Fathers and Sons

Sebastiano Caccaro
sebastiano.caccaro@studenti.unimi.it

Università degli Studi di Milano

1 Introduction

The aim of this project is compare the speeches of political leaders in the 1^{st}, 12^{th}, 17^{th} and 18^{th} legislature, in order to find out which political parties from different legislature speak in a more similar way.
The proposed solution consists in creating a model of speech for every political party by extracting the main topics from the speeches. The comparison between political party speeches is therefore performed using their topics. As such, two models are considered similar if their topics are similar.
Such topic extraction is performer with LDA (Latent Dirichlet Allocation), proposed in [1]. LDA is a generative statistical model that can be used on text corpora. In LDA, each document is described by a mixture of topic probabilities; in turn, each topic can be represented as set of word probabilities.

2 Research question and methodology

The goal of the project is to analyze the transcriptions of parliamentary debates of the 1^{st}, 12^{th}, $17^{th}$and 18^{th}legislature in order to find out which present political party's speeches resemble the most the speeches of ones in the past.
The problem can be destrucutred in the following steps:

- Text preprocessing
- Language modeling
- Models comparison

2.1 Text preprocessing

For each debate transcription, text needs to be divided into speeches. Parts of the document which are not speeches are discarded. Every speech gets tagged with its speaker name. This process results in every speaker having a collection of speeches. Every speech is then divided into tokens.

Data in the $1^{st}$and 12^{th}legislature is particularly noisy (see subsection 3.1). Therefore an additional correction phase has been deemed necessary.
For every legislature, every token (that is, a word) is checked against a vocabulary. This process eliminates words that cannot be corrected and weird OCR artifacts, especially in the 1^{st} and 12^{th} legislature; stopwords are also filtered out.

In the end, every token is stemmed. Therefore words that have the same root and thus have same meaning are effectively mapped into the same word. This is beneficial both in modeling topics and in classifying documents. Moreover, this approach will reduce the effective number of words, leading to a smaller document-word frequency matrix.

2.2 Language modeling

Each deputy's set of speeches is then vectorized in its own sparse document-word frequency matrix. The matrix is used to create a language model using LDA (Latent Dirichlet Allocation). For each model a perplexity score is calculated. The output of this phase, for each model, is:

- A list of topics: each topic is a set of the words in paired with their probability of being in the topic.
- For every topic, the probability of any document to be about a certain topic.

2.3 Models comparison

Model similarity is estimated taking into account the topic-word probability part of the previous output. Model similarity between two models M_1 and M_2 is calculated as follows:

- For every couple of topics in M_1 and M_2 a similarity score is calculated. Each score is put in a difference matrix where rows represent topics in M_1 and columns topics in M_2.
- Recursively the highest value of the matrix is summed to the final result, and the corresponding row and column are removed from the matrix.

The similarity score between two topics T_1 and T_2 is computed as follows:

- Each topic T_x gets vectorized: the i^{th} cell in the vector contains the probability that the i^{th} word in the set of words of $T_1 \cup T_2$ belongs to T_x.
- The similarity score corresponds to the cosine distance between the two topic vectors.

Results plotted in a table show three different "blocks" inside which similarity is greater that outside; this blocks are formed as follows:

1. 1^{st} legislature
2. 12^{th} legislature
3. 17^{th} and 18^{th} legislature

This is expected and it is almost certainly caused by the years separating the legislatures. Therefore, for every couple of blocks B_1 and B_2, the following normalization steps are applied:

- The combined average avg_b of all the cells in B_1 and B_2 is computed. The cells on the diagonal (the ones in which the same party models is confronted with itself) are not taken into account.

- The section of the table S_1 and S_2 are defined as the group of cells sitting in the same columns as B_1 and same rows as B_2 and vice versa. Since the table is symmetrical the average avg_s of their cells is the same.
- The absolute value of (avg_b - avg_s) is summed to all cells in S_1 and S_2.

3 Experimental results

3.1 Dataset

The dataset consists in the full transcriptions of parliamentary debates in the Italian House Of Representative of four different legislatures:

- 1^{st} legislature
- 12^{th} legislature
- 17^{th} legislature
- 18^{th} legislature

For every legislature the text is extracted from the PDF of the transcripts themselves. In the case of the 17^{th} and 18^{th} legislature, the PDF transcripts are native digital files. Therefore characters are encoded in UTF8 and the extracted text has been basically copy-pasted into the dataset.
In the case of the 1^{st} and 12^{th} legislature the extraction has been performed through OCR (Optical Character Recognition). This leads to many error and artifacts being present in the extracted text, such as misspelled, missing or fragmented words, and wrong or missing punctuation. As such, on this two legislatures a correction step is performed to try to correct some of the errors. Errors are detected by checking a word against a preexistent dictionary.
Since OCR errors are quite particular in the types of additions or deletions generated, standard distance-based algorithms for finding correction candidates do not work well. Therefore, an OCR-specific algorithm called TICCL [2] is used. To choose between generated candidates, a simple heuristic is used: the most frequent candidate in the corpus is the one that get chosen. Of course, a word-frequency list for the corpus needs to be calculated beforehand.
On the last two legislatures such correction step was deemed not necessary, as the amount of errors in the text it minimal.
In every legislature the extracted text also contains unwanted lines such as page headings and footings, titles, results of parliamentary votes and so on. Moreover, the text formatting follows different rules in every legislature: this requires a different parser to be written for every legislature.

3.2 Experimental methodology

In order to evaluate the models performances, the mandated metrics is perplexity. Therefore an attempt has been made to optimize the number of topics parameter through a grid search. For each party, speeches are shuffled and divided in train set and test set in a 85/15 split. The expected behaviour for train

perplexity would be to decrease as the number of topic increases, as stated in [1]. Unfortunately, the grid search showed increasing values of perplexity in both training and test sets. This could be likely caused by bug[1] in the LDA implementation of the used library. Because of this, an arbitrary number of topics of 40 is set.

3.3 Results

The models produced by LDA are obtained and put in table as described in previous sections. For each couple of models the score is a float number between 0 (totally different models) and 40 (the models are the same). Parties with less than 500 documents are discarded as results showed to be unreliable and differed too much from others.

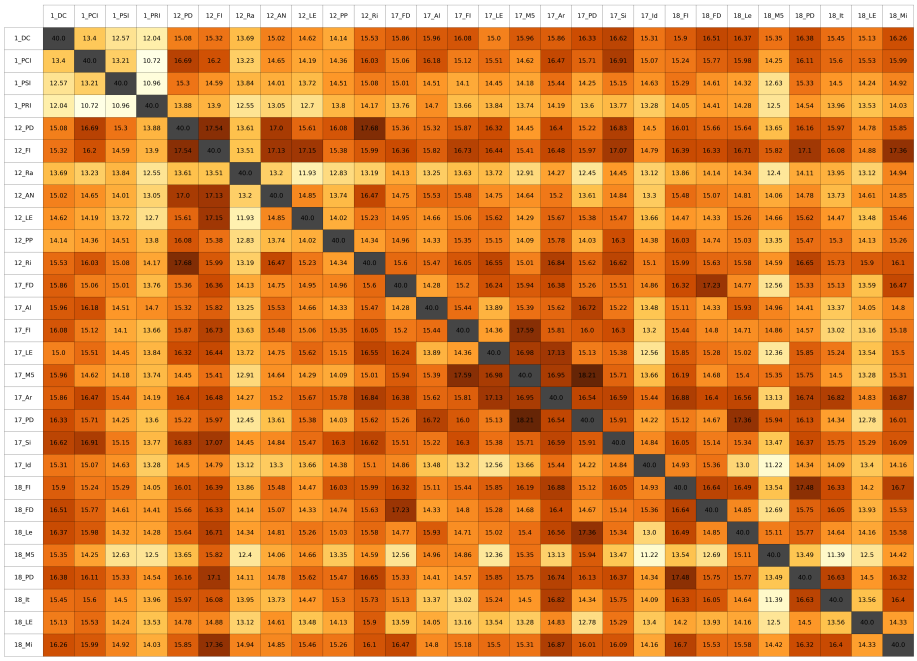

Fig. 1. Final comparison table. Some row and columns have been removed to better fit the paper. The full table can be found in the project folder.

[1] https://github.com/scikit-learn/scikit-learn/issues/6777

4 Concluding remarks

The results clearly show some marginally different scores for the parties. It is not sure, though, if some meaningful conclusions could be extracted from table. This considered, it is unlikely the proposed topic-extraction-based approach is the most suited to tackle the discussed problem.
However, some improvements could be made in future work to try to improve this approach's performance:

- More clear-cut distance metrics could be defined for topic and model comparison.
- Due to some problems with the scikit-learn, a proper grid search on the optimal number of topics parameter could not be conducted. Future work could use the gensim implementation of LDA to achieve some meaningful results.
- Due to some hardware and especially memory limitations, the project sticks to an unigram model. In fact, the limited amount of memory of the used PC (8GB), is not enough to work with n-grams models with $n > 1$. It may me interesting to try to use bigrams or even trigrams. The code already supports this option, as just a single constant would need to be changed.
- As text correction is not the main focus of this project, a simple heuristic is used to chose from some generated candidates. In future work this mechanism could be improved in order to obtain more accurate results. Moreover, the current canditate-genrating algorithm could be further optimized as it quite slow (and therefore is not used in the executable demo).

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3**, 993–1022 (2003)
2. Reynaert, M.: Non-interactive ocr post-correction for giga-scale digitization projects. In: International Conference on Intelligent Text Processing and Computational Linguistics. pp. 617–630. Springer (2008)