UE 803: Data Science

Project: Clustering and Classifying People based on Text and KB information

05 April 2021

About this project

In this project, you will collect information about a people belonging to different *categories* (such as artists, politician, sportspeople, etc.). Based on this information, you will then try to *automatically cluster and classify* these people into the correct categories.

We will use two sources of information, namely:

- The Wikipedia online encyclopedia[1]
- The Wikidata knowledge base[2]

[1] See `https://www.wikipedia.org`

[2] See `https://www.wikidata.org/wiki/Wikidata:Main_Page`

Deadline

The deadline for submission is May 15th, 2021. This is a strict deadline. Late submissions will be penalised (-0.20 points per day past the deadline).

Exercise 1 – Corpus extraction (10 points)

The goal of this exercise is to compile a text corpus from *the Wikipedia online encyclopedia*. This corpus will be made of plain text sentences, selected so as to have a *roughly* balanced corpus in terms of training data (each target category should be associated with the same number of sentences). We will focus on the 6 following categories: architects, mathematicians, painters, politicians, singers and writers.

Concretely, you are required to implement a Python program which takes as a an input:

- a number k of persons per category,

- a number n of sentences per person (persons whose wikipedia description is too short to have n sentences, should be ignored).

This extraction task can be realised through the following steps:

1. Create a list of persons you want to work with. Those persons should fall into two main types: artists (A) and non artists (Z). Singers, writers and painters are of type A while architects, politicians and mathematicians of type Z. For each category, select 30 persons of that category. So in total you should have a list of 180 persons (half of them are artists and half of them are not).

   You can use the wikidata warehouse to find persons of the expected categories. More precisely, the Wikidata collection can be filtered out using the SPARQL language and the following endpoint: `https://query.wikidata.org/`.

   You can thus use the `SPARQLwrapper` python library to apply a SPARQL query to the Wikidata warehouse and retrieve the required item identifiers.

2. for each selected person, retrieve his.her Wikidata description and Wikipedia page title. This can be done using the wikidata API along with the `wptools` python library.

3. Once you have a list of wikipedia page titles, fetch (if it exists) the corresponding English wikipedia page, and extract the n first sentences of its content.

At the end of this process, you will have in memory, for each person you selected:

- the text of the corresponding Wikipedia page ;
- the corresponding data type (A or Z) and category ;
- the WikiData description.

Store these into a csv or a json file and save it on your hard drive.

NB: Note there is some overlapping with lab session 1.

Question 1 is related to *lecture 6* of the course [link].

As an example of a SPARQL query on wikidata, the following clause retrieves the wikidata items corresponding to persons who were born in New York City:

```
select distinct ?item where {
  ?item wdt:P31 wd:Q5;  #Any instance of a human
        wdt:P19 wd:Q60. #Who was born in NYC
}
```

Question 2 is related to *lecture 3* of the course [link].

As an example of a wikidata extraction, here is the code snippet to get the wikidata description and wikipedia page title, given a wikidata item identifier:

```
>>> import wptools
>>> page = wptools.page(wikibase='Q715127')
>>> page.get_wikidata()
>>> page.data['description']
>>> page.data['title']
```

Question 3 is related to *lecture 3* of the course [link].

As an example of a wikipedia page content retrieval:

```
>>> import wikipedia
>>> page2 = wikipedia.page('Paul_Sweezy')
>>> page2.content
```

Exercise 2 – Pre-processing, Clustering and Classifying (10 points)

Pre-processing (3 points)

Before applying machine learning algorithm to your data, you first need to process text. For each Wikipedia text collected, do the following:

- tokenize the text

- lowercase the tokens
- remove function words

Do the same for the Wikidata description and store the results in a panda dataframe containing the following columns.

- column 1: person
- column 2: Wikipedia page text
- column 3: Wikipedia page text after preprocessing
- column 4: Wikidata description
- column 5: Wikidata description after preprocessing

Note. To improve clustering and classification results, feel free to add further pre-processing steps (eg Named entity recognition, pos-tagging and extraction of e.g., nouns and verbs).

Exercise 3 – Clustering (3 points)

The goal of this exercise is (i) to use the collected data (text and wikidata descriptions) to automatically cluster persons first, using 2 clusters and second, using 6 clusters and (ii) to compare the results obtained when using three ways of representing text (tokens, token frequency and tf-idf) and 2 vs. 6 clusters.

Your code should include the following functions:

- a function to train a clustering algorithm on some data using N clusters and some input representation method M (tokens, token frequency and tf-idf). Data, M and N should be parameters of that function.
- a function to compute both intrinsic (Silhouette coefficient) and extrinsic (homogeneity, completeness, v-measure, adjusted Rand index) evaluation scores for clustering results.
- a function to visualise those metrics values for each of the three input representations (tokens, token frequency and tf-id) and for 2 vs. 6 clusters (so your visualisation should display 5 scores for each of the 6 clustering results).

Exercise 4 – Classifying (4 points)

Since you know which category and subcategory each person in your dataset belongs to, you can also learn a classifier and check how well

it can predict the category and subcategory a person in your dataset belongs to.

Your code should include:

- a function which outputs accuracy, a confusion matrix, precision, recall and F1 for the results of your classification (when classifying into categories and when classifying into subcategories)
- a function which outputs a visualisation of the accuracy of your classifier per category and per subcategories

Note. For both clustering and classifying, feel free to experiment outwith the given requirements e.g., providing additional information about the data through vizualisation techniques of your choice, using additional features (e.g., learning a topic model and using topic information as an additional feature), analysing and visualising your results (eg plotting the loss for dev and train against the quantity of data used for training), comparing results when using all text data vs. using only the wikidata description or only the Wikipedia text etc.

Expected documents, Presentations and grading

At the end of the session, please provide us (for each group) with a zipped directory (zip archive) containing :

- your *commented* Python source code ;
- your extracted corpus ;
- a README file explaining how to install and run your code ;
- other optional useful information (e.g., figure representing the model of your database).

Please name your directory and zip file after the concatenation of your family names followed by "_UE803" (e.g., `MartinDupont_UE803.zip`) and send it to `claire.gardent` and `yannick.parmentier` at `loria.fr`

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 ?
- Corpus quality (expected content)
- Bonus points (up to 2 points): we'll reward code that goes beyond what is required and which either provides valuable additional information about the collected data and the clustering/classification results or improve clustering/classification results by investigating options not required by the project specification. 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 (Thursday 27 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.