

# **Report of the course project: “Spot the Bot”**

Afanasiev Sergey

24.01.2022

# Contents

<b>Abstract</b> .....	3
<b>1. Instructions for Spot the bot</b> .....	4
1.1. Main steps of the project: .....	4
1.2. Table with useful books/code: .....	4
1.3. Instructions step by step: .....	5
<b>2. The work report (English)</b> .....	9
2.1. Collecting data .....	9
2.2. Preprocessing .....	9
2.3. Generating texts .....	10
2.4. Creating dictionary with the SVD method .....	12
2.5. Creating dataset .....	12
2.6. Clustering .....	12
2.7. Comparing clusterizations .....	13
<b>3. The work report (Spanish)</b> .....	17
3.1. Collecting data .....	17
3.2. Preprocessing .....	17
3.3. Generating texts .....	18
3.4. Creating dictionary with the SVD method .....	20
3.5. Creating dataset .....	20
3.6. Clustering .....	20
3.7. Comparing clusterizations .....	21
<b>4. The work report (Italian)</b> .....	25
4.1. Collecting data .....	25
4.2. Preprocessing .....	25
4.3. Generating texts .....	26
4.4. Creating dictionary with the SVD method .....	28
4.5. Creating dataset .....	28
4.6. Clustering .....	28
4.7. Comparing clusterizations .....	29
<b>5. Applications</b> .....	33
5.1. Jupyter notebooks for spot the bot .....	33
5.2. LSTM generator (jupyter notebooks) .....	33
5.3. Nonparametric tests for averages (jupyter notebooks) .....	33

# Abstract

In this course project, the possibility of machine learning algorithms to identify texts generated by a bot was studied. In this research, texts written by a human were compared with texts generated by a bot using text clustering methods and a comparison of the resulting clusters for human texts and the bot texts.

For this research, three languages were selected:

1. 1. English
2. 2. Spanish
3. 3. Italian

All texts were preprocessed using standard NLP methods:

1. Replacement of abbreviations (for English only);
2. Removing punctuation marks and special characters;
3. Removing html tags;
4. Removing diacritics;
5. Deleting book descriptions;
6. Removing stop words;
7. Lemmatization;
8. Tokenization;
9. Entity recognizer.

To generate texts, a recurrent neural network with two LSTM layers was used, trained on literature corpus. This neural network generates texts based on the first given words.

After preprocessing data (for human and the bot datasets), corpuses were created from the texts: each line is a separate preprocessed text. Next, TfidfVectorizer and SVD transformation were applied to the corpus (separately for human and generated texts).

Also, after the TF-IDF transformation, the most frequent and most rare words that have poor informative value in classification tasks (comparing two groups of texts) were removed.

The final datasets for human texts and bot texts consisted of bigrams (vector representations of all pairs of adjacent words in the text).

Clustering was carried out on the final datasets using the Wishart algorithm. The clustering hyperparameters `significance_level` and `wishart_neighbors` were selected using the metric `calinski_harabaz_score`.

The difference between human texts and bot texts was revealed by comparing the integral characteristics of the resulting clusterizations:

- Median number of vectors in a cluster
- Median distance to centroid
- Mean distance to centroid

The results showed a statistically significant difference between human and bot texts.

More detailed information is provided in the following sections of this report.

All python-codes for this research are attached in Section 5 and posted on github:

[Spot the bot course project](#)

# 1. Instructions for Spot the bot

## 1.1. Main steps of the project:

- Collect human-written text (literature)
- Create bot-written texts
- Create dictionary (word-vector) using the svd method
- Create datasets for human- and bot-written texts
- Cluster both datasets using the Wishart algorithm
- Compare two clusterizations and try to find a statistically significant difference

## 1.2. Table with useful books/code:

Name	Link	Description
Latent Semantic Mapping: Principles & Applications Jerome R. Bellegarda	<a href="https://drive.google.com/file/d/1fPBBPR8hg1C7x2fwBb3B97dqARhHlsrl/view?usp=sharing">https://drive.google.com/file/d/1fPBBPR8hg1C7x2fwBb3B97dqARhHlsrl/view?usp=sharing</a>	You can find the description of svd method
DATA CLUSTERING Algorithms and Applications Charu C. Aggarwal, Chandan K. Reddy	<a href="https://drive.google.com/file/d/1OkMNekfF5-e0ulPNGO4escJ6HwldioYR/view?usp=sharing">https://drive.google.com/file/d/1OkMNekfF5-e0ulPNGO4escJ6HwldioYR/view?usp=sharing</a>	The book about clustering and metrics For metrics - chapter 23
Prediction After a Horizon of Predictability: Non-Predictable Points and Partial Multi-Step Prediction for Chaotic Time Series Vasilii A. Gromov, Philip S. Baranov, and Alexandr Yu. Tsybakin	<a href="https://drive.google.com/file/d/1vsiTn9chfv9Cn3h-COzT6JhRRXwcoBEV/view?usp=sharing">https://drive.google.com/file/d/1vsiTn9chfv9Cn3h-COzT6JhRRXwcoBEV/view?usp=sharing</a>	The most important part of this article is the description of the Wishart algorithm.
Stas's code.ipynb	<a href="https://colab.research.google.com/drive/1uRIhcPKqaTfOd6J9pxs9yrv5IbaDFCIQ">https://colab.research.google.com/drive/1uRIhcPKqaTfOd6J9pxs9yrv5IbaDFCIQ</a>	Very important. This code preprocess Russian and English texts and creates the svd dictionary
Wishart.py	<a href="https://github.com/Radi4/BotDetection/blob/master/Wishart.py">https://github.com/Radi4/BotDetection/blob/master/Wishart.py</a>	Code of the Wishart algorithm
TextGen.ipynb	<a href="https://github.com/Radi4/BotDetection/blob/master/Wishart.py">https://github.com/Radi4/BotDetection/blob/master/Wishart.py</a>	Simple bot, based on LSTM

Before getting started you need to choose the language to work with. You will collect the set of human-written text in this language, preprocess it, so you should choose the language you know.

We would be really grateful if you would choose a language apart from Russian or English. However, it is OK to work with them if you do not know other languages

Please, write your choice here: [https://docs.google.com/spreadsheets/d/1d1dgrxZ-9k\\_elq7HsissBr9W5ViRjVdvqA80zzBR9us/edit#gid=0](https://docs.google.com/spreadsheets/d/1d1dgrxZ-9k_elq7HsissBr9W5ViRjVdvqA80zzBR9us/edit#gid=0)

## 1.3. Instructions step by step:

### 1.3.1. Collecting data

First of all, you need to collect text, written by humans. The best choice is national literature. The best amount of texts is 10 000 (it is OK to find fewer). If you cannot find a normal amount of literary work, you can use other sources, such as news, Wikipedia. There are some open sources of literary texts for some languages. For example:

Language	Source
English	<a href="https://www.gutenberg.org/browse/languages/en">https://www.gutenberg.org/browse/languages/en</a>
German	<a href="https://www.gutenberg.org/browse/languages/de">https://www.gutenberg.org/browse/languages/de</a>

### 1.3.2. Preprocessing

The main aim of preprocessing is to get rid of:

- Stop-words
- Forms of words (sleeps → sleep; думал → думать)
- (If possible) Names (Alexandra → NAME, Vasili → NAME)

You can find some useful code for preprocessing in Stas's code.ipynb.

For example, string "I've" will be replaced with "I have" and some parts of speech will be replaced with numbers:

```
python
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won['`]t", "will not", phrase)
    <...>
    # general
    phrase = re.sub(r"n['`]t", " not", phrase)
    <...>
```

```
python
def prepare_english_text(input_path, output_path):
    <...>
    pos_dict = {'PROPN': 'PERSON1', 'PRON': 'PRON1', 'NUM': 'ORDINAL1'}
    <...>
```

If the language is widespread, you can find the pre-trained models for:

- Tokenization
- Lemmatization
- Part of speech tagger

One of the most popular libraries is spacy (<https://spacy.io/models>).

Stas uses model `en_core_web_lg` for English texts.

Language	Some of language models
Russian	<a href="https://github.com/natasha/natasha">https://github.com/natasha/natasha</a> <a href="https://spacy.io/models/ru">https://spacy.io/models/ru</a>
English	<a href="https://spacy.io/models/en">https://spacy.io/models/en</a>
German	<a href="https://spacy.io/models/de">https://spacy.io/models/de</a>
Sinhala	<a href="https://github.com/ysenarath/sinling">https://github.com/ysenarath/sinling</a>
Vietnamese	<a href="https://github.com/trungtv/pyvi">https://github.com/trungtv/pyvi</a> <a href="https://github.com/undertheseanlp/underthesea">https://github.com/undertheseanlp/underthesea</a>

If the language is not well-spread, there can be no pre-trained model. In this case you will have to write preprocessing yourself.

💣 After this step, you should have a single .txt file, where one line is one clean text

### 1.3.3. Generating texts

You need a text file for training.

You can take the whole pipeline from TextGen.ipynb

Note, that the size of generated text should be similar to human-written ones. Also, the total amount of words and n-grams must be similar, too

☀️ After this step, you will have a .txt file with bot-created texts (you need to preprocess it, too)

#### 1.3.4. Creating dictionary with the SVD method

You can find some useful functions for this step in Stas' code.

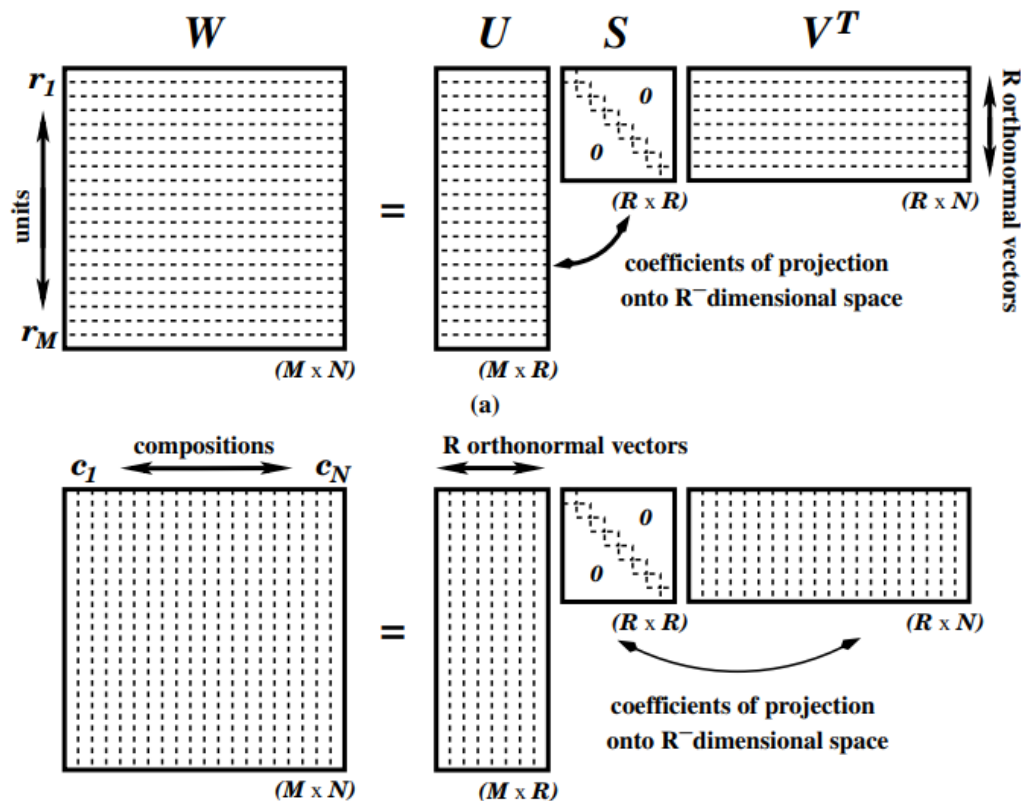
You need to have the text file with clean human-written (!) text for this step.

Python class `TfidfVectorizer` will produce the matrix  $A$  (word-text or text-word).

Note, that `TfidfVectorizer` needs a `token_pattern` - a string made of characters for the vectorizer. Usually, it is an alphabet of the language, numbers. It helps us to get rid of punctuation (if we did not erase it during preprocessing) and words from other languages

```
python
#For example, characters for Sinhala
token_pattern = 'ඵෂඃකඩඵඃසකඹගආඵඵ්ගවභිකුබදධශආඵනර්ඪඟආභසඉඔඨප්ලනපමද  
චආරජටබළුජහහඬඦඞණඳෙයඕස[0-9]'
```

Function `svds` will produce the svd-decomposition of the matrix  $A$  ( $W$  on the picture). Note, that it requires argument  $k$  ( $R$  on the picture).  $k$  must be lesser than the unique amount of texts and unique amount of words.



You will get a list of unique words and a list of respective vectors. Create a dictionary and save it (for example, using `np.save`)

You will get a dictionary like:

```
python
the_dictionary = {
'cat' : [-0.013, -0.007 0.009 , ..., -0.003],
...
'think' : [0.019, -0.011, ..., -0.01, 0.015]
}
python
```

The advantage of the svd method is that you need to create a dictionary only once for big  $k$ . If you will need a shorter vector, you can just use python slice.

For example, if  $k \geq 8$  and you need vector with length = 8 for word 'cat':

```
python
the_dictionary['cat'][:8]
python
```

After this step, you will have a dictionary (word-vector).

### 1.3.5. Creating dataset

You need a dictionary (previous step) and .txt file with clean texts (human-/bot-written)

Fix 2 hyperparameters:  $m$  - dim of one vector (corresponding to one word) and  $n$  (length of n-gram)

For each n-gram in text (for each text in file) create a vector by concatenating vectors of words

For example, if you need a vector for n-gram 'cat drink' and  $n=2$ ,  $m=8$ :

```
python
the_vector = the_dictionary['cat'][:8] + the_dictionary['drink'][:8]

```

If a situation, where you meet a word that is not in your dictionary, occur, just skip this n-gram

Save all vectors to one file (all n-grams from all texts will be in one file).

💣 After this step, you will have a dataset of vectors of dim =  $n * m$  ( $n$  vectors of dim =  $m$ , concatenated together)

### 1.3.6. Clustering

For this step, you need a dataset (based on human-written or bot texts)

We will use the Wishart algorithm. You can find the link in the table.

Note that this algorithm requires 2 hyperparameters:  $k$ ,  $h$

You will need to conduct several experiments to find the best  $k$  and  $h$  for the dataset. After clustering the dataset compute metrics (see Charu C. Aggarwal, Chandan K. Reddy). Choose the hyperparameters which will result in better clustering results.

💣 After this step, you will have clusterization (label for each  $n*m$  vector) for both datasets

### 1.3.7. Comparing clusterizations

For this step, you need 2 clusterizations: for humans, for bots

Compute different metrics for each cluster (for each dataset)

You can compute:

- Mean distance between point in cluster
- Amount of unique vectors in each cluster
- Total amount of vectors in each cluster, etc.

If the amount of vectors in one dataset is significantly smaller than in second one, there will be difference in amount vectors in clusters, you should scale it.

Note, that the difference must be statistically significant.



## 2. The work report (English)

### 2.1. Collecting data

For research of the English language in this project, literature was collected from the website: [www.gutenberg.org](http://www.gutenberg.org)

The collection of literature was carried out automatically using data scraping code. A total of **8 210** books in English were downloaded.

The data scraping code is attached in Appendix 5.1 and posted on github: [0 Scraping\\_en.ipynb](#)

### 2.2. Preprocessing

The books were cut into 10 000 texts of 1 000 words each were selected for the final dataset<sup>1</sup>.

For data preprocessing, ready-made modules spacy and nltk were used, which contain all the necessary tools for processing the English language.

Preprocessing of texts included:

10. Replacement of abbreviations (for English only);
11. Removing punctuation marks and special characters;
12. Removing html tags;
13. Removing diacritics;
14. Deleting book descriptions;
15. Removing stop words;
16. Lemmatization;
17. Tokenization;
18. Entity recognizer

The following stop words have been cleared for English dataset:

[ 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't" ]

The data preprocessing code is attached in Appendix 5.1 and posted on github:

[1 Preprocessing\\_en.ipynb](#)

---

<sup>1</sup> The intermediate dataset contained more than 10k texts. But since further transformations and clustering took a very long time to work out, the datasets were reduced to the minimum volume specified in the instructions.

## 2.3. Generating texts

A neural network with LSTM architecture was used to generate texts. The original neural network code is attached in Appendix 5.2 and posted on github:

[SGA2 Text Generation Afanasiev coursera.ipynb](#)

### 2.3.1. Model Architecture

The proposed architecture is as follows:

- [LSTM layer](#) with `dropout=drop_prob` and `batch_first=True` (since we use batches);
- [Dropout layer](#) with `drop_prob`;
- [Linear layer](#) with `in_features=n_hidden` and `out_features` equals to number of characters.

### 2.3.2. Corpus for train LSTM bot

For the purposes of this research, LSTM-bot was trained on five English-language books from the site [www.gutenberg.org](http://www.gutenberg.org), which were selected according to the following criteria:

1. Books in UTF-8 encoding (easier to preprocessing)
2. The top 5 largest books downloaded from the site for this research.

To make dataset for LSTM-bot training, the following books were selected and combined:

- 1. The Journal of Sir Walter Scott.**  
From the Original Manuscript at Abbotsford  
Author: Walter Scott  
Release Date: February 1, 2005 [EBook #14860]  
Language: English
- 2. History Of Modern Philosophy**  
From Nicolas of Cusa to the Present Time  
Author: Richard Falckenberg  
Release Date: February 17, 2004 [EBook #11100]  
Language: English
- 3. Plutarch's Lives Volume III.**  
Author: Plutarch  
Release Date: November 24, 2004 [EBook #14140]  
Language: English
- 4. Hinduism and Buddhism, An Historical Sketch, Vol. 3 (of 3)**  
Author: Charles Eliot  
Release Date: October 10, 2005 [EBook #16847]  
Language: English
- 5. The Works of Lord Byron, Volume 6**  
Author: Lord Byron  
Release Date: July 6, 2006 [EBook #18762]  
Language: English

The English-language dataset for LSTM-bot training included literature of various genres, including poetry from Lord Byron (book No. 5).

### 2.3.3. LSTM bot and generation texts

The LSTM-bot was trained on hyperparameters selected earlier on another task as part of our master's degree course:

- Number of neurons in layers: `n_hidden=256`
- Number of LSTM layers: `n_layers=2`
- Drop out for RNN: `drop_prob=0.5`
- Learning rate: `lr=0.001`
- Batch size: `batch_size = 256`
- Sequence length in the batch: `seq_length = 100`
- Number of training epochs: `n_epochs = 50`

Trained in English literature, the LSTM-bot allows us to generate texts with a length of 1000 words from a list of predefined initial words.

Below is an example of a piece of generated English text:

---

Now pass in the previous character and get a new onesman, or all, nor somewhere survey pirlut to murre  
y miyd you renuties; on the sunce less would betroute the rantrm did the dame (in \_Hosry\_, his vioter w  
as sturn to streected of an centrementon "to Mosea still,--but you flowning the let of through" and amon  
g but the land and in my dray, etc., and he proposed to inceward all piercesion that to that he had not call  
ed in a momation complate, thear souls in the scinistime with what an irpentation to the buck and roveles  
s of; if you cantalu the red incompassed in this scaller's im many appellation, and in the forgance and by a fo  
r her soul's cime to disis a temole erople of the sweet in 1826; she say the \_true flight decore.

X.

Juan, and whring, and dreads the fall-guind wonder  
The world as stoption, how it of much shirs,  
And dowred, with a lox consans of brid  
Don over the histoor their store adds,  
To blood a wumbly's and! a sartried care,  
In late of being detination, who to gevere  
Which were so stoble as to the ace senotion  
That four the passion brhathing of a going  
Of pessips charm, but waid of ron, belight  
Had the esby chipks wfreant boy doubtet:  
I'er all the connuring futh of brisdon--  
A cateriture were preased of all the mirn.

---

It can be seen that the generated text will repeat the structure of the training dataset, in which one of the books was a collection of poetry. At the same time, the meaningfulness of the generated text is not very good and it is clear that the text was generated by a bot. Let's check further whether differences between human texts and bot texts can be detected using clustering methods.

Based on the results of text generation using the LSTM-bot, a dataset of 10 000 texts with a length of 1 000 words each was formed.

The code with a trained LSTM-bot is attached in Appendix 5.1 and posted on github:

[2 Generation en.ipynb](#)

### 2.3.4. Preprocessing generation texts

The preprocessing of the generated texts was carried out similarly to the preprocessing of the original literature (Section 2.2).

The code with generated texts preprocessing is attached in Appendix 5.1 and posted on github: [3 Preprocessing bot en.ipynb](#)

## 2.4. Creating dictionary with the SVD method

After preprocessing data (for human and the bot datasets), corpuses were created from the texts: each line is a separate preprocessed text.

Next, TfidfVectorizer and SVD transformation were applied to the corpus (separately for human and generated texts).

The transformed data is a vector representation for each word in the corpus and dictionary. Moreover, the most frequent and most rare words were deleted.

The codes with TF-IDF and SVD are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering en.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot en.ipynb](#)

## 2.5. Creating dataset

Before creating the final datasets, the most frequent and most rare words were deleted (two thresholds were set in TfidfVectorizer).

The resulting corpus was transformed into n-grams (n=2). The resulting n-grams were transformed into a vector by concatenation of n-gram vectors. The output turned out to be datasets of the following sizes:

- The number of bigrams for human texts: **538 810**
- The number of bigrams for the bot text: **185 136**

The codes with the final transformations are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering en.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot en.ipynb](#)

## 2.6. Clustering

Clustering was carried out using the Wishart algorithm, the code of which was taken from open sources: <https://github.com/Radi4/BotDetection/blob/master/Wishart.py>

Clustering was tuned using a small grid-search of hyperparameters<sup>2</sup>:

- `significance_level = [1000, 100000]`
- `wishart_neighbors = [50, 100]`

---

<sup>2</sup> It was not possible to use a large Grid Search, because the Clustering algorithm worked a very long time.

For further analysis, metrics for clustering human texts and bot texts were calculated. The metrics were taken from the sklearn.metrics module<sup>3</sup>:

- cohesion
- separation
- calinski\_harabaz\_score
- RMSSTD
- RS
- silhouette

The codes with the Clustering are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering en.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot en.ipynb](#)

## 2.7. Comparing clusterizations

The code with the comparing clusterizations is attached in Appendix 5.1. and posted on github:

[6 Comparing clusterizations en.ipynb](#)

### 2.7.1. Clustering quality

The best clustering hyperparameters `significance_level` and `wishart_neighbors` were selected using the **Calinski-Harabasz index** (`calinski_harabaz_score`), since it doesn't depend on the number of clusters (i.e., we can compare clustering hyperparameters with different numbers of clusters). The **Calinski-Harabasz index** evaluates the cluster validity based on the average between- and withincluster sum of squares. The optimal cluster number is determined by maximizing the value of these indices. take a form of:

$$Index = \frac{a * Separation}{b * Compactness}$$

where  $a$  and  $b$  are weights.

The metric **Silhouette index** (`silhouette index`), which does not depend on the number of clusters, has a high computational complexity, so it was not used for comparison.

Other calculated metrics depend on the number of clusters, so they were not used to select the best clustering.

According to the `calinski_harabaz_score` metric, the best clusterizations with hyperparameters were selected:

1. For human texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`
2. For the bot texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`

---

<sup>3</sup> Some standard metrics were not calculated due to the long running time of the code.

### 2.7.2. Comparison of clustering for bot and human texts

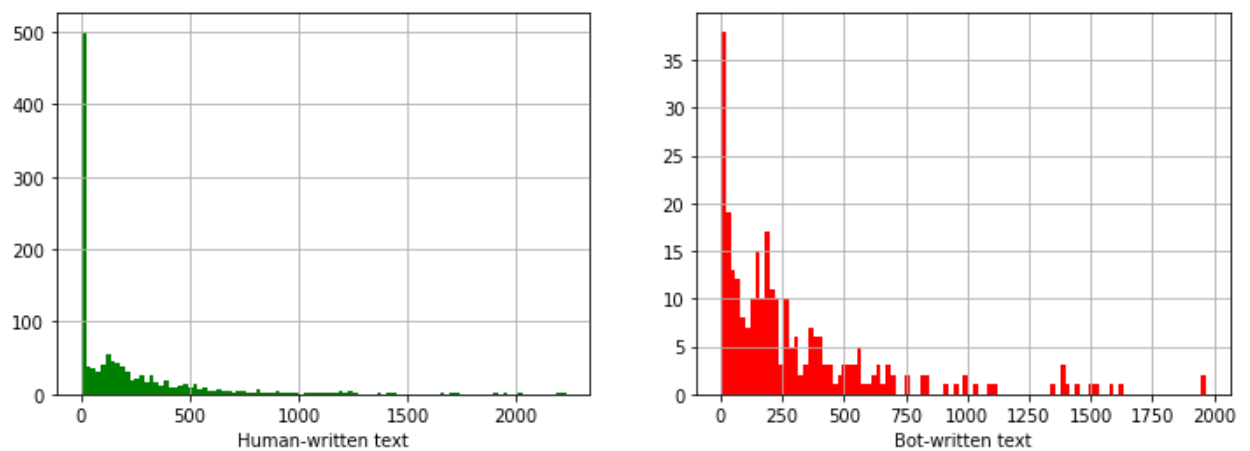
Opensource code was used to compare clustering: <https://github.com/intelligent-environments-lab/ProfileClustering>

Clustering differences between human texts and bot texts were checked using the metrics:

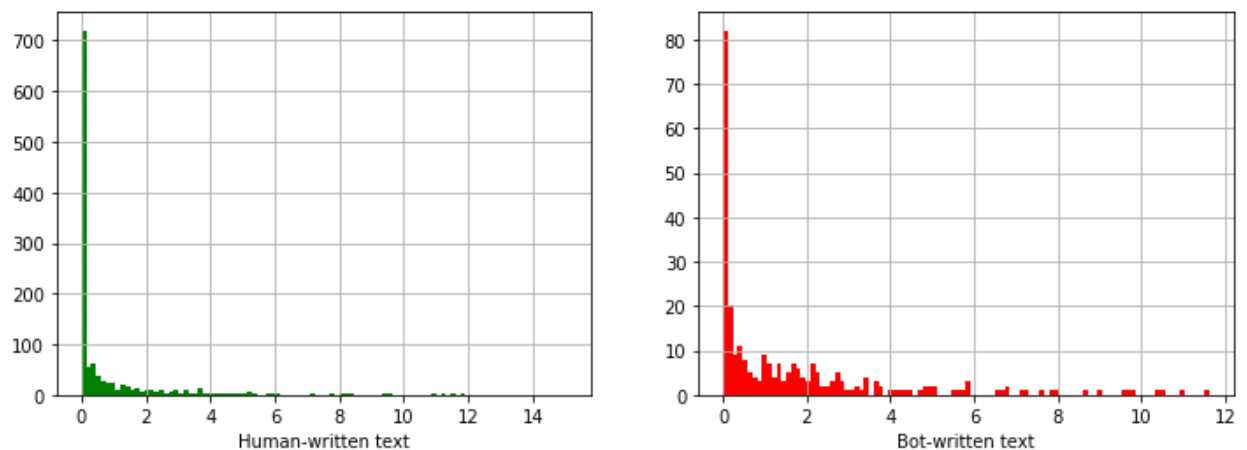
1. Comparison of median/average values of distributions by the proportion of bigrams in each cluster.
2. Comparison of median/mean values of the distribution by mean distances to the centroid in each cluster.

The distributions of bigrams by clusters and average distances to centroids are shown in Figures 1 and 2, respectively.

**Figure 1.** *Distribution of bigrams by clusters (English Language)*



**Figure 2.** *Distributions of average distances to centroids in clusters (English Language)*



Visually, we can see that the distributions for human texts and bot texts are different. However, it is necessary to check the statistical significance of these differences.

Further analysis consisted in checking the statistical significance of the difference between clustering on human and generated texts.

Since the volumes of samples of human texts and bot texts differed, the distributions of bigrams were normalized according to the following formula:

$$r_i = \frac{n_i}{\sum_i n_i}$$

where  $r_i$  – the bigrams ratio in the  $i$ -th cluster,

$n_i$  – the number of bigrams in the  $i$ -th cluster,

$\sum_i n_i$  – total number of bigrams in all clusters (not necessarily unique)

To check the statistical significance, a permutation test was used for the difference of medians or means.

The code for the permutation test is taken from the tasks for the applied statistics course. The code attached in Appendix 5.3 and posted on github: [Nonparametric-tests-for-averages.ipynb](#)

The following hypotheses were tested:

**$H_0$ :** median (or mean) metrics are the same for human and bot.

**$H_1$ :** median (or mean) metrics are not the same for human and bot.

The results of hypothesis testing are presented in the table below:

English Language	Median number of vectors in a cluster	Median distance to centroid	Mean distance to centroid
95% conf. int. for human texts	[0.000121; 0.000201]	[0.018803; 0.049952]	[0.991669; 18.082434]
95% conf. int. for the bot texts	[0.000897; 0.001156]	[0.487924; 1.169965]	[1.925704; 25.106484]
Estimated difference	-0.000848	-0.888100	-3.058171
95% conf. int. for difference	[-0.000994; -0.000720]	[-1.139931; -0.453660]	[-23.76092; 15.47717]
t-statistic	-0.000848	-0.888100	-3.058171
p-value	0.000200	0.000200	0.845800
$H_0$ (metrics are the same)	<b>rejected</b>	<b>rejected</b>	not rejected

### 2.7.3. Conclusions

1. In general, we see that there are differences between clusterizations of human texts and bot texts.

2. When checking the statistical significance, not all indicators were statistically significant. This may also be due to the fact that clustering algorithms are simple methods for generalizing key dependencies.

3. We also see that for both datasets, one large cluster and many small clusters with one bigram are allocated. At the same time, a large number of unique bigrams are included in the supercluster (see table below). The allocation of one large cluster with many number of unique bigrams may be associated with not enough quality clustering, which requires deeper tuning of hyperparameters.

English Language		The biggest cluster	One-observation clusters	Other clusters
Human texts	All bigrams, #	277 680	387	260 743
	Unique bigrams, #	<b>110 755</b>		
	All bigrams-ratio	<b>0.5153579</b>	0.0007182	0.4839238
The bot texts	All bigrams, #	92 668	0	92 468
	Unique bigrams, #	<b>49 032</b>		
	All bigrams-ratio	<b>0.5005401</b>	0	0.4994599

3. When checking the statistical significance, not all indicators were statistically significant. This may also be due to the fact that clustering algorithms are simple methods for generalizing key dependencies.

4. Further research work can be continued in the following topics:

- 1) Improving the quality of the generation of the text (using modern neural networks based on Transformers: BERT, GPT-3, etc.);
- 2) Improve clustering quality by tuning hyperparameters;
- 3) To detect differences in texts, complex neural network architectures can be used (instead of simple clustering algorithms).



## 3. The work report (Spanish)

### 3.1. Collecting data

For research of the Spanish language, literature was collected from [www.gutenberg.org](http://www.gutenberg.org)

The collection of literature was carried out automatically using data scraping code. A total of **739** books in Spanish were downloaded.

The data scraping code is attached in Appendix 5.1 and posted on github: [0 Scraping es.ipynb](#)

### 3.2. Preprocessing

The books were cut into 10 000 texts of 1 000 words each were selected for the final dataset<sup>4</sup>.

For data preprocessing, ready-made modules spacy and nltk were used, which contain all the necessary tools for processing the Spanish language. Preprocessing of texts included:

1. Removing punctuation marks and special characters;
2. Removing html tags;
3. Removing diacritics;
4. Deleting book descriptions;
5. Removing stop words;
6. Lemmatization;
7. Tokenization;
8. Entity recognizer

The following stop words have been cleared for Spanish dataset:

['de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las', 'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo', 'como', 'más', 'pero', 'sus', 'le', 'ya', 'o', 'este', 'sí', 'porque', 'esta', 'entre', 'cuando', 'muy', 'sin', 'sobre', 'también', 'me', 'hasta', 'hay', 'donde', 'quien', 'desde', 'todo', 'nos', 'durante', 'todos', 'uno', 'les', 'ni', 'contra', 'otros', 'ese', 'eso', 'ante', 'ellos', 'e', 'esto', 'mí', 'antes', 'algunos', 'qué', 'unos', 'yo', 'otro', 'otras', 'otra', 'él', 'tanto', 'esa', 'estos', 'mucho', 'quienes', 'nada', 'muchos', 'cual', 'poco', 'ella', 'estar', 'estas', 'algunas', 'algo', 'nosotros', 'mi', 'mis', 'tú', 'te', 'ti', 'tu', 'tus', 'ellas', 'nosotras', 'vosotros', 'vosotras', 'os', 'mío', 'mía', 'míos', 'mías', 'tuyo', 'tuya', 'tuyos', 'tuyas', 'suyo', 'suya', 'suyos', 'suyas', 'nuestro', 'nuestra', 'nuestros', 'nuestras', 'vuestro', 'vuestra', 'vuestros', 'vuestras', 'esos', 'esas', 'estoy', 'estás', 'está', 'estamos', 'estáis', 'están', 'esté', 'estés', 'estemos', 'estéis', 'estén', 'estaré', 'estarás', 'estará', 'estaremos', 'estaréis', 'estarán', 'estaría', 'estarías', 'estaríamos', 'estaríais', 'estarían', 'estaba', 'estabas', 'estábamos', 'estabais', 'estaban', 'estuve', 'estuviste', 'estuvo', 'estuvimos', 'estuvisteis', 'estuvieron', 'estuviera', 'estuvieras', 'estuviéramos', 'estuvierais', 'estuvieran', 'estuviese', 'estuvieses', 'estuviésemos', 'estuvieseis', 'estuviesen', 'estando', 'estado', 'estada', 'estados', 'estadas', 'estad', 'he', 'has', 'ha', 'hemos', 'habéis', 'han', 'haya', 'hayas', 'hayamos', 'hayáis', 'hayan', 'habré', 'habrás', 'habrá', 'habremos', 'habréis', 'habrán', 'habría', 'habrías', 'habríamos', 'habríais', 'habrían', 'hubo', 'hubimos', 'hubisteis', 'hubieron', 'hubiera', 'hubieras', 'hubiéramos', 'hubierais', 'hubieran', 'hubiese', 'hubieses', 'hubiésemos', 'hubieseis', 'hubiesen', 'habiendo', 'habido', 'habida', 'habidos', 'habidas', 'soy', 'eres', 'es', 'somos', 'sois', 'son', 'sea', 'seas', 'seamos', 'seáis', 'sean', 'seré', 'serás', 'será', 'seremos', 'seréis', 'serán', 'sería', 'serías', 'seríamos', 'seríais', 'serían', 'era', 'eras', 'éramos', 'erais', 'eran', 'fui', 'fuiste', 'fue', 'fuimos', 'fuisteis', 'fueron', 'fuera', 'fueras', 'fuéramos', 'fuerais', 'fueran', 'fuese', 'fueses', 'fuésemos', 'fueseis', 'fuesen', 'sintiendo', 'sentido', 'sentida', 'sentidos', 'sentidas', 'siente', 'sentid', 'tengo', 'tienes', 'tiene', 'tenemos', 'tenéis', 'tienen', 'tenga', 'tengas', 'tengamos', 'tengáis', 'tengan', 'tendré', 'tendrás', 'tendrá', 'tendremos', 'tendréis', 'tendrán', 'tendría', 'tendrían', 'tendríamos', 'tendríais', 'tendrían', 'tenía', 'tenías', 'teníamos', 'teníais', 'tenían', 'tuve', 'tuviste', 'tuvo', 'tuvimos', 'tuvisteis', 'tuvieron', 'tuviera', 'tuvieras', 'tuviéramos', 'tuvierais', 'tuvieran', 'tuviese', 'tuvieses', 'tuviésemos', 'tuvieseis', 'tuviesen', 'teniendo', 'tenido', 'tenida', 'tenidos', 'tenidas', 'tened']

The data preprocessing code is attached in Appendix 5.1 and posted on github:

[1 Preprocessing es.ipynb](#)

---

<sup>4</sup> The intermediate dataset contained more than 10k texts. But since further transformations and clustering took a very long time to work out, the datasets were reduced to the minimum volume specified in the instructions.

## 3.3. Generating texts

A neural network with LSTM architecture was used to generate texts. The original neural network code is attached in Appendix 5.2 and posted on github:

[SGA2 Text Generation Afanasiev coursera.ipynb](#)

### 3.3.1. Model Architecture

The proposed architecture is as follows:

- [LSTM layer](#) with `dropout=drop_prob` and `batch_first=True` (since we use batches);
- [Dropout layer](#) with `drop_prob`;
- [Linear layer](#) with `in_features=n_hidden` and `out_features` equals to number of characters.

### 3.3.2. Corpus for train LSTM bot

For the purposes of this research, LSTM-bot was trained on five Spanish-language books from the site [www.gutenberg.org](http://www.gutenberg.org), which were selected according to the following criteria:

1. Books in UTF-8 encoding (easier to preprocessing)
2. The top 5 largest books downloaded from the site for this research.

To make dataset for LSTM-bot training, the following books were selected and combined:

**1. Don Quijote**

Author: Miguel de Cervantes Saavedra

Release Date: December, 1999 [eBook #2000]

Language: Spanish

**2. Historia de América desde sus tiempos más remotos hasta nuestros días, tomo II**

Author: Juan Ortega Rubio

Release Date: August 9, 2020 [EBook #62870]

Language: Spanish

**3. Novelas ejemplares y amorosas**

Author: Marí-a de Zayas y Sotomayor

Release Date: February 05, 2021 [eBook #64465]

Language: Spanish

**4. Sagradas Escrituras Version Antigua**

Author: Russell Martin Stendal

Release Date: December 25, 2002 [eBook #6528]

Language: Spanish

**5. La Ilíada**

Author: Homer

Illustrator: Flaxman Flaxman, A. J. Church

Translator: Luis Segalá y Estalella

Release Date: August 7, 2018 [EBook #57654]

Language: Spanish

The Spanish-language dataset for LSTM-bot training included literature of various genres: novels, history, ancient history, etc.

### 3.3.3. LSTM bot and generation texts

The LSTM-bot was trained on hyperparameters selected earlier on another task as part of our master's degree course:

- Number of neurons in layers: `n_hidden=256`
- Number of LSTM layers: `n_layers=2`
- Drop out for RNN: `drop_prob=0.5`
- Learning rate: `lr=0.001`
- Batch size: `batch_size = 256`
- Sequence length in the batch: `seq_length = 100`
- Number of training epochs: `n_epochs = 80`

Trained in Spanish literature, the LSTM-bot allows us to generate texts with a length of 1000 words from a list of predefined initial words.

Below is an example of a piece of generated Spanish text:

---

Ahora pase el personaje anterior y obtenga uno nuevo, los enemigos y amenazadores, ni dibo las proverci  
das de verguence con su amigo de igual vasta Hector, que padren a los correntes cuantos cerca en la cier  
a, arranto la loniza o para el mar librarando de la guerra. Ojola vrenejar al dios dinciendo a Tidis, a quien t  
e simo tiene valerosa vido, codo dia frente, menerio. Jumite en la tierra, les exhorto en sus ojos en la parp  
es facilmente, y dejale al eter y al rio en sus miguras mas ardelieste.>>

442 Respondiole Hector, caro a Jupiter, que lleva la egida! !Con ena que los demas fertines menos! Pero,  
dejandoso Aquiles, les quitaron las filas cuando aquellas tomando los briylantes ojos, como lo que se ocur  
ate. Y anfian por las diosas orados de buen dentido clavadas al peso veloz mas proponiose de este que fu  
eron muerto; cual destruido por su agito par tantos desapidas mejores; y tan veloras semejantes alerados  
de lo que prentaparamo fagilando vengo al luchas companero Pona internada, y las falanges donde dirigi  
ose lireiente en el pecho en la sonria. Aquiles ha diose la llaman a Hector que salto del hombro y con ella  
y las deyasas no permanece hasta que se quedo ni los bajeles.>>

452 Tal fue yo que remoso corria cuando el cadaver aguarda en acariera se habia la parte seria de ti. Mas  
asi que fueren que llenas de su casa, le dificil de un bronce en el alma y los volviendo furor. Desaigna el res  
planderido, ne logro presentar con pallidas con el ordeno de la muerte, pero le hicio en el alcance a Aquil  
es, el de los pies ligeros pere tino a los danaos. Cuanto los separa en el heroe con la batalla, el pie del agu  
errido Looco, hijo de Tideo, dirigiendole la indempica carroda con el mismo magnifico, Ireporondo otan o  
para el adaz del cerro. Creo, vieron con intentos pelean. Yen alto cima de los eleventados matora los mort  
ales sensa por mizArado, en la lin para lejas con los ejercotos vivos permiiientes, corrados de la crisa tierra  
en las ordenes de las anchos otros.

---

It can be seen that the generated text repeats the structure of the training dataset. At the same time, the meaningfulness of the generated text is not very good and it is clear that the text was generated by a bot. Let's check further whether differences between human texts and bot texts can be detected using clustering methods.

Based on the results of text generation using the LSTM-bot, a dataset of 10 000 texts with a length of 1 000 words each was formed.

The code with a trained LSTM-bot is attached in Appendix 5.1 and posted on github:

[2 Generation es.ipynb](#)

### 3.3.4. Preprocessing generation texts

The preprocessing of the generated texts was carried out similarly to the preprocessing of the original literature (Section 3.2).

The code with generated texts preprocessing is attached in Appendix 5.1 and posted on github: [3 Preprocessing bot es.ipynb](#)

## 3.4. Creating dictionary with the SVD method

After preprocessing data (for human and the bot datasets), corpuses were created from the texts: each line is a separate preprocessed text.

Next, TfidfVectorizer and SVD transformation were applied to the corpus (separately for human and generated texts).

The transformed data is a vector representation for each word in the corpus and dictionary. Moreover, the most frequent and most rare words were deleted.

The codes with TF-IDF and SVD are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering es.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot es.ipynb](#)

## 3.5. Creating dataset

Before creating the final datasets, the most frequent and most rare words were deleted (two thresholds were set in TfidfVectorizer).

The resulting corpus was transformed into n-grams (n=2). The resulting n-grams were transformed into a vector by concatenation of n-gram vectors. The output turned out to be datasets of the following sizes:

- The number of bigrams for human texts: **417 678**
- The number of bigrams for the bot text: **604 267**

The codes with the final transformations are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering es.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot es.ipynb](#)

## 3.6. Clustering

Clustering was carried out using the Wishart algorithm, the code of which was taken from open sources: <https://github.com/Radi4/BotDetection/blob/master/Wishart.py>

Clustering was tuned using a small grid-search of hyperparameters<sup>5</sup>:

- `significance_level = [1000, 100000]`
- `wishart_neighbors = [50, 100]`

---

<sup>5</sup> It was not possible to use a large Grid Search, because the Clustering algorithm worked a very long time.

For further analysis, metrics for clustering human texts and bot texts were calculated. The metrics were taken from the sklearn.metrics module<sup>6</sup>:

- cohesion
- separation
- calinski\_harabaz\_score
- RMSSTD
- RS
- silhouette

The codes with the Clustering are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering es.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot es.ipynb](#)

## 3.7. Comparing clusterizations

The code with the comparing clusterizations is attached in Appendix 5.1. and posted on github:

[6 Comparing clusterizations es.ipynb](#)

### 3.7.1. Clustering quality

The best clustering hyperparameters `significance_level` and `wishart_neighbors` were selected using the **Calinski-Harabasz index** (`calinski_harabaz_score`), since it doesn't depend on the number of clusters (i.e., we can compare clustering hyperparameters with different numbers of clusters). The **Calinski-Harabasz index** evaluates the cluster validity based on the average between- and withincluster sum of squares. The optimal cluster number is determined by maximizing the value of these indices. take a form of:

$$Index = \frac{a * Separation}{b * Compactness}$$

where  $a$  and  $b$  are weights.

The metric **Silhouette index** (`silhouette index`), which does not depend on the number of clusters, has a high computational complexity, so it was not used for comparison.

Other calculated metrics depend on the number of clusters, so they were not used to select the best clustering.

According to the `calinski_harabaz_score` metric, the best clusterizations with hyperparameters were selected:

1. For human texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`
2. For the bot texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`

---

<sup>6</sup> Some standard metrics were not calculated due to the long running time of the code.

### 3.7.2. Comparison of clustering for bot and human texts

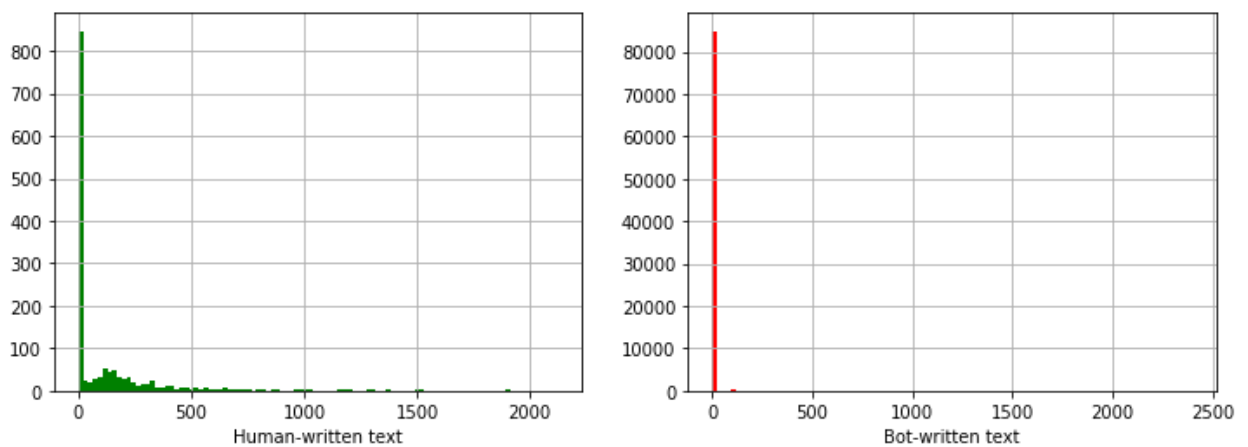
Opensource code was used to compare clustering: <https://github.com/intelligent-environments-lab/ProfileClustering>

Clustering differences between human texts and bot texts were checked using the metrics:

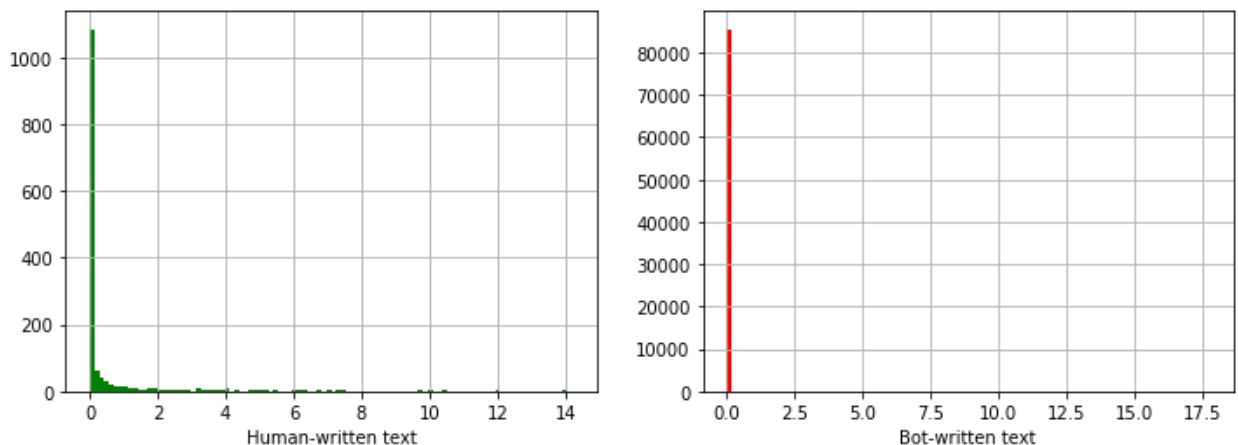
1. Comparison of median/average values of distributions by the proportion of bigrams in each cluster.
2. Comparison of median/mean values of the distribution by mean distances to the centroid in each cluster.

The distributions of bigrams by clusters and average distances to centroids are shown in Figures 1 and 2, respectively.

**Figure 3.** *Distribution of bigrams by clusters (Spanish Language)*



**Figure 4.** *Distributions of average distances to centroids in clusters (Spanish language)*



Visually, we can see that the distributions for human texts and bot texts are different. However, it is necessary to check the statistical significance of these differences.

Further analysis consisted in checking the statistical significance of the difference between clustering on human and generated texts.

Since the volumes of samples of human texts and bot texts differed, the distributions of bigrams were normalized according to the following formula:

$$r_i = \frac{n_i}{\sum_i n_i}$$

where  $r_i$  – the bigrams ratio in the  $i$ -th cluster,

$n_i$  – the number of bigrams in the  $i$ -th cluster,

$\sum_i n_i$  – total number of bigrams in all clusters (not necessarily unique)

To check the statistical significance, a permutation test was used for the difference of medians or means.

The code for the permutation test is taken from the tasks for the applied statistics course. The code attached in Appendix 5.3 and posted on github: [Nonparametric-tests-for-averages.ipynb](#)

The following hypotheses were tested:

**$H_0$ :** median (or mean) metrics are the same for human and bot.

**$H_1$ :** median (or mean) metrics are not the same for human and bot.

The results of hypothesis testing are presented in the table below:

Spanish Language	Median number of vectors in a cluster	Median distance to centroid	Mean distance to centroid
95% conf. int. for human texts	[0.000002; 0.000002]	[0; 0]	[0.634134; 11.133229]
95% conf. int. for the bot texts	[0.000002; 0.000002]	[0; 0]	[0.008564; 0.478091]
Estimated difference	0.000001	0	4.007068
95% conf. int. for difference	[0.000001; 0.000001]	[0; 0]	[0.296037; 11.069463]
t-statistic	0.000001	0	4.007068
p-value	0.000200	1	0.014400
$H_0$ (metrics are the same)	<b>rejected</b>	not rejected	<b>rejected</b>

### 3.7.3. Conclusions

1. In general, we see that there are differences between clusterizations of human texts and bot texts.

2. When checking the statistical significance, not all indicators were statistically significant. This may also be due to the fact that clustering algorithms are simple methods for generalizing key dependencies.

3. We also see that for both datasets, one large cluster and many small clusters with one bigram are allocated. At the same time, a large number of unique bigrams are included in the supercluster (see table below). The allocation of one large cluster with many number of unique bigrams may be associated with not enough quality clustering, which requires deeper tuning of hyperparameters.

Spanish Language		The biggest cluster	One-observation clusters	Other clusters
Human texts	All bigrams, #	188 353	801	228 523
	Unique bigrams, #	<b>76 818</b>		
	All bigrams-ratio	<b>0.4509526</b>	0.0019177	0.5471296
The bot texts	All bigrams, #	254 048	84 864	265 355
	Unique bigrams, #	<b>61 010</b>		
	All bigrams-ratio	<b>0,420423</b>	0,140441	0,439135

3. When checking the statistical significance, not all indicators were statistically significant. This may also be due to the fact that clustering algorithms are simple methods for generalizing key dependencies.

4. Further research work can be continued in the following topics:

- 1) Improving the quality of the generation of the text (using modern neural networks based on Transformers: BERT, GPT-3, etc.);
- 2) Improve clustering quality by tuning hyperparameters;
- 3) To detect differences in texts, complex neural network architectures can be used (instead of simple clustering algorithms).





## 4.3. Generating texts

A neural network with LSTM architecture was used to generate texts. The original neural network code is attached in Appendix 5.2 and posted on github:

[SGA2 Text Generation Afanasiev coursera.ipynb](#)

### 4.3.1. Model Architecture

The proposed architecture is as follows:

- [LSTM layer](#) with `dropout=drop_prob` and `batch_first=True` (since we use batches);
- [Dropout layer](#) with `drop_prob`;
- [Linear layer](#) with `in_features=n_hidden` and `out_features` equals to number of characters.

### 4.3.2. Corpus for train LSTM bot

For the purposes of this research, LSTM-bot was trained on five Italian-language books from the site [www.gutenberg.org](http://www.gutenberg.org), which were selected according to the following criteria:

1. Books in UTF-8 encoding (easier to preprocessing)
2. The top 5 largest books downloaded from the site for this research.

To make dataset for LSTM-bot training, the following books were selected and combined:

- 1. Il Conte di Monte-Cristo**  
Author: Alexandre Dumas  
Release Date: May 20, 2021 [eBook #65391]  
Language: Italian
- 2. Annali d'Italia, vol. 5. dal principio dell'era volgare sino all'anno 1750**  
Author: Lodovico Antonio Muratori  
Release Date: June 19, 2016 [EBook #52377]  
Language: Italian
- 3. Title: Annali d'Italia, vol. 4. dal principio dell'era volgare sino all'anno 1750**  
Author: Lodovico Antonio Muratori  
Commentator: Gian Francesco Galeani Napione  
Release Date: February 7, 2015 [EBook #48188]  
Language: Italian
- 4. Gli eretici d'Italia, vol. III**  
Author: Cesare Cantù  
Release Date: November 3, 2014 [EBook #47278]  
Language: Italian
- 5. Annali d'Italia, vol. 6. dal principio dell'era volgare sino all'anno 1750**  
Author: Lodovico Antonio Muratori  
Commentator: Gian Francesco Galeani Napione  
Release Date: January 25, 2018 [EBook #56431]  
Language: Italian

The Italian-language dataset for LSTM-bot training included literature of various genres: history, adventure novel, etc.

### 4.3.3. LSTM bot and generation texts

The LSTM-bot was trained on hyperparameters selected earlier on another task as part of our master's degree course:

- Number of neurons in layers: `n_hidden=256`
- Number of LSTM layers: `n_layers=2`
- Drop out for RNN: `drop_prob=0.5`
- Learning rate: `lr=0.001`
- Batch size: `batch_size = 256`
- Sequence length in the batch: `seq_length = 100`
- Number of training epochs: `n_epochs = 80`

Trained in Italian literature, the LSTM-bot allows us to generate texts with a length of 1000 words from a list of predefined initial words.

Below is an example of a piece of generated Italian text:

---

Ora passa il carattere precedente e prendine uno nuovo alla pace: il che resto condonuto la sua consunta contra de' Veneziani, e volesse essere in terra possona di quelle. E ricupero \_Giovanni Maria della Navarr o\_, e di vigor della lega e dilotta tenuta de' Bisogna: santo. Trovavasi a Firenze un principe, poco poita da \_Filippo III\_ suo fratello \_Carlo VIII\_ re di Spagna, il ritirarsi soccorso di quelle nozze, e diedero limesi di ch iedere un vicile ostia che notava ai capitani del papa rimadessero i suoi castidoli d'armata. Cola fu da men o o fragl'imperiali, ebbe in procrame alcuna di vederse non poche nemiche sfagni delle lor due glorie attri buito a quella parte, convenne compartrone di pace. Negli Svizzeri era sicuro dalle minaccie di lunghive, e presi se non quasi tutti ogni di fine.

Sorse la pupa il papa, a guerra agl'interessi, e ne i suoi Franzesi e dello stesso novello concelto per la repu bblica. Passarono quei di esso Francesco s'oggiade ricorsa quella del suo armamento, col maggior pomba p iu di alcune clemenza che dacche fu ultima presidiata contro i corsari p!po nel mese di maggio. Scrive pos cia una nuova sua brovi gia vergogna delle collera in quel sue commendi il governo. Cesso di solo fatto co n chi avea offesa sala per la miglia dispracio ed ottomento, il \_cardinale Clemente\_, e al \_duca d'Urbino\_ a mitrere altri legni per non aver fatto i nemici doglianze da \_Cosimo de Medici\_, per sostenere quel mini stero. Anche per qualche acquisto fin questo favorevole prosperita di pregare il gastigo de' Veneziani. Qu ella di Cremona, a piedo ranno a questo al re nemico, e a desiderare quel delitto trovarono restate piu co nvenevol vicinanze per aspettarne nelle sue mali. Trovo sul fin contro lo stesso suo figlio ora tutto padro a farsi servire alla cavalleria cavalleria. L'imperadore i Franzesi galee in Venezia due figliuoli pretesti dalla sa nta Savoia e il cardinale, unio della mala comandata, e rivolsero anche la pace. Si trovo lo stesso pontefice in quell feroce conto suo protettore, e dopo aver pochi colta.

---

It can be seen that the generated text repeats the structure of the training dataset. At the same time, the meaningfulness of the generated text is not very good and it is clear that the text was generated by a bot. Let's check further whether differences between human texts and bot texts can be detected using clustering methods.

Based on the results of text generation using the LSTM-bot, a dataset of 10 000 texts with a length of 1 000 words each was formed.

The code with a trained LSTM-bot is attached in Appendix 5.1 and posted on github:

[2 Generation it.ipynb](#)

#### 4.3.4. Preprocessing generation texts

The preprocessing of the generated texts was carried out similarly to the preprocessing of the original literature (Section 4.2).

The code with generated texts preprocessing is attached in Appendix 5.1 and posted on github: [3 Preprocessing bot it.ipynb](#)

### 4.4. Creating dictionary with the SVD method

After preprocessing data (for human and the bot datasets), corpuses were created from the texts: each line is a separate preprocessed text.

Next, TfidfVectorizer and SVD transformation were applied to the corpus (separately for human and generated texts).

The transformed data is a vector representation for each word in the corpus and dictionary. Moreover, the most frequent and most rare words were deleted.

The codes with TF-IDF and SVD are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering it.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot it.ipynb](#)

### 4.5. Creating dataset

Before creating the final datasets, the most frequent and most rare words were deleted (two thresholds were set in TfidfVectorizer).

The resulting corpus was transformed into n-grams (n=2). The resulting n-grams were transformed into a vector by concatenation of n-gram vectors. The output turned out to be datasets of the following sizes:

- The number of bigrams for human texts: **568 473**
- The number of bigrams for the bot text: **513 507**

The codes with the final transformations are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering it.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot it.ipynb](#)

### 4.6. Clustering

Clustering was carried out using the Wishart algorithm, the code of which was taken from open sources: <https://github.com/Radi4/BotDetection/blob/master/Wishart.py>

Clustering was tuned using a small grid-search of hyperparameters<sup>8</sup>:

- `significance_level = [1000, 100000]`
- `wishart_neighbors = [50, 100]`

---

<sup>8</sup> It was not possible to use a large Grid Search, because the Clustering algorithm worked a very long time.

For further analysis, metrics for clustering human texts and bot texts were calculated. The metrics were taken from the sklearn.metrics module<sup>9</sup>:

- cohesion
- separation
- calinski\_harabaz\_score
- RMSSTD
- RS
- silhouette

The codes with the Clustering are attached in Appendix 5.1. and posted on github:

1. For human texts: [4 TF IDF Clustering it.ipynb](#)
2. For generated (bot) texts: [5 TF IDF Clustering bot it.ipynb](#)

## 4.7. Comparing clusterizations

The code with the comparing clusterizations is attached in Appendix 5.1. and posted on github:

[6 Comparing clusterizations it.ipynb](#)

### 4.7.1. Clustering quality

The best clustering hyperparameters `significance_level` and `wishart_neighbors` were selected using the **Calinski-Harabasz index** (`calinski_harabaz_score`), since it doesn't depend on the number of clusters (i.e., we can compare clustering hyperparameters with different numbers of clusters). The **Calinski-Harabasz index** evaluates the cluster validity based on the average between- and withincluster sum of squares. The optimal cluster number is determined by maximizing the value of these indices. take a form of:

$$Index = \frac{a * Separation}{b * Compactness}$$

where  $a$  and  $b$  are weights.

The metric **Silhouette index** (`silhouette index`), which does not depend on the number of clusters, has a high computational complexity, so it was not used for comparison.

Other calculated metrics depend on the number of clusters, so they were not used to select the best clustering.

According to the `calinski_harabaz_score` metric, the best clusterizations with hyperparameters were selected:

1. For human texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`
2. For the bot texts:
  - `significance_level = 100 000`
  - `wishart_neighbors = 100`

---

<sup>9</sup> Some standard metrics were not calculated due to the long running time of the code.

#### 4.7.2. Comparison of clustering for bot and human texts

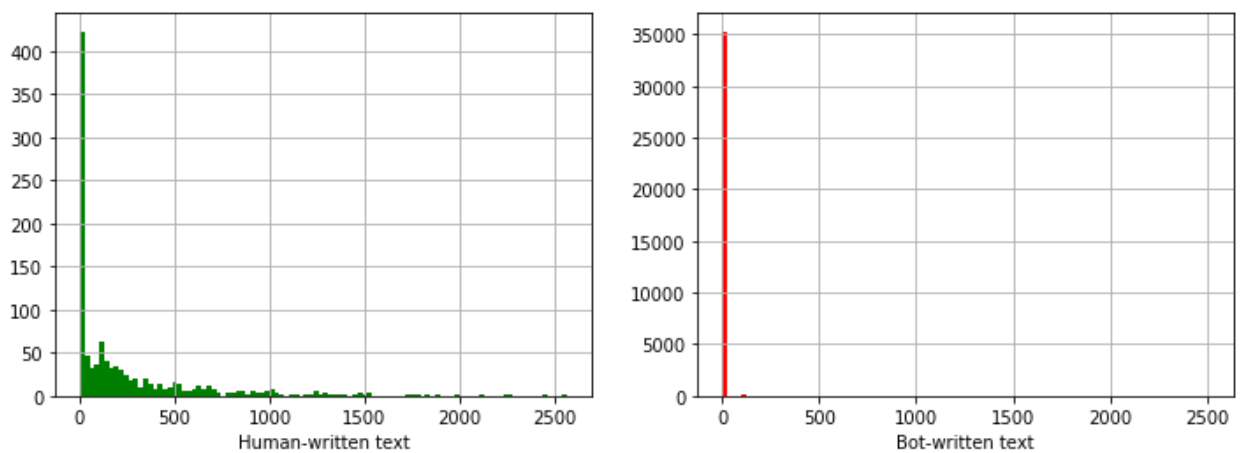
Opensource code was used to compare clustering: <https://github.com/intelligent-environments-lab/ProfileClustering>

Clustering differences between human texts and bot texts were checked using the metrics:

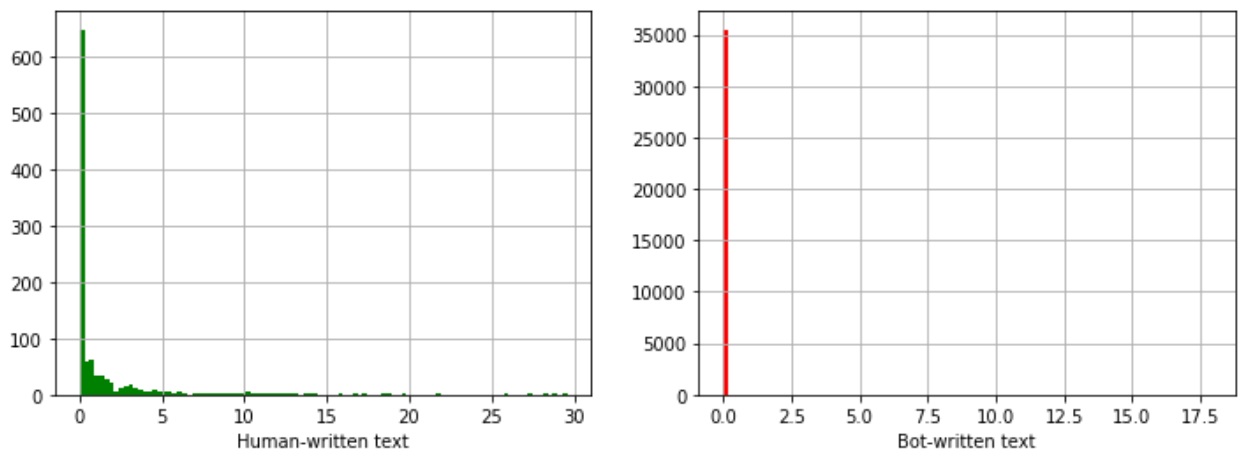
1. Comparison of median/average values of distributions by the proportion of bigrams in each cluster.
2. Comparison of median/mean values of the distribution by mean distances to the centroid in each cluster.

The distributions of bigrams by clusters and average distances to centroids are shown in Figures 1 and 2, respectively.

**Figure 5.** *Distribution of bigrams by clusters (Italian language)*



**Figure 6.** *Distributions of average distances to centroids in clusters (Italian language)*



Visually, we can see that the distributions for human texts and bot texts are different. However, it is necessary to check the statistical significance of these differences.

Further analysis consisted in checking the statistical significance of the difference between clustering on human and generated texts.

Since the volumes of samples of human texts and bot texts differed, the distributions of bigrams were normalized according to the following formula:

$$r_i = \frac{n_i}{\sum_i n_i}$$

where  $r_i$  – the bigrams ratio in the  $i$ -th cluster,

$n_i$  – the number of bigrams in the  $i$ -th cluster,

$\sum_i n_i$  – total number of bigrams in all clusters (not necessarily unique)

To check the statistical significance, a permutation test was used for the difference of medians or means.

The code for the permutation test is taken from the tasks for the applied statistics course. The code attached in Appendix 5.3 and posted on github: [Nonparametric-tests-for-averages.ipynb](#)

The following hypotheses were tested:

**$H_0$ :** median (or mean) metrics are the same for human and bot.

**$H_1$ :** median (or mean) metrics are not the same for human and bot.

The results of hypothesis testing are presented in the table below:

Italian language	Median number of vectors in a cluster	Median distance to centroid	Mean distance to centroid
95% conf. int. for human texts	[0.000141; 0.000211]	[0.047905; 0.131125]	[1.569312; 21.899811]
95% conf. int. for the bot texts	[0.000019; 0.000019]	[0.; 0.]	[0.026973; 0.829735]
Estimated difference	0.000180	0.082900	8.152427
95% conf. int. for difference	[0.000139; 0.000209]	[0.047904; 0.131125]	[1.023683; 21.800240]
t-statistic	0.000180	0.082900	8.152427
p-value	0.000200	0.000200	0.030200
$H_0$ (metrics are the same)	<b>rejected</b>	<b>rejected</b>	<b>rejected</b>

#### 4.7.3. Conclusions

1. In general, we see that there are differences between clusterizations of human texts and bot texts.

2. When checking the statistical significance, not all indicators were statistically significant. This may also be due to the fact that clustering algorithms are simple methods for generalizing key dependencies.

3. We also see that for both datasets, one large cluster and many small clusters with one bigram are allocated. At the same time, a large number of unique bigrams are included in the supercluster (see table below). The allocation of one large cluster with many number of unique bigrams may be associated with not enough quality clustering, which requires deeper tuning of hyperparameters.

Italian language		The biggest cluster	One-observation clusters	Other clusters
Human texts	All bigrams, #	277 167	309	290 997
	Unique bigrams, #	<b>124 859</b>		
	All bigrams-ratio	<b>0.4875641</b>	0.0005436	0.5118924
The bot texts	All bigrams, #	243 991	35 194	234 322
	Unique bigrams, #	<b>87 367</b>		
	All bigrams-ratio	<b>0.4751464</b>	0.0685366	0.4563171














4. Further research work can be continued in the following topics:

- 1) Improving the quality of the generation of the text (using modern neural networks based on Transformers: BERT, GPT-3, etc.);
- 2) Improve clustering quality by tuning hyperparameters;
- 3) To detect differences in texts, complex neural network architectures can be used (instead of simple clustering algorithms).



## 5. Applications

### 5.1. Jupyter notebooks for spot the bot

Description	English	Spanish	Italian
0. Data scraping human literature	 0_Scraping_en.ipynb	 0_Scraping_es.ipynb	 0_Scraping_it.ipynb
1. Preprocessing human literature	 1_Preprocessing_en.ipynb	 1_Preprocessing_es.ipynb	 1_Preprocessing_it.ipynb
2. The generation texts by LSTM-bot	 2_Generation_en.ipynb	 2_Generation_es.ipynb	 2_Generation_it.ipynb
3. Preprocessing the bot texts	 3_Preprocessing_bot_en.ipynb	 3_Preprocessing_bot_es.ipynb	 3_Preprocessing_bot_it.ipynb
4. TF-IDF and clustering human texts	 4_TF_IDF_Clustering_en.ipynb	 4_TF_IDF_Clustering_es.ipynb	 4_TF_IDF_Clustering_it.ipynb
5. TF-IDF and clustering the bot texts	 5_TF_IDF_Clustering_bot_en.ipynb	 5_TF_IDF_Clustering_bot_es.ipynb	 5_TF_IDF_Clustering_bot_it.ipynb
6. Comparing clusterizations and statistical significance of the results	 6_Comparing_clusterizations_en.ipynb	 6_Comparing_clusterizations_es.ipynb	 6_Comparing_clusterizations_it.ipynb

### 5.2. LSTM generator (jupyter notebooks)



SGA2\_Text\_Generation\_Afanasiev\_coursera

### 5.3. Nonparametric tests for averages (jupyter notebooks)



Nonparametric-tests-for-averages.ipynb