NLP Deliverable 2 Guide

1 Introduction: Named Entity Recognition

The objective of this project is to fully understand the structured perceptron algorithm applied to Named Entity Recognition (NER). NER problems are very useful in many contexts, from information retrieval to question answering systems. The goal of this project is not to achieve the best results, but to fully understand all the details about a simple solution.

2 Deliverable zip file

This project has to be submitted by a single member of each group using a specific task created in the webpage of the course. The project needs to be in a `zip` file containing all the code to reproduce the results. The `zip` file containing all the work needs to have the following form:

- `name1_name2_name3.zip`. Where `name1,name2,...` are the names of the members of each group. Please write your full name. Example: If your name is "John Doe" write `JohnDoe`.
- Groups can be up to 5 people. There is no penalization for multiple people in a group.

3 Basic rules of the deliverable

- The `zip` file needs a `maindoc.pdf` file with a description of your work.
- The `zip` needs a notebook `reproduce_results.ipynb` that loads the work and evaluates it.
  - Accuracy on the train set and test set but taking into account only those predictions where the ground truth is not "O".
  - Confusion matrix on the train and test set.
  - F-score (weighted version, see https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)
  - The predictions for the TINY_TEST (look below) and the accuracy in such dataset.
- The data used for this project is `train_data_ner.csv`, `test_data_ner.csv` containing training data and test data respectively
- The **deadline for the project will be the 20 of june at 12 pm**.
- The `zip` file does not need to include the training data (but you can include it if you want).

Each requirement from this document that is not satisfied in your work will imply an reduction on your final mark.

4 Code format

Your project must be in a folder with the following tree structure:

```
> name1_name2_name3
    > main_doc.pdf
    > data
        > train_data_ner.csv
        > test_data_ner.csv
        > tiny_test.csv
    > train_models.ipynb
    > reproduce_results.ipynb
    > fitted_models
        > model1
        > model2
        ...
    > utils
        > utils.py
```

Some details:

- `main_doc.pdf` is a document containing basic explanations of your work and your results. In particular you need to explain how the structure perceptron works in detail. For instance, explain what happens if a word that is never seen during training is presented to the model at test time.
- `train_models.ipynb` is a notebook containing all the code required to train the models and store them in fitted_models
- `reproduce_results.ipynb` is a notebook that must: load the data, load the fitted models from disk and evaluate the models, following the rules for the deliverable described in this document.
- `fitted_models` is a folder where you will save your trained models. Any model used in reproduce_results.ipynb should be imported from `fitted_models`.
- `utils` is a folder containing functions that might be used in the `train_models.ipynb` and/or `reproduce_results.ipynb`. The goal is to keep the notebooks clean (avoid writing large functions inside the notebook, if you do it you can take them out, write them in `utils.py`, import them and use them).

4.1 What models to test

You need to test:

- The structured perceptron provided in a previous session. You can think about adding or modifiying feature templates to improve results.
- Another model that you choose, such as deep learning models (optional).

4.2 TINY_TEST

Notebook `reproduce_results.ipynb` should print the predicted tags for the following `TINY_TEST` in the format

```
w1/t1 w2/t2 w3/t3 w4/t4
```

where `wi` is a word and `ti` the tag associated to `wi`.

```
# TINY_TEST (these sentences and the ground truth labels are given in the
#            tiny_test.csv file)

The programmers from Barcelona might write a sentence without a spell
checker.

The programmers from Barchelona cannot write a sentence without a spell
checker.

Jack London went to Parris.

Jack London went to Paris.

Bill gates and Steve jobs never thought Microsoft would become such a big company.

Bill Gates and Steve Jobs never thought Microsof would become such a big company.

The president of U.S.A thought they could win the war.

The president of the United States of America thought they could win the war.

The king of Saudi Arabia wanted total control.

Robin does not want to go to Saudi Arabia.

Apple is a great company.

I really love apples and oranges.

Alice and Henry went to the Microsoft store to buy a new computer during their trip to
    New York.
```

5 Extra exercise

This extra exercise can increment your mark on this deliverable giving you up to 2 extra points.

Improve the provided code for the structured perceptron. In particular:

- Find where it spends more time during training and inference.
- Improve the code to make it faster.
- Benchmark the improvement.
- Explain your improvements in the `pdf` file.

Hint: We have seen how cython can be used to dramatically improve the speed of python. You can benefit from cython to do this.