MahleModel: An approach for topic modeling based on TF-IDF and Clustering

Lucas Sérgio Mahle[1] **, Denio Duarte**[1]

[1]Federal University of Fronteira Sul (UFFS)
Campus Chapecó
Chapecó – SC – Brazil

lucassmahle@gmail.com, duarte@uffs.edu.br

Abstract. *Due the growth of technology, there has been a considerable increase of the information capture and storage. Such large volume of data, mainly textual ones, brings a gap in data processing. Topic modeling is a class of machine learning algorithms used to extract latent topics in document collections. Latent Dirichlet Allocation (LDA) is the most used model. The idea of LDA is to use the Dirichlet distribution to define the probability of occurrence of each topic in the document. Topic modeling can be very well applied in processing large volumes of data, and for each type of data – such as images, short texts, or long documents – there is a model that is best applied for the extraction of latent topics. The present work proposes the development of a new approach for topic modeling, called MahleModel, using the TF-IDF concept with the K-means clustering tool. Some experiments are conducted to show the effectiveness of our approach regarding two state-of-art baselines: LDA and BERTopic. Results show that MahleModel is a promossing approach to extract latent topics.*

Keywords: *topic modeling, TF-IDF, topic coherence, k-means, clustering*

1 Introduction

The amount of information captured and stored has been growing considerably over the years. Often, this data is collected and stored without any practical use, but it can be processed, separated, categorized, and interpreted. However, performing these operations manually becomes extremely costly and ineffective. Therefore, there are computational approaches to assist with this type of process.

When it comes to textual data (documents), topic modeling techniques are widely used to group documents into topics [1]. Different topic extraction approaches behave differently across various collections. Some approaches perform better on certain types of collections compared to others. For instance, a topic extraction approach designed for a collection composed of scientific articles in a more general field will not be as effective as an approach aimed at extracting topics from short texts, such as tweets.

Using the example of a notary's database, which may consist of thousands of documents, and the need to categorize the entire database to separate the data into topics, a short-text approach will not yield good results, as notary documents are typically not short texts. Moreover, a complex model might perform poorly due to the large volume of data a notary might hold. In this type of scenario, a model that aims to cluster documents based on topics could perform better than the other models mentioned.

Each approach has its own strategy for estimating the relevance of each word in the context of the collection, and this is one of the factors that determine the effectiveness of the approach in generating topics. Some strategies for extracting these numerical features include the Bag of Words (BoW), which is used by LDA, or TF-IDF (term frequency/inverse document frequency), which extracts the numerical feature by calculating the relationship of the word with the corpus.

The objective of this work is to implement a topic extraction approach based on TF-IDF and clustering to determine whether the use of these combined techniques produces acceptable results compared to state-of-the-art approaches. The work uses some techniques present in LDA [1] and BERTopic [2], so the results will be compared with these methods.

The data used for the experiments was processed using the MahleModel, LDA, and BERTopic approaches, with a varied number of topics, K. The results were evaluated using the C_v metric, which generally measures the quality of the generated topics. The results indicate that the MahleModel proved to be promising compared to LDA and faster in terms of execution time compared to BERTopic.

2 Background

This chapter presents important concepts for understanding the proposed tool. In topic modeling, different strategies can be used to extract numerical features. In the case of the proposed tool, TF-IDF is used, which will also be explained in this chapter. The last concept covered will be the clustering tool, K-means.

2.1 Topic modeling

Topic modeling [3] is a set of algorithms used to discover the thematic structure of a document collection. Given this collection, the result is a set of topics formed by a group of interpretable words, which makes it possible to define a meaning for each topic.

The main objective of topic modeling [4] is that a document can be represented as a mixture of latent topics,

where each topic has a probabilistic distribution over the vocabulary.

Documents are a collection of topics, where each topic has a probabilistic distribution over certain words [5]. A topic model is a generative model for documents that specifies the probability with which a document could have been generated.

A document can be defined as a sequence of words **D**=$\{w_1, w_2, \ldots, w_n\}$, where n is the number of words in **D**. Similarly, a corpus (or collection) is a set of documents **C**=$\{D_1, D_2, \ldots, D_m\}$, where m is the number of documents in **C**. Additionally, a document can be any text-based content, such as an article or a social media comment.

Most approaches consider a document as a set of words, regardless of the order in which they appear. Preprocessing must be performed on the collection before carrying out the topic extraction step. This process generally consists of the following steps [5]:

- (i) Removal of stop words and spurious words, i.e., removing irrelevant words from the collection;
- (ii) Tokenization, transforming the collection into a list of words;
- (iii) Stemming, reducing words to their root form;
- (iv) Lemmatization, in other words, grouping the inflected forms of a word.

In Figure 1, you can see four examples of topics derived from the TASA corpus (a collection of over 37,000 passages of educational material collected by Touchstone Applied Science Associates). The table presents 16 words with the highest probability for each topic. Documents with different contents may have been generated from different topic distributions.

Topic modeling is an unsupervised machine learning technique, so it is necessary to identify the best number of topics for each collection. For this task, there are various techniques to evaluate the number of topics. The coherence metric [6] is a technique that provides a value for the generated topics, which can be compared with other numbers of topics to assess the results.

In this work, coherence metrics using C_v will be employed. This metric is the most sensitive and utilizes the sliding window technique to calculate coherence. This technique uses a window with a certain number of words and slides over other words (maintaining the number of words). In each round, the vector related to the topic word being evaluated is calculated with the words in the window. At the end of the calculation, the higher the value, the better the result.

2.2 TF-IDF

TF-IDF calculation is a commonly used approach to determine the importance of each word (or term) in the document relative to the corpus [7].

Starting from the definition of topic modeling, where a document d_i is represented by a set of words (w_1, w_2, ..., w_n) where each w_i is a word that makes up the document d_i and has a meaning and relevance in the text. This relevance of the word is calculated based on its frequency in the text (TF) relative to the frequency with which the term appears in the documents (IDF), resulting in a statistical value represented as r_i (which is the product of TF(w_i) $\times$ IDF(w_i)). Thus, a d_i can be represented as an n-dimensional vector:

$$d_i = [r_1, r_2, ..., r_n]$$

The relevance value implies that: (1) the more frequently a word appears in a document, the greater its representativeness in the text; (2) the more frequently a word appears in the documents, the greater its representativeness in the corpus [8].

Topic 247

word	prob.
DRUGS	.069
DRUG	.060
MEDICINE	.027
EFFECTS	.026
BODY	.023
MEDICINES	.019
PAIN	.016
PERSON	.016
MARIJUANA	.014
LABEL	.012
ALCOHOL	.012
DANGEROUS	.011
ABUSE	.009
EFFECT	.009
KNOWN	.008
PILLS	.008

Topic 5

word	prob.
RED	.202
BLUE	.099
GREEN	.096
YELLOW	.073
WHITE	.048
COLOR	.048
BRIGHT	.030
COLORS	.029
ORANGE	.027
BROWN	.027
PINK	.017
LOOK	.017
BLACK	.016
PURPLE	.015
CROSS	.011
COLORED	.009

Topic 43

word	prob.
MIND	.081
THOUGHT	.066
REMEMBER	.064
MEMORY	.037
THINKING	.030
PROFESSOR	.028
FELT	.025
REMEMBERED	.022
THOUGHTS	.020
FORGOTTEN	.020
MOMENT	.020
THINK	.019
THING	.016
WONDER	.014
FORGET	.012
RECALL	.012

Topic 56

word	prob.
DOCTOR	.074
DR.	.063
PATIENT	.061
HOSPITAL	.049
CARE	.046
MEDICAL	.042
NURSE	.031
PATIENTS	.029
DOCTORS	.028
HEALTH	.025
MEDICINE	.017
NURSING	.017
DENTAL	.015
NURSES	.013
PHYSICIAN	.012
HOSPITALS	.011

Figure 1: An illustration of four (out of 300) topics extracted from the TASA corpus

2.3 *K-means*

K-means is a clustering algorithm that evaluates and groups data according to its characteristics into a predetermined number of groups. The K-means algorithm consists of two phases [9]. The first phase generates k centroids randomly (where k is the number of centroids). The following phase involves associating each data point with the nearest centroid, typically using Euclidean distance.

Figure 2 illustrates an example of the initial steps

in the K-means processing [10], where each circle represents a centroid and the diamonds represent data points. The first step (step a) is the creation of k centroids in the hyper-space (in the case of the figure, there are three centroids). After this definition, all data entries are associated with a centroid (step b). In the next step (step c), the centroids are repositioned based on the data points associated with them; the tendency is for them to be moved to areas with a higher density of data points. Finally (step d), the data entries are reassigned to the repositioned centroids, and the data points associated with each centroid may vary compared to the previous execution.

Thus, it can be stated that the initial generation of centroids massively influences the outcome of the processing, as well as the number of clusters, and consequently the number of centroids.

3 Related Work

This chapter presents and describes two tools related to the work.

3.1 LDA

Latent Dirichlet Allocation (LDA) is a static document collection model that aims to capture the semantics [1].

In Figure 3, several words are highlighted in different colors, illustrating the composition of some topics. Words related to data analysis (such as computer and prediction) are highlighted in blue, words related to evolutionary biology (such as life and organism) are highlighted in pink, and words related to genetics (such as sequenced and genes) are highlighted in yellow. If these highlights were applied to all the words in the document—excluding words with little meaning, such as and, but, and if (stopwords)—it can be observed that the document covers all three topics (genetics, data analysis, and evolutionary biology) in varying proportions. This allows us to categorize which scientific field the document belongs to.

To understand its generative process, we assume that the topics are defined before any data is generated, and that a topic is a distribution over a fixed vocabulary. That is, the genetics topic contains words related to the subject of genetics. Once the topics are generated, only then are the words for each document in the collection generated in a two-stage process:

1. a distribution over a topic is randomly chosen;
2. for each word in the document:
    (a) randomly choose a topic from the distribution in step 1;
    (b) randomly choose a word according to the distribution over the vocabulary;

Each document displays topics in different proportions (Step 1), and each word in each document is generated based on a topic (Step 2b), which is chosen from a distribution of topics (Step 2a). In this example, the topic distribution would allocate probabilities to genetics, data analysis, and evolutionary biology, and each word would be drawn from one of these three topics. If a new document were generated using topics such as data analysis and neuroscience, the topic distribution would be placed over these two topics instead of the three used in the previous example. This is the distinctive feature of LDA: all documents in the collection share a set of topics, but each document displays the topics in different proportions.

3.2 Bertopic

BERTopic is a topic modeling approach that leverages transformers (document manipulation libraries) and *c-TF-IDF* (class-based TF-IDF) to create dense clusters, allowing for easily interpretable topics and maintaining relevant words in the topic description [2]. Figure 4 shows a diagram of how BERTopic works, and it is possible to see that the process is divided into three stages.

1. Incorporation of documents:
    (a) Extraction of documents and embedding with the BERT model (*Bidirectional Encoder Representations from Transformers*), although other embedding techniques can also be used;
2. Clustering documents:
    (a) Using UMAP (*Uniform Manifold Approximation and Projection for Dimension Reduction*), to reduce the dimensionality of the embedding;
    (b) HDBSCAN is used to cluster and reduce the embedding by creating clusters with similar semantics;
3. Creating topic representations:
    (a) Extracting and reducing topics with c-TF-IDF;
    (b) Improving coherence and diversity of words with *Maximal Marginal Relevance* (a technique that calculates relevance between words in the document);

3.2.1 Incorporation of documents

In the first stage, the documents are embedded, where the incorporation of documents is done using libraries that utilize the BERT modeling. [2] recommends using this model because it is pre-trained for various languages.

3.2.2 Clustering documents

In the second stage, before clustering begins, a dimensionality reduction of the embedded documents is applied. For this step, UMAP is used, and subsequently, HDBSCAN is applied to identify the outliers.

3.2.3 Creating topic representations

The third stage is creating the topic representation. For this stage, class-based TF-IDF is applied (Figure 5), which involves applying the TF-IDF algorithm within each cluster. The goal is to extract the words with the highest relevance within each cluster, as these words will better describe the topic.

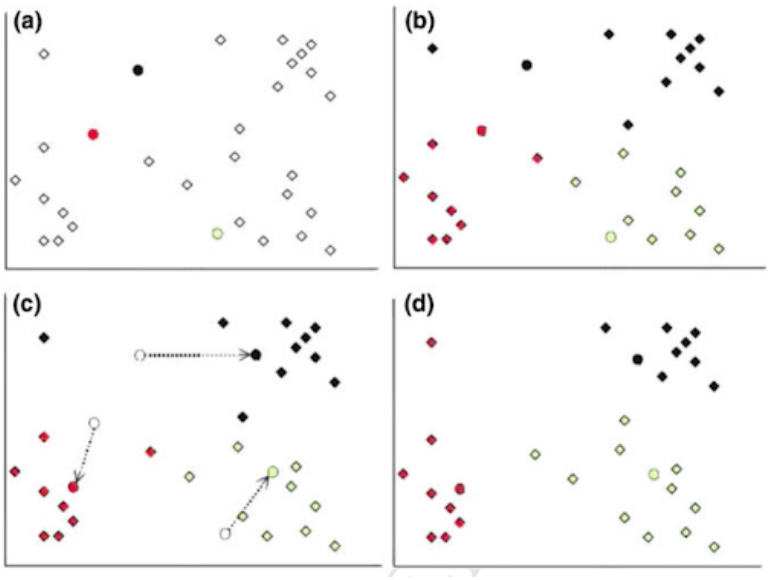

Figure 2: Initial steps of the K-means algorithm

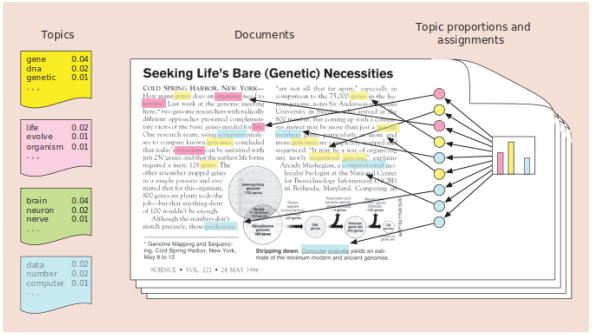

Figure 3: Example of the distribution of words into topics and topics within the document [1]

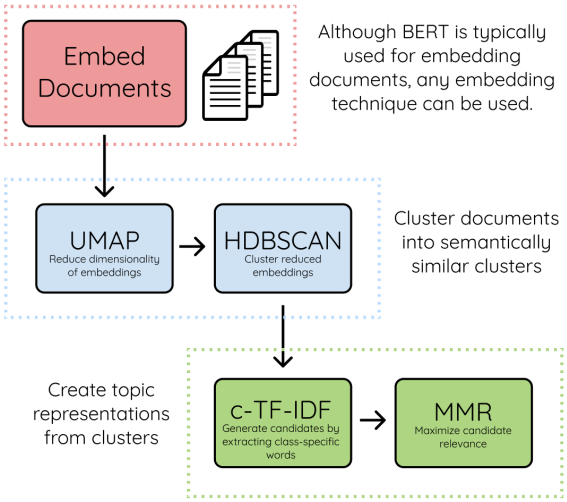

Figure 4: Representation of the operational stages of BERTopic [2]

Each cluster is converted into a single document instead of maintaining the set of documents. Then, the word frequency (represented as x in Figure 5) in class c (where c refers to the cluster that has just been created) is extracted. Next, the logarithm of one plus the average number of words per class A divided by the word frequency x across the classes is calculated. One is then added to the logarithm to ensure the value is positive. The result is the class-based IDF representation. As with classical TF-IDF, the TF value is multiplied by this result to obtain the importance score of each word in each class.

4 MahleModel

This section presents the contribution of this work: a new topic modeling model called *MahleModel*. The operation and structure of the model are discussed.

The model was implemented using the Python language and the libraries *gensim*[1], *sklearn*[2] e *spacy*[3]. The flowchart of the model is illustrated in Figure 6 and Figure 7 and is explained below.

4.1 MahleModel

In MahleModel, it is necessary to use a collection of documents for the tool to extract topics. MahleModel provides a method where you can input the original (untreated) collection, in which case the tool itself processes the documents (this step is explained in more detail in Section 4.2). Another way to train the model is by directly providing the manipulated collection (this step is explained in more detail in Section 4.3).

4.2 Collection

The input collection for MahleModel must undergo several necessary processes, such as removing stop words and lemmatization, converting spurious words (similar words

[1] https://radimrehurek.com/gensim/
[2] https://scikit-learn.org/stable/
[3] https://spacy.io/

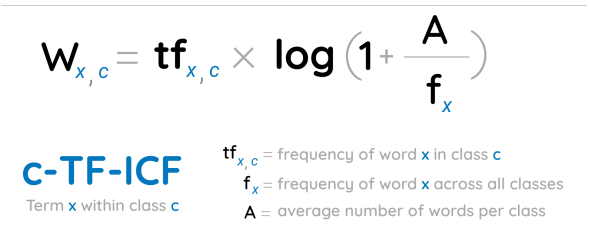

Figure 5: Formula for c-TF-IDF Calculation [2]

lacking genuine qualities), and generating bigrams. The purpose of this preprocessing is to ensure that the document is separated by words, with words without meaning (e.g., stop words) removed, while lemmatization is a process of reducing words to their root form, removing inflections. For example, occurrences of words like "amigos", "amiga" and "amizades" will be converted to "amigo", "amigo" e "amizade" respectively.

In the process of removing spurious words, the tool removes accents from words as well as removes words below a minimum length.

In the bigram process, compound words with a combined meaning are joined together. For example, the term "ser_vivo" when separated generates two words ("ser" and "vivo"), each with its own meaning, but together they convey a different meaning. In this case, the bigram step merges these words so that they are treated as one.

4.3 Dictionary and bag of words

After the documents are processed, the collection is passed to the dictionary process. In this stage, each word in the document is assigned an identification number (*id*), creating a large matrix that maps a number to a word. This stage generates a global dictionary, meaning each word receives an *id*, and this *id* is used across all documents in the trained collection. This step is crucial so that subsequent stages use the *id* of the word rather than the word itself.

Immediately after generating the dictionary, the Bag of Words (BoW) is created, which literally means 'bag of words'. The idea of BoW is to count the occurrence of each word and generate a matrix with the word's identification and the number of times it has been repeated. In this case, the *id* generated in the dictionary stage is used. However, this assignment occurs for each document, meaning the number of BoWs corresponds to the number of trained documents. The *id* of the word is relative to the dictionary, considering the global scope of the documents.

4.4 TF-IDF

Considering the BoW list and the dictionary, MahleModel applies this information in the TF-IDF model (included in the *gensim* library). The TF-IDF model performs the calculation (described in Section 2.2) for each document, generating a TF-IDF matrix where the *id* of the word and its calculated weight are present. This is the final stage (identified in the last box of Figure 6) of the training. With this, topic extraction can be performed.

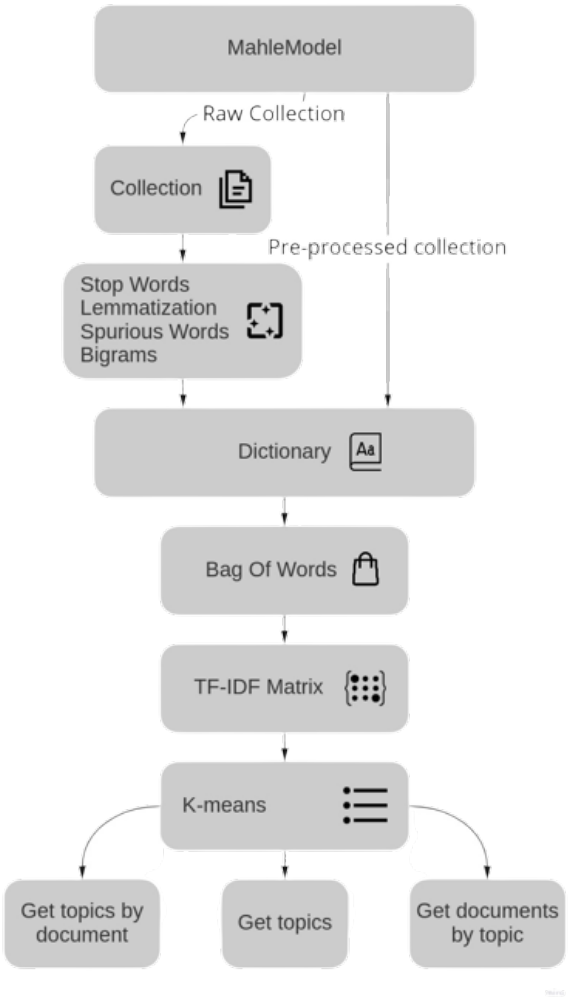

Figure 6: Representation of the stages of the MahleModel

4.5 *K-means*

The second part of the MahleModel is dedicated to topic extraction from the trained documents. In the K-means processing step, MahleModel applies the TF-IDF matrix obtained from the previous stage and also uses the desired number of topics as the number of clusters to be generated.

After the K-means processing, each cluster contains a list of pairs (<word ID>, <TF-IDF weight>).It is from this list that the MahleModel retrieves the most relevant words for each topic. The number of words forming the topic varies according to what is specified by the user for the model through a parameter.

4.6 Topics

At the end of the topic extraction processing, MahleModel provides several methods for the user.

The `get topics` method returns a list with the number of generated topics. Each item in this list is a sublist containing the most relevant words for each topic. From this list, the user can interpret and describe each topic.

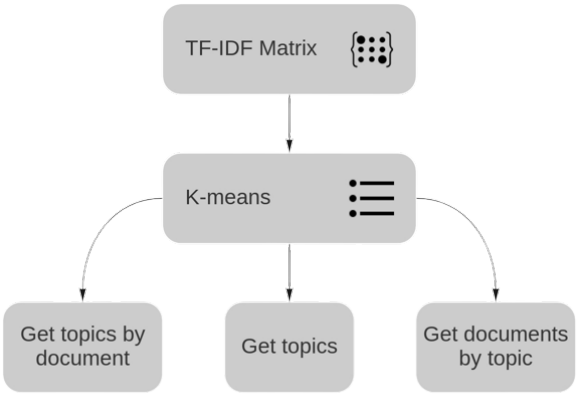

Figure 7: Representation of the MahleModel extraction steps

Pseudocode 1 Topic extraction

Require: wordsCount; KClusters
1: // For each cluster
2: **for** k = 1 to $KClusters$ **do**
3: // Sorts by TF-IDF values
4: *sortedWords* = $sortTopicWords(k)$
5: // Starts list to return word
6: *topicWord[k]* = [ ]
7: // *wordsCount* defined by the user
8: **for** $i = 1$ to $wordsCount$ **do**
9: // Get word id
10: *wordId[k]* = $sortedWords[i]$
11: // Gets the word from the dictionary
12: *word* = $dicionario[wordId]$
13: // Adds to the topic list
14: $topicWord[k].add(word)$
15: **return** $topicWord$

The pseudocode in Algorithm 1 describes how the method for obtaining the list of topics works. Initially, all clusters (topics) are iterated over (`for` on Line 2). For each topic *k*, the function `sortTopicWords` (Line 3) sorts the words in the documents of *k* in descending order of TF-IDF values. This function not only sorts the values but also processes the determination of the words that describe the topics. This process is based on the average TF-IDF value of each word within the context of the cluster, and the sorting is performed based on this average. Based on `wordsCount` (defined by the user), the words with the highest TF-IDF (loop starting from Line 7) for each topic are returned.

Another method available is `get topics by document`, described in Pseudocode 2. In this method, the user provides the index of a document from the collection, and the tool returns a list of topics that compose this document, ordered from most important to least important. This degree of importance is calculated based on the Euclidean distance from the document to all the centroids of the clusters generated by K-means. Not all topics generated are associated with the document, as a filter is applied based on a minimum distance threshold defined by the user.

Pseudocode 2 Topics by Document

Require: docId; minDistance
1: // Obtain K-means data for the document
2: *documentData* = $kmeans(docId)$
3: // Initialize composition
4: *docComp* = [ ]
5: **for** $cluster$ in $centroids$ **do**
6: *distance* = $euclidean(cluster, documentData)$
7: $docComp.add((cluster, distance))$
8: // Calculate weight inversely proportional to distance
9: $totalDistance = sum(docComp)$
10: **for** $topic$ in $docComp$ **do**
11: $docComp[topic] = docComp[topic]/totalDistance$
12: **if** $minDistance$ **then**
13: $docComp = filter(docComp, minDistance)$
14: $docCompSorted = sort(docComp)$
15: **return** $docCompSorted$

Pseudocode 3 Documents by Topic

Require: topicId; minDistance
1: // Obtain K-means centroid
2: *topicData* = $kmeansCentroids(topicId)$
3: // Initialize composition
4: *topicDocs* = [ ]
5: **for** $document$ in $documents$ **do**
6: *distance* = $euclidean(topicData, document)$
7: $topicDocs.add((document, distance))$
8: // Calculate the weight inversely proportional to the distance
9: $totalDistance =$ sum$(topicDocs)$
10: **for** doc in $topicDocs$ **do**
11: $topicDocs[doc] = topicDocs[doc]/totalDistance$
12: **if** $minDistance$ **then**
13: $topicDocs =$ filter$(topicDocs, minDistance)$
14: $topicDocsSorted =$ sort$(topicDocs)$
15: **return** $topicDocsSorted$

Finally, the method `get documents by topic` (described in Pseudocode 3) returns all documents in which the specified topic (identified by its index) is present. This method also uses Euclidean distance to measure the relevance of the topic to each document. As with the `get topics by document` method, a filter based on a user-defined minimum distance threshold is applied to determine which documents are considered in the return.

5 Experiments

This section presents the experiments to validate the MahleModel. Three approaches from the literature are used: LDA with BoW, LDA with TF-IDF, and BERTopic. Information is provided about the datasets used, as well as the preprocessing applied to them. Finally, the results of the experiments using each model on the datasets are presented, along with an analysis of the results.

5.1 Collection

With the aim of covering various scenarios of collection characteristics concerning the documents they con-

tain, three collections were chosen: IMDB, Movies, and Medium. These collections are described as follows.

5.1.1 IMBD

The *Internet Movie Database* (IMDB)[4] dataset contains 50,000 reviews of various movie genres. The dataset includes only two fields: the review and the sentiment of the review. For this experiment, only the review field will be considered. Table 1 presents some statistics of the collection. It can be observed that this dataset has a small number of words per document, with the largest quartile containing 280 words. Despite having a large volume of documents, each document is relatively short in length.

	Characters	Words
Mean	1309	230
Standard Deviation	990	170
Minimum	32	4
25%	699	126
50%	970	172
75%	1590	280
Maximum	13704	2469

Table 1: Statistics of the IMDB Collection

5.1.2 Movies

The Movies dataset[5] contains 2,000 movie reviews and was obtained through the *nltk*[6] library, in this case, the content of the texts will be used, not the library itself. Table 2 shows that the dataset has a median number of words per document, with the first quartile containing 126 words.

	Characters	Words
Mean	3904	668
Standard Deviation	1719	295
Minimum	91	17
25%	2745	470
50%	3640	628
75%	4730	810
Maximum	15097	2462

Table 2: Statistics of the Movies Collection

5.1.3 Medium

The Medium dataset[7] contains over 70,000 articles related to artificial intelligence. Due to the large number of articles, this experiment was limited to only 18,000 articles, with most documents containing a large number of words. Table 3 shows a higher average word count compared to the Movies dataset and a significantly larger number of documents. The largest document contains over 23,000 words, whereas in the other datasets, the largest documents do not exceed 2,500 words.

[4]https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

[5]http://www.cs.cornell.edu/people/pabo/movie-review-data/

[6]https://www.nltk.org/nltk_data/

[7]https://www.kaggle.com/datasets/aiswaryaramachandran/medium-articles-with-content

	Characters	Words
Mean	5158	824
Standard Deviation	5049	832
Minimum	1	0
25%	2337	358
50%	4011	641
75%	6476	1044
Maximum	144509	23720

Table 3: Statistics of the Medium Collection

5.2 Pre-processing

Before using the collections, it is necessary to pre-process their documents (as mentioned in section 4.2), which will be detailed below.

The first step in the pre-processing of the collection is the removal of stop words and the generation of lemmas (lemmatization) from the words. For both cases, the processing is performed using the *spacy* library. This library is used for natural language processing. After this step, each document becomes a list of tokens. This understanding is important, as from this point on, documents are no longer treated as text but as a list of words (or tokens).

O segundo passo do pré-processamento é a remoção de *spurious words*. Após transformar os textos dos documentos para minúsculo, é removido acentuação das palavras e considerado apenas palavras com no mínimo três e no máximo quinze caracteres, estes valores forma definidos com base em uma média de tamanho de palavras.

The second step in the pre-processing is the removal of spurious words. After converting the texts of the documents to lowercase, accents are removed from the words, and only words with a minimum of three and a maximum of fifteen characters are considered. These values were defined based on the average word length.

5.3 Execution

With all the collections pre-processed, the experiment runs one base at a time, using all the models on each collection. All models are executed using the same manipulated corpus as described in the previous section. The models were tested with 30 to 100 topics (in increments of 10, i.e., 30, 40, 50 ... up to 100). By default, a topic is formed by the ten words closest to the topic's centroid. However, this number can be adjusted, although increasing it may make the interpretation of the topic's semantics more complex due to weaker relationships between the words. For each iteration of k topics and for each approach, the training was performed using the pre-processed dataset. After training the model, coherence is calculated from the generated topics using the *gensim* library and the c_v metric (see Section 2.1). The results of the metric are stored to generate a graph showing the performance of each approach. The values on the Y-axis represent the coherence obtained from the topics generated by the model, while the X-axis represents the number of topics that the model was set to

generate. A higher value on the Y-axis indicates better results presented by the model.

5.3.1 IMBD

The results obtained from processing the IMDB dataset are illustrated in Figure 8.

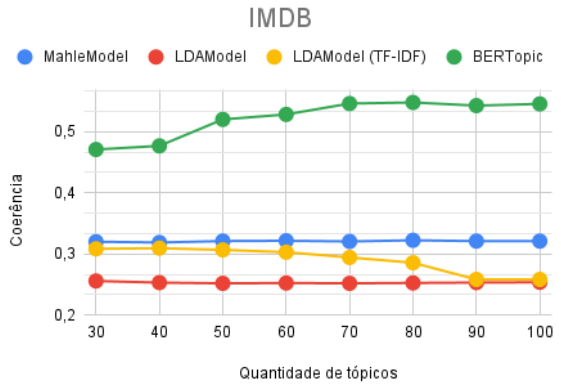

Figure 8: Results Chart Using the IMDB Dataset

MahleModel achieved its best result with 80 topics. The topic with the highest relevance (highest number of associated documents) across the entire collection was topic number 4. This topic is composed of the following words: *scene*, *say*, *life*, *star*, *woman*, *role*, *old*, *point*, *actually*, *director* and *long*.

LDA using BoW achieved its best result with 30 topics. The topic with the highest relevance across the entire collection was topic number 4. This topic is composed of the following words: *movie*, *like*, *good*, *see*, *think*, *thing*, *want*, *well*, *try* and *lot*.

LDA with TF-IDF achieved the best performance with 40 topics, and the topic with the highest relevance was topic number 3. This topic is composed of the following words: *movie*, *great*, *good*, *film*, *watch*, *think*, *well*, *see*, *like* and *story*.

Like MahleModel, BERTopic achieved its best result with 80 topics, with the most relevant topic being number 1. This topic is composed of the following words: *scary*, *horror*, *horrormovie*, *freddy*, *scare*, *movie*, *horrorfilm*, *creepy*, *house* and *film*.

The BERTopic approach had the best performance in the c_v metric for all numbers of topics. MahleModel consistently performed slightly better than the LDA-based experiments. The LDA using TF-IDF outperformed the LDA based on BoW.

Analyzing the relationship between the words of the *hot topics* from each model, only the BERTopic results showed cohesion among the words. Apparently, the most relevant topic in the results covers horror movie terms, whereas the other models produced results with little cohesion among the words. A downside of BERTopic in this experiment was that in some cases, the processing took up to 3 hours, while LDA with BoW took an average of 10 minutes, LDA with TF-IDF took an average of 25 minutes, and MahleModel performed better, taking an average of 3 minutes to produce results.

5.3.2 Movies

The results obtained from processing the Movies dataset are illustrated in Figure 9.

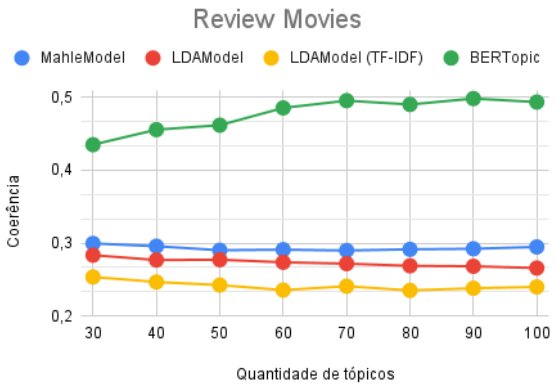

Figure 9: Results Chart Using the Movies Dataset

MahleModel achieved its best result with 30 topics, with the most relevant topic being topic number 12, composed of the words: *black*, *boy*, *mother*, *murder*, *cut*, *hollywood*, *true*, *pay*, *remember*, *break* and *case*.

The two LDA approaches achieved the best result with 30 topics. LDA using BoW had the highest relevance for topic number 4, composed of the following words: *play*, *good*, *character*, *think*, *like*, *big*, *way*, *actually*, *know* and *act*.

LDA using TF-IDF had the highest relevance for topic number 4, composed of the following words: *life*, *scene*, *story*, *bad*, *man*, *character*, *play*, *funny*, *plot* and *action*.

BERTopic achieved its best result with 90 topics, with the most relevant topic across the entire dataset being topic number 1. This topic is composed of the following words: *wrestling*, *flubber*, *murphy*, *wcw*, *comedy*, *funny*, *movie*, *wrestler*, *joke* and *laugh*.

BERTopic was the best-performing approach in the c_v metric for all numbers of topics. MahleModel consistently outperformed the LDA-based experiments, showing only slight variations. LDA using BoW performed better than LDA using TF-IDF.

Analyzing the relationship between the words of the *hot topics* for each model, only the results from BERTopic showed cohesion among the words. Apparently, the most relevant topic in the results pertains to comedy and humor, while the other models produced results with less cohesion among the words. In this experiment, MahleModel was the fastest, generating results in under a minute, whereas the other models took approximately three minutes each.

5.3.3 Medium

The results obtained from processing the Medium dataset are illustrated in Figure 10.

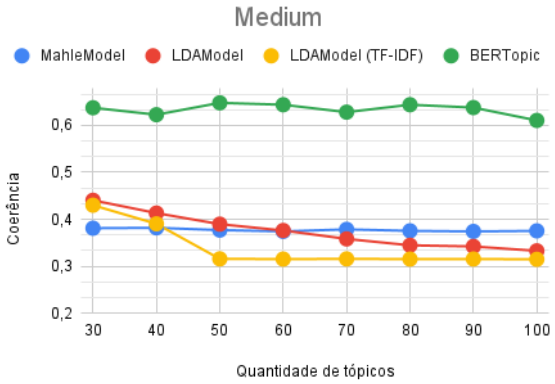

Figure 10: Results Chart Using the Medium Dataset

MahleModel achieved its best result with 40 topics, and the most relevant topic across the entire dataset was topic number 5. This topic is composed of the following words: *online*, *smart*, *internet*, *post*, *global*, *deal*, *trend*, *combine*, *currently*, *device* and *meet*.

The two LDA approaches achieved their best results with 30 topics. LDA using BoW had the highest relevance in topic number 4, which is composed of the following words: *like*, *time*, *know*, *work*, *think*, *want*, *way*, *good*, *thing* and *need*.

LDA using TF-IDF, on the other hand, had the highest relevance in topic number 18, which is composed of the following words: *people*, *think*, *thing*, *know*, *life*, *say*, *human*, *like*, *world* and *want*.

BERTopic achieved the best result with 50 topics, and its most relevant topic across the entire dataset was topic number 1. This topic is composed of the following words: *model*, *algorithm*, *datum*, *machinelearning*, *value*, *probability*, *feature*, *variable*, *example* and *point*.

BERTopic was the best performing approach in the c_v metric across all numbers of topics. MahleModel maintained consistent results, while the LDA-based experiments showed better performance with fewer topics, with performance declining as the number of topics increased. LDA with BoW experienced a more pronounced drop, while LDA with TF-IDF declined gradually until reaching results similar to LDA with BoW.

Analyzing the relationship between the words of the *hot topics* from each model, all models displayed a more cohesive topic composition compared to previous experiments. BERTopic produced results with the best cohesion between words, making it easier to understand the topic, which was related to machine learning models and algorithms. However, it took 1 hour to generate these results. MahleModel took approximately 5 minutes to generate its best result, a topic related to devices and the internet. The LDA-based models took between 13 to 17 minutes, and both models identified topics related to knowledge and thinking.

5.4 Final Considerations

The goal of the experiments was to demonstrate the viability of MahleModel as a topic extraction approach.

Considering the comparison with LDA, MahleModel performed well on the Movies and IMDB collections, and was competitive on the Medium collection. BERTopic proved to be more robust but came with a higher processing time cost.

The best performance of BERTopic on the used metric compared to MahleModel was expected. Despite MahleModel being based on the same principle as BERTopic (document clustering), BERTopic has additional layers of data processing before clustering, unlike LDA, which is based on word probabilities.

The experiments demonstrate that the proposed approach is a promising method for topic extraction from document collections, serving as a lighter alternative to BERTopic.

One point to consider is that in all collections, MahleModel exhibited very little variation in the c_v metric across different numbers of topics. A theory to explain this phenomenon is that the words generated by the model are ”generic” words from the collection, resulting in a median coherence value. However, this aspect warrants further investigation.

6 Conclusions

This article presented an approach for topic modeling called MahleModel, which is based on TF-IDF calculation and uses the K-means library for document clustering. The development of this approach was carried out using the Python programming language.

The effectiveness of the model was evaluated through experiments comparing MahleModel with three baselines: LDA BoW, LDA TF-IDF, and BERTopic. The results showed that MahleModel is a promising approach for topic extraction, being competitive with LDA and faster in terms of execution time compared to BERTopic.

Future work that can be applied to MahleModel includes conducting new experiments with collections of different natures and comparing it with other topic modeling approaches, such as Biterm Topic Models, Correlated Topic Models, and Latent Semantic Analysis.

Another aspect to consider for modifying MahleModel is the use of other distance calculation methods, such as Manhattan distance. Additionally, implementing c-TF-IDF for obtaining the words that describe the topics is also a possible modification.

A gap left in this article was understanding why MahleModel results had little variation across the number of topics in all experiments conducted, making this verification also a future research direction.

References

[1] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77, April 2012.

[2] Maarten Grootendorst. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics., 2020.

[3] Paul DiMaggio, Manish Nag, and David Blei. Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of u.s. government arts funding. *Poetics*, 41(6):570–606, 2013.

[4] Zengchang Qin, Yonghui Cong, and Tao Wan. Topic modeling of chinese language beyond a bag-of-words. *Computer Speech & Language*, 40:60–78, 2016.

[5] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[6] João Marcos Campagnolo, Denio Duarte, and Guilherme Dal Bianco. Topic coherence metrics: How sensitive are they? *Journal of Information and Data Management*, to appear(0), 2022.

[7] Zhang Yun-tao, Gong Ling, and Wang Yong-cheng. An improved tf-idf approach for text classification. *Journal of Zhejiang University-SCIENCE A*, 6:49–55, 2005.

[8] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, mar 2002.

[9] Kristina P. Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 8:80716–80727, 2020.

[10] Denio Duarte and Niclas Ståhl. Machine learning: a concise overview. In Alan Said and Vicenç Torra, editors, *Data Science in Practice*, pages 27–58. Springer, 2019.