NLP Deliverable 1 Guide

1 Quora Objective:

Make a basic model to solve the Quora challenge. The deliverable should contain a simple solution and an improved solution.

- Try a simple solution
  - What problems/limitations do you think the model has?
  - What type of errors do you get?
  - What type of features can you build to improve the basic naive solution?
- Improve your simple solution:
  - Build features for each input and use them to compute distances between the inputs.
  - Investigate and code a feature vector or a distance between two strings. Use you implementation to define a feature to capture the similarity between two documents.
  - Implement from scratch the feature vector or the distance function for two input documents.
  - Split implementations between members in the group (do not code the same thing twice).
  - Explain the implemented code in `main.pdf`.

2 Format and Delivery Rules

2.1 General Rules

- The project can be done in groups of up to 4 people.
- The project documentation should contain a section briefly describing how the work has been distributed between the group members.
- The project has to be uploaded to the virtual campus. If it is too heavy to upload, it has to be sent by email by a single member of each group (in case it is too heavy send a link to dropbox or similar) to `daniel.ortiz@ub.edu`. The deadline for the project is May 4 at 11.59 pm (delivering on the 5 of May is already too late).
- The project needs to be in a zip file containing all the code to reproduce the results.
- The zip file has to be self contained and with the following form: `name1_name2_name3.zip` Where `name1`, `name2`, ... are the names of the members of each group. Please write your full name in `CamelCase` form.
- If “Name1” and “Name2” make a team the zip filename has to be `Name1_Name2.zip`.

2.2 Content of the zip File

The zip file should contain something analogous to:

```
Name1_Name2.zip
|
|___ main.pdf
|
|___ models
|
|___ train_models.ipynb
|
|___ reproduce_results.ipynb
|
|___ utils.py
|
|___ utils_Name1.ipynb
|
|___ utils_Name2.ipynb
|
|___ requirements.txt
```

Explanation:

- The zip file `name1_name2_name3.zip` **DOES NOT** have to include train or test data.
- The data has to be read from `$HOME/Datasets/QuoraQuestionPairs`, ensure that the code can be run in other computers as long as the data is in the same path.
- The `name1_name2_name3.zip` file must contain:
  - `main.pdf`: A description of your work. It should contain a brief section describing the work carried out by each group member.
  - `models`: A folder containing the trained models. This folder should be created by `train_models.ipynb` and models should be stored there after running `train_models.ipynb` notebook. The code should check if the folder is there and in such a case do not overwrite/store the models.
  - `utils.py`: A python module with the functions used in `train_models.ipynb` and `reproduce_results.ipynb`.
  - `train_models.ipynb`: Notebook with the code needed to train and store models to disk. This notebook has to be clean (do not define functions here, do them in an external `utils.py` and import them). The notebook has to be reproducible (if you run it twice, the same output has to be displayed and stored to disk).
  - `reproduce_results.ipynb`: This notebook needs to load models from disk, run evaluations and make a dataframe with the evaluations of the results.
  - Some notebooks `utils_name.ipynb` containing an explanation of the functions from `utils.py` that person `name` created. They can be used to show/explain the usage of functions in `utils.py`. Only person `name` can write and is the owner/responsible for `utils_name.ipynb`.
  - `requirements.txt` file: requirements file to ensure that your code is runable in another machine. See more details about this in Section 3.1.

2.3 Evaluation Procedure

Below there is a list of the evaluation steps for this deliverable:

- Create an environment from your requirements:

  ```
  conda create --name quora test env --file requirements.txt
  ```

- Run `reproduce_results.ipynb` within `quora_test_env` environment
- Check that the results match what you report on `main.pdf`

3 Notes on Deliverable Files

3.1 `requirements.txt`

In order to build your reproducible project, before you start you can do the following:

1. `conda create --name quora_challenge_env python=3.9`
2. `conda activate quora_challenge_env`
3. Install all your dependencies ensuring that you are in the environment
4. Make all your project code.
5. Do a `conda list -e requirements.txt`.
6. Run `conda deactivate` to go outside the `quora_challenge_env` environment.
7. Then, to ensure that everything works do:

   ```
   conda create --name quora_test_env --file requirements.txt
   ```

8. Run `reproduce_results.ipynb` within your `quora_test_env` environment

3.2 `train_models.ipynb`

The `train_models.ipynb` notebook:

- Is a responsability of all members of a group. All of you should execute this and ensure it works as expected.
- Has to use the code done by each member in the group to generate features for the challenge.

This is a Kaggle challenge: There is no validation/test data with labels.

Therefore **you have to create the following split** in order to share the same train validation and test splits across teams:

```
quora_df = pd.read_csv("./quora_data.csv")
A_df, test_df = sklearn.model_selection.train_test_split(quora_df, test_size=0.05,
    random_state=123)
train_df, val_df = sklearn.model_selection.train_test_split(A_df, test_size=0.05)
print('train_df.shape=',train_df.shape)
print('val_df.shape=',val_df.shape)
print('test_df.shape=',test_df.shape)
```

```
tr_df.shape= (291897, 6)
va_df.shape= (15363, 6)
te_df.shape= (16172, 6)
```

3.3 `reproduce_results.ipynb`

If there are random parts in the code, make sure to have seeds to make your results reproducible.

The `reproduce_results.ipynb` notebook:

- Is a responsability of all members of a group. All of you should execute this and ensure it works as expected.
- Contain train/validation/test results of ROC AUC (it is available in Scikit-learn: `sklearn.metrics.roc_auc_score`) as well as precision and recall.

Note that the teacher will only load and run this notebook, unless there is something very unclear requiring the execution of `train_models.ipynb` as well.

Additional notes:

- This notebook does not have to train anything.
- It should be relatively fast to execute (probably less than 10 minutes since there is no training).
- This notebook should only load previously trained models from disk. After loading the models, it should make predictions and compute metrics.