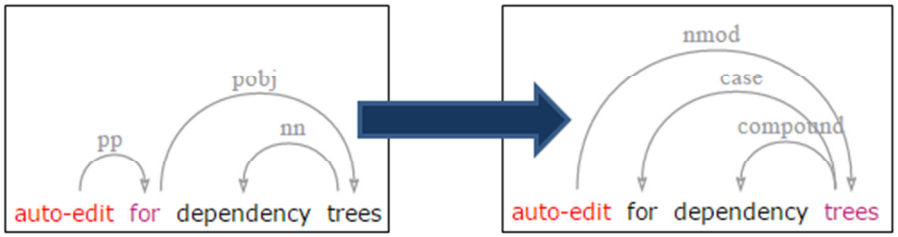

DepEdit
User Guide – Version 1.2.0

title:	DepEdit User Guide
software version:	1.2.0
guide version:	1.0.0
date:	29 December 2015

author:	Amir Zeldes
e-mail:	amir.zeldes@georgetown.edu
homepage:	http://corpling.uis.georgetown.edu/depedit/

Introduction

DepEdit is a simple configurable tool for manipulating dependency trees, written in Python 2.X. To run it you need Python 2.X, a configuration file describing the manipulations (see Configuration), and an input file in the CoNLL 10 column dependency format (see Format). Basic usage is:

```
Python depedit.py -c config_file.ini INPUT.conll10 > OUTPUT.conll10
```

DepEdit can also be imported as a module into other projects to preprocess dependency trees (see Importing as a module).

Format

DepEdit uses the 10 column version of the CoNLL dependency format, with some specific names given to fields in the configuration files. There have been different ways of using the CoNLL format in different projects, e.g. whether or not each second column in a pair is used for gold vs. prediction or some other purpose, fine/coarse grained tags (in MALT), lemmas in the third column, non-projective vs. projective dependencies, etc. DepEdit uses the following column names/mappings, though you may use these column names to run the script even if your own use differs. The following two examples show a minimal and elaborate use of the columns.

Basic input:

The 10 columns are tab-separated, sentences are separated by a blank line. Columns are (bold italics indicate column can be manipulated directly, see Configuration):

1. Token ID within sentence
2. ***text*** - Token text
3. blank (underscore)
4. ***pos*** - Part of speech
5. blank (underscore)
6. blank (underscore)
7. ID of head token
8. ***func*** **–** dependency function
9.-10. – reserved for alternate trees with multiple parentage (not used)

1	Wikinews	_	NP	_	_	2	nsubj	_	_
2	interviews	_	VVZ	_	_	0	root	_	_
3	President	_	NN	_	_	2	dobj	_	_
4	of	_	IN	_	_	3	prep	_	_
5	the	_	DT	_	_	7	det	_	_
6	International	_	NP	_	_	7	amod	_	_
7	Brotherhood	_	NP	_	_	4	pobj	_	_

8	of	_	IN	_	_	7	prep	_	_
9	Magicians	_	NPS	_	_	8	pobj	_	_
1	Wednesday	_	NP	_	_	0	root	_	_
2	,	_	,	_	_	0	punct	_	_
3	October	_	NP	_	_	4	nn	_	_
4	9	_	CD	_	_	1	appos	_	_
5	,	_	,	_	_	0	punct	_	_
6	2013	_	CD	_	_	3	tmod	_	_

More elaborate input

This German example also encodes lemmas and morphology in the format, using column 2 (***lemma***) a and column 6 (***morph***).

1	Die	die	ART	ART	ART.Def.Nom.Pl.*	2	DET	_	_
2	Jugendlichen	Jugendliche	NN	NN	N.Reg.Nom.Pl.*	5	SUBJ	_	_
3	in	in	APPR	APPR	APPR	2	PP	_	_
4	Zossen	Zossen	NN	NN	N.Name.Dat.Sg.Neut	3	PN	_	_
5	wollen	wollen	VMFIN	VMFIN	VFIN.Mod.3.Pl.Pres.Ind	0	S	_	_
6	ein	eine	ART	ART	ART.Indef.Acc.Sg.Neut	7	DET	_	_
7	Musikcafé	Café	NN	NN	N.Reg.Acc.Sg.Neut	5	OBJA	_	_
8	.	.	$.	$.	SYM.Pun.Sent	0	ROOT	_	_
1	Das	die	PDS	PDS	PRO.Dem.Subst.Acc.Sg.Neut	2	OBJA	_	_
2	forderten	fordern	VVFIN	VVFIN	VFIN.Full.3.Pl.Past.Ind	0	S	_	_
3	sie	sie	PPER	PPER	PRO.Pers.Subst.3.Nom.Pl.*	2	SUBJ	_	_
4	bei	bei	APPR	APPR	APPR	2	PP	_	_
5	der	die	ART	ART	ART.Def.Dat.Sg.Fem	8	DET	_	_
6	ersten	erst	ADJA	ADJA	ADJA.Pos.Dat.Sg.Fem	8	ATTR	_	_
7	Zossener	Zossener	NN	NN	ADJA.Pos.*.*.*	8	ATTR	_	_
8	Runde	Runde	NN	NN	N.Reg.Dat.Sg.Fem	4	PN	_	_
9	am	an	APPRART	APPRART	APPRART.Dat.Sg.Masc	2	PP	_	_
10	Dienstagabend	Abend	NN	NN	N.Reg.Dat.Sg.Masc	9	PN	_	_
11	.	.	$.	$.	SYM.Pun.Sent	0	ROOT	_	_

Configuration

The manipulations carried out by the script are defined in a configuration file. This is a simple text file with one instruction per line and optional blank lines and comments (beginning with ';' or '#'). Each instruction contains 3 columns, as in the following example:

config_file.ini

```
;Connect nouns to a preceding article or possessive pronoun with the 'det' function
pos=/DT|PRP\$/;pos=/NNS?/      #1.#2           #2>#1;#1:func=det

;Change to-infinitive from aux to mark
text=/^[Tt]o$/&func=/aux/      none            #1:func=mark
```

The first column describes the tokens to be matched using regular expressions.

- Constraints are given as regular expressions over the fields:
    - ***text*** (column 2 of CoNLL10)
    - ***lemma*** (column 3)
    - ***pos*** (column 4)
    - ***morph*** (column 6)
    - ***func*** (dependency function, column 8)
- Multiple tokens are separated by ';'
- You can specify multiple criteria using '&', as in the second rule
- You may specify negative criteria using !=, e.g. lemma!=/able/

The middle column defines relationships between tokens. It refers to each token in the definition by number (#1, #2...) and specifies:

- Adjacency (.): #1.#2 means the first token in column 1 is followed by the second
- Distance (.n or .n,m): #1.4#2 means 4 tokens distance, and #1.1,4#2 means a distance of 1-4
- Parentage (>): #1>#2 means the first token in column 1 is the head of the second token
- If the instruction refers to only one token, as in the second example, the middle column says 'none'.

The third column specifies what to do if a rule matches:

- Change a property of token:
    - text
    - part of speech
    - dependency function
    - morphological analysis
    - lemma

- Make some token in the definitions the head of another

Importing as a module

Starting in version 1.1.0 you can import depedit into other projects using the entry point function run_depedit, which expects two file handles for the input and configuration files:

```
from depedit import run_depedit


infile = open("path/to/infile.txt")
config_file  = open("path/to/config.ini")
result = run_depedit(infile, config_file)
```