# Data Science Lab
Lab #5

Politecnico di Torino
November 6-7, 2019

## Intro

The main objective of this laboratory is to put into practice what you have learned on clustering techniques and itemset mining. You will mainly work on textual data, a domain where the data preparation phase is crucial to any subsequent task. Specifically, you will try to detect topics out of a set of real-world news data. Then, you will describe each cluster through frequent itemset mining.

**Important note.** For what concerns this laboratory, you are encouraged to upload your results to our online verification platform, even though the submission will not count on your final exam mark. Doing so, you can practice with the same system that you will use for the final exam. Reference Section 3 to read more about it.

## 1 Preliminary steps

### 1.1 Useful libraries

As you may have already understood, the Python language comes with many handy functions and third-party libraries that you need to master to avoid boilerplate code. In many cases, you should leverage them to focus on the analysis process rather than its implementation.

That said, we listed a series of libraries you can make use of in this laboratory:

- NumPy
- scikit-learn
- Natural Language Toolkit
- SciPy

We will point out their functions and classes when needed. In many cases, their full understanding decreases significantly your programming effort: take your time to explore their respective documentations.

**Warning:** we have noticed from previous laboratories that in some cases copying snippets of code directly from the PDF file leaded to wrong behaviours in Jupyter notebooks. Please consider to write them down by yourself.

### 1.2 wordcloud

Make sure you have this library installed. As usual, if not available, you need to install it with `pip install wordcloud` (or any other package manager you may be using). The `wordcloud` library is a word cloud generator. You can read more about it on its official website.

## 1.3 Datasets

For this laboratory, a single real-world dataset will be used.

### 1.3.1 20 Newsgroups

The 20 Newsgroups dataset was originally collected in Lang 1995. It includes approximately 20,000 documents, partitioned across 20 different newsgroups, each corresponding to a different topic.

For the sake of this laboratory, we chose $T \leq 20$ topics and sampled uniformly only documents belonging to them. As a consequence, you have $K \leq 20,000$ documents uniformly distributed across $T$ different topics. You can download the dataset at:

`https://github.com/dbdmg/data-science-lab/blob/master/datasets/T-newsgroups.zip?raw=true`

Each document is located in a different file, which contains the raw text of the news. The name of the file in an integer number and corresponds to its ID.

## 2 Exercises

Note that exercises marked with a (*) are optional, you should focus on completing the other ones first.

### 2.1 Newsgroups clustering

In this exercise you will build your first complete data analytics pipeline. More specifically, you will load, analyze and prepare the newsgroups dataset to finally identify possible clusters based on topics. Then, you will evaluate your process through any clustering quality measure.

1. Load the dataset from the root folder. Here the Python's os module comes to your help. You can use the `os.listdir` function to list files in a directory.

2. Focus now on the data preparation step. As you have learned in laboratory 2, textual data needs to be processed to obtain a numerical representation of each document. This is typically achieved via the application of a weighting schema.
   Choose now one among the weighting schema that you know and transform each news into a numerical representation. The Python implementation of a simple *TFIDF* weighting schema is provided in section 2.1.1, you can use it as starting point.

   This preprocessing phase is likely going to influence the quality of your results the most. Pay enough attention to it. You could try to answer the following questions:

   - Which weigthing schema have you used?
   - Have you tried to remove stopwords?
   - More generally, have you ignored words with a document frequency lower than or higher than a given threshold?
   - Have you applied any dimensionality reduction strategy? This is not mandatory, but in some cases it can improve your results. You can find more details in Appendix 3.3.

3. Once you have your vector representation, choose one clustering algorithm of those you know and apply it to your data.

4. You can now evaluate the quality of the cluster partitioning you obtained. There exists many metrics based on distances between points (e.g. the Silhouette or the Sum of Squared Errors (SSE)) that you can explore. Choose one of those that you known and test your results on your computer.

5. Consider now that our online system will evaluate your cluster quality based on the real cluster labels (a.k.a. the *ground truth*, that you do not have). Consequently, it could happen that a cluster subdivision achieves an high Silhouette value (i.e. *geometrically* close points were assigned to the same cluster) while the matching with the real labels gives a poor score (i.e. real labels are heterogeneous within your clusters).
   In order to understand how close you came to the real news subdivision, upload your results to our online verification system (you can perform as many submission as you want for this laboratory, the only limitation being a time limit of 5 minutes between submissions). Head to Section 3 to learn more about it.

#### 2.1.1 A basic TFIDF implementation

The transformation from texts to vector can be simplified by means of ad-hoc libraries like Natural Language Toolkit and scikit-learn (from now on, `nltk` and `sklearn`). If you plan to use the *TFIDF* weighting schema, you might want to use the sklearn's `TfidfVectorizer` class. Then you can use its `fit_transform` method to obtain the *TFIDF* representation for each document. Specifically, the method returns a SciPy sparse matrix. You are encouraged to exhaustively analyze TfidfVectorizer's constructor parameters since they can significantly impact the results. Note for now that you can specify a custom `tokenizer` object and a set of stopwords to be used.

For the sake of simplicity, we are providing you with a simple tokenizer class. Note that the TfidfTokenizer's tokenizer argument requires a callable object. Python's callable objects are instances of classes that implement the `__call__` method. The class makes use of two nltk functionalities: word_tokenize

and the class WordNetLemmatizer. The latter is used to lemmatize your words after the tokenization. The *lemmatization* process leverages a morphological analysis of the words in the corpus with the aim to remove the grammatical inflections that characterize a word in different contexts, returning its base or dictionary form (e.g. {am, are, is} ⇒ be; {car, cars, car's, cars'} ⇒ car).
For what concerns the stop words, you can use again a nltk already-available function: stopwords.
The following is a snippet of code including everything you need to get to a basic *TFIDF* representation:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords as sw


class LemmaTokenizer(object):
    def __init__(self):
        self.lemmatizer = WordNetLemmatizer()

    def __call__(self, document):
        lemmas = []
        for t in word_tokenize(document):
            t = t.strip()
            lemma = self.lemmatizer.lemmatize(t)
            lemmas.append(lemma)
        return lemmas


lemmaTokenizer = LemmaTokenizer()
vectorizer = TfidfVectorizer(tokenizer=lemmaTokenizer, stop_words=sw.words('english'))
tfidf_X = vectorizer.fit_transform(corpus)
```

## 2.2 Cluster characterization by means of word clouds and itemset mining

In many real cases, the *real* clustering subdivision is not accessible at all[1]. Indeed, it is what you want to discover by clustering your data. For this reason, it is commonplace to add a further step to the pipeline and try to characterize the clusters by inspecting their points' characteristics. This is especially true while working with news, where the description can lead to the identification of a topic shared among all the documents assigned to it (e.g. one of your clusters may contain news related to sports).

In this exercise you will exploit word clouds and frequent itemset algorithms to characterize the clusters obtained in the previous exercise.

1. Split your initial data into separate chunks accordingly to the cluster labels obtained in the previous exercise. For each of them, generate a Word Cloud image using the `wordcloud` library. Take a look at the library documentation to learn how to do it. Can you figure out the topic shared among all the news of each cluster?

2. (*) Provide a comment for each word cloud and upload the images and the comments. Head to section 3 to know how.

3. (*) One further analysis can exploit the frequent itemset algorithms. Choose one algorithm and run it for each cluster of news. Try to identify the most distinctive set of words playing around with different configurations of the chosen algorithm. Based on the results, can you identify any topic in any of your clusters?

[1]Or worse, it might not *exist* at all.

## 3 Submitting you work

For this laboratory, you should upload two files to two different web sites. The first file contains the clustering results, the second file contains a report on the identified clusters. The following sections provide further details on that.

**Deadline.** You can submit your work until **Monday November 11, 11:59PM**. Later submissions will not be evaluated.

### 3.1 Upload clustering results

You are required to upload a single CSV file. Please respect the following requirements:

- use the format UTF-8 (see `open()`'s `encoding` parameter)
- the first line must contain a two columns header equal to: `Id,Predicted`
- each of the $N$ following lines must contain the document ID followed by the cluster ID assigned to it. IDs must be increasing integer numbers starting from $0$.

The file must be uploaded to our submission platform located at http://35.158.140.217/. You can find an example file on the course website.
Please reference the guide from the course website to go through the submission procedure.

### 3.2 Upload your report

You are required to upload a single PDF file. Please respect the following requirements:

- state clearly how many clusters you have found;
- for each of them, report the description that you obtained in exercise 2.2.

If you have developed your solution on a Jupyter notebook, you can export it as a PDF and use it for the submission. However the file must be sufficiently commented.
The file must be uploaded to the "Portale della Didattica", under the Homework section of the course. Please use as description: `report_lab_5`.

### 3.3 Evaluation

Your clustering results will be evaluated via the Rand-index score. You can read more about it on Wikipedia. Your report will be evaluated via the old, always-working human reading.

## Appendix

### Notions on linear transformations and dimensionality reduction

In many real cases, your data comes with a large number of features. However, there is often a good chance that some of them are uninformative or redundant and have the only consequence of making your analysis harder. The simplest example could be features that are linearly dependent with each other (e.g. a feature that describes the same information with different, yet correlated units, like degrees Celsius and Fahrenheit).

One additional detail can be addressed in your preprocessing step, other than the dimensionality reduction. There might be cases where the distribution of your data has hidden underlying dynamics that could be enhanced by choosing different features (i.e. dimensions). Figure 1 shows several points distributed in a Gaussian cluster (see Laboratory 4). Let's make now an assumption: quantitatively we assess that directions with largest variances in our space contain the dynamics of interest. In Figure 1 the direction with the largest variance is not (1,0) nor (0,1), but the direction along the long axis of the cluster.

As you will soon learn, in the known literature there is a proven method that addresses the aforementioned problems: the *Principal Component Analysis* (PCA). This technique frames the problem as a linear change of basis. The final basis vector are commonly termed *principal components*. Let's qualitatively understand why and how it is made by means of a few algebraic notions.

In PCA we assume that there exist a more meaningful basis to re-express our data. The hope is that this new basis will filter out the noise and reveal hidden dynamics. Following the assumption presented beforehand, the new basis must align the directions with the highest variance. Also, the change of basis follows another strict, yet powerful assumption: it is assumed that the new basis is a linear combination of the original one (studies expanded on this to non linear domains). In Figure 1 PCA would likely identify a new basis in the directions of the two black arrows.
In other words, if we call **X** the original set of data, in PCA we are interested in finding a matrix **P** that stretches and rotates our initial space to obtain a more convenient representation **Y**:

$$\boldsymbol{PX} = \boldsymbol{Y} \tag{1}$$

Now that foundations are laid, we know that we are looking for a new basis that highlights the inner dynamics and we assume that the change of basis can be achieved with a simple linear transformation. This linearity assumption let us solve analytically the problem with matrix decomposition techniques. Even if the simpler eigendecomposition can be used, the state-of-the-art solution is obtained through the *Singular Value Decomposition* (SVD).

Many, many theoretical and technical details have been left behind in this short summary. If you are willing to learn more, you can find a thorough tutorial about PCA, SVD and their relationship in Shlens 2014.

The use of the PCA algorithm via SVD decomposition in Python is straightforward. The following lines show how you can apply the change of basis transforming your data.

```python
from sklearn.decomposition import TruncatedSVD

# X: np.array, shape (1000, 20)
svd = TruncatedSVD(n_components=5, random_state=42)
red_X = svd.fit_transform(X)     # red_X will be: np.array, shape (1000, 5)
```

Note that the `TruncatedSVD` class lets you choose how many top-principal components to retain (they are ranked by explained variance). Doing so, you will be applying the dimensionality reduction at the same time.

## References

[1] Ken Lang. "Newsweeder: Learning to filter netnews". In: *Proceedings of the Twelfth International Conference on Machine Learning*. 1995, pp. 331–339.

[2] Jonathon Shlens. "A Tutorial on Principal Component Analysis". In: *CoRR* abs/1404.1100 (2014). arXiv: 1404.1100. URL: http://arxiv.org/abs/1404.1100.

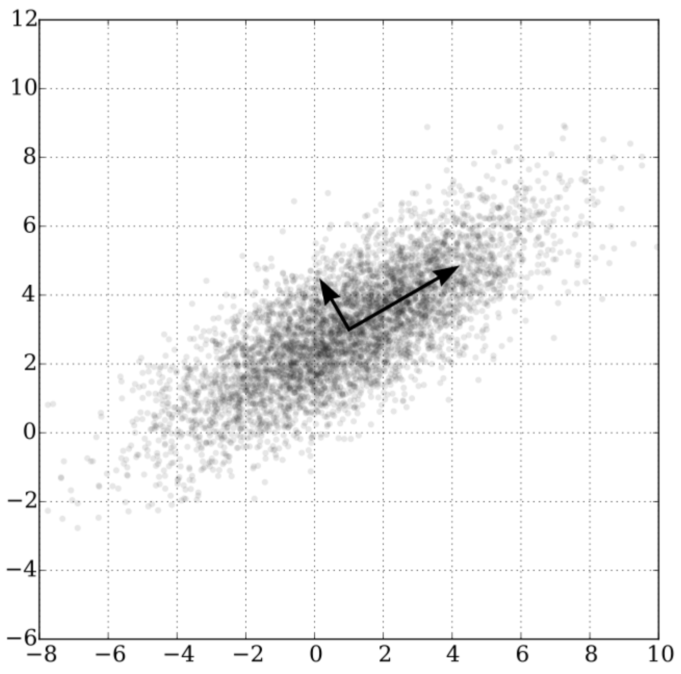

Figure 1: Points distributed in a Gaussian cluster with mean (1,3) and standard deviation 3. Source: Wikipedia