UE 803: Data Science

Project: Clustering and Classifying Articles based on Text and knowledge-base information

March 30, 2022

About this project

In this project, you will collect information about articles belonging to different *categories* (such as airports, artists, politicians, sportspeople, etc.). Based on this information, you will then try to *automatically cluster and classify* these articles 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, 2022. 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 23rd, 2022. The defense is composed of a 10-minute presentation (using slides) followed by a 10-min discussion with the jury.

Exercise 1 – Corpus extraction (10 points)

The goal of this exercise is to compile a **parallel corpus** from *the Wikipedia online encyclopedia*. This corpus will be made of :

(i) plain text sentences[3],

(ii) key-value pairs corresponding to articles' infobox (if any),

[3] 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)

(iii) triples coming from articles' corresponding wikidata page.

We will focus on the following categories:

- `Airports`
- `Artists`
- `Astronauts`
- `Building`
- `Astronomical_objects`
- `City`
- `Comics_characters`
- `Companies`
- `Foods`
- `Transport`
- `Monuments_and_memorials`
- `Politicians`
- `Sports_teams`
- `Sportspeople`
- `Universities_and_colleges`
- `Written_communication`

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

- a number k of articles per category,
- a number n of sentences per article (articles whose content is too short should be ignored).

This extraction task can be realised through the following steps:

1. For each category c from the list above, retrieve k wikipedia articles belonging to c.

   Note that you can use DBpedia to find Wikipedia articles belonging to a given category.

2. For each selected Wikipedia article, retrieve:

   (a) the corresponding Wikipedia page content,

   (b) the corresponding infobox,

   (c) the corresponding wikidata statements (triples).

At the end of this process, you will have in memory, for each article you selected in step 1:

- the text of the corresponding Wikipedia page ;
- the infobox content ;
- the wikidata statements (triples).

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.

Step 1 is related to *lecture 6* of the course.
The following clause for instance retrieves 100 articles which belong to the Comedy_films category:

```
PREFIX dcterms:
  <http://purl.org/dc/terms/>
PREFIX dbc:
  <http://dbpedia.org/resource/Category:>

SELECT ?film WHERE {
  ?film
    dcterms:subject/skos:broader*
      dbc:Comedy_films .
}
LIMIT 100
```

Step 2(a) is related to *lecture 3* of the course.
The following code snippet for instance retrieves the wikipedia page content of a given article:

```
>>> import wptools
>>> page=wptools.page('Stephen Fry',silent=True)
>>> page.get_query()
>>> page.data['extract']
'<p><b>Stephen Fry</b>...'
```

Step 2(b) is related to *lecture 3* of the course.
The following code snippet for instance retrieves the wikipedia infobox of a given article:

```
>>> import wptools
>>> page=wptools.page('Stephen Fry',silent=True)
>>> page.get_parse()
>>> page.data['infobox']
{'name': 'Stephen Fry', 'image': 'Stephen ...'}
```

Step 2(c) is related to *lecture 3* of the course.
The following code snippet for instance retrieves the wikidata statements of a given article:

```
>>> import wptools
>>> page=wptools.page('Stephen Fry',silent=True)
>>> page.get_wikidata()
>>> page.data['wikidata']
{...
  u'place of birth (P19)': u'Hampstead (Q25610)',
  u'religion (P140)': u'atheism (Q7066)',
  u'sex or gender (P21)': u'male (Q6581097)',
  ...}
```

Exercise 2 – Pre-processing, Clustering and Classifying (20 points)

Pre-processing (8 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 punctuation and 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); or/and to also pre-process the wikidata statements and the infobox content (note that these are not standard text however).

Here is an example query which retrieves Barack Obama's (Q76) description from Wikidata:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX schema: <http://schema.org/>

SELECT ?o
WHERE
{
  wd:Q76 schema:description ?o.
  FILTER ( lang(?o) = "en" )
}
```

Output:
44th president of the United States, from 2009 to 2017

Clustering (8 points)

The goal of this exercise is to use the collected data (text and wikidata descriptions) to automatically cluster the Wikipedia documents first, using 16 clusters and second, experimenting with different numbers of clusters.

Your code should include the following functions:

- a function to train a clustering algorithm on some data using N clusters
- 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 values of N ranging from 2 to 16.

Classifying (4 points)

Since you know which category each document in your dataset belongs to, you can also learn a classifier and check how well it can predict the category of each document in your dataset.

Your code should include:

- a function which outputs accuracy, a confusion matrix, precision, recall and F1 for the results of your classifier
- a function which outputs a visualisation of the accuracy of your classifier per category

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 ;
- 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 ;
- 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 ?
- 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 (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.