Train, parse and evaluate using UDPipe

Download and compile UDPipe, if you have not done so already.

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

Then download a treebank of your choosing from Universal Dependencies (http://universaldependencies.org).

Train a model on the `-train` portion of the treebank, and test it on the `-test` portion.

```
$ udpipe  --tokenizer none --tagger none --train SOMELANGUAGE.udpipe <
TRAININGFILE.conllu
```

Replace `SOMELANGUAGE` with the language code of the language you're using and `TRAININGFILE` with the filename of the training data.

Then parse the test data using:

```
$ udpipe --parse SOMELANGUAGE.udpipe < TESTINGFILE.conllu >
TESTINGOUT.conllu
```

You should use the evaluation script (http://universaldependencies.org/conll17/eval.zip) from the CoNLL-17 shared task to evaluate the performance of the parser you've trained. Remember to evaluate on the test data!

Inspect 10 trees, what kind of errors were made?

Upload the results and a report (in Markdown (https://en.wikipedia.org/wiki/Markdown)) of the errors to your GitHub repository.

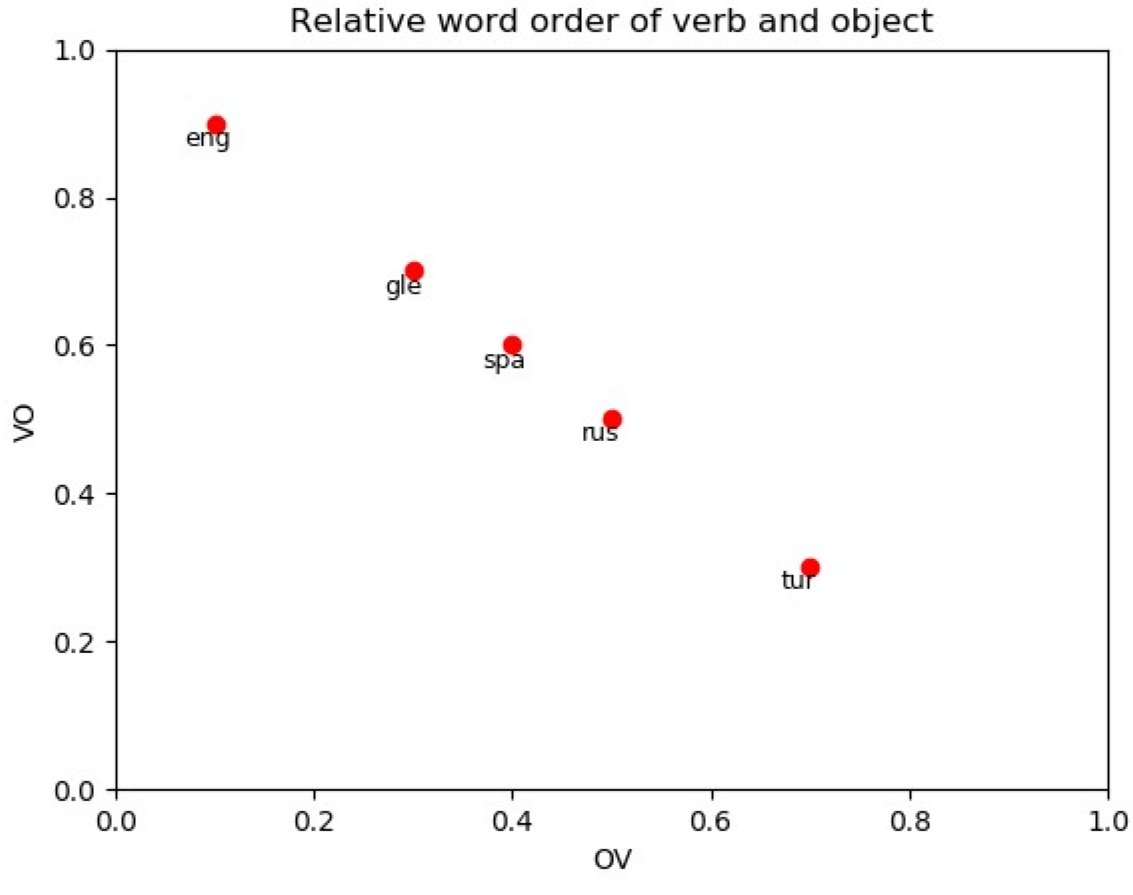
Relative word order study

As you are probably aware, languages of the world vary (https://wals.info/feature/83A#2/18.0/152.9) in the order they give to constituents like subject, object and verb.

We can use treebanks to study this variation.

You should write a script that will collect information about relative order of object and verb in 10 treebanks from the Universal Dependencies (http://universaldependencies.org) project.

You should create a visualisation in two axes, object—verb and verb—object:



Relative word order of verb and object

To do that you can adapt the following code using `matplotlib`:

```
import sys
import matplotlib.pyplot as plt

labels = {0:'eng', 1:'rus', 2:'gle',3:'tur',4:'spa'}
x = [0.1, 0.5, 0.3, 0.7, 0.4]  # proportion of OV
y = [0.9, 0.5, 0.7, 0.3, 0.6]  # proportion of VO
plt.plot(x, y, 'ro')
plt.title('Relative word order of verb and object')
plt.xlim([0,1]) # Set the x and y axis ranges
plt.ylim([0,1])
plt.xlabel('OV') # Set the x and y axis labels
plt.ylabel('VO')
for i in labels:  # Add labels to each of the points
    plt.text(x[i]-0.03, y[i]-0.03, labels[i], fontsize=9)
plt.savefig(sys.argv[1])
plt.show()
```

Upload your code and the plot to your GitHub repository.