Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingegneria Informatica

Adaptive Systems

Practical Assignment 1

Evaluating Topic-Specific Ratio Quality: TF-IDF vs LDA

Supervisor:

Ramírez Jaime

Authors:

Fosse Cloé
Franzoso Antonio

Madrid, October 29, 2024

Contents

1 Introduction

This assignment explores content-based filtering techniques through natural language processing and topic modeling. The aim is to evaluate the thematic similarity between articles on specific topics within a dataset, using TF-IDF (Term Frequency-Inverse Document Frequency) and LDA (Latent Dirichlet Allocation) vectors to calculate a metric called *ratio quality*.

This ratio helps measure the degree to which topically similar articles are clustered together, providing insights into the coherence of specific topics across the dataset.

To achieve this, we calculated *ratio quality* using a provided pseudocode approach, where:

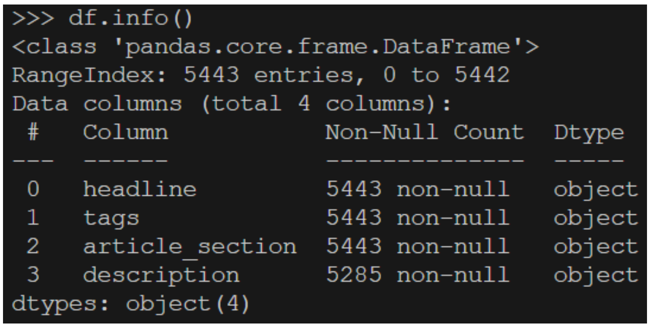
- Articles tagged under certain topics are identified.
- Each article is compared to others in the dataset to find the top-10 most similar ones.
- The proportion of top-10 articles sharing the same topic is determined for each target topic.

Through separate timings, we analyze the efficiency of TF-IDF and LDA models, from vector creation to similarity comparison, and assess differences in execution and clustering quality between the two approaches.

1.1 Dataset Overview

The dataset used in this analysis contained news articles with a total of 5443 entries. Each entry contains the following columns:

- **headline**: The headline or title of the article.
- **tags**: Keywords or tags associated with the article.
- **article section**: The section or category of the article.
  - Used to identify if an article is about a certain topic
- **description**: A brief description or summary of the article's content.
  - Used to vectorize the documents and create the *corpus*.



```
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5443 entries, 0 to 5442
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   headline         5443 non-null   object
 1   tags             5443 non-null   object
 2   article_section  5443 non-null   object
 3   description      5285 non-null   object
dtypes: object(4)
```

Figure 1: Original dataset information.

As we can see in Figure 1 , we encountered a challenge with the *description* column, as some entries (5285 out of 5443) contained null values.

Since these descriptions will be crucial for our text-based analysis, we opted to remove any entries lacking a valid description. Consequently, our final dataset only includes entries with non-null descriptions, for a total of 5285 entries.

1.2 Code Overview

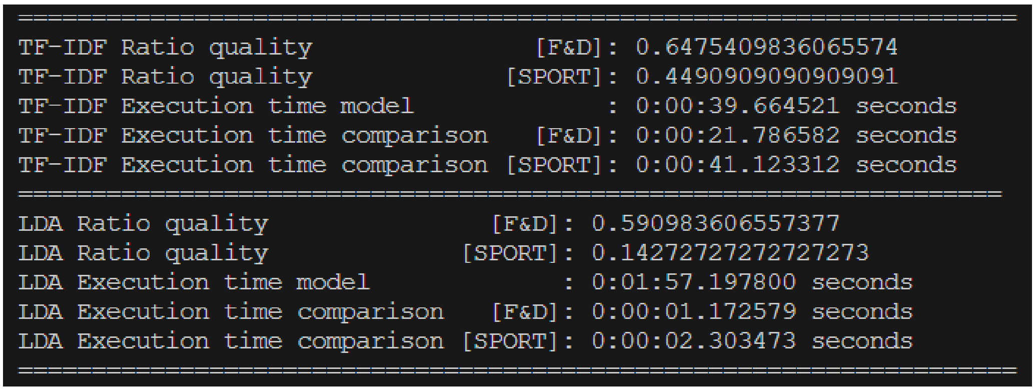
The most important part of the code is the *run* function, that performs the main processing pipeline for evaluating article similarity based on either TF-IDF or LDA models, and calculates a ratio quality score for a specified target topic (or a list of them).

It can be divided in two sub-parts:

1. **Data Loading, Preprocessing and Model Creation**
   - Reads the dataset from a CSV file (‘news1.csv‘), removing rows with missing values in the specified text column to ensure all entries have valid descriptions.
   - Calls the function `create_model` to generate a topic model or TF-IDF model, depending on the value of the `isLDA` parameter. The output of this function includes:
     - `vectors`: vector representations of the documents,
     - `matrix`: the matrix representation of the document-topic (or document-term) relationships.
   - Records the time taken for the model creation.
2. ***Ratio Quality* calculation for topics:**
   - Uses the `calculate_ratio_quality` function to compute *ratio quality* based on the cosine similarity of documents to the target topics, using the matrix associated to the chosen approach.
   - Records the time taken for the topic comparison.

2 Results

An overview of the final results can be found in Figure 2. It shows ratio quality, creation model time and computing similarities time for both TF-IDF and LDA methods, each applied on two category of articles (”Food and Drink” first and ”Sports” later).



```
====================================================================
TF-IDF Ratio quality               [F&D]: 0.6475409836065574
TF-IDF Ratio quality             [SPORT]: 0.4490909090909091
TF-IDF Execution time model             : 0:00:39.664521 seconds
TF-IDF Execution time comparison   [F&D]: 0:00:21.786582 seconds
TF-IDF Execution time comparison [SPORT]: 0:00:41.123312 seconds
===================================================================
LDA Ratio quality               [F&D]: 0.590983606557377
LDA Ratio quality             [SPORT]: 0.14272727272727273
LDA Execution time model             : 0:01:57.197800 seconds
LDA Execution time comparison   [F&D]: 0:00:01.172579 seconds
LDA Execution time comparison [SPORT]: 0:00:02.303473 seconds
====================================================================
```

Figure 2: Experiment results.

A brief discussion of the results is now presented. In particular Section 2.1 will analyze the differences between TF-IDF and LDA approaches only focusing on one topic (Question 1 and 2), Section 2.2 will do the same but comparing performance based on the different topic (Question 3) and finally Section 2.3 will try to propose a possible approach to improve the performances taking advantage of the *tags* column in the dataset (Question 4).

2.1 TF-IDF vs LDA

2.1.1 Ratio Quality Comparison

When dealing with ”Food and Drink” articles, TF-IDF (0.648) achieves a higher ratio quality than LDA (0.591).

- TF-IDF calculates similarities based on the full text of the articles, processing the term frequencies of potentially many unique words, making it better at retrieving similar articles, especially in a corpus where topics have distinct vocabularies, like ”Food and Drink”.
- LDA’s reliance on topic distributions rather than explicit term frequencies (as TF-IDF does) can lead to less sensitivity to specific term usage and may not perform as well as TF-IDF.
- Moreover, it is possible that with the current setup (30 topics and 2 passes), the model might not fully capture the correct meaning of the documents, and a more precise cross-validation phase would be necessary to correctly tune those parameters.

2.1.2 Execution Time Comparison

- **Model Creation Time**: The TF-IDF model creation takes about 39.66 seconds, while LDA requires significantly longer (1 minute and 57.2 seconds).

  - LDA's complexity is higher due to its iterative nature, where it repeatedly assigns words to topics over multiple passes. This process is computationally more intensive than TF-IDF's straightforward calculation of term frequencies and inverse document frequencies, resulting in a much longer model creation time.

- **Similarity Computation Time**: For TF-IDF, computing similarities takes 21.79 seconds for ”Food and Drink” whereas LDA completes this task faster, taking only 1.17 seconds.

  - Once the LDA model is built, it allows for quicker similarity checks as each article is represented by a topic probability vector of fixed length (30 topics). In contrast, TF-IDF operates over a much larger vector space based on the vocabulary size, making pairwise similarity calculations more computationally intensive.

2.1.3 Summary

In summary, TF-IDF seems to perform better than LDA in terms of ratio quality due to its ability to capture specific vocabulary matches, which is especially useful in a corpus where distinct words define topic boundaries. However, LDA demonstrates a trade-off with much faster similarity computations once the model is built, at the cost of longer initial model creation time and lower retrieval quality.

2.2 ”Food and Drink” vs ”Sports”

2.2.1 Ratio Quality Comparison

Both TF-IDF and LDA achieve higher ratio quality scores for ”Food and Drink” (0.648 and 0.591, respectively) compared to ”Sports” (0.449 and 0.143).

- ”Food and Drink” articles likely have more consistent and specific vocabulary, making them easier to match with similar articles. Words associated with this topic may be more distinctive and less ambiguous (e.g., ”recipe,” ”ingredient,” ”restaurant”), allowing both TF-IDF and LDA to identify relevant articles effectively.

- In contrast, ”Sports” articles cover a broader range of subtopics (e.g., different Sports teams, and events) and are likely to use more general terms that overlap with other topics or subcategories (e.g., double meaning of ”goal”). This makes it harder for both TF-IDF and LDA to identify precise matches, leading to a lower ratio quality. Additionally, the diversity within ”Sports” articles may cause LDA to distribute words across multiple topics, reducing its effectiveness in capturing strong similarity within the top 10 articles.

2.2.2 Execution Time Comparison

The TF-IDF similarity computation for ”Sports” articles takes significantly longer (41.12 seconds) compared to ”Food and Drink” (21.79 seconds). For LDA, similarity computation times are generally low and differ only slightly between ”Food and Drink” (1.17 seconds) and ”Sports” (2.3 seconds).

- The increased time for ”Sports” is likely due to the larger and more diverse vocabulary within this category. The TF-IDF model must compute similarities across more varied terms, which increases computational complexity, while LDA might suffer from the less concentrated topic distributions compared to ”Food and Drink” articles. Thus, the presence of broader vocabulary results in more computations and longer processing times.

2.2.3 Summary

In summary, ”Food and Drink” articles have better ratio quality and lower execution times, especially with LDA, due to their relative homogeneity. The ”Sports” category, due to its diversity and varied vocabulary, poses challenges for both models in effectively clustering similar articles, leading to a lower quality ratio.

2.3 Improvement Proposal: Combined Similarity

To improve the quality of similarity detection, we propose a method that combines the similarities calculated from article descriptions with those obtained from the associated tags. The underlying intuition is that while the description provides a nuanced view of each article’s content, the tags offer a concise summary of its core topics or categories, which may enhance the accuracy of the similarity calculation.

In this approach, we calculate two separate similarity scores for each article:

- **Description-based similarity**: Using either TF-IDF or LDA, we calculate the similarity between articles based on the content in their descriptions.
- **Tag-based similarity**: Using a simple method such as Jaccard similarity or Cosine similarity, we compute the similarity between articles based on their tags.

After computing these two similarity scores, we combine them to produce a final similarity score for each article pair, assigning a weight to each score to control their influence on the final result. This approach allows for flexibility in emphasizing either the content of the article or the categorization provided by the tags.

The combined similarity calculation can be expressed in Equation 1:

$$\text{final\_similarity}(a, b) = \alpha \times \text{similarity\_desc}(a, b) + (\beta) \times \text{similarity\_tags}(a, b) \tag{1}$$

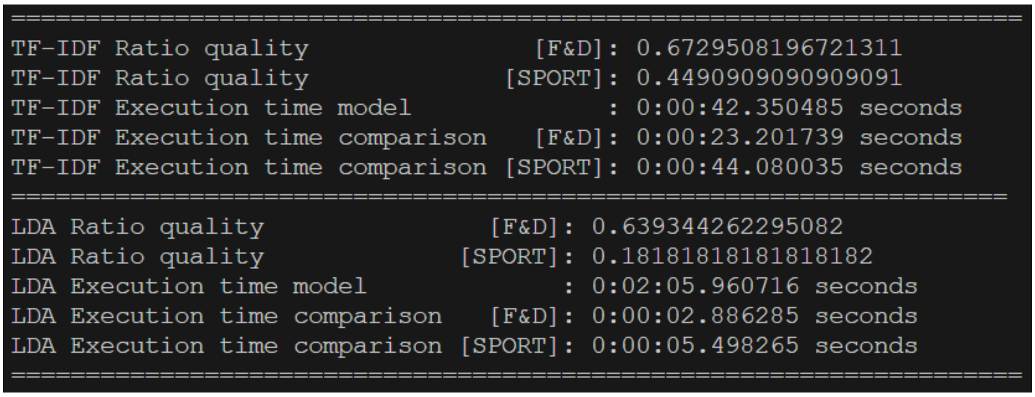
where:

- α and $\beta = 1 - \alpha$ are weight parameters, that controls how much relative importance we want to assign respectively to the description-based similarity and to the tags-based one.
  - Since description provide us with indeed more information about the article than the tags, a value like $\alpha = 0.7$ seems reasonable enough.
- $\text{similarity\_desc}(a, b)$ is the similarity score between articles a and b based on descriptions.
- $\text{similarity\_tags}(a, b)$ is the similarity score between articles a and b based on tags.

2.4 Implementation

To explore the effectiveness of combining description and tag-based similarities, we implemented the proposed approach described in Section 2.3. For consistency and brevity, and to leverage our existing code structure, we calculated tag-based similarities using TF-IDF model as we did with the article descriptions, even though a simpler similarity measure like Jaccard similarity might be enough when dealing with tags.

In this implementation, we computed separate similarity matrices for the descriptions and tags, then combined them using the formula in Equation 1.



```
====================================================================
TF-IDF Ratio quality               [F&D]: 0.6729508196721311
TF-IDF Ratio quality             [SPORT]: 0.4490909090909091
TF-IDF Execution time model             : 0:00:42.350485 seconds
TF-IDF Execution time comparison   [F&D]: 0:00:23.201739 seconds
TF-IDF Execution time comparison [SPORT]: 0:00:44.080035 seconds
===================================================================
LDA Ratio quality               [F&D]: 0.639344262295082
LDA Ratio quality             [SPORT]: 0.18181818181818182
LDA Execution time model             : 0:02:05.960716 seconds
LDA Execution time comparison   [F&D]: 0:00:02.886285 seconds
LDA Execution time comparison [SPORT]: 0:00:05.498265 seconds
====================================================================
```

Figure 3: New implementation results ($\alpha = 0.7$).

Comparing the new results (Figure 3) with the original ones shown in Figure 2, is possible to underline a minor but still evident improvement in terms of ratio quality.

More specifically, the combined approach achieves a 6-7% ratio quality increment when dealing with ”Food and Drink” articles (for both TF-IDF and LDA) indicating an improvement in matching relevant articles within this category, while ”Sports” articles seems to remain almost unaffected by the new approach (LDA-only slight improvement). However, this improvement comes with a trade-off, as the model requires 5-6% more time to build.

In summary, even though many more advanced proposals could be implemented, we are satisfied by the results obtained through our method, since it allows a ratio quality performance boost at a very cheap cost in terms of development effort and computational time.