Natural Language Processing – Project 1: ARABIC FINE-GRAINED DIALECT IDENTIFICATION

Submission Instructions:

1. Upload your solution files to Canvas on **Sunday 22nd of November 2020**.
2. Go to the Repeat_Project_1 folder in Canvas to upload your files.
3. Once you have submitted your file you should verify that you have correctly uploaded it. It is your responsibility to make sure you upload the correct file.
4. Please make sure you **fully comment your code** that will reflect your **OWN** work.
5. You need to use **Jupyter Notebook** and run the code before the submission and you can submit a .ipynb, .pdf and/or .html files.
6. Please put **your student name and number** as comments **at the top of your file**.

This project is worth 50% of the Natural Language Processing Module.

Project Description

The objective of this project is to apply NLP techniques in order to improve Arabic dialect identification on user generated content dataset.

In this assignment, you are provided with a large-scale collection of parallel sentences in the travel domain covering the dialects of 25 cities from the Arab World plus Modern Standard Arabic (MSA). The task is to build systems that predict a dialect class among one of the 26 labels (25+ MSA) for given sentences.

Dataset

The data of this assignment is the same reported on in the following papers.

Bouamor, H., Habash, N., Salameh, M., Zaghouani, W., Rambow, O., et al. (2018). The MADAR Arabic Dialect Corpus and Lexicon. In Proceedings of the 11th International Conference on Language Resources and Evaluation. (PDF: http://www.lrec-conf.org/proceedings/lrec2018/pdf/351.pdf)

Salameh, M., Bouamor, H. & Habash, N. (2018). Fine-Grained Arabic Dialect Identification. In Proceedings of the 27th International Conference on Computational Linguistics. (PDF: http://aclweb.org/anthology/C18-1113)

Systems' Evaluation Details

Evaluation will be done by calculating microaveraged F1 score (F1μ) for all dialect classes on the submissions made with predicted class of each sample in the Test set. To be precise, we define the scoring as following:

$$P\mu = \Sigma TPi / \Sigma(TPi + FPi) \forall i \{Happy,Sad,Angry\}$$
$$R\mu = \Sigma TPi / \Sigma(TPi + FNi) \forall i \{Happy,Sad,Angry\}$$

where TPi is the number of samples of class i which are correctly predicted, FNi and FPi are the counts of Type-I and Type-II errors respectively for the samples of class i.

The final metric F1μ will be calculated as the harmonic mean of Pμ and Rμ.

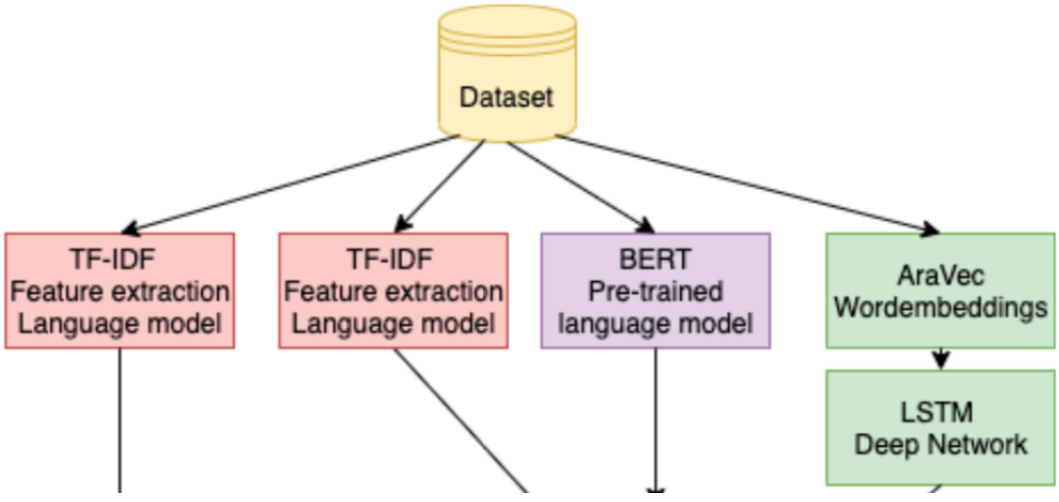
Implementation

Step1 (30%) : Data preprocessing

Download the Training and Development Data Canvas. And pre-process the dataset, i.e. cleaning, tokenization, etc.
You can find an Arabic text tokenizer (Farasa) within the Project_1 folder.
You can, also, use Camel Tools (already installed in C127 machines).
https://github.com/CAMeL-Lab/camel_tools

Step2 (70%): System implementation

Your task is to implement and compare three different text classification methods as follows.



1- Feature-Based Classification for Dialectal Arabic (20%)

Use and compare **two different feature-based** classification methods (classical Machine Learning techniques) in order to implement your Arabic dialect identification system.
Your models should apply various n-gram features as follows:

- Word-gram features with uni-gram, bi-gram and tri-gram;
- Character-gram features with/without word boundary consideration, from bi-gram and up to 5-gram.

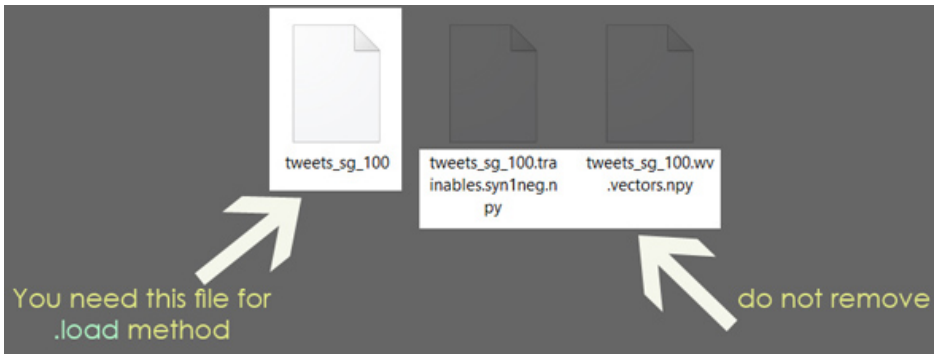
2- LSTM Deep Network (20%)

Use the Long Short Term Memory (LSTM) architecture with **AraVec** pre-trained word embeddings models. These models were built using gensim Python library. Here's the steps for using the tool:

1. Install `gensim` >= **3.4** and `nltk` >= **3.2** using either `pip` or `conda`

pip install gensim nltk
conda install gensim nltk

2. extract the compressed model files to a directory [e.g. `Twittert-CBOW`]
3. keep the **.npy** files. You are gonna to load the file with no extension, like what you'll see in the following figure.
4. run the python code to load and use the model



You can find a simple code for loading and using one the models by following these steps in the following link:

https://github.com/bakrianoo/aravec

And an example of using LSTM for text classification in the following guide:

https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17

3- BERT for Dialectal Arabic (20%)

BERT[1] or Bidirectional Encoder Representations from Transformers (BERT), has recently been introduced by Google AI Language researchers (Devlin et al., 2018). It replaces the sequential nature of RNN (LSTM & GRU) with a much faster Attention-based approach. The model is also pre-trained on two unsupervised tasks, masked language modeling and next sentence prediction. This allows you to use a pre-trained BERT model by fine-tuning the same on downstream specific tasks such as Dialectal Arabic classification.

You can employ the multi-lingual BERT that has been pre-trained on MSA and then fine tune it for dialectal Arabic.
I recommend to read the following Blog Multi-label Text Classification using BERT
https://medium.com/huggingface/multi-label-text-classification-using-bert-the-mighty-transformer-69714fa3fb3d

4- Evaluation (10%)

Use the MADAR-DID-Scorer.py scrip to evaluate your systems.

Steaming and Classifying Arabic Tweets from Twitter (Bonus)

For those of you that might be interested in applying the Dialectal Arabic classifier in real classification of Arabic tweets, it is possible to create a developer account with Twitter. You can use the Twitter API to stream tweets based on a specific keyword.
Go to the following for a guide.
https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/
Please note it can take a little bit of time to get this working. You can pipe these tweets through your model in order to classify sentiment. For example, you could increase the probability threshold for both positive and negative and classify sporting events and monitor/graph sentiment throughout the event.

Reference:

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirec- tional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[1] For more technical details see (Devlin et al., 2018).
Source: https://mc.ai/bert-explained-state-of-the-art-language-model-for-nlp/
Source: http://jalammar.github.io/illustrated-bert/
BERT multi-lingual mdoel is presented in https://github.com/google-research/bert/blob/master/multilingual.md

Submission

You are required to submit only via Canvas (no email submissions):

- An electronic copy of all source code developed (.tar.gz or .zip archive).
- Please put **your student name and number at the top of your file(s).**
- It is your responsibility to make sure you upload the correct file.

All files can be submitted with Canvas before **Sunday 22nd November 2020**. Please ensure to include **your name and student number** on the **Jupyter notebook document**.

Code

All code should be completed using Python as the programming language.
Your code should have a logical structure and a high level of readability and clarity. Please comment your code and put all code into functions. You code should be efficient and should avoid duplication.

Late submissions

If you don't get the assignments done to your satisfaction and don't meet the minimum requirements by the deadline, you have the option (as with any assignment at CIT) of submitting up to 1 week late for a penalty of 10%.

This penalty is subtractive. Work that would have earned 55% if on time, would get 45% (not 49.5%) if late.

The penalty is applied weekly. So, 1 day late costs the same 6.

If you have a specific reason for submitting a late assignment (sickness, etc) please submit directly a medical certificate in the department secretary.

Plagiarism

Please read and strictly adhere to the CIT Honesty, Plagiarism and Infringements Policy Related to Examinations and Assessments. Note that reports are **checked** against each other and against external web sources for plagiarism. Any suspected plagiarism will be treated seriously and may result in penalties and a zero grade.

Grading

The assignment is worth 50% of the overall mark for the module. Marks will be awarded based on the quality of the code and the results. In particular, I will be checking to see if you are handling and preprocessing data correctly, carrying out exploratory analysis to gain insights, correctly performing model implementation, and critically, documenting everything in a clear and concise way. The submitted code will also be checked to ensure that the work is your own.