a, 20c.

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* 18a ⟩ ≡

```
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 17c ⟩
  if [ $LOOPCOUNTER -ge 10 ]
  then
    cp $auxfil $oldaux
  fi;
done
```

◇

Fragment referenced in 16e.
Uses: `auxfil` 17a, 20c, `indexfil` 17a, `oldaux` 17a, 20c, `oldindexfil` 17a.

A.5.4 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* 18b ⟩ ≡

```
html : m4_htmltarget

```

◇

Fragment defined by 15b, 18b, 22ab.
Fragment referenced in 12a.

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

Make lists of the graphics files and copy them.

⟨ *parameters in Makefile* 18c ⟩ ≡

```
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 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Uses: `FIGFILES` 13c.

⟨ *impliciete make regels* 19a ⟩ ≡

```
m4_htmldocdir/%.pstex : %.pstex
        cp  $< $@

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

◇

Fragment defined by 14b, 15c, 19a.
Fragment referenced in 12a.

Copy the nuweb file into the html directory.

⟨ *expliciete make regels* 19b ⟩ ≡

```
m4_htmlsource : dutch-nlp-modules-on-Lisa.w
        cp  dutch-nlp-modules-on-Lisa.w m4_htmlsource
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

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* 19c ⟩ ≡

```
m4_4htfildest : m4_4htfilsource
        cp m4_4htfilsource m4_4htfildest
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

Copy the bibliography.

⟨ *expliciete make regels* 19d ⟩ ≡

```
m4_htmlbibfil : m4_anuwebdir/dutch-nlp-modules-on-Lisa.bib
        cp m4_anuwebdir/dutch-nlp-modules-on-Lisa.bib m4_htmlbibfil
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

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

⟨ *expliciete make regels* 19e ⟩ ≡

```
m4_htmltarget : m4_htmlsource m4_4htfildest $(HTML_PS_FIG_NAMES) $(HTML_PST_NAMES) m4_htmlbibfil
        cp w2html /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin
        cd /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin && chmod 775 w2html
        cd m4_htmldocdir && /home/phuijgen/nlp/dutch-nlp-modules-on-Lisa/bin/w2html dutch-nlp-modules-
```

◇

Fragment defined by 13ab, 15a, 16c, 19bcde.
Fragment referenced in 12a.

Create a script that performs the translation.

"w2html" 20a≡
```
#!/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=/home/phuijgen/usrlocal/bin/nuweb

⟨ filenames in w2html 20c ⟩

⟨ perform the task of w2html 20b ⟩

◇
```
Uses: nuweb 17c.

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 20b ⟩ ≡
```
⟨ run the html processors until the aux file remains unchanged 21a ⟩
⟨ remove the copy of the aux file 17b ⟩
◇
```
Fragment referenced in 20a.

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 20c ⟩ ≡
```
nufil=$1
trunk=${1%%.*}
texfil=${trunk}.tex
auxfil=${trunk}.aux
oldaux=old.${trunk}.aux
indexfil=${trunk}.idx
oldindexfil=old.${trunk}.idx
◇
```
Fragment referenced in 20a.
Defines: auxfil 17a, 18a, 21a, nufil 17ac, 21b, oldaux 17ab, 18a, 21a, texfil 17ac, 21b, trunk 17ac, 21bc.
Uses: indexfil 17a, oldindexfil 17a.

⟨ *run the html processors until the aux file remains unchanged* 21a ⟩ ≡

```
while
  ! cmp -s $auxfil $oldaux
do
  if [ -e $auxfil ]
  then
   cp $auxfil $oldaux
  fi
  ⟨ run the html processors 21b ⟩
done
⟨ run tex4ht 21c ⟩

◇
```

Fragment referenced in 20b.
Uses: `auxfil` 17a, 20c, `oldaux` 17a, 20c.

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

⟨ *run the html processors* 21b ⟩ ≡

```
$NUWEB -o -n $nufil
latex $texfil
makeindex $trunk
bibtex $trunk
htlatex $trunk
◇
```

Fragment referenced in 21a.
Uses: `bibtex` 17c, `makeindex` 17c, `nufil` 17a, 20c, `texfil` 17a, 20c, `trunk` 17a, 20c.

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* 21c ⟩ ≡

```
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 21a.
Uses: `bibtex` 17c, `makeindex` 17c, `trunk` 17a, 20c.

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* 21d ⟩ ≡

```
MKDIR = mkdir -p

◇
```

Fragment defined by 11, 12d, 13c, 14a, 16b, 18c, 21d.
Fragment referenced in 12a.
Defines: `MKDIR` 22a.

⟨ *make targets* 22a ⟩ ≡

```
DIRS = ⟨ directories to create 3a, ... ⟩

$(DIRS) :
        $(MKDIR) $@
```

◇

Fragment defined by 15b, 18b, 22ab.
Fragment referenced in 12a.
Defines: DIRS 22b.
Uses: MKDIR 21d.

⟨ *make targets* 22b ⟩ ≡

```
sources : dutch-nlp-modules-on-Lisa.w $(DIRS)
        $(NUWEB) dutch-nlp-modules-on-Lisa.w
        ⟨ make scripts executable 4a, ... ⟩

jetty : sources
        cd .. && mvn jetty:run
```

◇

Fragment defined by 15b, 18b, 22ab.
Fragment referenced in 12a.
Uses: DIRS 22a.

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