UE 803: Data Science

Project 2023

April 3, 2023

About this project

This project falls into two main parts.

In the first part, you will compare and classify two types of texts. The emphasis in that part is on data collection, statistics, visualisation, linguistic processing and classification. You will collect wikipedia biographies for two categories of people, compute statistics about the two sets of biographies, produce vizualisation of their content and train/test a binary classifier which should automatically predict the category of a given text.

In the second part, you will compare the behaviour of two linguistic processing libraries (e.g., Stanza and Spacy) on the same set of texts. Different libraries will yield different results e.g., different tokenization, pos tags, named entities and syntactic parses. The goal of this part of the project is to analyse these differences.

Deadline

The deadline for submission is May 15th, 2023. This is a strict deadline. Late submissions will be penalised (-0.20 points per day past the deadline).

Defense

You will defend your project on May 24th, 2022. The defense is composed of a 10-minute presentation (using slides) followed by a 10-min discussion with the jury.

Part 1 – Comparing and classifying Texts

In this part, you will compare and classify two types of texts. There are three main subtasks to implement.

Data Collection (10 points)

Your code should support the extraction from the web of biographies for people belonging to two distinct categories e.g., Wikipedia biographies for astronauts vs writers[1]

[1] Cf. Exercise sessions 3 and 4.

The code should include a function which (i) returns the two sets of texts together with their classes, (ii) stores each (text, category) pair in a pandas dataframe and (ii) stores each text into a file with filename person_category.txt (e.g., AnnaKikina_Astronaut.txt), JaneAusten_Writer.txt). These output files should be included in your submission.

Data Analysis (20 points)

Using linguistic processing, this part of the code should provide an analysis of the differences between the two categories of texts. Minimally, it should provide the following statistics and visualisations.

- Vocabulary: 50 most frequent words and word cloud for each category
- Sentences: Min/max/avg number of sentences per category together with the corresponding histograms and box plots
- Tokens: Total number of token occurrences per category. Min/max/avg number of token occurrences per sentence per category.

Statistical results and visualisations should be included in your report.

Remarks. Bonus points will be granted if you consider other features (e.g. named entities, vocabularies of nouns/verbs/adjectives), statistics (e.g., correlation significance testing or correlations) and vizualisations.

Also note that you'll need to figure out which preprocessing steps are useful here e.g., should you first segment into sentences, remove stop words, normalise using lower cases etc. Make sure to report these decisions in the report as they will impact statistical results.

Classification (15 points)

Train two binary classifiers (e.g., perceptron and logistic regression) which predict which category a text belongs to and compute its results in terms of accuracy, precision, recall, F1 and confusion matrix. Discuss the results (in your report) and make sure to first split your data into train and test.

Remark. Here you'll need to think about how much data to use (you could for instance decide to use more data i.e., retrieve a higher nb of articles for each category, and inspect how your classifiers improves with increading datasize); which features and which classifiers to use (feel free to experiment with the various classifiers available in sklearn). In the report, make sure to provide a clear description of your choices, to include the results and to comment on them.

Part 2

In this part of the project, you will compare the output of two linguistic processing libraries (e.g., Stanza vs. Spacy). Using a given set of texts, you will run both libraries on these texts and report statistics on how often they (dis)agree with respect to the different levels of linguistic description.

Minimally, your code should include the following functionalities and outputs.

Data collection (5 points).

- a function for creating a dataset of texts. This function should include two parameters *src* and *n* where *src* permits specifying the source (archive, url etc.) from which the data is retrieved and *n* is the size of the data to be extracted which can be either a number of texts, of sentences or of tokens.
- The text dataset extracted by your function (or an illustrating extract if it is very large)

Sentence segmentation (5 points). We want to know which sentences are recognised by both libraries and how often they agree. Your code should:

- Compute and output the respective number of sentences produced by your two libraries for your text data. If your text data is separated into files, you can also provide the respective number of sentences per file produced by your two libraries.
- Output the set of sentences (SharedSentences) recognized by both libraries. The code should store this set into a Pandas Dataframe and into a file.

Tokenization (10 points). Your code should compute and output:

- the vocabulary (set of unique tokens) recognised by each library and their respective size

- the intersection of these two vocabularies (which tokens are part of both vocabularies?)
- the sets of tokens that is specific to each library (i.e., SpacyOnly: tokens that are in Spacy vocabulary but not in Stanza's vs. Stanza-Only: tokens that are in Stanza vocabulary but not in Spacy's).
- the set of shared token **occurrences** i.e., tokens that are identified by both libraries before (SharedTokensNosentences) and after (SharedTokensInSentences) sentence segmentation. Here we want to find in a given text, all tokens that are found by both libraries.

Remark: The first three items refer to the vocabulary ie the set of unique tokens (token types) found by each library while the fourth item focuses on token occurrences i.e., how often given a text, the two tokenizers find the same tokens.

POS tagging (30 points). For each token in SharedTokensInSentences, your code should compute and output:

- the ratio and the number of times, the token is assigned the same UPOS (universal postag[2]) by both libraries
- for each UPOS postag, the frequency mapping from one library to the other i.e., for the set of tokens labelled t by a library, the set of labels and their frequency found to this set of tokens by the other library. E.g., for all tokens labelled ADJ by Stanza, the labels found by Spacy are: (ADJ: 80%, ADV:15%, NOUN:5%)

[2] https://universaldependencies.org/u/pos/

Bonus points will be granted for extending the approach to dependency relations i.e., examining how often the two libraries agree on a given subject relation and when they disagree, finding out the frequency mapping for each dependency relation.

Expected documents, Presentations and grading

When submitting, please provide us (for each group) with a zipped directory (zip archive) containing :

- your *commented* Python source code ;
- a README file explaining how to install and run your code ;
- a REQUIREMENT file listing the libraries used by your code and their version number ;
- your extracted corpus (or an extract if too large to upload to Arche);

- other optional useful information (e.g., figure representing the model of your database).

Your zipped archive should be uploaded to Arche in the Group Project repository. Please name this zip archive using your logins at UL (example : `dupont2_martin5_toussaint9.zip`). Please also write down your first and last names within your code!

Grading will take into account the following aspects:

- Replicability (how easily your code can be run / adapted to other input data). **Make sure we can run your code and inspect results easily by giving us precise information about how to run the program, hardcoded file names (if any) and where to find input/output.**
- Code quality / readability (highly related to code comments). **Make sure we can understand your code easily. Use meaningful variable and procedure names. Include clear comments.**
- Code efficiency: how much of the required functionalities does your code covered and how well ?
- Bonus points (up to 2 points): we'll reward code that goes beyond what is required. If you want to claim bonus points, please provide a clear description of what you think justifies your claim (which part of the code goes beyond the project specification and what additional information does it provide).

Each group will present their results in a 10 minute presentation (with slides) during the last session of the class (Monday 23 May, 9-12h15). Presentations should be divided so that each student in the group presents something. The grade for the presentation is individual, the grade for the code is the same for all members of the group.

The project grade will be calculated as follows:

- 90% for the code
- 10% for the presentation

Bonus points, if any, are added to this grade.

Course Grade

The course grade will be calculated as follows:

- 20% for each of the 3 exams
- 40% for the project