INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Heládio Sampaio Lopes

NATURAL LANGUAGE PROCESSING FOR TREND FORECASTING

Final Paper
2020

Course of Computer Engineering

CDU 621.38

Heládio Sampaio Lopes

NATURAL LANGUAGE PROCESSING FOR TREND FORECASTING

Advisor

Prof. Dr. Filipe Alves Neto Verri (ITA)

COMPUTER ENGINEERING

São José dos Campos
Instituto Tecnológico de Aeronáutica

2020

Cataloging-in Publication Data
Documentation and Information Division

Lopes, Heládio Sampaio
Natural Language Processing for Trend Forecasting / Heládio Sampaio Lopes.
São José dos Campos, 2020.
62f.

Final paper (Undergraduation study) – Course of Computer Engineering– Instituto Tecnológico de Aeronáutica, 2020. Advisor: Prof. Dr. Filipe Alves Neto Verri.

1. Natural Language Processing. 2. Topic Modeling. 3. Machine Learning. I. Instituto Tecnológico de Aeronáutica. II. Natural Language Processing for Trend Forecasting.

BIBLIOGRAPHIC REFERENCE

LOPES, Heládio Sampaio. **Natural Language Processing for Trend Forecasting**. 2020. 62f. Final paper (Undergraduation study) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Heládio Sampaio Lopes
H8A St., 113
12228-460 – São José dos Campos–SP

NATURAL LANGUAGE PROCESSING FOR TREND FORECASTING

This publication was accepted like Final Work of Undergraduation Study

Heládio Sampaio Lopes

Author

Filipe Alves Neto Verri (ITA)

Advisor

Prof. Dr. Inaldo Capistrano Costa
Course Coordinator of Computer Engineering

São José dos Campos: November 20, 2020.

To my lovely mother, who has always supported me through this tough journey.

Acknowledgments

"Quanto mais difícil for a vitória, maior será sua felicidade em ganhar."

— Pelé

Resumo

Com avanço cientifico-tecnológico em escalas globais, a disponibilidade de dados para serem analisados cresce de maneira exponencial. A partir desse crescimento acelerado, técnicas de inteligência artificial, como o Processamento de Linguagem Natural (PLN), ajudam a automatizar o processo de se extrair informação de documentos. Com intuito de realizar previsões sobre as tendências que surgirão em meio ao crescimento de dados, é possível utilizar técnicas de modelagem de tópicos para agrupar documentos em temas semelhantes e estudar a incidência temporal desses temas.

Assim, este trabalho busca aplicar técnicas de PLN combinadas com aprendizado supervisionado para agrupar documentos em um conjunto de temas e a partir disso aprender a identificá-los em novos documentos em um período futuro. Utilizando publicações cientificas da conferência de Sistemas de Processamento de Informação Neural, fomos capazes de modelar essa tarefa e obter bons resultados. Também estudamos a capacidade dos classificadores de manter desempenho satisfatório com o passar do tempo.

Abstract

With scientific and technological advances on global scales, the availability of data to be analyzed grows exponentially. Based on this accelerated growth, artificial intelligence techniques, such as Natural Language Processing (NLP), increasingly help to automate the process of extracting information from documents. In order to make predictions about the trends that will arise from data growth, it is possible to use topic modeling techniques to group documents on similar clusters and study the temporal incidence of those topics.

Thereby, this work aims to apply NLP techniques combined with supervised learning to group documents into a set of themes and from there learn to identify them in new documents in a future period. Using scientific publications from the Conference on Neural Information Processing Systems, we were able to model this task and obtain good results. We also studied the classifiers' ability to keep a satisfactory performance over time.

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	Artificial Intelligence
BoW	Bag of Words
CBOW	Continuous Bag of Words
FN	False Negative
FP	False Positive
IT	Information Technology
LDA	Latent Dirichelt Allocation
LOWESS	Locally Weighted Scatterplot Smoothing
ML	Machine Learning
NIPS	Neural Information Processing Systems
NLP	Natural Language Processing
NPMI	Normalized Pointwise Mutual Information
OLS	Ordinary Least Squares
TF-IDF	Term Frequency Inverse Document Frequency
TN	True Negative
TP	True Positive

Contents

1 Introduction

Recently, the growth of artificial intelligence has been helping us to solve problems in most various areas, including linguistics. With natural language processing techniques, computers can process text in order to extract information faster than humans.

1.1 Motivation

Every kind of expression, verbal or in writing, brings us much information to be interpreted. Whether the topic is chosen, the tone used, the choice of words, everything can be interpreted, and then generate some valuable information.

Over the years, more and more knowledge is generated and we humans are unable to process such an amount of information. A study made by Beath *et al.* (2012) reveals that the enterprises' volume of data is expanding by 35%-50% every year, including textual data. Despite this, they also find that the companies could not extract information from this huge amount of data.

In that way, extracting valuable information from this data has become more and more an important skill. Therefore, Natural Language Processing emerges as a technology capable of assisting us in the task of deal with textual data.

Khurana *et al.* (2017) defines Natural Language Processing, abbreviated by NLP, as a branch of artificial intelligence capable of making computers comprehend and extract information from human language. In fact, NLP can perform easy tasks but not quite scalable for a human, such as translate text from one language to another, identify different topics for a set of documents, classify text on predefined subjects, and beyond that extract the sentiment to know what people are saying about something.

In recent years a particular kind of NLP application is drawing a lot of attention. The voice assistants like Siri, Cortana, Alexa and Google Assistant are the hot AI topics of the moment. Together, all digital assistants attempted to answer more and more questions each year, (KRAYEWSKI, 2019).

In our particular scope, the topic discovery was widely explored in the literature. Hurtado *et al.* (2016) and Jelodar *et al.* (2020) use it to group, respectively, papers abstracts and comments in social media in similar clusters. The first one uses it to study the topics' evolution and correlation between the founded topics. In contrast, the second group comments to analyze the general sentiment about thenm.

1.2 Objective

Curious about the fast world's evolution, this work aims to implement Natural Language Processing techniques to propose a framework capable of modeling in real-time the topics' evolution over time, using an annual granularity. With this in mind, we must evaluate the ability of those models in modeling the themes occurrence over the years.

By the end of this work, a series of tasks must be done. First, we must normalize the documents to segment them into time-ordered subsets, *Past*, *Present* and *Future*, respectively. With the eldest subset, we must find meaningful subjects to train classification models based on these topics. Then, evaluate the predictions for *Present* set and how the models' performance degenerates over the years.

1.3 Organization of this work

The remaining of this work is organized as follows: Chapter 2 will cover the theory behind Natural Language Processing and basics Classification. Chapter 3 will describe some previous works which use topic discovery and trend forecast. Chapter 4 will better explain the problem and the methodology that will be used to handle it. Chapter 5 will show the discovered results for the proposed experiment and some discussion about them. Finally, Chapter 6 will present some conclusions and suggestions for future work.

2 Literature Review

In this chapter, we will introduce the general concepts and techniques behind Natural Language Processing. We will cover all the necessary steps for extracting meaningful topics from texts. Finally, a brief discussion about machine learning for tasks like classification and regression.

2.1 Natural Language Processing

In artificial intelligence, NLP appears as the field responsible for study the interactions between computers and natural human language. Thus, it is capable of the program a computer to extract significant information from the text corpus, called documents, and use this information to apply a machine learning model and use it in several applications.

2.1.1 Text Processing Techniques

The key task to several machine learning problems consists in make good data processing before applying any model. A clean data set can allow a model to increase its performance in the learning process, making a better identification of the patterns present in the variables. Therefore, in the following sections, it will be discussed a few techniques to clear the text and prepare it for machine learning algorithms.

2.1.1.1 Normalization

There is no right way to normalize text. This process is crucially important to put all text at the same level. A normalization process consists of a series of steps to be followed consecutively, all of then can be seen as 4 big tasks: stemming, lemmatization, stop word removal, and everything else.

1. Stemming: is the process of reducing inflected words to a primitive form, the stem. This method can remove the word's affixes to capture its base meaning and still

reducing the number of variations to save memory space. Figure 2.1 shows how some inflections for "connect" can be converted to its root form.

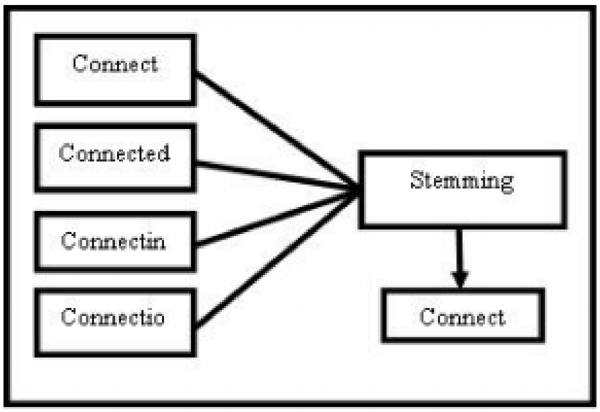


FIGURE 2.1 – Stemming process for "connect" variations, Figure from (VIJAYARANI *et al.*, 2015).

2. Lemmatization: similar to stemming, this step also reduces words to some primitive form, but with a little improvement. Lemmatization can return the words to his dictionary form, based on its part of speech context. Hence, it is possible to discriminate words with the same spelling but different meanings depending on the context.

3. Remove stop words: many words can occur a several time in a document without adding any meaningful information, such as *the*, *is*, *at*, *which*, and *on*. Their high frequency can be identified as an obstacle to perform good results on NLP models, (KANNAN; GURUSAMY, 2014).

   The classic and easier method for eliminating stop words is based on using a pre-compiled list of know words and removing them from the text. But there still are some other techniques to remove stop words, most based on evaluating the frequency of words in the corpus, as described by (VIJAYARANI *et al.*, 2015).

   A good way to perform it is by the Luhn's cut. From Zipf's law, the medium frequency words add more information, so we can cut off high and low frequency words, as shown in Figure 2.2.

4. Everything else: different from the previous steps, the last one doesn't need any grammar rules or even a frequency analysis, it's purely text manipulation. It involves set all character to lowercase; remove numbers or convert them to word form; remove punctuation; expand contractions; convert special characters to ASCII form; and any other conversion needed.

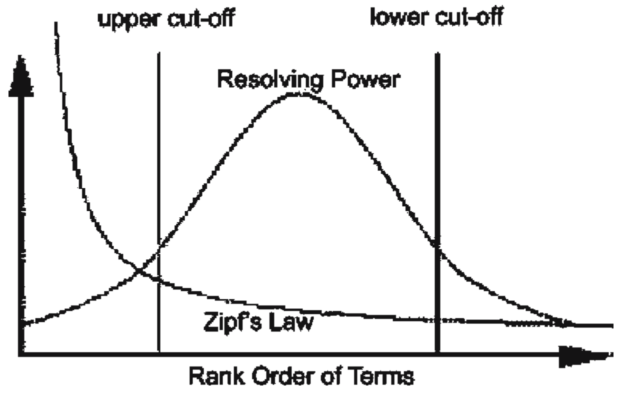


FIGURE 2.2 – Zipf's law and Luhn's proposed cut-off points, Figure from (CUMMINS; RIORDAN, 2006).

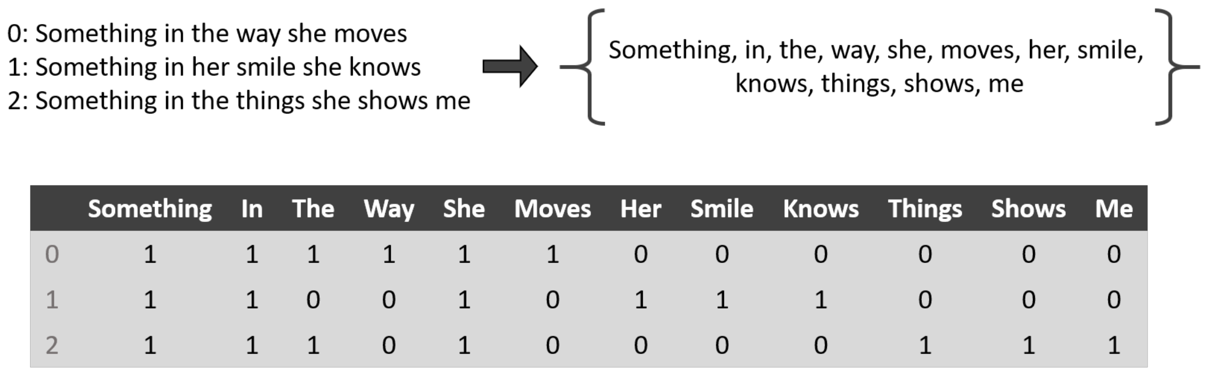
2.1.1.2 Tokenization

Once the data is normalized, we need to know how to represent it. The tokenization process consists of splitting longer strings into meaningful small pieces called tokens. The most common way to tokenize a text is chunking it into words, i.e., given a piece of text, the tokenization process will return a list of words.

2.1.1.3 Bag of Words

The machine learning algorithms take numerical features as input, hence, it will be necessary to represent the text in numerical form. With the Bag of Words model, we can represent in matrix form a set of documents.

With the tokenization output, we will have the lists representations for all documents in the data set. Those lists can be interpreted as vectors over the vector space of all unique tokens, also called by vocabulary. So, for a given sentence, we mark how many times its words appear in the list indexes where each entry corresponds to a word in the vocabulary. Figure 2.3 shows a simple example of how three sentences can be represented with the BoW model.


0: Something in the way she moves
1: Something in her smile she knows
2: Something in the things she shows me

{ Something, in, the, way, she, moves, her, smile, knows, things, shows, me }

	Something	In	The	Way	She	Moves	Her	Smile	Knows	Things	Shows	Me
0	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	1	1	1	0	0	0
2	1	1	1	0	1	0	0	0	0	1	1	1

FIGURE 2.3 – Bag of Words example.

2.1.1.4 TF-IDF

Term Frequency Inverse Document Frequency, TF-IDF for short, it is applied to a BoW to determine the relative frequency for words in a specific document when compared to the inverse proportion of that word over all documents in the collection. So, we can determine how important are the words in a specific document.

From BoW, for the i^{th} vocabulary's word in the j^{th} document, its TF-IDF weight is:

$$w_{i,j} = \text{tf}_{i,j} \times \log\left(\frac{N}{\text{df}_i}\right). \tag{2.1}$$

Where, the term frequency, $\text{tf}_{i,j}$, is how many time i^{th} word appears in the j^{th} document. The document frequency, df_i, is the number of documents in which th i_{th} vocabulary words is present. And, finally, N is the size of the document collection, with a large number of documents this term can explode, so the logarithmic function is applied to dampen this effect.

2.1.2 Word Embedding

The vectorization methods like BoW and TF-IDF can be very useful, but they can not represent the context of the words. This means the same words used in different contexts have the same representation, just as different words used with the same meaning are represented differently. Besides that, an one-hot encoding method, like BoW, presents a very sparse representation with high dimensionality.

Word Embedding is a technique to represent words in vectors capable of capture the words context in a document. It is also capable to smooth the high dimensionality effect by using a much more compact vector to represent the words.

There are three most known ways to perform a good word embedding. We will describe briefly each one of them below.

2.1.2.1 Word Representations in Vector Space

The first great word embedding technique emerged when Google researchers proposed two architectures to build continuous vector representations of words. Word's context can be observed as the words that surround it in a sentence. Then, using shallow neural networks, it is possible to calculate the word vector space based on the word's context, (MIKOLOV *et al.*, 2013).

The first suggested approach is the continuous bag of words or CBOW, the left side of

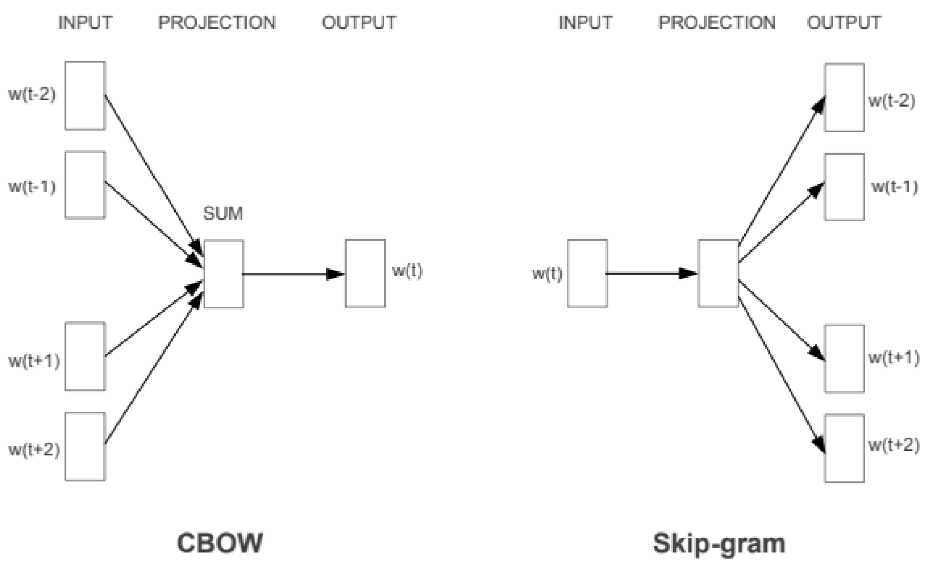


FIGURE 2.4 – Word2Vec architectures, Figure from (MIKOLOV *et al.*, 2013).

Figure 2.4 shows its architecture. Here the neural networks are designed to predict, given the context, which word is most likely to appear. So, words with the same probability to appear can assume a shared dimension in the words vector space.

The second approach is known by Skip-Gram, architecture at right in Figure 2.4. Very similar to CBOW, but instead of predicting the current word the Skip-Gram uses the current word as an input to a neural network to predict its context.

After the network training process, we can use the hidden layer weight matrix as a lookup table to build the word embedding representation. The dimension for the vector space is managed by the number of neurons in the hidden layer.

2.1.2.2 Global Vectors for Word Representation

Just a year later Pennington *et al.* (2014) arrives with a new approach to represent words in a vector space. The Global Vectors for Word Representation, or GloVe, method emerged by the need to consider some factors ignored by Skip-Gram.

Methods such as Skip-Gram learn their embedding by targeting words to their respective context, ignoring the fact that some words appear more in a context than others. Thus, this co-occurrence of words only adds more useless training examples, increasing the training complexity without adding relevant information.

GloVe, however, proposes to use the corpus statistics more efficiently. Using a weighted least squares model trained on a global word-word co-occurrence counts matrix. Thereby, it is possible to build a lookup table for the words in vocabulary and use it to represent them in a vector space.

2.1.2.3 Word Vectors with Subword Information

Both Skip-Gram and GloVe provide a good vector representation for words, but there still is an unsolved problem, What to do with unknown words? To solve this question was proposed a new embedding technique which uses subword units to build a vector space, (BOJANOWSKI *et al.*, 2017).

Similar to Skip-Gram, this new method, the FastText, train its embedding by using a target to predict the context. However, instead of using the full words, FastText goes a level deeper, breaking the words in n-grams, i.e., the word becomes its own context. The Figure 2.5 shows how the word "apple" can be broken into n-grams.

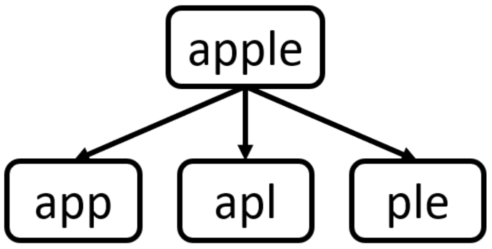


FIGURE 2.5 – Tri-gram representation for "apple" word.

There are a couple of great advantages to using this method. It is now possible to generalize new words or unseen in training data since they have the same characters as known ones. Although it is possible to use available pre-trained models, FastText requires less text to be trained, it can extract much more information from small pieces of text.

2.1.3 Topic Modeling

In text-mining, we often have document collections that we want to split into similar groups or even classify them related to each other. In this way, topic modeling is an unsupervised classification tool frequently used in text-mining to identifying the hidden patterns, called "topics", in this document collection which carries consistent semantic meaning.

Clustering methods, like hierarchical clustering, can be used to group the document collection into similar clusters. However, with a large amount of data a simple document clustering is not enough, because the semantic level of topics is not taken into account. To accomplish the topic modeling task a more sophisticated method is required, so methods like Latent Dirichlet allocation.

Latent Dirichlet allocation, or LDA for short, is a probabilistic model that uses a Dirichlet distribution to model both the topics and the words. In LDA, each item in the collection is modeled as a mixture over an underlying set of topics. And the topics, in this way, is modeled as a mixture of words over the vocabulary, (BLEI *et al.*, 2003).

Without going too deep into the math behind the model, we can easily understand the two most basic principles that guide LDA. The first one said the documents are a mixture of topics which means that in a topic space each document could be interpreted as a linear combination of these topics, however, each one of the documents must be the most homogeneous as possible. The second principle said the topics are formed as a combination of words, i.e., documents about the same topics must share similar words, and also words distribution must be the most homogeneous as possible.

Take the Figure 2.6 as an example. It shows a little fictitious document set compounded for six words "ball" and "match" which can represent the Sports topic, "pizza" and "pasta" for the Food topic end, finally "computer" and "phone" for the Technology topic. In this case, each word belongs to one topic, but that is not a rule, each topic is formed by a mix of words, and the documents a combination of topics.

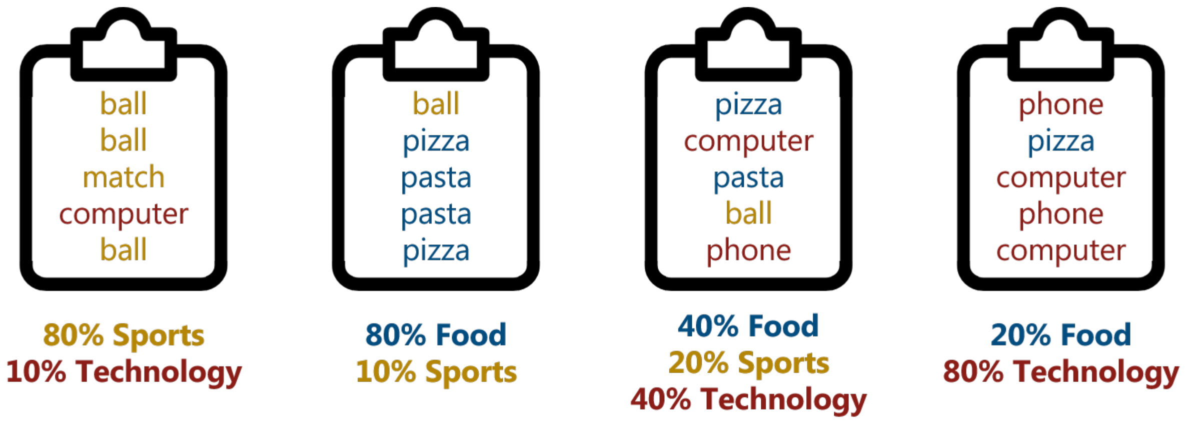


FIGURE 2.6 – Document collection modeled by topics.

A final remark for the topic modeling algorithms is that the topic is not discovered by names, the machine only discovers the distribution of the words by topics but not which one is. It's human work to label the topic once they are found.

2.1.3.1 Generative Process

To better understand this model, we will analyze the generative process for creating the document collection illustrated by Figure 2.7. Let's assume we have specified who the available topics are and their respective word distributions (far left), besides the topic proportions for each of the documents in the collection (the histogram at right).

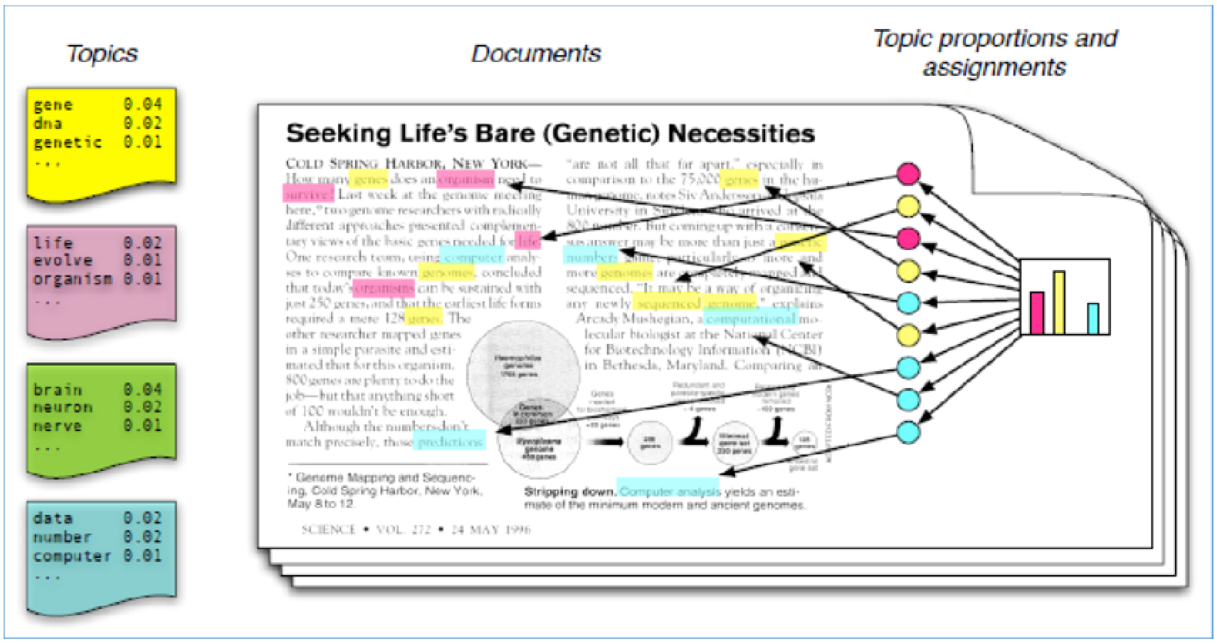


FIGURE 2.7 – The intuition behind LDA Generative process, Figure from (BLEI, 2012).

To generate a single document we choose its words as follows. Randomly choose a topic assignment from the topic distribution, then we choose the word from the corresponding topic. As shown by Figure 2.8 we can describe the generative process for a given α and β parameters.

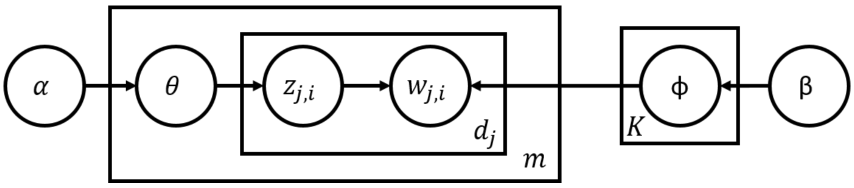


FIGURE 2.8 – Mathematical representation for LDA Generative process.

With that, we can model the generative process by:

$$p(w|\alpha,\beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{j=1}^{M} p(\theta_j|\alpha) \left(\prod_{i=1}^{V} p(z_{j,i}|\theta_j) p(w_{i,j}|z_{i,j},\phi_{z_{j,i}}) \right). \tag{2.2}$$

Where ϕ is the distribution of V words over the K topics and θ is the topic distribution over the M documents. The term inside the parentheses represent the choice of a word w for a document given the assignment of topics z for the specific document.

2.1.3.2 Model Inference

The inference process for LDA consists of finding the best fit of the distribution ϕ, for words over topics, the distribution *theta*, for topics over documents, and the assignment

z for each document in the collection given the words vectors representation, usually a bag of words model, and the hyperparameters α and β, who are the Dirichlet parameters for respective θ and ϕ distributions.

Given the expensive computational cost, we can use some techniques to infer the LDA, the most common one is the Gibbs Sampling. This method is an algorithm for obtaining a sequence of observation which are approximated from a specified multivariate probability distribution. In easy words, showed by Faleiros *et al.* (2016), the Gibbs sampler will iterate over the documents and its words to compute the relations:

$$\theta_k = \frac{C^{NWZ}_{w,k} + \beta_w}{\sum_{w_{-i}} \left(C^{NWZ}_{w_{-i},k} + \beta_{w'} \right)}, \tag{2.3}$$

$$\phi_d = \frac{C^{NZM}_{d,k} + \alpha_k}{\sum_{k'}^{K} \left(C^{NZM}_{d,k'} + \alpha_{k'} \right)}. \tag{2.4}$$

Where $C^{NWZ}_{w,k}$ is the counter of times a word w is assigned to topic k; and $C^{NZM}_{d,k}$ is the counter of times a topic k is assigned to a word in a document d. The Algorithm 1 shows in a simple way how the Gibbs sampling works to infer the LDA distributions.

Algorithm 1 Gibbs sampler

```
 1: input: number of topics $K$, document collection, $\alpha$ and $\beta$
 2: begin
 3:     Initialize the counters randomly;
 4:     while Did not converge do
 5:         for document $m \in [1, D]$ do
 6:             for word $n \in [1, N_m]$ in document $m$ do
 7:                 $nwz[w_{m,n}, z_{m,n}] - -$; $nz[z_{m,n}] - -$; $nzm[z_{m,n}, m] - -$;
 8:                 Sample a topic $z_{m,n}$ according to the previous equation
 9:                 $nwz[w_{m,n}, z_{m,n}] + +$; $nz[z_{m,n}] + +$; $nzm[z_{m,n}, m] + +$;
10:             end for
11:         end for
12:         if converged $L$ since the last sampling then
13:             calculate the distributions $\theta_k$ and $\phi_d$;
14:         end if
15:     end while
16: end
```

2.1.3.3 Evaluating Topic Models

After approximating LDA's distribution, we will have K topics represented like distributions over the vocabulary. To interpret the found topics, we use the most likely words to co-occur, usually the top 10 or 15 with high probability. However, the K parameter is a model input, so the wrong choice for it could result in unmeaningful topics.

To evaluate properly the good interpretability for a topic model, there are some topic coherence metrics. The concept behind topic coherence consists of measuring the degree of semantic similarity between high scoring words in the topic. Then, a topic is considered to be coherent if its main words are related.

There are several topic coherence metrics to evaluate interpretability for topic models, Newman *et al.* (2010) presents and compares some of them. Despite this, according to Syed e Spruit (2017), the coherence metric, labeled by C_{V}, achieves the highest correlation with all available human topic raking data. The calculation for C_{V} metric is based on four steps:

1. Segmentation of the data into word pairs, combining each topic's top-N words with every other top-N word;

2. Calculation of word or word pair probabilities, $p(w_i)$ or $p(w_i, w_j)$, respectively. That probability is calculated as the percentage of documents in which (w_i) or (w_i, w_j) occurs, ignoring the frequencies and distances between words;

3. Calculation of a confirmation measure ϕ that quantifies how strongly a word set supports another one, based on the normalized pointwise mutual information NPMI, for each segmented pair S_i. These confirmation measures can be calculated by:

$$\vec{v}(\mathrm{W}') = \left\{ \sum_{w_i \in \mathrm{W}'} \mathrm{NPMI}(w_i, w_j)^{\gamma} \right\}_{j=1,\ldots,|\mathrm{W}|}, \tag{2.5}$$

$$\mathrm{NPMI}(w_i, w_j)^{\gamma} = \left(\frac{\log \frac{P(w_i,w_j)+\epsilon}{P(w_i)\cdot P(w_j)}}{-\log(P(w_i, w_j) + \epsilon)} \right)^{\gamma}, \tag{2.6}$$

$$\phi_{S_i}(\vec{u}, \vec{w}) = \frac{\sum_{i=1}^{|\mathrm{W}|} u_i \cdot w_i}{||\vec{u}||_2 \cdot ||\vec{w}||_2}. \tag{2.7}$$

   Where ϵ and γ are parameters to, respectively, account the zero logarithm and place more weight on higher NPMI values. The vector $\vec{v}(\mathrm{W}')$ is called by context vectors.

4. Finally, aggregation of individual confirmation measures into an overall coherence score, i.e., apply the arithmetic mean for all confirmation measures.

2.2 Machine Learning

The basics behavior for a Machine Learning (ML) algorithm consists of building mathematical models based on data, usually known by training data, to provide predictions without being explicitly programmed to this, i.e., just knowing the training data, the model learns to make new predictions.

When the entries for our training data have a label, used to guide the learning algorithm, we are facing a supervised learning method. Digging a little deeper, our problem could consist in classification when the labels are restricted to a limited set of values; or regression when the labels could have any numerical value in range.

2.2.1 Classification

As a usual ML problem, a classification task consists of building a model to fit some training data set. In a classification model, we make predictions for the class of given data points, when we have only two classes, let's call then by the positive and negative, we are facing a binary classification problem.

There are several algorithms that we can use to build a classifier from the simple k-nearest neighbor until the more sophisticated ones like decision trees, naive bayes, support vectors machines, and artificial neural networks. Although each of these methods has its particularities in their formulation, they are capable of fitting the training data to make new predictions based on them.

2.2.2 Evaluating a classifier

After the training step, we need to evaluate our classifier to verify how good our model is fitting the training data. The easiest way to evaluate our model is by reserving a split of the training set, the holdout set, exemplified in Figure 2.9. So, we use this unseen data to test the model and compute a metric.

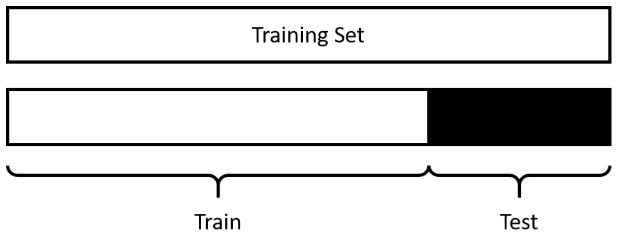


FIGURE 2.9 – Train test split for holdout method.

Another way to evaluate the model is through the k-fold cross-validation method, shown in Figure 2.10. Dividing the data set into k folds with the same size, we can train the model with $k-1$ folds and use the remaining one to test. Then we repeat this process to each fold and average the metric to evaluate the model.

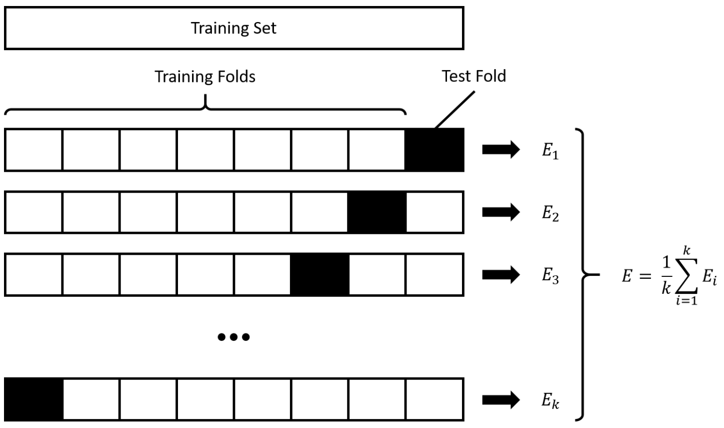


FIGURE 2.10 – Cross-validation process.

2.2.2.1 Metrics

Given the evaluation techniques, we still need a metric to properly evaluate the model. To present the main metrics, let's first take a brief look at the confusion matrix. Table 2.1 exemplifies the confusion matrix for binary classification, each row represents the entries in a predicted class and the columns represent the actual class for the entry points.

TABLE 2.1 – Binary classification confusion matrix.

		Actual Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Thus, we can have the true positives (TP) and true negatives (TN) when our model makes a good prediction about the positive and negative classes, respectively, when the model misses the prediction we call false positive (FP) or false negative (FN).

1. **Accuracy:** The simplest metric, extracted from the confusion matrix, shows us the percentage of correct predictions by the relation:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{2.8}$$

2. **Precision:** In some cases, like a very imbalanced scenario, the accuracy is not good because even if the model predicts all the instances by the predominating class we

will still have high accuracy. We define the precision of the positive class by:

$$Precision(P) = \frac{TP}{TP + FP}. \tag{2.9}$$

Thus, we will analyze the true predictions only by the positive predicted class.

3. **Recall:** This metric represents the fraction from a class that is correctly predicted. We define the recall for the positive class by:

$$Recall(P) = \frac{TP}{TP + FN}. \tag{2.10}$$

Similar to precision the recall for the positive class will compare the true predictions by the total count of true actual classes.

4. **F1 Score:** In some cases, we may want to give the same importance to both precision and recall. A popular metric to combine then is called F1-score, which is the harmonic mean of precision and recall. We define the F1-score for the positive class by:

$$F1(P) = \frac{2 \times Precision(P) \times Recall(P)}{Precision(P) + Recall(P)}. \tag{2.11}$$

3 Related Works

In the last chapter, we saw the theoretical foundation of NLP techniques, topic modeling and binary classification. In this chapter, we will review in the literature some works that employ the NLP techniques described to discover topics in a data set. In addition, we will show some applications for this type of task. And, finally, some final remarks to continue this work.

3.1 Topics Discovery

Finding meaningful topics in a document collection has been used for many authors for the most various applications. For example, Hurtado *et al.* (2016) use topic modeling to inspect research publications, patents, and technical reports aiming to model the evolution of the direction of research and forecast the near future trends in IT industry.

Using the titles and abstracts of a data set with more than six thousand academic papers between 2002 and 2010, mostly collected by Tang *et al.* (2008), they proposed a sentence-level association rule to discover meaningful topics. After categorizing the documents in topics, they were capable of building time series for each found topic, marking how many times that topic was cited in a given year. So, they were able to build an ensemble of forecasters to study the patterns and relationships among topics over the years.

For a better understanding, Figure 3.1 has a flowchart with their proposed framework for the topic discovery and forecasting.

This framework involves some well-known major steps of NLP processing. First, they convert the documents into a transactional form, i.e., the phrases in each document will be considered individually during the process. Next, they perform the basic normalization steps which include case conversion, tokenization, removing step words, part of speech tagging and stemming. It is also performed an additional step, specific to their application, removing verbs such as “exploiting”, “adapting” and “propose”, because they are very common in scientific publications and do not add much meaningful information.

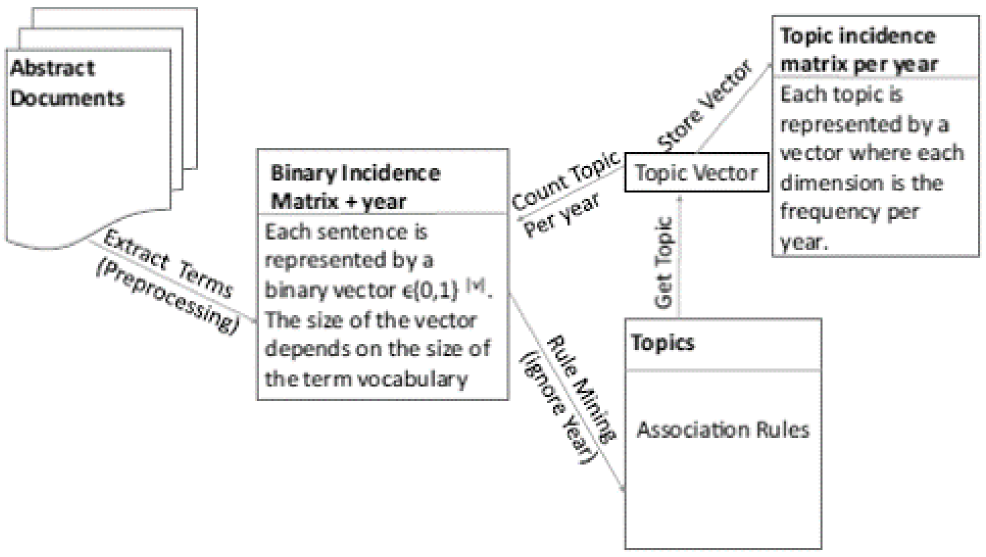

Abstract Documents
Extract Terms (Preprocessing)
Binary Incidence Matrix + year
Each sentence is represented by a binary vector ∈{0,1}^|M|. The size of the vector depends on the size of the term vocabulary
Count Topic Per year
Rule Mining (ignore Year)
Topic Vector
Store Vector
Get Topic
Topic incidence matrix per year
Each topic is represented by a vector where each dimension is the frequency per year.
Topics
Association Rules

FIGURE 3.1 – Flowchart of the proposed framework, Figure from (HURTADO *et al.*, 2016).

To vectorize the transactions, it is used a slight variation of BoW. Instead of word counting, it is only checked whether a word belongs to a transaction, this is called the binary incidence matrix. The topic discovery step comes afterward, applying an association rule mining to the transactions and discovery their patterns. In order to avoid different topics with redundant words is applied a rule refinement process that allows similar topics to be combined. At the end of this process a topic incidence matrix per year is built like Table 3.1 below.

TABLE 3.1 – Topic incidence matrix per year.

Topic	**2002**	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**
Regress_logist	2	2	10	11	8	11	12	8	5
Random_walk	0	0	4	4	4	10	9	11	19
Time_seri	45	13	10	38	29	29	36	42	37
Neural_network	14	4	2	2	2	5	7	12	7
Social_network	1	6	6	15	19	32	41	62	72
Compon_princip	2	3	9	5	12	8	11	7	9
Mixtur_model	5	8	17	19	14	22	22	6	12

Online forums and social media are excellent platforms for people to discuss and share information about a variety of subjects. Recently, the topic discovery technique was used to summarize different topics related to COVID-19 disease and perform a sentiment analysis on them (JELODAR *et al.*, 2020).

Reddit is a discussion website in which its users can submit posts and start discussions with other community members. The posts are organized in the called "subreddits", boards created by users to discuss a specific subject. Using over half a million comments from 10 health-related subreddits with information about COVID-19, Jelodar *et al.* (2020) performed a topic discovery to group similar comments. So, applying a sentiment analysis on each comment it was possible to summarize the average opinion about the discovered topics.

3.2 Trend Forecast

Predicting future trends can be very helpful in various applications, like to model the evolution of research. Following the topic discovery process from Hurtado *et al.* (2016), a forecast trend was used to predict the near future related to each discovered topic. With all documents belonging to at least one identified topic in the set, was created a topic incidence matrix that contains the count of times a topic is mentioned over the years. Finally, they make an ensemble forecasting to predict the future topic counting using the framework shown in Figure 3.2.

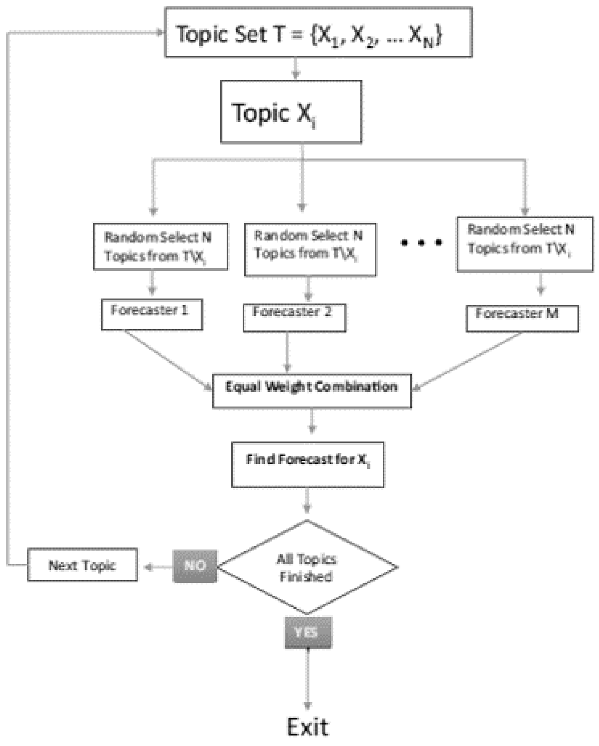

Topic Set T = {X1, X2, ... XN}
Topic Xi
Random Select N Topics from T\Xi
Random Select N Topics from T\Xi
Random Select N Topics from T\Xi
Forecaster 1
Forecaster 2
Forecaster M
Equal Weight Combination
Find Forecast for Xi
All Topics Finished
NO
Next Topic
YES
Exit

FIGURE 3.2 – Ensemble forecaster framework, Figure from (HURTADO *et al.*, 2016).

Given a specific topic, they generate M forecasters which target X_i along with N randomly chosen fields, excluding X_i. Then, the predicted value, $\hat{X}_i(t+1)$, is an average of each individual forecast, $\hat{X}_{i,F_k}(t+1)$, calculated by:

$$\hat{X}_i(t+1) = \frac{1}{M}\sum_{k=1}^{M}\hat{X}_{i,F_k}(t+1). \qquad (3.1)$$

By evaluating metrics like the coefficient of determination (R-squared) and mean squared error (MSE) they were able to conclude the ideal number N of variables to predict more accurately the future for all topics in the set.

Shen (2018) also predicted trends by analyzing the exponential growth in the volume of scholarly articles published over the years. However, he skips the NLP process step by using a pre-labeled data set from Springer containing the number of works above 14 subjects in 25 years. Using an ensemble forecast based on neural networks and support vector regression he was able to study the topics' growth and codependency between them.

3.3 Final Remarks

As we saw earlier topic discovery and trend forecast were subjects widely explored in the literature. With that in mind, we wish, in this work, to reproduce these techniques. However, in addition to what has been presented, we want to be able to explore some modifications.

Assuming a system that receives documents in real-time it would be very computationally expensive to redo all the topic discovery process with each new news. It would be much simpler if there was a process that, given an input document, would be able to identify which topics were covered by it.

So, aiming at this type of application, we will propose a system capable of such activity to model the topics evolution over time.

4 Materials and Methods

In the previous chapters, we presented the motivation of this work, the literature review showing the main concepts behind the work, then, we briefly described some related works. In this chapter, we will describe the problem, the whole experimentation setup, and the tools used to accomplish the proposed objectives.

4.1 Database

The first necessary task is to find a database spread over several years. For this, the Neural Information Processing Systems database was chosen, (NIPS, 2020). With 7241 entries spread in a thirty-year range, this base contains academic papers from the NIPS conference, one of the most prestigious yearly events in the machine learning community. Table 4.1 contains the description of the main columns present in the data set.

TABLE 4.1 – Main feature description for NIPS data set.

Feature	Description	Data Type
id	Paper identifier	Integer
year	Year of publication	Integer
title	Title of publication	Text
abstract	Publication abstract	Text
paper_text	Publication corpus	Text

4.2 Pre-processing the Data

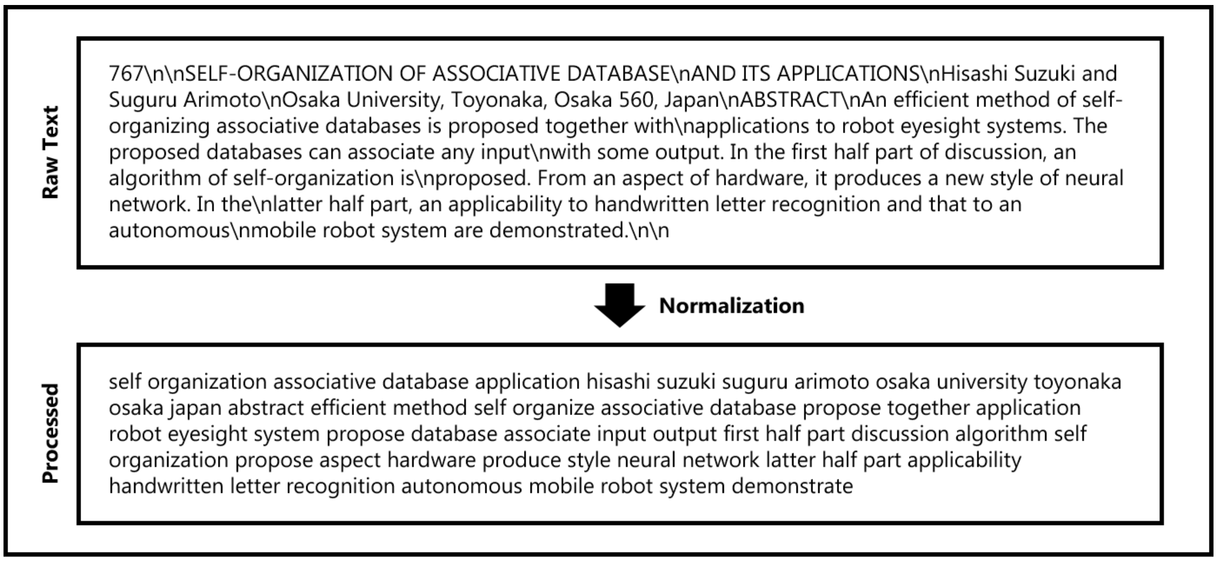
With the chosen database, we must define a data pre-processing pipeline to normalize the documents, putting all of them at the same pattern. For this, we can use the normalization techniques shown in Chapter 2 like stemming, lemmatization, stop words removal, and all necessary textual manipulations to obtain the best text representation for the documents.

4.2.1 Normalization Pipeline

To be more specific about the normalization process, let us describe it in more detail. First of all, we must concatenate the title, the abstract, and the paper text to treat all the paper information like a single document in our corpus. Done that, we proceed with the other normalization steps in the order as follows.

1. **Drop links:** drop all possible links, by regular expression, in the text that could disturb the other steps;
2. **Remove numbers:** remove all numbers, by regular expression, in the text;
3. **Expand contractions:** Convert some english contractions into their full form;
4. **Remove punctuation:** Replace all punctuation marks to empty space;
5. **Convert special characters:** Replace special characters, accented letters for example, with their ASCII form;
6. **Case conversion:** Change all characters by their lower forms;
7. **Lemmatization:** Convert the words to their root form by lemmatizing them according to their part of speech tag;
8. **Remove stop-words:** Remove the english well-known stop words.

Figure 4.1 illustrates an example of the application of the above mentioned language processing techniques to a given input.



767\n\nSELF-ORGANIZATION OF ASSOCIATIVE DATABASE\nAND ITS APPLICATIONS\nHisashi Suzuki and Suguru Arimoto\nOsaka University, Toyonaka, Osaka 560, Japan\nABSTRACT\nAn efficient method of self-organizing associative databases is proposed together with\napplications to robot eyesight systems. The proposed databases can associate any input\nwith some output. In the first half part of discussion, an algorithm of self-organization is\nproposed. From an aspect of hardware, it produces a new style of neural network. In the\nlatter half part, an applicability to handwritten letter recognition and that to an autonomous\nmobile robot system are demonstrated.\n\n

self organization associative database application hisashi suzuki suguru arimoto osaka university toyonaka osaka japan abstract efficient method self organize associative database propose together application robot eyesight system propose database associate input output first half part discussion algorithm self organization propose aspect hardware produce style neural network latter half part applicability handwritten letter recognition autonomous mobile robot system demonstrate

FIGURE 4.1 – Demonstration about the normalization process.

4.2.2 Segmentation

After processing the data, we need to index them over time and, then, split the full treated data set into three subsets. They must be time ordered, as shown in Figure 4.2, this means that for each document in T_i its time index $t_x^{(i)}$ must be lower then any index $t_x^{(m)}$ in a document contained in T_m, and so on.

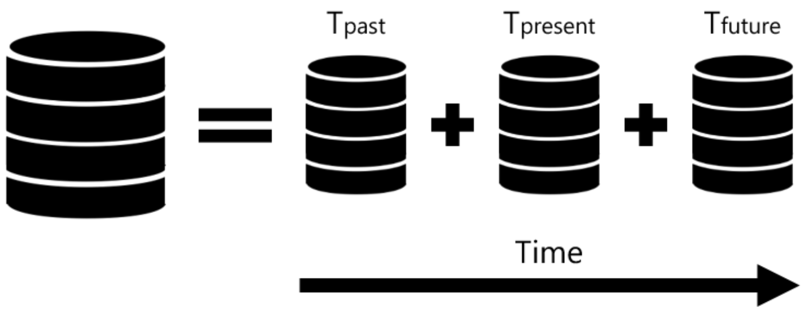
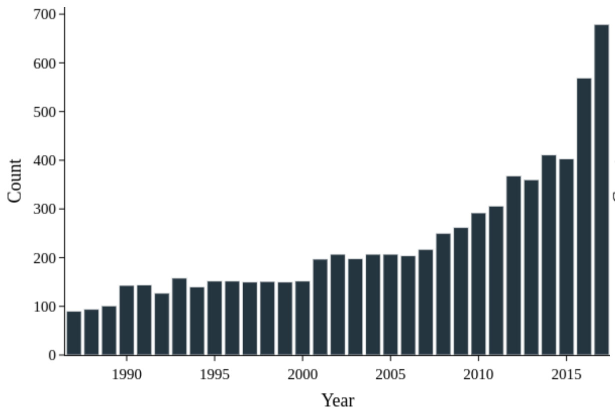
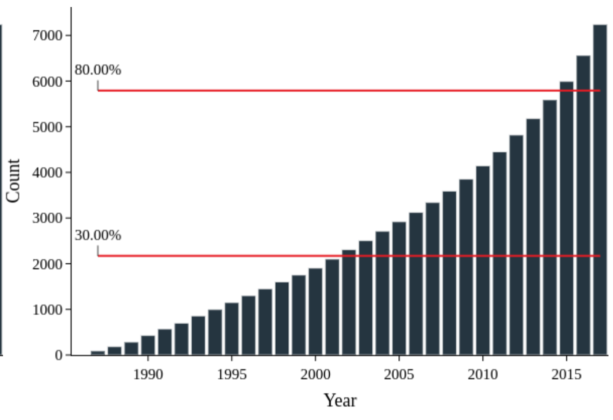


FIGURE 4.2 – Database time split representation.

Before proceeding, let us give a name to the subsets. The initial subset will be called *Past* from now on, the middle and final ones will be respectively designated by *Present* and *Future*. To segment these three subsets, let us first analyze the yearly distribution of documents. Figure 4.3a shows how the documents are spread over the years, and Figure 4.3b shows the cumulative distribution of documents with the percentiles marks of 30% and 80%.



(a) Histogram for year distribution.



(b) Cumulative histogram.

FIGURE 4.3 – Yearly distribution of documents.

After this division, shown in Figure 4.3, we approach the limits to 2002 and 2015. Then, the temporal division for the subsets was done as indicated in Table 4.2.

TABLE 4.2 – Subsets description after segmentation.

Subset	No. of documents	Years
Past	2308	1987 - 2002
Present	3685	2003 - 2015
Future	1248	2016 - 2017

4.3 Topic Modeling

With the *Past* set, techniques of topic modeling will be applied to identify the discussed subjects in the documents. Figure 4.4 illustrates the sequence of steps for this task. Next to the topic identification, we will obtain a topic set, then each document contained in *Past* can be labeled with at least one topic from the set, this new database will be called by *labeled Past*.

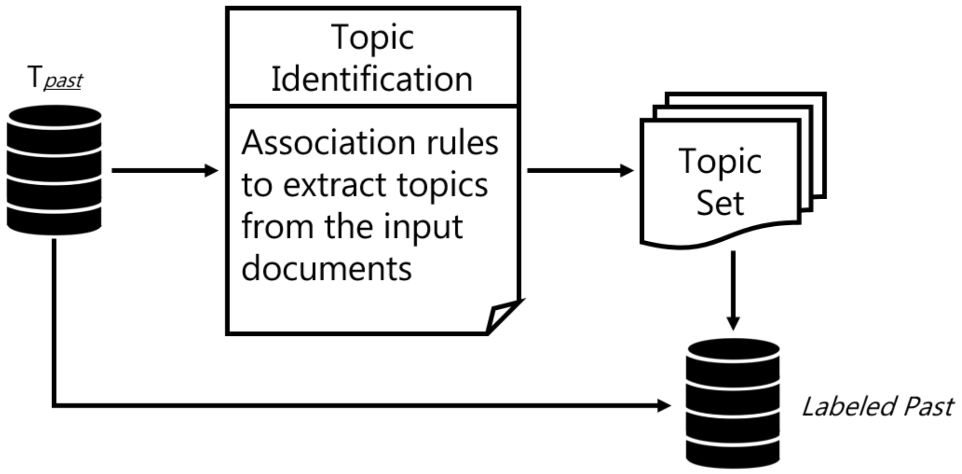

T*past*
Topic Identification
Association rules to extract topics from the input documents
Topic Set
Labeled Past

FIGURE 4.4 – Topic identification process.

To perform this, in the first place, we need to make a few adjustments to our data by choosing the proper vocabulary. We already cut off some well-known stop words, like articles, adverbs, and conjunctions. However, there are still some words outside the conventional stop word lists that can disturb the performance of the NLP algorithms. Thus, from Zipf's law, we will apply a Luhn's cut to eliminate both the most frequent and the least frequent words.

After cutting off these words, we can finally advance to the topic modeling step. The LDA algorithm needs three parameters to execute, the Dirichlet distribution parameters α and β, and the numbers of topics K. To choose the best set we must run a hyperparameter optimization, on the *Past* set, guided by the C_v topic coherence metric.

Found the best parameters, we re-run the LDA on *Past* set to fit the word distribution over topics and the topics distribution over documents. To label the data, we will proceed as follows, if for a topic the document has a probability greater than a certain threshold, we say the document has a positive class for that topic. So, at the end of this process, we will have for each K topics a class to label the documents.

4.3.1 Evaluating the *Present*

In order to evaluate the ability for a model with the *Past* make good predictions about the *Present*, we must find some correspondence between the topics from both sets. We repeat the topic modeling on *Present* set with their own vocabulary, after the Lunh's cut, by performing LDA with the same hyperparameters found by the optimization. Finally, we assign the found topics to the documents.

To associate the discovered topics from both data sets we will combine two techniques. The first one consists of comparing the intersection between the N most likely words as described by:

$$\text{Sim}_1\left(\text{Past}_i, \text{Present}_j\right) = \frac{|\text{Top}_N(\text{Past}_i) \cap \text{Top}_N(\text{Present}_j)|}{N}. \tag{4.1}$$

Where $\text{Top}_N(\text{Topic}_k)$ represents the set of N words with the higher probability to belong to Topic_k. Thus, the greater this score the greater the similarity between topics.

For the second technique, we need to represent the distribution of the words over topics only by the vocabularies intersection, so we measure the similarity by applying:

$$\text{Sim}_2\left(\text{Past}_i, \text{Present}_j\right) = \sqrt{\sum_{v=1}^{V} \left(w_v^{\text{Past}_i} - w_v^{\text{Present}_j}\right)^2}. \tag{4.2}$$

Where V is the intersection vocabulary length and $w_v^{\text{Past}_i}$ represent the normalized probability for the v^{th} word over the topic Past_i.

4.4 Document Classification

For each topic in our set of discovered topics, we must be able to identify which topics are covered by a new document. Thus, we will build a classifier to perform this verification. Knowing that a document can be about several topics, so we must have a multi-class classifier. We can see this as an individual binary classifier for each topic that tells us whether the document has it. Using the *labeled Past* set it is possible to build this classifier. Figure 4.5 illustrates it in detail.

Let us forget the multi-class problem for a while, let us focus in build a single classifier that predicts if a document talks about a certain topic. In this way, at the end of this

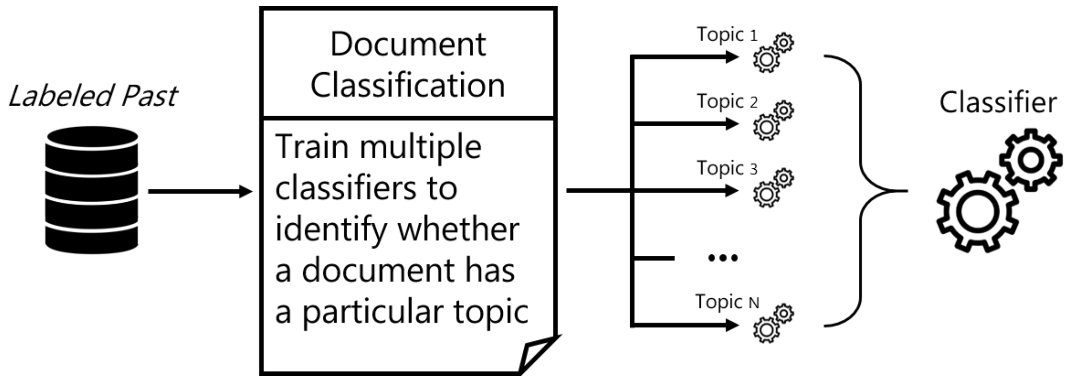

Labeled Past
Document Classification
Train multiple classifiers to identify whether a document has a particular topic
Topic 1
Topic 2
Topic 3
...
Topic N
Classifier

FIGURE 4.5 – Multi-class classifier from *Labeled Past* set.

process, we just repeat the training methods for all topics. With the *Labeled Past* data set we will try different classification models with some vectorization techniques, like BoW, TF-IDF, and embedding methods. To evaluate the good fit for our models, we will perform cross-validation to evaluate the main metrics presented in section 2.2.2.1.

Chosen the best model for our data, we must study how these models, build from the past, behave in presence of present data. For this, we encode the *Present* documents with the past vocabulary to guarantee that only the words already known by the model are used to decide the topics. Thus, we use these predictions in combination with the main topic correspondence to proceed with the analysis.

4.5 Tools

The source code for this work was done mainly with Python 3.8, (ROSSUM; DRAKE, 2009). This language has support for several machine learning libraries that will facilitate the code's implementation. Let us briefly describe some of the main packages used so far.

The pre-processing pipeline requested lots of packages, the *re* to apply regex cleaning methods, *unidecode* to convert some special characters, *contractions* to expand the contractions, *nltk* to extract the preliminary stop-words, and *spacy* to perform the lemmatization. For the topic modeling, we used mostly the *gensim* models package. It provides user-friendly methods to perform and evaluate the LDA algorithm. For the classification, it was used the *scikit-learn* both to vectorize the data and to build the models. Besides this some other mode commons packages were used, such as *pandas* and *numpy*, to array manipulation, and *plotly* to display quality visualizations.

5 Results and Discussions

In the last chapter, we detailed all the methodology steps behind the experimentation. In this one, we will show the proper results obtained by the topic modeling and classification steps.

5.1 Topic Modeling

In this first section, the results are related to the topic modeling step, divided into four stages: the vocabulary selection; the topic coherence optimization; applying LDA in past and present set; and the combination of topics from the past and future.

5.1.1 Vocabulary Selection

After the normalization, we had performed the Luhn's cut in order to remove words with document frequency higher than 60% and words that appear in less than 5 documents. The cut was performed in both *Past* and *Present* sets, the Table 5.1 shows a brief description for the vocabularies dimension before and after the selection, and for their intersection.

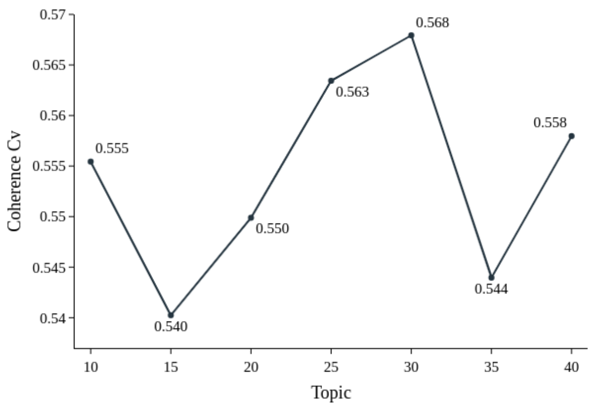
TABLE 5.1 – Vocabulary description.

	Raw	**High DF**	**Low DF**	**Filtered**
Past	77512	104	65480	11928
Present	90122	183	70356	19583
Intersect	-	-	-	9745

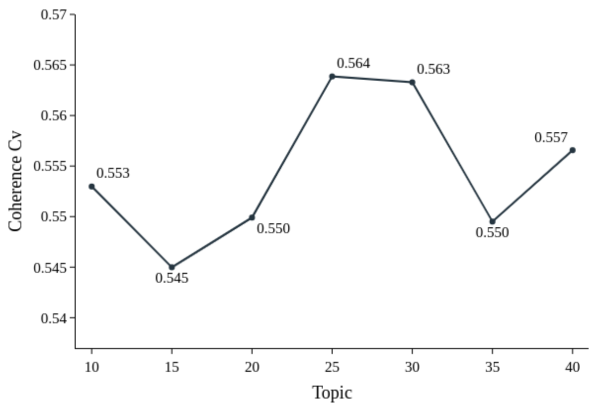
With this cut, words like “abstract”, “show” and “reference” were removed from the vocabulary. They are quite common in this type of scientific publication and are not relevant for analysis. The low frequency includes very specific words and mostly misspellings.

5.1.2 Topic Coherence Optimization

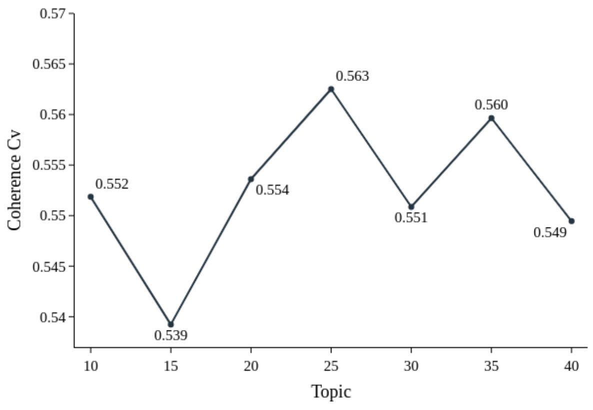
To find the best topic model for our data, an hyperparameter optimization has been performed over the three LDA parameters: $\alpha \in \{0.1, 0.5, 1\}$, $\beta \in \{0.05, 0.1, 0.5, 1\}$ and $K \in \{10, 15, 20, 25, 30, 35, 40\}$. The dataset used was the *Past* set with its vocabulary as described in the previous section. Figure 5.1 shows the coherence score C_V by the number of topics for the main α and β combinations.



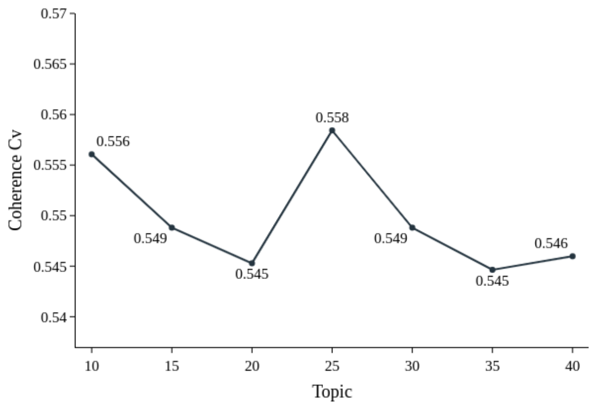
(a) $\alpha = 0.5$ and $\beta = 1.0$.



(b) $\alpha = 0.1$ and $\beta = 1.0$.



(c) $\alpha = 1.0$ and $\beta = 1.0$.



(d) $\alpha = 1.0$ and $\beta = 0.1$.

FIGURE 5.1 – Topic Coherence Optimization.

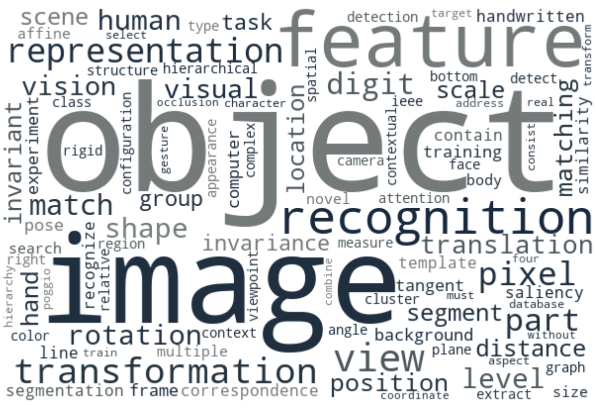
Although the combination for $\alpha = 0.5$ and $\beta = 1.0$ has it maximum for 30 topics, we can clearly detect a maximum trend for 25 topics. From the application point of view, the smaller the number of topics the better for a human to follow their evolution. In this manner, we decided on 25 topics to proceed with the experiment, with this value for K, the α and β parameters with the best coherence score are 0.1 and 1.0, respectively.

5.1.3 Finding Topics

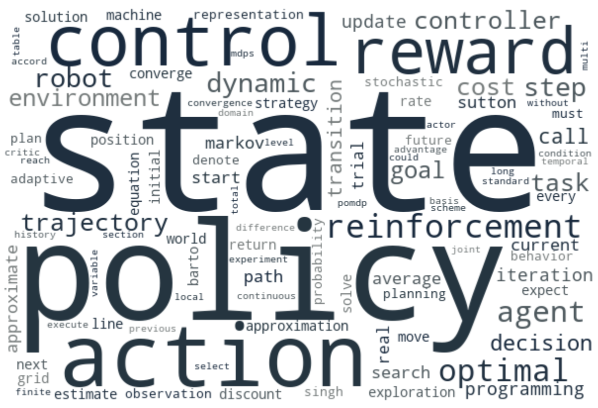
At this stage we had performed the LDA algorithm over the *Past* and *Present* sets, with the parameters established in the previous stage and their respective vocabularies, to find their topics. After the modeling the *Present* model has presented a coherence of 0.614, this indicates that the *Present* model is slightly better than the *Past* one.

5.1.3.1 Past

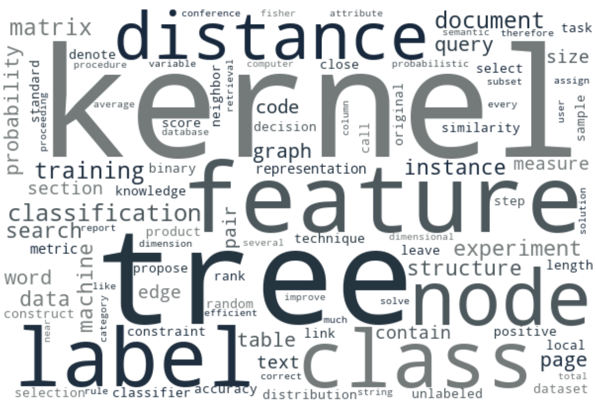
Figure 5.2 shows some word clouds for *Past* topics that are easier to identify what they talk about. For example, topic 0 talks about image recognition, topic 10 clearly is about reinforcement learning, topic 16 can be about supervised learning, and topic 19 about facial recognition. To assess all the topics in more detail check Appendix A.



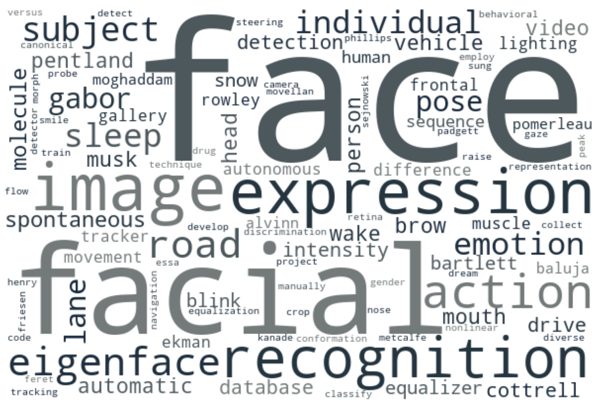
(a) Past topic number 0.



(b) Past topic number 10.



(c) Past topic number 16.



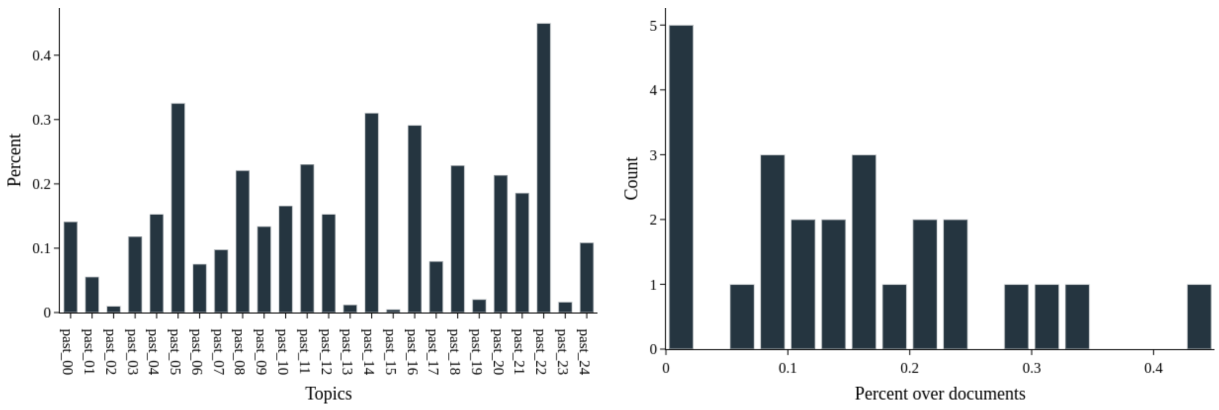
(d) Past topic number 19.

FIGURE 5.2 – *Past* topics wordclouds.

Besides getting the topics we also have the distribution of the documents of topics. Using these distributions we had labeled the documents as positive or negative for the topics with a threshold of $1/(2K)$, i.e., for a document covers a given topic his probability has to be at least 0.04.

Figure 5.3 shows the distribution of topics over the documents. In Figure 5.3a we have the percentage of documents that is about the 25 past topics, Figure 5.3b shows the histogram for these percentages, each bin with 2.5%. According to the figures, none of the topics are in more than 50% of the documents, some of them are present in less than

2.5%, but mostly they are distributed between 10% and 35%.



(a) Percentage of documents with topics. (b) Histogram for the topic percentages.

FIGURE 5.3 – Distribution of topics over the documents.

Figure 5.4 shows the distribution of topics in each document. This histogram shows that most documents have more than one topic, a few have only one topic, even fewer have more than 7, and none document has more than 10 topics.

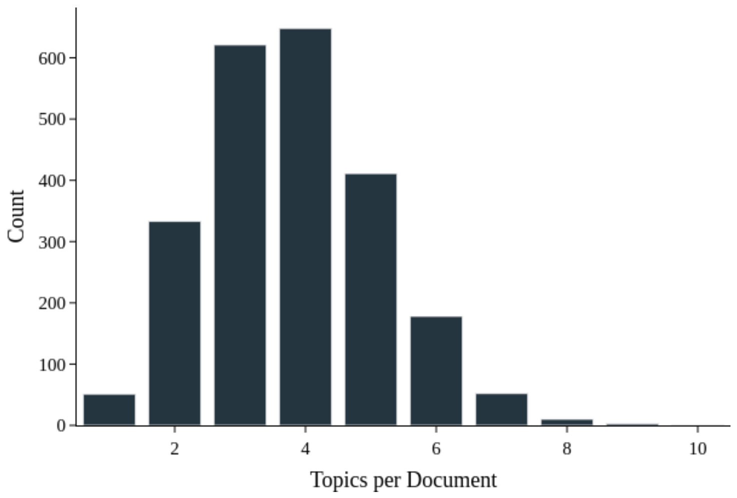
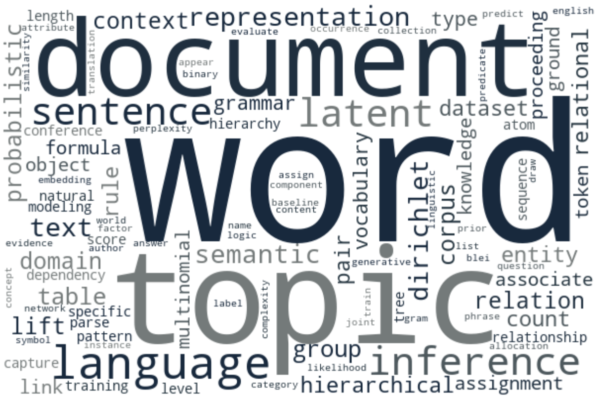


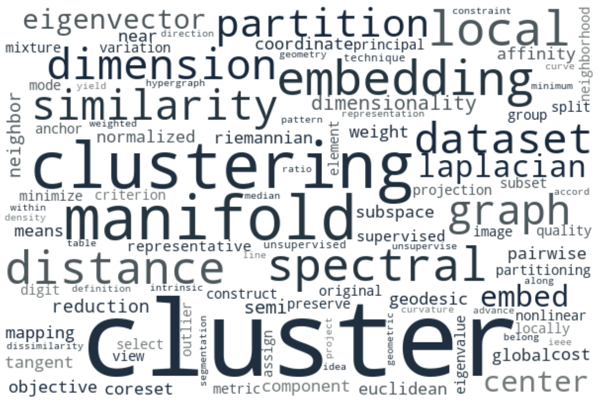
FIGURE 5.4 – Histogram for the number of topics per document.

5.1.3.2 Present

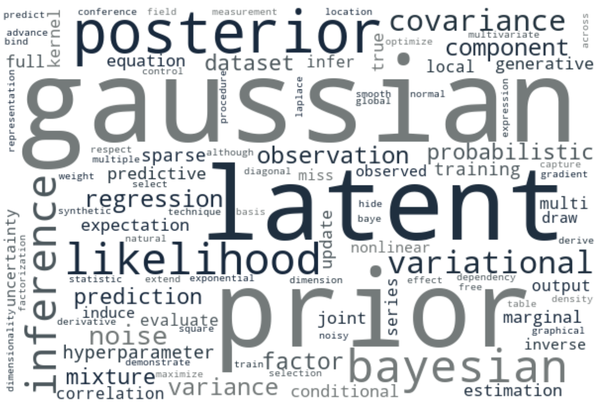
Figure 5.5 shows some word clouds for *Present* topics that are easier to identify what they talk. For example, topic 0 clearly is about NLP and topic modeling, topic 13 is about clustering, topic 16 can be about statistical inference, and topic 24 about classification. To access all the topics in more detail check Appendix B.



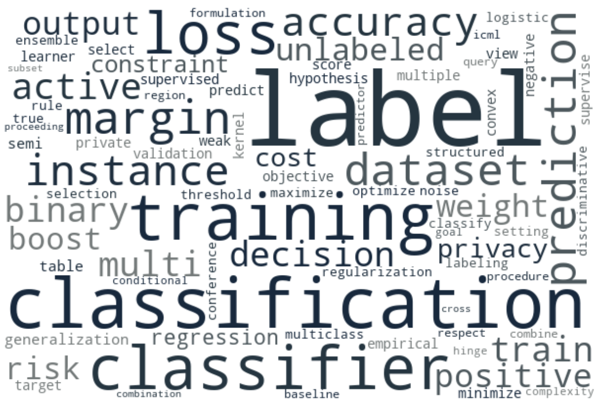
(a) Present topic number 0.



(b) Present topic number 13.



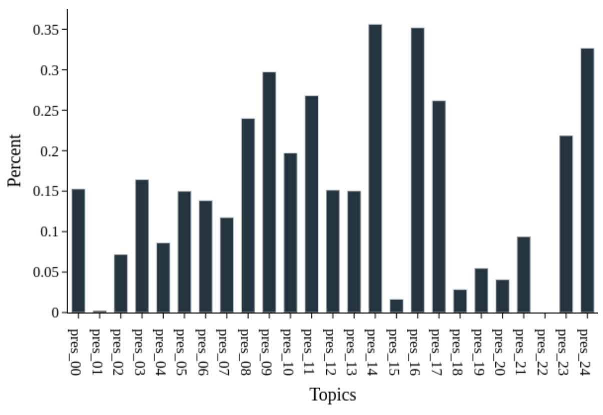
(c) Present topic number 16.



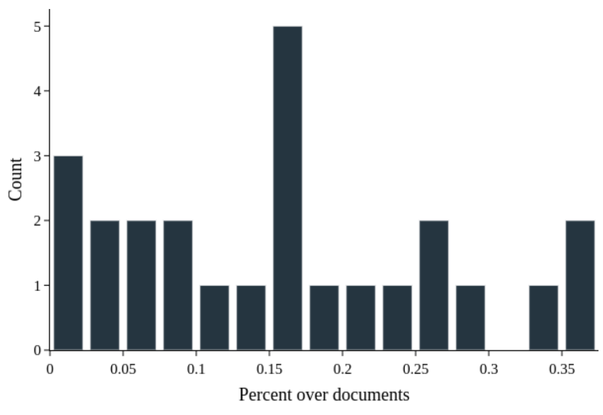
(d) Present topic number 24.

FIGURE 5.5 – *Present* topics wordclouds.

Figures 5.6 and 5.7 have the same idea as the Figures 5.3 and 5.4. The 5.6 shows that the topics are well distributed over the documents, the topic present in more documents is about 35% of them. Figure 5.7 shows the distribution of *present* topics over the docuements, similarly to the *past* distribution, the documents have mostly about 3 and 4 topics and none of then have more than 10 topics.



(a) Percentage of documents with topics.



(b) Histogram for the topic percentages.

FIGURE 5.6 – Distribution of topics over the documents.

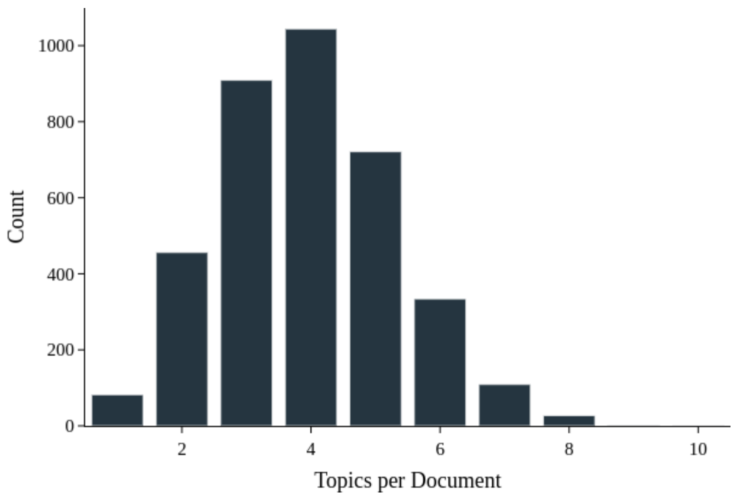


FIGURE 5.7 – Histogram for the number of topics per document.

5.1.4 Past-Present Combination

With the *past* and *present* topics defined, the similarity metrics described in section 4.3.1 were applied to obtain the topic correspondence. For Sim_1, we use the top 50 words, i.e., $N = 50$. As Sim_1 and Sim_2 are negatively correlated, to aggregate then, we divide Sim_1 for Sim_2 to capture the best from both metrics. Table 5.2 illustrates the main correspondences between *past* and *present* topics.

TABLE 5.2 – Main association for *past* and *present* topics.

Past	**Present**	**Sim_1**	**Sim_2**	**$\text{Sim}_1/\text{Sim}_2$**
11	06	58%	0.026	1119.71
10	05	56%	0.031	911.39
14	16	44%	0.040	555.48
20	09	38%	0.036	528.22
18	11	40%	0.043	467.59
07	12	34%	0.038	445.64
04	02	54%	0.061	440.94
00	08	46%	0.055	417.94
05	03	32%	0.047	341.65
22	07	36%	0.057	313.72

By aggregating the two metrics, we are choosing topics whose common word distribution is similar while giving priority to topics with the most similar top words. For example, take the combinations 18-11 and 07-12, the second one is the closest in terms of word distribution, but the first has the top 50 more similar making this combination stronger. Table 5.3 shows the intersection for those top 10 topic combinations.

Through the intersections we see that the topics of the *past* continued in the *present.* For example, the combination 10-05 talks about reinforcement learning, and 22-07 is about

TABLE 5.3 – Words intersection for the main combinations.

Present	**Past**	**Words**				
11	06	neuron	spike	cell	response	stimulus
10	05	policy	action	control	reward	reinforcement
14	16	gaussian	mixture	likelihood	prior	bayesian
20	09	gradient	constraint	convergence	update	cost
18	11	bound	theorem	loss	bind	proof
07	12	subject	stimulus	human	response	trial
04	02	signal	filter	noise	frequency	channel
00	08	object	image	recognition	pixel	shape
05	03	markov	inference	sequence	likelihood	bayesian
22	07	unit	training	layer	train	hide

deep learning. From that, we evaluate how the models of the *past* behave with the *present*.

5.2 Document Classification

This section will cover the results behind the classification model. First, the training process and model selection with the *Past* set, then, the evaluation of those models at *Present* according to the combination of topics.

5.2.1 Model Selection

We now have the *Labeled Past* dataset. From that, we can train a classification model for each topic. For this was performed cross-validation with 10 folds for each combination of vectorization + model, the raw results for those models are in Appendix C.

To evaluate which model is the best, we will compare the average of the model scores for all topics. Table 5.4 shows the comparison between the precision, recall, and f1 for the models. As marked in the table, the best model is the TF-IDF with SVM.

TABLE 5.4 – Classification models comparison.

Type	**Statistics**		
	Precision	**Recall**	**F1 Score**
BOW + Naive Bayes	0.656	0.781	0.695
TF-IDF + Naive Bayes	0.719	0.483	0.553
BOW + SVM	0.863	0.539	0.648
TF-IDF + SVM	**0.906**	**0.683**	**0.769**

Following these results, we identified that we have a bias for higher precision than recall. These results show us the models tend not to predict more positive classes than the existent ones, on the other hand, the smaller recall indicates the models are mispredicting the documents that really belongs to that class. Tables C.1 to C.4, from Appendix C, also show the more imbalanced the class, the predictions tend to be less accurate.

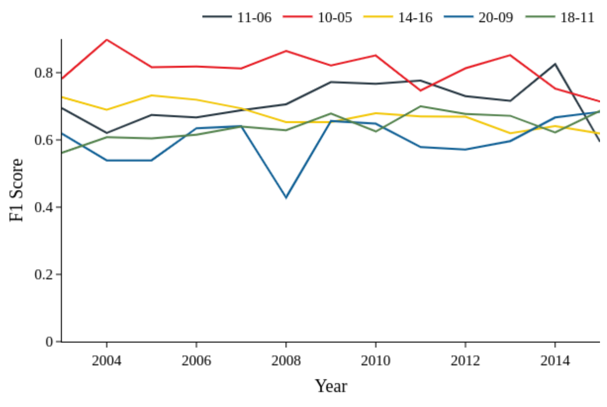
5.2.2 Evaluating Present Predictions

With the built classifiers from the *Labeled Past* set, we apply it in the *present* set and compare the output with its labels. Using just the top-10 combinations in Table 5.2 we have obtained the average results for this predictions shown in Table 5.5 below.

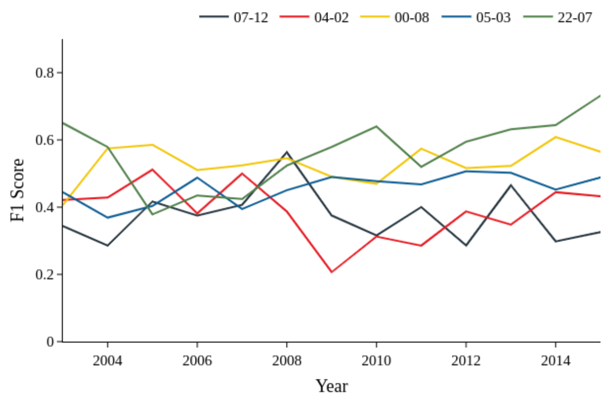
TABLE 5.5 – Metrics for the top-10 topic combinations.

Past	**Present**	**Precision**	**Recall**	**F1 Score**
11	06	0.976	0.563	0.714
10	05	0.936	0.709	0.807
14	16	0.640	0.697	0.667
20	09	0.699	0.547	0.614
18	11	0.552	0.795	0.651
07	12	0.790	0.249	0.379
04	02	0.454	0.355	0.398
00	08	0.967	0.368	0.533
05	03	0.324	0.820	0.464
22	07	0.680	0.550	0.608

In addition to this comparison, we want to evaluate the behavior of the models over the years. For this, we will evaluate the f1 score for each year isolated. In Figure 5.8 is presented the charts for them over the years, Figure 5.8a shows the first to fifth, and Figure 5.8b shows the remaining combinations.



(a) Combinations 1 to 5.



(b) Combinations 6 to 10.

FIGURE 5.8 – F1 Score for the main combinations.

From Table 5.5, we see a high negative correlation between the position of combinations and their f1 scores, -0.775 for the record. This result said to us that is a decreasing relationship between the predictions of these combinations. Figure 5.9 shows a scatter plot for this relationship with an ordinary least squares (OLS) trend line.

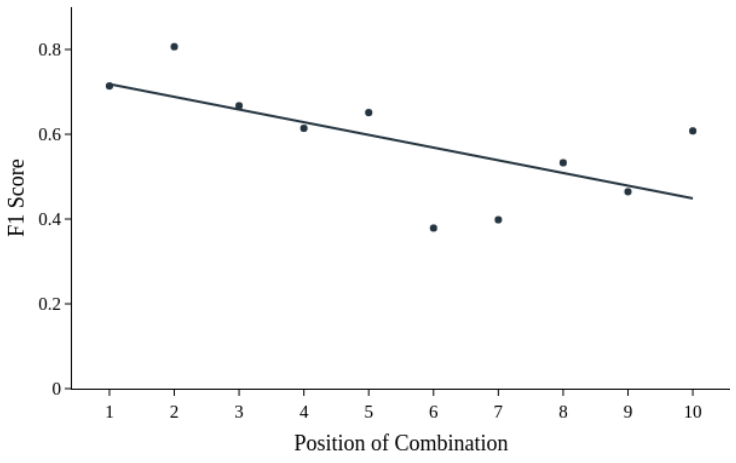
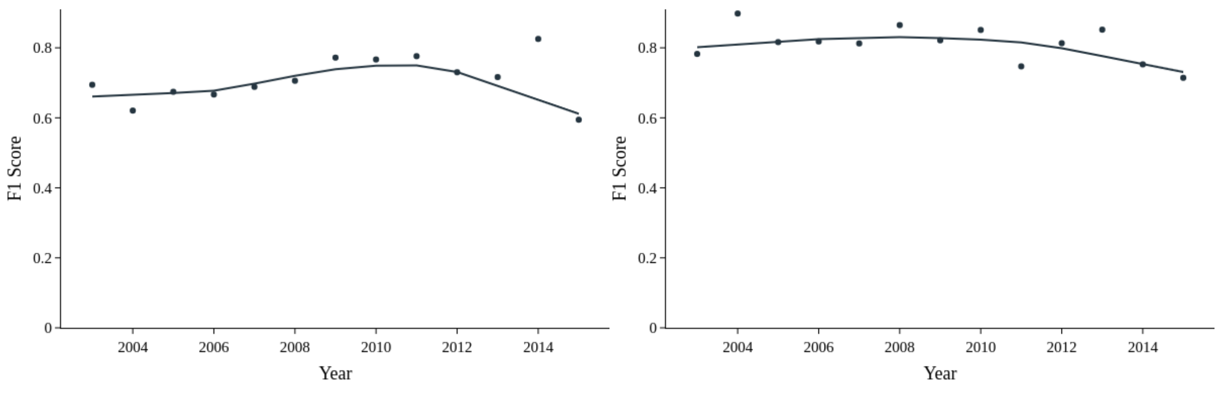


FIGURE 5.9 – F1 Score correlation with the combination position.

This decreasing trend reflects directly in score curves in Figure 5.8. Despite slight fluctuations, the decreasing trend in combinations scores could indicate that even in the best combinations the predictions are not so accurate. The terms evolution over the years to develop the topics maybe have changed and the models can not learn these new terms.

Let us observe a little deeper at the relationship between the score and the time evolution. In Figure 5.10 we have a scatter plot with the local regression trendline of LOWESS type for the two best combinations. Notice that despite a small increase, there is a natural tendency of constancy followed by a decrease in the models' efficiency over the years. Again, this inclination suggests that the rising of new terms for new techniques affects the classification.



(a) Topics combination *Past 11 - Present 06.* (b) Topics combination *Past 10 - Present 05.*

FIGURE 5.10 – F1 Score for the main combinations.

Realize that combinations under the top 5 may have some growth trend. However, in general, these combinations already have low efficiency in relation to the best ones, decreasing the result's reliability.

6 Conclusions, Recommendations, and Future Works

In this last chapter, we wrap up this work sharing conclusions. We also suggest new research lines inspired by the results of this work.

6.1 Conclusions

The main goal of this work was to evaluate the ability of classification models built from the topic modeling data, obtained by Latent Dirichlet Allocation. With a subset of documents, initial on time, called by *Past*, we obtained their natural topics, then we do the same for intermediate data, called by *Present*. After discovering the best combinations for these topics, we built classification models for each one of them and evaluate the predictions over the years.

In chapter 1, we show a brief introduction, presenting the motivation and objective of this work. In chapter 2 we described some important concepts of Natural Processing Language and classification techniques. In chapter 3 we presented related works that implement steps used in this work.

Chapter 4 described the whole methodology used to conduct the experimentation in this work, describing carefully each individual step. Finally, 5 presented and discussed the results obtained in each step of the experiment.

In synthesis, we saw from the results that are really possible to classify documents in short term, from a little subset of data. However, the topics evolve as time goes forward, evidenced by the discrepancy between the main closest combinations and those that follow right away.

We identified two aspects of these classification models. Firstly, we have a tendency of constancy in the efficiency of the model, because of the slow evolution of a given subject, mostly when dealing with scientific publications, as in our case. The second aspect shows a decreasing trend over time, as the model tends to unlearn about themes as new terms are used, such as neural networks, for example, recently the use of deep learning for these kinds of applications has become very popular.

Given this decreasing aspect, our models gradually lose their ability to predict. Therefore, a re-train routine could be fundamental to always keep achieving good results.

6.2 Future Work

From now, we propose other research lines some improvements directly by the results previously obtained:

- *Forecast Step:*

  The immediate continuation given the chronological order is to develop the forecast step. Following the flowchart shown in Figure 6.1, first, using the classifier we can build the incidence topic incidence matrix over time, with *Past* and *Present* sets. Then, apply a forecaster process for those time series with a Long Short-Term Memory neural network (HOCHREITER; SCHMIDHUBER, 1997). Finally, with the *Future* set, perform an evaluation for the built forecaster.

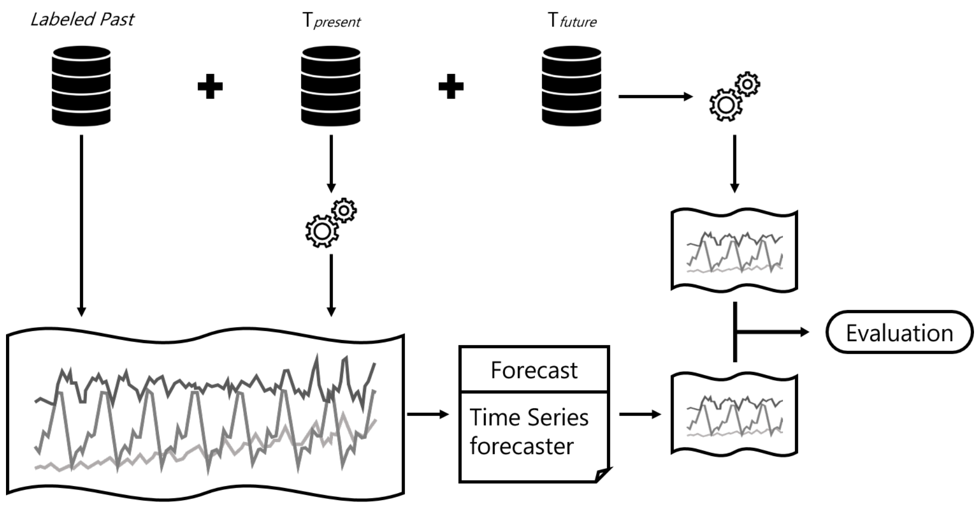


FIGURE 6.1 – Flowchart to evaluate the time series model.

- *Retraining point:*

  Seeking to use this forecasting framework in a production environment, we will need to reduce the time granularity to month or even days. But, as we saw in Figure 5.10 the classification models are losing efficiency over time. This research line aims to propose and evaluate metrics to identify an optimal threshold for retraining the topic model in order to update them with the new terms.

- *Feature extraction improvement:*

  In section 2.1.1 we saw many techniques for normalization and vectorizing, but it doesn't stop there. Several unexplored techniques can improve our topic models and classification such as n-grams, hashing vectorizer and polynomial features, the word embedding itself presented a bad result due to the lack of a pre-trained model based on scientific publication context. It is even possible to explore other metrics to better combine the past and present with topics such as cosine similarity. This research line aims to use more sophisticated techniques for improving the topic and classification models.

Bibliography

BEATH, C.; BECERRA-FERNANDEZ, I.; ROSS, J.; SHORT, J. Finding value in the information explosion. **MIT Sloan Management Review**, Massachusetts Institute of Technology, Cambridge, MA, v. 53, n. 4, p. 18, 2012.

BLEI, D. M. Probabilistic topic models. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 4, p. 77–84, 2012.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. **Journal of machine Learning research**, v. 3, n. Jan, p. 993–1022, 2003.

BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, v. 5, p. 135–146, 2017. ISSN 2307-387X.

CUMMINS, R.; RIORDAN, C. O. Evolving local and global weighting schemes in information retrieval. **Information Retrieval**, v. 9, p. 311–330, 06 2006.

FALEIROS, T. d. P.; LOPES, A. d. A. *et al.* Modelos probabilísticos de tópicos: desvendando o latent dirichlet allocation. São Carlos, SP, Brasil., 2016.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HURTADO, J. L.; AGARWAL, A.; ZHU, X. Topic discovery and future trend forecasting for texts. **Journal of Big Data**, Springer, v. 3, n. 1, p. 7, 12 2016.

JELODAR, H.; WANG, Y.; ORJI, R.; HUANG, H. Deep sentiment classification and topic discovery on novel coronavirus or covid-19 online discussions: Nlp using lstm recurrent neural network approach. **arXiv preprint arXiv:2004.11695**, 2020. Disponível em: <https://arxiv.org/abs/2004.11695>.

KANNAN, S.; GURUSAMY, V. Preprocessing techniques for text mining. **International Journal of Computer Science & Communication Networks**, v. 5, n. 1, p. 7–16, 2014.

KHURANA, D.; KOLI, A.; KHATTER, K.; SINGH, S. Natural language processing: State of the art, current trends and challenges. 08 2017.

KRAYEWSKI, K. **The Current State of NLP, Voice Search, and Voice Assistants**. nov 2019. Disponível em: <https://www.ultimate.ai/blog/ai-automation/the-current-state-of-nlp-voice-search-voice-assistants>.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

NEWMAN, D.; LAU, J. H.; GRIESER, K.; BALDWIN, T. Automatic evaluation of topic coherence. In: **Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics**. [S.l.: s.n.], 2010. p. 100–108.

NIPS. **Neural Information Processing Systems Foundation Inc**. 2020. Accessed: 2020-10-13. Disponível em: <https://papers.nips.cc/>.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543. Disponível em: <https://www.aclweb.org/anthology/D14-1162>.

ROSSUM, G. V.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

SHEN, Q. Topic discovery and future trend prediction in scholarly networks. In: . Shanghai, China: [s.n.], 2018.

SYED, S.; SPRUIT, M. Full-text or abstract? examining topic coherence scores using latent dirichlet allocation. In: **2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. [S.l.: s.n.], 2017. p. 165–174.

TANG, J.; ZHANG, J.; YAO, L.; LI, J.; ZHANG, L.; SU, Z. Arnetminer: extraction and mining of academic social networks. In: **Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2008. p. 990–998.

VIJAYARANI, S.; ILAMATHI, M. J.; NITHYA, M. Preprocessing techniques for text mining-an overview. **International Journal of Computer Science & Communication Networks**, v. 5, n. 1, p. 7–16, 2015.

Appendix A - Past Topics

Table A.1 contains the top 10 words for the 25 topics obtained from the LDA algorithm in the *Past* dataset.

TABLE A.1 – Top 10 words for the *Past* topics.

Topic 00	**Topic 01**	**Topic 02**	**Topic 03**	**Topic 04**
object	memory	hippocampus	circuit	signal
image	capacity	hippocampal	chip	filter
feature	associative	conditioning	analog	motion
recognition	store	episode	current	noise
view	representation	animal	voltage	frequency
transformation	recall	lesion	neuron	channel
representation	item	replay	vlsi	temporal
pixel	storage	neocortical	gate	velocity
part	retrieval	consolidation	implement	visual
digit	bind	dayan	implementation	video

Topic 05	**Topic 06**	**Topic 07**	**Topic 08**	**Topic 09**
probability	cluster	subject	image	classifier
distribution	language	stimulus	component	classification
state	grammar	auditory	matrix	class
variable	clustering	human	pixel	training
estimate	rule	response	local	feature
markov	parse	trial	dimensional	rate
belief	structure	sound	feature	detection
inference	sentence	user	basis	decision
approximation	symbol	target	representation	train
sample	string	task	natural	classify

Topic 10	**Topic 11**	**Topic 12**	**Topic 13**	**Topic 14**
state	neuron	field	instruction	gaussian
policy	spike	visual	program	mixture
action	cell	cell	processor	estimate
control	response	direction	block	distribution
reward	stimulus	receptive	schedule	sample
reinforcement	synaptic	orientation	execution	density
optimal	rate	spatial	schema	likelihood
agent	activity	position	rollout	data
robot	firing	location	parallel	prior
dynamic	cortical	center	simd	bayesian

Topic 15	**Topic 16**	**Topic 17**	**Topic 18**	**Topic 19**
packet	kernel	motor	bound	face
route	tree	movement	theorem	facial
load	feature	control	margin	image
traffic	label	oscillator	loss	expression
send	class	feedback	bind	recognition
slot	node	phase	class	action
routing	distance	trajectory	proof	eigenface
queue	classification	muscle	machine	road
router	document	axon	generalization	subject
switch	training	velocity	training	sleep

Topic 20	**Topic 21**	**Topic 22**	**Topic 23**	**Topic 24**
gradient	dynamic	unit	game	speech
solution	equation	training	player	word
constraint	state	layer	protein	recognition
matrix	field	train	play	speaker
optimization	attractor	hide	gene	training
convergence	solution	task	equilibrium	feature
update	theory	architecture	nash	state
energy	line	node	strategy	character
equation	neuron	hidden	structure	sequence
minimize	limit	activation	sequence	train

Appendix B - Present Topics

Table B.1 contains the top 10 words for the 25 topics obtained from the LDA algorithm in the *Present* dataset.

TABLE B.1 – Top 10 words for the *Present* topics.

Topic 00	**Topic 01**	**Topic 02**	**Topic 03**	**Topic 04**
word	wasserstein	signal	sampling	rank
topic	orbit	source	chain	item
document	transport	event	markov	user
language	lattice	frequency	inference	query
inference	conical	speech	gibb	ranking
latent	sinkhorn	sensor	posterior	permutation
sentence	simplicial	speaker	sampler	preference
representation	cone	noise	mcmc	score
text	cuturi	audio	particle	worker
dirichlet	barycenter	channel	carlo	rating

Topic 05	**Topic 06**	**Topic 07**	**Topic 08**	**Topic 09**
policy	neuron	network	image	gradient
action	spike	layer	object	update
reward	network	deep	pixel	iteration
agent	cell	training	detection	convergence
game	stimulus	train	segmentation	stochastic
control	activity	unit	recognition	constraint
trajectory	population	output	patch	objective
player	response	convolutional	visual	convex
dynamic	dynamic	architecture	scene	descent
reinforcement	synaptic	representation	face	message

Topic 10	**Topic 11**	**Topic 12**	**Topic 13**	**Topic 14**
norm	bound	human	cluster	kernel
convex	regret	response	clustering	rank
regression	bind	motion	manifold	sparse
regularization	theorem	trial	embedding	tensor
sparse	loss	stimulus	local	theorem
lasso	online	subject	spectral	column
sparsity	lemma	decision	distance	norm
group	proof	target	dimension	subspace
selection	bandit	prediction	dataset	noise
penalty	setting	visual	similarity	component

Topic 15	**Topic 16**	**Topic 17**	**Topic 18**	**Topic 19**
causal	gaussian	node	movement	gene
influence	prior	graph	motor	sequence
cascade	latent	tree	classi	protein
intervention	posterior	edge	subject	alignment
discovery	bayesian	network	cation	expression
teaching	likelihood	graphs	brain	site
identifiability	inference	vertex	hand	network
effect	variational	submodular	trial	individual
independence	covariance	graphical	signal	prediction
diffusion	noise	partition	control	cell

Topic 20	**Topic 21**	**Topic 22**	**Topic 23**	**Topic 24**
brain	distance	contour	estimator	label
region	metric	illusory	density	classification
subject	domain	saund	estimation	training
patient	neighbor	ridge	statistic	classifier
functional	similarity	contours	gaussian	loss
voxel	near	accidental	mixture	margin
fmri	hash	palo	entropy	prediction
disease	target	alto	family	dataset
group	search	maximality	variance	instance
connectivity	dataset	permissible	likelihood	accuracy

Appendix C - Raw Classification Results

C.1 Naive Bayes with Bag of Words

Cross-validation results for multinomial naive bayes from sklearn, hyperparamter: $\alpha = 0.1$, with bag of words.

TABLE C.1 – Naive Bayes + BoW.

Topic	**Precision Train**	**Precision Test**	**Recall Train**	**Recall Test**	**F1 Train**	**F1 Test**
00	0.769	0.676	0.965	0.779	0.856	0.721
01	0.741	0.687	0.984	0.687	0.846	0.654
02	0.737	0.658	1.000	0.783	0.848	0.710
03	0.792	0.706	0.987	0.872	0.879	0.774
04	0.644	0.542	0.929	0.804	0.761	0.646
05	0.775	0.723	0.914	0.786	0.839	0.727
06	0.725	0.658	0.913	0.760	0.808	0.690
07	0.671	0.543	0.977	0.801	0.795	0.641
08	0.792	0.717	0.941	0.790	0.860	0.746
09	0.606	0.534	0.968	0.793	0.745	0.627
10	0.947	0.895	0.927	0.763	0.937	0.822
11	0.820	0.780	0.915	0.874	0.865	0.822
12	0.662	0.592	0.915	0.794	0.768	0.674
13	0.731	0.713	0.964	0.733	0.831	0.677
14	0.717	0.679	0.916	0.828	0.804	0.732
15	0.769	0.533	1.000	0.650	0.869	0.550
16	0.736	0.644	0.905	0.810	0.811	0.706
17	0.504	0.459	0.968	0.886	0.662	0.599
18	0.719	0.667	0.916	0.835	0.806	0.728
19	0.649	0.550	1.000	0.720	0.787	0.601
20	0.651	0.572	0.946	0.789	0.772	0.651
21	0.795	0.681	0.957	0.762	0.868	0.715
22	0.938	0.859	0.918	0.805	0.928	0.819
23	0.769	0.606	1.000	0.658	0.868	0.605
24	0.830	0.735	0.971	0.777	0.895	0.749

C.2 Naive Bayes with TF-IDF

Cross-validation results for multinomial naive bayes from sklearn, hyperparamter: $\alpha = 0.1$, with TF-IDF.

TABLE C.2 – Naive Bayes + TF-IDF.

Topic	Precision Train	Precision Test	Recall Train	Recall Test	F1 Train	F1 Test
00	0.926	0.873	0.749	0.530	0.828	0.652
01	1.000	0.927	0.454	0.180	0.624	0.293
02	1.000	0.000	0.226	0.000	0.361	0.000
03	0.954	0.889	0.840	0.704	0.893	0.781
04	0.885	0.720	0.741	0.539	0.807	0.614
05	0.878	0.798	0.835	0.699	0.856	0.715
06	0.980	0.955	0.610	0.421	0.752	0.578
07	0.975	0.803	0.703	0.407	0.817	0.532
08	0.923	0.838	0.776	0.612	0.843	0.700
09	0.954	0.864	0.726	0.457	0.825	0.586
10	0.989	0.960	0.776	0.653	0.869	0.776
11	0.869	0.830	0.871	0.825	0.869	0.826
12	0.827	0.717	0.772	0.657	0.798	0.678
13	1.000	0.100	0.238	0.033	0.381	0.050
14	0.818	0.758	0.843	0.747	0.830	0.735
15	0.000	0.000	0.000	0.000	0.000	0.000
16	0.887	0.779	0.828	0.703	0.856	0.724
17	0.915	0.827	0.802	0.620	0.855	0.687
18	0.919	0.843	0.839	0.718	0.877	0.758
19	1.000	0.500	0.255	0.110	0.406	0.180
20	0.911	0.822	0.752	0.533	0.824	0.633
21	0.970	0.903	0.752	0.517	0.847	0.654
22	0.943	0.864	0.886	0.761	0.913	0.795
23	1.000	0.500	0.351	0.125	0.517	0.200
24	0.982	0.922	0.739	0.550	0.843	0.682

C.3 Support Vector Machine with Bag of Words

Cross-validation results for support vector classifier from sklearn, hyperparamter: $C = 1$, kernel = linear and gamma = auto, with bag of words.

TABLE C.3 – Support Vector Machine + BoW.

Topic	**Precision Train**	**Precision Test**	**Recall Train**	**Recall Test**	**F1 Train**	**F1 Test**
00	1.000	0.928	0.946	0.611	0.972	0.731
01	1.000	1.000	0.925	0.390	0.961	0.544
02	1.000	0.100	0.850	0.033	0.919	0.050
03	1.000	0.982	0.930	0.633	0.964	0.768
04	0.997	0.943	0.940	0.576	0.967	0.710
05	0.996	0.901	0.967	0.742	0.981	0.805
06	1.000	0.992	0.872	0.518	0.932	0.677
07	1.000	0.982	0.941	0.468	0.969	0.622
08	0.998	0.901	0.957	0.671	0.977	0.767
09	1.000	0.921	0.930	0.576	0.964	0.701
10	1.000	0.962	0.942	0.630	0.970	0.758
11	0.994	0.935	0.969	0.789	0.981	0.853
12	0.997	0.940	0.956	0.632	0.976	0.754
13	1.000	0.700	0.889	0.267	0.941	0.380
14	0.995	0.912	0.972	0.725	0.983	0.801
15	1.000	0.200	0.828	0.200	0.905	0.200
16	0.996	0.896	0.946	0.658	0.970	0.746
17	1.000	0.990	0.924	0.457	0.960	0.617
18	0.998	0.936	0.958	0.646	0.978	0.757
19	1.000	0.900	0.915	0.425	0.956	0.564
20	0.999	0.902	0.961	0.631	0.979	0.739
21	0.999	0.955	0.907	0.559	0.951	0.704
22	0.993	0.922	0.978	0.795	0.986	0.847
23	1.000	0.800	0.974	0.258	0.987	0.383
24	1.000	0.985	0.955	0.574	0.977	0.722

C.4 Support Vector Machine with TF-IDF

Cross-validation results for support vector classifier from sklearn, hyperparamter: $C = 1$, kernel = linear and gamma = auto with TF-IDF.

TABLE C.4 – Support Vector Machine + TF-IDF.

Topic	**Precision Train**	**Precision Test**	**Recall Train**	**Recall Test**	**F1 Train**	**F1 Test**
00	0.994	0.902	0.952	0.684	0.972	0.772
01	1.000	0.960	0.984	0.594	0.992	0.714
02	1.000	0.600	1.000	0.467	1.000	0.510
03	1.000	0.950	0.982	0.736	0.991	0.827
04	0.994	0.885	0.981	0.683	0.988	0.768
05	0.987	0.898	0.967	0.771	0.977	0.822
06	1.000	0.978	0.980	0.696	0.990	0.808
07	1.000	0.947	0.949	0.615	0.974	0.740
08	0.993	0.888	0.958	0.729	0.975	0.799
09	1.000	0.922	0.967	0.664	0.983	0.767
10	0.997	0.937	0.966	0.708	0.981	0.805
11	0.994	0.926	0.980	0.827	0.987	0.870
12	0.999	0.920	0.975	0.708	0.987	0.799
13	1.000	1.000	0.964	0.567	0.982	0.703
14	0.992	0.885	0.958	0.739	0.975	0.801
15	1.000	0.700	0.839	0.650	0.912	0.667
16	0.989	0.868	0.954	0.728	0.972	0.785
17	1.000	0.948	0.979	0.664	0.989	0.773
18	0.996	0.926	0.957	0.733	0.976	0.814
19	1.000	0.950	0.960	0.570	0.979	0.706
20	0.995	0.883	0.956	0.704	0.975	0.781
21	1.000	0.926	0.953	0.650	0.976	0.762
22	0.991	0.903	0.969	0.813	0.980	0.850
23	1.000	0.967	0.997	0.692	0.999	0.782
24	1.000	0.988	0.962	0.689	0.981	0.809

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO TC	2. DATA July 7th, 2020	3. DOCUMENTO N° DCTA/ITA/DM-018/2015	4. N° DE PÁGINAS 62

5. TÍTULO E SUBTÍTULO:
Natural Language Processing for Trend Forecasting

6. AUTOR(ES):
Heládio Sampaio Lopes

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):
Aeronautics Institute of Technology – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:
Natural Language Processing; Topic Modeling; Machine Learning

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:
Natural Language Processing; Topic Modeling; Machine Learning

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**
ITA, São José dos Campos, 2020. Trabalho de Graduação. 62 páginas.

11. RESUMO:

Com avanço cientifico-tecnológico em escalas globais, a disponibilidade de dados para serem analisados cresce de maneira exponencial. A partir desse crescimento acelerado, técnicas de inteligência artificial, como o Processamento de Linguagem Natural (PLN), ajudam a automatizar o processo de se extrair informação de documentos. Com intuito de realizar previsões sobre as tendências que surgirão em meio ao crescimento de dados, é possível utilizar técnicas de modelagem de tópicos para agrupar documentos em temas semelhantes e estudar a incidência temporal desses temas.
Assim, este trabalho busca aplicar técnicas de PLN combinadas com aprendizado supervisionado para agrupar documentos em um conjunto de temas e a partir disso aprender a identificá-los em novos documentos em um período futuro. Utilizando publicações cientificas da conferência de Sistemas de Processamento de Informação Neural, fomos capazes de modelar essa tarefa e obter bons resultados. Também estudamos a capacidade dos classificadores de manter desempenho satisfatório com o passar do tempo.

12. GRAU DE SIGILO:
(X) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**