# Big data computing - Homework 2

Giulia Muscarà 1743261

December 23, 2020

# Objectives

The purpose of this homework was to process and analyse a dataset of reviews in order to cluster them with respect to two predominant topics. The dataset is made of english written documents, with noise introduced by non-english words, special characters, typing mistakes and english abbreviations, for a total of 314808 reviews.

# Data preprocessing and sampling

Given that the corpus of the reviews and the respective classification labels were provided in two different txt files, they were merged into one single csv file with headers "line" and "label" in order to have all the needed information in one place. This conversion was carried out in the python code "file_conversion.py", taking in input the original files "corpus.txt" and "labels.txt" and outputting the merged csv file "csv_file.csv".
Then the whole set of reviews was preprocessed through lemmatization, in order to reduce the noise in the text and to pave the way to a more successful clustering. Lemmatizion was preferred to stemming because it led to better results. Indeed, stemming just removes or stems the last few characters of a word, often leading to incorrect meanings and spelling. Lemmatization, instead, considers the context and converts the word to its meaningful base form thanks to the use of pos tags. Pos tags are special labels assigned to each token in the text corpus to indicate the part of speech and other grammatical categories, to allow a more meaningful lemmatizion. To start with, all occurrences of dots, dashes and backslahses were replaced with spaces. This is due to the fact that in many cases, spaces were missing after the end of a sentence or that these symbols appeared often in between of two consecutive terms. Then, english stop words were removed using the set provided by NLTK library. Other words or english typing abbreviations that were particulary recurrent in the text and not relevant to the distinction of the topics were added to the set of stop words so as to remove them as well from the tokenized reviews. Words were lower-cased and every token that was not entirely made of literals was not taken into consideration to exclude digits, leftover symbols or noisy words. The dataset was lemmatized in the "lemmatization.py" code, taking as input the "csv_file.csv", that weighted 149 $MB$, and outputting the "lemmatized_file.csv", of 77 $MB$. Lemmatization was performed on the whole dataset because in some cases using the entire lemmatized dataset led to better results in terms of accuracy and topics clustering and was still feasible in terms of memory usage and running time. This is why sampling was done after and not before lemmatization.
Sampling was carried out in the "sampling.py" code, taking in input the "lemmatized_file.csv" and producing the "sampled_file.csv". First of all, the size of the sample was chosen to be large enough to have confidence that it could accurately represent the original population. The chosen confidence level was of 95%, while the accepted margin of error $e$ was of 1%. The assumption that the target proportion $p$ was of 50% was made. Through Cochran's formulae [1] the size $n$ of the sample that fulfills the desired confidence level and margin of error could be computed statistically, where $n$ is:

$$n = \frac{n_0}{1+\frac{n_0-1}{N}}, \quad n_0 = Z^2 \frac{p(1-p)}{e^2}$$

N is the size of the original dataset and Z is the z-score that can be computed from the confidence level. In this case it amounts to 1.96. Using the formulae, the sample size should be approximately 3% of the original dataset. After computing the size, sampling was carried out using a hash function that maps each line uniformly into one-hundred buckets and keeps the reviews that were hashed in the first three buckets, so as to keep about the 3% of the total. The number of documents in the sampled file is 9484.

# KMeans and MiniBatchKMeans

Two variants of KMeans were used in the file "kmeans.py", classic KMeans and MiniBatchKMeans. MiniBatchKMeans reduces the computational cost by not using all the dataset at each iteration but a subsample of a fixed size [2]. Given that it was feasible to perform both algorithms over the entire lemmatized file in terms of memory usage, the code was executed on the entire dataset. The set of documents was vectorized and transformed into a matrix of Tf-Idf feature vectors through Tf-Idf vectorizer. The metrics retrieved both from kmeans and mini batch kmeans were computed by comparing the original labels with those newly generated by the algortihms. Kmeans does not assign specific labels to clusters, so at every execution clustering labels may be reversed. This could lead to mistaken values of the accuracy, in the case when the labels of kmeans and the original ones are inverted. This is why $1 - accuracy$ was printed if accuracy was lower than 50%, as it would be sufficient to invert the labels to get the right value. The code takes in input the "lemmatized_file.csv" and

it outputs the ten most relevant words of both clusters. Accuracy rate for both algorithms floats depending on which local minimum the algorithm starts its computation from, as they are not deterministic. Nontheless, accuracy of kmeans is stable on 80% with a minor centesimals error. Instead, even if the obtained accuracy with mini batch KMeans is generally higher, it is subject to an error of about 0.5 and it is around 84%. Topics are well separated in both cases and the first ten terms for each cluster are shown in Fig. 1 and Fig. 2. However, MiniBatchKMeans gives a better accuracy and has a lower running time.

## Truncated Singular Value Decomposition

The purpose of applying truncated singular value decomposition was to reduce the large number of features of the dataset after vectorization to a dataset containing significantly fewer values, but still containing a large fraction of the variability present in the original one. Truncated SVD computes a reduced rank approximation to the input matrix by setting all but the first k largest singular values equal to zero and using only the first k columns of U and V. Contrary to PCA, data is not centered beforehand and this is why it can work with sparse matrices efficiently. This is why it was possible to run the "svd.py" code on the whole dataset in terms of memory usage. Indeed, the code takes as input the entire "lemmatized_file.csv". To produce labels for documents and compute an accuracy rate, kmeans was performed over the SVD components. Even though after applying only SVD the first ten terms were initially mixed for the two topics, as both contained terms about animals, after performing kmeans topics division improved and the accuracy amounted to 84.2% with a minor decimals error. Fig. 3 shows the plotting of the cloud words for the ten most important terms for each cluster.

## Principal Component Analysis

Principal component analysis (PCA) is a technique for reducing the dimensionality of the dataset after vectorization, increasing interpretability and minimizing information loss, creating uncorrelated variables that maximize variance [3]. Due to the fact that PCA algorithm does not take as input sparse matrices, the matrix resulting from vectorization was turned into a ndarray, an array with the same shape and containing the same data represented by the sparse matrix with the requested memory order. The algorithm only operates on dense matrices because the covariance matrix must be computed for PCA, which requires operating on the entire matrix. However, creating a dense matrix starting from the entire dataset caused a memory error, because of its high dimensions. This is why the code "pca.py" takes as input the "sampled_file.csv". Then, the matrix was not scaled through the Standard Scaler, as input data did not belong to different ranges of values. Indeed, applying the scaler's $fit\_transform$ function, the accuracy rate would decrease significantly. To produce labels for documents and compute an accuracy rate, kmeans was then performed over the PCA components. The accuracy after applying PCA and kmeans on the sampled data amounts to 81% with a minor centesimals error and, indeed, topics are clearly defined by the ten most important words of each cluster as shown in Fig. 4.
PCA was also applied through data centering and singular value decomposition in the code "svd+centering.py", taking in input the sampled data as well. After vectorization, the resulting matrix was turned into a ndarray for the same reasons as above and data was centered by subtracting the mean of a variable from each data point such that the new variable's means was zero. Then Truncated SVD was applied to the resulting matrix, followed by kmeans. The accuracy rate obtained from kmeans is 87%. The result is stable with minor oscillations in centesimals. Topics are neatly separated and defined by the ten most important words for each cluster, shown in Fig. 5.

## Conclusions

Basing on the ten most important words obtained by applying all the different dimensionality reduction and clustering techniques, it was clear that the first topic concerned animals and the second one was about babies. The most reliable technique for their clustering was PCA implemented through data centering and SVD, with a stable accuracy rate of 87% and with a neat distinction between the two topics that were identified. The cloud words produced from this version of PCA in Fig. 5 show the best topic division obtained. In the cloud words representation, the bigger the size of the words, the higher their importance in the cluster.


Figure 1: KMeans cloud words


Figure 2: MiniBatchKMeans cloud words


Figure 3: Truncated SVD cloud words


Figure 4: PCA cloud words

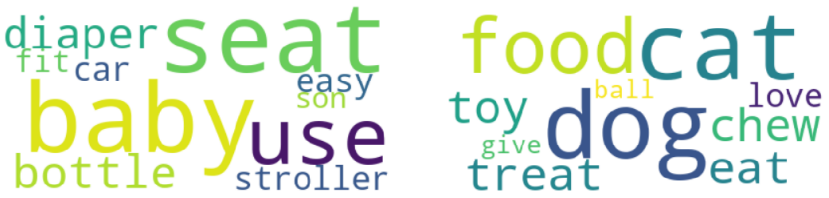


Figure 5: SVD and data centering cloud words

# References


[1] Cochran, W.G., *Sampling techniques*, Harvard University, John Wiley & sons, 1977.

[2] Bejar, J., *K-means vs Mini Batch K-means: A comparison*, Universitat Politècnica de Catalunyas, `https://upcommons.upc.edu/bitstream/handle/2117/23414/R13-8.pdf`, 2013.

[3] Leskovec, J., *Mining Massive Datasets*, Stanford University, 2011.