Tagger comparison

Choose three part of speech taggers of your choice (from wherever) and compare them on the same data set.

For example, let's say we want to test the performance of UDPipe (https://github.com/ufal/udpipe) for Finnish:

First checkout UDPipe:

```
$ git clone https://github.com/ufal/udpipe
$ cd udpipe/src
$ make
```

This will compile UDPipe, now put it (`cp`) somewhere in your `$PATH`, e.g. `/usr/local/bin`. You can find out what directories are in your `$PATH` by doing: `echo $PATH`.

Now download the `UD_Finnish-TDT` treebank.

```
$ git clone https://github.com/UniversalDependencies/UD_Finnish-TDT
```

If you go into the directory `UD_Finnish-TDT`, you will be able to train UDPipe with the following command:

```
$ cat fi_tdt-ud-train.conllu | udpipe --tokenizer=none --parser=none
--train fi.udpipe
```

This will produce a model file `fi.udpipe` which you can then use for tagging:

```
$ cat fi_tdt-ud-test.conllu | udpipe --tag fi.udpipe > fi_tdt-ud-
test_output.conllu
```

You can use the CoNLL-2017 evaluation script (http://universaldependencies.org/conll17/eval.zip) to evaluate the tagger performance:

```
$ python3 conll17_ud_eval.py --verbose fi_tdt-ud-test.conllu fi_tdt-ud-test_output.conllu
```

You're interested in the row, `UPOS`.

Now process the same dataset with some other tagger, for example:

- hunpos (https://code.google.com/archive/p/hunpos) trigram HMM tagger
- You could also try a simple Perceptron-based tagger (https://github.com/ftyers/conllu-perceptron-tagger)
- There are also a number of taggers in NLTK — you will probably need to write code to convert the input format to/from CoNLL-U.
- Another possibility is to write your own simple unigram tagger.

You should upload a report (in Markdown or HTML) describing the comparison.

Constraint Grammar

The objective of this task is to write some constraint grammar rules to disambiguate a sentence. You can use your own morphological analyser to analyse the sentence, or you can use one I wrote that makes an analyser from a UD corpus. Get that code:

```
$ git clone https://github.com/ftyers/ud-scripts.git
```

Now get a UD corpus:

```
$ git clone https://github.com/UniversalDependencies/UD_Russian-SynTagRus
```

Make the morphological analyser:

```
$ cat UD_Russian-SynTagRus/*.conllu | python3 ud-scripts/conllu-analyser.py -t ru-analyser.tsv
```

And you can analyse new sentences as follows:

```
$ echo "Однако стиль работы Семена Еремеевича заключался в том, чтобы
принимать всех желающих и лично вникать в дело." |\
  python3 ud-scripts/conllu-analyser.py ru-analyser.tsv
"<Однако>"
    "однако" ADV Degree=Pos
"<стиль>"
    "стиль" NOUN Animacy=Inan Case=Nom Gender=Masc Number=Sing
    "стиль" NOUN Animacy=Inan Case=Acc Gender=Masc Number=Sing
"<работы>"
    "работа" NOUN Animacy=Inan Case=Gen Gender=Fem Number=Sing
    "работа" NOUN Animacy=Inan Case=Nom Gender=Fem Number=Plur
    "работа" NOUN Animacy=Inan Case=Acc Gender=Fem Number=Plur
"<Семена>"
    "семен" PROPN Animacy=Anim Case=Gen Gender=Masc Number=Sing
    "семен" PROPN Animacy=Anim Case=Acc Gender=Masc Number=Sing
"<Еремеевича>"
    "еремеевич" PROPN Animacy=Anim Case=Gen Gender=Masc Number=Sing
"<заключался>"
    "заключаться" VERB Aspect=Imp Gender=Masc Mood=Ind Number=Sing
Tense=Past VerbForm=Fin Voice=Mid
"<в>"
    "в" ADP
"<том>"
    "тот" DET Case=Loc Gender=Neut Number=Sing
    "то" PRON Animacy=Inan Case=Loc Gender=Neut Number=Sing
    "тот" DET Case=Loc Gender=Masc Number=Sing
"<,>"
    "," PUNCT
"<чтобы>"
    "чтобы" SCONJ Mood=Cnd
"<принимать>"
    "принимать" VERB Aspect=Imp VerbForm=Inf Voice=Act
"<всех>"
    "весь" DET Case=Gen Number=Plur
    "весь" DET Case=Loc Number=Plur
```

```
    "весь" DET Case=Acc Number=Plur
    "все" PRON Animacy=Anim Case=Acc Number=Plur
    "все" PRON Animacy=Anim Case=Gen Number=Plur
"<желающих>"
    "желать" VERB Aspect=Imp Case=Gen Number=Plur Tense=Pres
VerbForm=Part Voice=Act
"<и>"
    "и" CCONJ
    "и" PART
"<лично>"
    "лично" ADV Degree=Pos
"<вникать>"
    "*вникать"
"<в>"
    "в" ADP
"<дело>"
    "дело" NOUN Animacy=Inan Case=Nom Gender=Neut Number=Sing
    "дело" NOUN Animacy=Inan Case=Acc Gender=Neut Number=Sing
"<.>"
    "." PUNCT
```

Next you'll need to install CG:

```
$ wget http://apertium.projectjj.com/apt/install-nightly.sh -O - | sudo
bash
$ sudo apt-get install cg3
```

Note: If you're using some non-Linux operating system you can try the CG3 IDE (https://visl.sdu.dk/cg3ide.html).

Next we can try to write a simple rule:

- Remove `DET` if the following token is `PUNCT`

Make a new file called `rus.cg3` and paste the following in:

```
DELIMITERS = "." ;


LIST DET = DET ;
LIST PUNCT = PUNCT ;


SECTION


REMOVE DET IF (1C PUNCT) ;
```

Now you can run the rules on your input as follows:

```
$ echo "Однако стиль работы Семена Еремеевича заключался в том, чтобы
принимать всех желающих и лично вникать в дело." |\
  python3 ~/source/ud-scripts/conllu-analyser.py ru-analyser.tsv |
vislcg3 -t -g rus.cg3
"<Однако>"
    "однако" ADV Degree=Pos
"<стиль>"
    "стиль" NOUN Animacy=Inan Case=Nom Gender=Masc Number=Sing
    "стиль" NOUN Animacy=Inan Case=Acc Gender=Masc Number=Sing
"<работы>"
    "работа" NOUN Animacy=Inan Case=Gen Gender=Fem Number=Sing
    "работа" NOUN Animacy=Inan Case=Nom Gender=Fem Number=Plur
    "работа" NOUN Animacy=Inan Case=Acc Gender=Fem Number=Plur
"<Семена>"
    "семен" PROPN Animacy=Anim Case=Gen Gender=Masc Number=Sing
    "семен" PROPN Animacy=Anim Case=Acc Gender=Masc Number=Sing
"<Еремеевича>"
    "еремеевич" PROPN Animacy=Anim Case=Gen Gender=Masc Number=Sing
"<заключался>"
    "заключаться" VERB Aspect=Imp Gender=Masc Mood=Ind Number=Sing
Tense=Past VerbForm=Fin Voice=Mid
"<в>"
    "в" ADP
"<том>"
    "то" PRON Animacy=Inan Case=Loc Gender=Neut Number=Sing
;   "тот" DET Case=Loc Gender=Neut Number=Sing REMOVE:6
;   "тот" DET Case=Loc Gender=Masc Number=Sing REMOVE:6
"<,>"
    "," PUNCT
"<чтобы>"
    "чтобы" SCONJ Mood=Cnd
"<принимать>"
    "принимать" VERB Aspect=Imp VerbForm=Inf Voice=Act
"<всех>"
    "весь" DET Case=Gen Number=Plur
```

```
    "весь" DET Case=Loc Number=Plur
    "весь" DET Case=Acc Number=Plur
    "все" PRON Animacy=Anim Case=Acc Number=Plur
    "все" PRON Animacy=Anim Case=Gen Number=Plur
"<желающих>"
    "желать" VERB Aspect=Imp Case=Gen Number=Plur Tense=Pres
VerbForm=Part Voice=Act
"<и>"
    "и" CCONJ
    "и" PART
"<лично>"
    "лично" ADV Degree=Pos
"<вникать>"
    "*вникать"
"<в>"
    "в" ADP
"<дело>"
    "дело" NOUN Animacy=Inan Case=Nom Gender=Neut Number=Sing
    "дело" NOUN Animacy=Inan Case=Acc Gender=Neut Number=Sing
"<.>"
    "." PUNCT
```

Note how the rule has applied (search for `REMOVE:6`).

Now write three more rules.

You should upload the rule file and your input file.

Improve perceptron tagger

The objective of this task is to download and run a very basic averaged perceptron tagger (less than 300 lines of Python). After successfully running it, try and improve it by changing the feature specifications.

First download the code:

```
$ git clone https://github.com/ftyers/conllu-perceptron-tagger.git
```

Then download some data, feel free to replace `UD_Portuguese` with any language in UD.

```
$ git clone https://github.com/UniversalDependencies/UD_Portuguese-GSD.git
```

Then download the CoNLL shared task 2017 official evaluation script and unzip it:

```
$ wget http://universaldependencies.org/conll17/eval.zip
$ unzip eval.zip
```

Finally enter the directory of the perceptron tagger:

```
$ cd conllu-perceptron-tagger
```

You can train the tagger using the following command:

```
$ cat ../UD_Portuguese-GSD/pt_gsd-ud-train.conllu | python3 tagger.py -t pt-ud.dat
222070
Iter 0: 200635/222070=90.3476381321205
222063
Iter 1: 209720/222070=94.43869050299455
222033
Iter 2: 212814/222070=95.83194488224433
222068
Iter 3: 214833/222070=96.74111766560094
222021
Iter 4: 215973/222070=97.25446931147836
```

Now you can run the tagger

```
$ cat ../UD_Portuguese-GSD/pt_gsd-ud-test.conllu | python3 tagger.py pt-ud.dat > pt-ud-test.out
```

And evaluate:

```
$ python3 ../evaluation_script/conll17_ud_eval.py --verbose
../UD_Portuguese-GSD/pt_gsd-ud-test.conllu pt-ud-test.out
Metrics    | Precision |    Recall |  F1 Score | AlignedAcc
-----------+-----------+-----------+-----------+-----------
Tokens     |    100.00 |    100.00 |    100.00 |
Sentences  |    100.00 |    100.00 |    100.00 |
Words      |    100.00 |    100.00 |    100.00 |
UPOS       |     95.73 |     95.73 |     95.73 |     95.73
XPOS       |    100.00 |    100.00 |    100.00 |    100.00
Feats      |    100.00 |    100.00 |    100.00 |    100.00
AllTags    |     95.73 |     95.73 |     95.73 |     95.73
Lemmas     |    100.00 |    100.00 |    100.00 |    100.00
UAS        |    100.00 |    100.00 |    100.00 |    100.00
LAS        |    100.00 |    100.00 |    100.00 |    100.00
```

Note: The interesting value here is the `UPOS` column as that is the only one we have changed. The other columns are 100% because the script hasn't changed them.

In the code, feature extraction is done by `_get_features()` (https://github.com/ftyers/conllu-perceptron-tagger/blob/master/tagger.py#L154) in `tagger.py` (https://github.com/ftyers/conllu-perceptron-tagger/blob/master/tagger.py). Try playing around with this and see if you can improve the performance. Remember to try and improve your features on the `pt_gsd-ud-dev.conllu` file, not on the `pt_gsd-ud-test.conllu` file.

Submit a report describing what features your changed and what difference it made on the tagging performance.