UE 803: Data Science

Project 2024

April 18, 2024

About this project

This project falls into two main parts.

In the first part, you will carry out clustering using two different types of input formats. The emphasis in that part is on data collection, statistics, visualisation, linguistic processing and clustering. You will collect wikipedia biographies as well as knowledge graph facts for two categories of people, compute statistics about the two sets of persons. You will then produce visualisation of their content and train/test a clustering model which should automatically segment the data you collected.

In the second part, you will compare the behaviour of two linguistic processing libraries (e.g., Stanza and Spacy) on the task of named entity recognition (NER) on the same set of texts. Different libraries will yield different results and the goal of this part of the project is to analyse these differences. You will also develop methods for comparing the Named Entities found in a text with the RDF entities present in the corresponding graph.

Deadline

The deadline for submission is June 14th, 2024. 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 June 21st, 2024. The defense is composed of a 10-minute presentation (using slides) followed by a 10-min discussion with the jury.

Part 1 – Comparing and clustering Texts

In this part, you will compare and cluster texts from two categories of persons. There are three main subtasks to implement.

1. Data Collection (15 points)

Your code should support the extraction (i) from the web of biographies **(Text)** for people belonging to two distinct categories[1] e.g., Wikipedia biographies for Sculptors vs Computer Scientists[2]; and (ii) correspondingly, pieces of facts **(KG Graph)** about the same people from an open-source knowledge graph e.g. RDF triples from DBpedia/Wikidata etc.[3] [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., FernadoBotero_Sculptor.txt), RajReddy_ComputerScientist.txt), and (iii) a json file that holds the KG Graph for each person (which you can easily map back to each person). These output files should be included in your submission.

2. *Data Analysis (15 points)*

This part of the code should provide an analysis of the differences between the two categories of texts and graphs. Minimally, it should provide the following statistics and visualisations.

Text

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

Graphs

- RDF properties: 50 most frequent properties and property cloud for each category
- Facts: Min/max/avg number of facts per category together. Histograms and box plots for number of facts per graph for each category

Statistical results and visualisations should be included in your report.

[1] You should ensure that you have at least 100 persons for each category

[2] Cf. Exercise sessions 3 and 4.

[3] The KG graph for a person should include all facts that mention the person as a subject or an object.

[4] If you face storage constraints, you may limit the number of facts for each person to 100.

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.

3. Clustering (15 points)

Train a clustering model (e.g. KMeans) which predicts two clusters. Do this using (i) the text for each person that you collected; and separately, (ii) the fact graph for each person that you collected. [5]. Compare the clusters predicted between using the Text as input vs using the KG Graph as input; do this by computing both supervised and unsupervise metrics as well as visualisations comparing the predicted clusters. 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 number of articles for each category, and inspect how it affects the clustering results with increasing data size); which features and which algorithm to use (feel free to experiment with the various clustering algorithms 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 task of named entity recognition. For ease of comparison, ensure that both pipelines use the same tokenization.[6]

You will use the data that you collected in Part 1. Minimally, your code should include the following functionalities and outputs.

1. Named Entity Recognition (10 points)

- Write a function that will process one document (i.e. a biography) and return the set of named entities (NE) detected within it. Do this for both Spacy and Stanza. Store your results where (e.g. Pandas dataframe/json file) you can easily retrieve them to do comparisons.

- Report basic statistics about the predicted set of NEs by each package (i.e. Spacy vs Stanza). This should include:

[5] You can linearise the facts as a single string and treat it as a text e.g. (⟨ Raj Reddy, award received, AAAI Fellow ⟩, ⟨ Raj Reddy, member of, National Academy of Engineering ⟩...) may be represented as "Raj Reddy award received AAAI Fellow. Raj Reddy member of National Academy of Engineering... "

[6] `https://stanfordnlp.github.io/stanza/tokenize.html#start-with-pretokenized-text`, `https://spacy.io/api/doc#init`

(a) avg/min/max number of NEs. Report this per category per package.

(b) avg/min/max number of words in each NE. Report this per category per package.

(c) Use visualisation to compare the above 2 statistics, per category per package.

2. Named Entity Recognition: analysis by entity type (15 points)

- Write a function that checks one document (i.e. a biography) for the following:

(a) the number of spans (i.e. token(s)) where both packages agree and predict is an NE (i.e. complete overlap in span predicted).

(b) the number of spans where there is a partial agreement between both packages (i.e. partial overlap in spans predicted).

(c) for each package, the number of spans that a package predicted as an NE, but the other package did not predict as an NE.

(d) for the spans with full and partial agreement, was there an agreement in the NE type (e.g. Person, Location, Organisation etc)

(e) Use visualisation to compare the above statistics, per category per package (i.e. Spacy vs Stanza)

3. Named Entity Recognition: verification against knowledge graph (20 points)

- Write a function that checks the following for each person:

(a) for each set of NEs predicted by each package (i.e. Stanza and Spacy), how many of them can be found mentioned in the KG Graph you collected for the person?[7]

(b) return for each package (i.e. Stanza and Spacy), the ratio of predicted NEs that you can be confidently said to be in the KG graph for the person.

Bonus points

We will 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). Bonus points (up to 5 points) will be granted for:

[7] note that you should take into consideration the NE span in the text may not be an exact match with KG entity label. Here you can use regular expressions to match the NEs found in the text with the RDF entities present in the associated texts.

- Part 1: extending your data collection to more languages besides English (i.e. your method can specify which languages will be collected). (1 point)
- Part 2: extending your code (only on the verification, i.e. Q3) to consider co-reference resolution [8]. (3 points)

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 ?

Each group will present their results in a 10 minute presentation (with slides) during the last session of the class (Friday June 21st, 2024, 9h00-12h15). Presentations should be divided so that each student in the group presents something. The grade for the presentation

[8] `https://stanfordnlp.github.io/stanza/coref.html`, `https://spacy.io/api/coref`, `https://staedi.github.io/posts/coref`

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