Statistical models for natural language processing and Topic Modelling in R

Written by

Efthimios Ioannis Kavour

(p3622114)

Supervised by

Dr Panagiotis Papastamoulis

A thesis composed and presented for the degree of

Master of Applied Statistics

Athens University

of Business and Economics

Greece

19/02/2024

Acknowledgements

I would like to take a moment to personally thank all the professors of the Master's program I attended at the Athens University of Business and Economics. Each one of you taught me something unique and exceeded my highest expectations. I am especially grateful to my supervisor, Dr. Panagiotis Papastamoulis. His door was always open whenever I encountered difficulties or needed guidance. Though writing a thesis is primarily an individual task, he consistently steered me in the right direction, fostering my academic growth. Thank you, Professor Papastamoulis, for this invaluable experience and for doing me the honor of your mentorship.

On a personal note, I would also like to thank my friends, both within and outside the university, for the passionate conversations and the fun we shared. I am deeply grateful to my family-Sofia, Stelios, Evangelia, and Christos—for their unwavering support in every decision that led me to complete this Master's thesis. Last but certainly not least, I want to express my heartfelt thanks to Theodosia, who was always there when I needed her. Thank you for your patience and support as I pursued a second master's degree and for engaging in discussions about the new concepts I learned, even though they were outside your field of interest.

Contents

1 Introduction

Topic modeling is a powerful technique within the domain of machine learning and more specifically natural language processing (NLP) that has as main goal to uncover the underlying themes or subjects found within a set of documents. In this thesis, we are going to explore the application of topic modeling, specifically Latent Dirichlet Allocation (LDA) in the context of genre clustering for a diverse range of books. Furthermore we will provide a brief introduction to Latent Semantic Analysis (LSA) so that the reader can clearly see the difference of how LDA works and how topic modeling is established aside from the fact that the model does not consider the meaning of the words.

The application of topic modeling holds significant promise across various domains, including but not limited to information retrieval, recommendation systems, and content analysis. By automatically organizing documents into coherent topics, topic modeling techniques offer a systematic approach to handling large volumes of unstructured text data. Furthermore, the insights derived from topic modeling can inform decision-making processes in fields such as marketing, social sciences, and healthcare.

In this thesis, we delve into the practical implementation of topic modeling, focusing on its application to genre clustering of books. Genres serve as a fundamental organizational structure for textual content, providing a means to categorize anything that can be categorized as in our case books. By employing topic modeling techniques, we aim to create clusters representing genre when the information is missed and we can not open and categorize the documents one by one. We are going to use books' descriptions in order to successfully reach our goal. A more accurate way would be to use the actual text of each book, but this is a hard resource to find.

The structure of this thesis is the following. In Section 2 we will provide to you an overview Machine Learning in general. We will examine the different kinds of learning more closely, and give some examples per category.Then in Section 3 it is going to be presented, basic word statistics manipulation and how to build matrices that can help us answer simple data exploratory questions. Moving forward in Section 3 we will present a brief introduction on Sentiment Analysis and different ways to enhance our calculations in order to acquire information toward the sentiment of a textual data obtained. The reason for this inclusion is because we believe that may interests the reader in the following way. Initially, LDA or LSA could be used to create clusters upon reviews of other textual data and then the one may be interested in using sentiment analysis in order to further research what are the sentiments the textual data contain per cluster created. For example having a set of reviews, it would be interesting to create clusters upon the topics or features contained in the review (e.g. applications, features, bugs etc) and then extract the sentiments upon the category that one could be interested. Moving forward in Section 4 we will get into Topic Modeling, the main part of this thesis. This is divided into two subcategories, the first is about Latent Dirichlet Allocation topic modeling method in which we are interested and how we can set it to usage. The second subcategory of this chapter is an introduction to LSA method, where we aim to create clusters based on the semantics. Finally, Section 6 contains applications of the topics discussed in this analysis. Initially, we present the way of collecting the data used for our applications. Then, we present a detailed application on the topic of LDA where we try to create clusters of book genre. Apart from that we continue with an application of LSA on the same data so that we can later on compare the results of both methods. Finally, section 7 includes useful material for better understanding of the processes.

Please note: The scripts developed during this thesis, along with the data, the thesis itself, and the presentation, have all been uploaded for further analysis and review at

$$https : //github.com/KavourEI/AUEB - Thesis$$

2 Machine Learning and Natural Language Processing

Machine learning (ML) focuses on creating algorithms and statistical models that allow computers to carry out tasks without specific instructions. Instead, these algorithms learn from data, iteratively improving their performance over time. Lately due to extreme computational evolution, the improvements in algorithmic techniques and the storing and gathering of huge amount of data ML field has went through a rapid growth. In other words, some would say that Machine learning is a developing field of computational algorithms aimed at mimicking human intelligence through learning from the surrounding environment.[12]

Machine learning lies at the intersection of statistics, computer science, engineering and often other disciplines. Primarily, ML uses statistics. The main goal of machine learning is to create algorithms that can generalize patterns from data and use them to make predictions or decisions when presented with new, unseen data. This ability to generalize is what distinguishes ML from rule-based programming and other traditional human attention methods. This is adaptability and the ability to work on different domains is what makes ML algorithms so popular nowadays.

Machine learning can be categorized into three main types:

- Supervised learning ml methods
- Unsupervised learning ml methods
- Reinforcement learning ml methods.

2.1 Supervised Learning

At its core, supervised learning involves learning a mapping from input features to output labels based on examples provided in the training data by the user/researcher in need. The training dataset consists of input-output pairs, where the first are feature vectors representing the characteristics of data, and the latter are corresponding labels or target values.

Some of the types of Supervised learning tasks can be categorized into two main tasks: classification and regression.

Classification tasks involve predicting discrete class labels for input features as discussed above. The goal is to learn a decision boundary that separates different classes from the same pool of data. Common classification algorithms include *logistic regression*, *decision trees*, *random forests*, *support vector machines (SVM)*, and *neural networks*. Some examples of classification, just to name a few to have in mind, are email spam detection, sentiment analysis, medical diagnosis and image recognition.

Regression tasks, on the other hand, involve predicting continuous numeric values. The objective is to learn a function that maps input features to a continuous output space. *Linear regression, polynomial regression, decision trees, support vector regression (SVR), and neural networks* are popular regression techniques. Regression finds applications in areas such as stock price prediction, house price estimation, and demand forecasting.

Without getting to details, we find it useful to provide a small description of the methods mentioned earlier:

- **Linear models** are a class of algorithms that assume a linear relationship between the input features and the target variable. Some examples include linear regression for regression tasks and logistic regression for classification tasks. Though those models are characterized as simple, linear models are often effective and computationally efficient, making them popular choices for many supervised learning problems. After all the most beautiful solutions are the simplest ones.

- **Decision trees** are hierarchical tree-like structures used for both classification and regression tasks. They partition the feature space into regions based on feature values, making decisions at each node to predict the target variable. Decision trees are interpretable and can handle both numerical and categorical data. However, they are prone to overfitting, which can be mitigated using techniques like pruning and ensemble methods such as random forests and gradient boosting.

- **Support Vector Machine** is a powerful supervised learning algorithm for both classification and regression tasks. It works by finding the optimal hyperplane that separates different classes in the feature space (classes that the user/researcher defined as we are in the supervised case) while maximizing the margin between classes. SVMs are effective in high-dimensional spaces and are robust to overfitting. They can also handle nonlinear relationships through the use of kernel functions.

- **Neural networks**, particularly deep learning models, have gained prominence in recent years due to their ability to learn complex patterns from data, and once again the high speed of technological evolution. Deep neural networks consist of multiple layers of interconnected neurons, enabling them to automatically extract hierarchical features from raw data. Convolutional neural networks (CNNs) are widely used for image classification and object detection, while recurrent neural networks (RNNs) are popular for sequential data tasks such as natural language processing and time series prediction.

As we have mentioned earlier, Machine Learning is primarily evolved statistical models. As a result, evaluation is process that can not be omitted and of course the results can not be taken as absolute truth without checking first. To asses the performance of a model and ensure its generalization to unseen data, the definition of evaluation and validation methods are a crucial step. Common evaluation metrics for classification tasks include *accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve*. For regression tasks, metrics such as *mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared* are commonly used. Apart from evaluation metrics common techniques such as *cross-validation, holdout validation, and bootstrapping* are employed to estimate the model's performance on unseen data and detect potential issues such as overfitting.

2.2 Unsupervised Learning

On the other hand, we have Unsupervised learning which is a branch of machine learning where the algorithm learns patterns and structures from unlabeled data. So unlike supervised learning, the user has to provide the data but not paired with a label and there are no explicit target variables during the training process. Instead, the algorithm aims to discover data structures, relationships, or groupings within the data by itself. The user at most can prepare the data as good as possible.

The primary goal of unsupervised learning is to explore the underlying structure of the data without the guidance of labeled examples. Unsupervised learning tasks typically involve clustering, dimensionality reduction, and density estimation.

At this point we could not skip providing a summary for each of the types of Unsupervised Learning Tasks:

- **Clustering** is a common unsupervised learning task that involves partitioning data into groups, or clusters, such that data points within the same cluster are more similar to each other than to data points appearing in other clusters generated. Popular clustering algorithms include K-means clustering, hierarchical clustering, and density-based clustering methods like DBSCAN. Clustering finds applications in customer segmentation, document categorization, and anomaly detection.

  – **K-means** clustering is a popular algorithm for partitioning data into K clusters based on similarity measures such as Euclidean distance. The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids until convergence. K-means clustering is widely used for data segmentation and pattern recognition tasks.

- **Dimensionality reduction** techniques aim to reduce the number of features or variables in the dataset while preserving its essential characteristics. Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are common dimensionality reduction methods. By reducing the dimensionality of data, dimensionality reduction facilitates visualization, feature selection, and computational efficiency in subsequent analysis tasks.

  – **Principal Component Analysis (PCA)** is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional representation while preserving the variance in the data. It achieves this by identifying orthogonal axes, called principal components, along which the data exhibits the maximum variance. PCA is commonly used for feature extraction, data visualization, and noise reduction.

- **Density estimation** techniques model the probability distribution of the data in the feature space. Kernel density estimation (KDE), Gaussian mixture models (GMM), and self-organizing maps (SOM) are examples of density estimation methods. These techniques are useful for understanding data distributions, identifying outliers, and generating synthetic data.
  - **Gaussian Mixture Models (GMM)** is a probabilistic model that represents the data as a mixture of multiple Gaussian distributions. It learns the parameters of these distributions, including the means and covariances, using the Expectation-Maximization (EM) algorithm. GMMs are versatile models that can capture complex data distributions and are often used in clustering and density estimation tasks.

Clearly defining evaluation and validation methods for the results for unsupervised learning algorithms can be challenging due to the absence of ground truth labels. Instead, qualitative assessments such as *visual inspection, silhouette scores, Davies-Bouldin index, and gap statistics* are commonly used to evaluate clustering algorithms. For dimensionality reduction techniques, *visualization and reconstruction error metrics* are often employed to assess the quality of the reduced representation.

2.3 Reinforcement Learning

Finally, reinforcement learning (RL) is a branch of machine learning concerned with how agents ought to take actions in an environment to maximize some notion of cumulative reward. Unlike supervised learning, where the algorithm is trained on labeled data, and unsupervised learning, where the algorithm learns patterns without explicit guidance, reinforcement learning is about learning to make decisions sequentially through trial and error, guided by feedback from the environment.

In reinforcement learning, an agent (built by the user/researcher) interacts with an environment by taking actions and receiving feedback in the form of rewards or penalties. The agent's objective is to learn a rule (or rules) hidden within the data that maximizes the cumulative reward over time. Reinforcement learning problems are typically modeled as Markov Decision Processes (MDPs), where the agent observes the current state of the environment and selects actions based on this information.

Reinforcement learning involves three main components:

- An Agent which is the learner or decision-maker that interacts with the environment.
- An Environment which is the external system with which the agent interacts and receives feedback.
- A Reward Signal which is a scalar feedback signal received by the agent after each action, indicating the immediate desirability or utility of the action taken.

Reinforcement learning tasks as well as the rest of the learning methods, can be categorized into certain types. Here we have two main types:

- **Episodic Tasks**: Tasks where the agent interacts with the environment for a finite number of time steps, with each interaction leading to the termination of an episode. Examples include games with a fixed number of moves or tasks with a specific goal to achieve.
- **Continuous Tasks**: Tasks where the agent interacts with the environment indefinitely, with no predetermined termination. Examples include continuous control problems like robot locomotion or autonomous driving.

2.4 Applications of Machine Learning

This analysis is around a certain application of Machine learning. But before diving into it, let us take a step back and provide some wider examples of ML applications. Machine learning has found applications across a wide range of fields, revolutionizing industries and driving innovation. Some of them are:

- Healthcare: ML algorithms are used for disease diagnosis, personalized treatment plans, drug discovery, and medical imaging analysis.

- Finance: ML techniques are applied in algorithmic trading, credit scoring, fraud detection, and risk management.
- Marketing and Advertising: ML algorithms power recommendation systems, targeted advertising, customer segmentation, and sentiment analysis.
- Autonomous Vehicles: ML plays a crucial role in enabling self-driving cars to perceive their environment, make decisions, and navigate safely.
- Natural Language Processing (NLP): ML models are used for language translation, sentiment analysis, speech recognition, and text generation.

While machine learning offers tremendous potential, it also presents several challenges, after all the saying goes like ”nothing good comes without a price”. These challenges include issues related to data quality, interpretability, fairness, privacy, and scalability. Additionally, the black-box nature of some ML algorithms raises concerns about their trustworthiness and reliability, especially in critical applications such as healthcare and finance.

In summary, machine learning has emerged as a powerful tool for solving complex problems and extracting insights from data. With ongoing advancements in algorithms, hardware, and data availability, the potential applications of ML continue to expand, promising to reshape industries and transform the way we interact with technology.

2.5 Natural Language Processing

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence and Machine Learning with main purposes being the understanding, interpretation and generation of human language in a way that is meaningful and of course relevant to the context that the case demands. The case of course is not defined by anyone else apart from the user/researcher that has the intentions to use it. Lately as the topics described in the sub-chapters written above has met an exponential growth due to the great gathering of textual data on the internet, the continuous technological upgrades on the field and of course the high demand for automated language processing tasks. After all it is the demand that drives the advancements occur.

Some of the basic tasks that NLP method are challenged are tokenization, the process of segmenting text into individual words, phrases and/or symbols, morphological analysis, syntax analysis (e.g. grammatical structures), semantic analysis (which is the extraction of meaning and understanding what is the relationship of the words in contrast to the given text corpus), entity extraction/recognition (which is identifying and categorizing names, location, dates and even custom list of entities), sentiment analysis (which is the categorization of text chunks based on the sentiments hidden behind the words that the author has chosen to compose the piece of text that is under analysis. Note that the sentiments may be positive, negative neutral or any other custom list of levels the researcher decided appropriate for the occasion)

In order NLP to successfully complete the tasks mentioned earlier, techniques and algorithms are built so that linguistic knowledge, statistical methods, and machine learning approaches are gathered to a field. Rule-based systems are built using linguistic rules to perform tasks such as parsing and information extraction. Different statistical methods use probabilistic models such as Markov Models for language modeling and sequence labeling tasks. Machine learning techniques encompass supervised, unsupervised, and reinforcement learning algorithms for numerous of NLP tasks, ranging from classification and clustering modeling. Deep learning architectures, including Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformer models, wield immense power in tasks such as machine translation, sentiment analysis, and text generation, driven by their capacity to automatically learn hierarchical representations from data.

3 Modelling and Visualizing Word Count Data

Analyzing natural language data, can be quite a demanding process. The challenges begin early on, at the exploration part, where we want to answer simple questions like, what is the distribution[36] that is followed or what are the most and least frequent words in the corpus and many more simple, yet intriguing questions[1]. As a results, in this section we are going to present certain ways to represent linguistic data and how to explore them in order to gain insights upon further research.

Understanding human language is a formidable challenge, and word vectorization emerges as a powerful technique to bridge the gap between the richness of language and computational models. A first part for assisting this process is to disclose the concept of *word vectors*. In this part, words are transformed into numerical vectors[2] in a multi-dimensional space. This process is not a random process, whereas it is a carefully directed and well established process taking under consideration semantic relationships and contextual meanings of words. Words that share similar or somehow similar meaning are located closer with those that don't in the multi-dimensional vector space.

Vectors, or distributional models of meaning, are typically derived from a co-occurrence matrix. We can represent word vectors in many different shapes and sizes, depending the goal we aim to achieve. Namely, we have

1. Term - Document matrix (column wise)
2. Term - Document matrix (row wise)
3. Term - Term matrix

In the first case, where terms and documents co-occur in a matrix, each row represents a word and each column represents a document. The cells filling the matrix present the number of times a particular word occurs in a particular document. This specific way of representation was initially defined as part of the vector space[3]

[1]Those and many more questions help us answer library zipfr[15] an open-source easy to use R-package.
[2]Vectors for representing words are called **embeddings**
[3]A **vector space** is a collection of vectors, characterized by their dimension.

model of information retrieval[4]. The intuition behind this is that two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar. A matrix of this sort has dimensions $|V|$ rows (as each row represents a word in the vocabulary V) and D columns (as each column represents a documents from the collection of D documents).

In the second case instead of creating a vector per column, we decide to create a vector per row. This change has as a result to compare the similarity (or to be more precise the correlation) of words instead of documents. The exact same principle applies here and we can say that similar or correlated words will tend to have similar vectors because they tend to occur in the same context and therefore in the same documents. By this small change we are now able to extract the meaning of each word or at least get an intuition of it by looking at the vectors from the term-document matrix and compare them with similar words appearing close to each other in the vector space.

Finally, we have the term-term matrix. The dimensions of which are $|V| \times |V|$ as its rows and columns present words from the collected vocabulary $|V|$. Depending on the analysis that we want to complete we can fill this particular table with a different information. Some examples are the count of times that both words co-occurred in a document of our corpus, or they both existed in the same paragraph/sentence. The most usual case here is to fill the table by creating a window of n-words (for example 4) and count the number of times, each word existed in the previous or next four words of the targeted one.

Another way of presenting and analyzing linguistic data is by using Term Frequency-Inverse Document Frequency (tf-idf) matrix. Tf-idf can be derived from both Term-Document Matrix and Term-Term matrix. Tf-idf is used for document classification, clustering or relevance ranking as it aims to reflect the importance of terms in a collection of documents. Let us now take a closer look at each part of the matrix separately.

- Term Frequency (tf) is used to get an idea how often a term appears in a docu-

[4]**Information retrieval** is the process of finding the document d from a collection of D documents that best matches to a query q. See appendix for more information.

ment. For term t in a document d, it is calculated as followed:

$$TF(t,d) = \frac{\text{Number of times } t \text{ appears in } d}{\text{Total number of terms in } d}$$

- Inverse document frequency (idf) is used to check how important/unique a term is within the corpus. For a corpus built from D documents, some of which contain term t, idf is calculated as followed:

$$IDF(D,t) = log\frac{\text{Total number of documents } D}{\text{Number of documents containing term } t}$$

When both values are calculated, we can calculate the tf-idf value for a word in a corpus composed by documents D as followed

$$TF-IDF(t,d,D) = TF(t,d) \times IDF(D,t)$$

The tf-idf matrix is constructed by representing each document as a vector where each element corresponds to the tf-idf score of a term in the vocabulary generated by the corresponding corpus. Therefore, the matrix has dimensions $m \times n$, where m is the number of terms in the vocabulary and n is the number of documents.

A high tf-idf score for a term in a particular document indicates that the term is both frequent in that document and relatively rare across all documents in the corpus, suggesting it is important or characteristic to that document.

As tf-idf is a key element for our analysis, it seems wise to provide a simple yet important example to clear any doubt that may has risen. Support we have the following three documnts:

1. The cat sat on the mat.

2. The dog played in the garden.

3. The cat and the dog are friends.

The process is as followed. Initially we tokenize the documents and calculate the term frequency for each term in each of the documents that our corpus is consisted from (as shown in Table 1). Next, we need to calculate the inverse document frequency, as shown above, for each term in the available vocabulary generated from the corpus in hand (as shown in Table 2). Finally, we are ready to calculate the corresponding tf-idf scores for every term per document (as shown in Table 3)

Document 1		Document 2		Document 3	
Vocabulary	Score	Vocabulary	Score	Vocabulary	Score
cat	1/6	dog	1/6	cat	1/7
sat	1/6	played	1/6	and	1/7
mat	1/6	in	1/6	the	2/7
on	1/6	the	1/6	doc	1/7
the	1/6	garden	1/6	are	1/7
-	-	-	-	friends	1/7

Table 1: Tern Frequency for vocabulary of given corpus

Terms:	cat, dog	sat, mat, on, played, in, garden, and, are, friends	the
Score:	$log(3/2) \approx 0.176$	$log(3/1) \approx 0.585$	$log(3/3) \approx 0$

Table 2: Inverse document frequency for each term of the vocabulary

Document 1		Document 2		Document 3	
Vocab	Score	Vocab	Score	Vocab	Score
cat	$1/6 \times 0.176 \approx 0.029$	dog	$1/6 \times 0.176 \approx 0.029$	cat	$1/7 \times 0.176 \approx 0.025$
sat	$1/6 \times 0.585 \approx 0.098$	played	$1/6 \times 0.176 \approx 0.029$	and	$1/7 \times 0.585 \approx 0.083$
mat	$1/6 \times 0.585 \approx 0.098$	in	$1/6 \times 0.585 \approx 0.098$	the	$2/7 \times 0 = 0$
on	$1/6 \times 0.585 \approx 0.098$	the	$1/6 \times 0 = 0$	doc	$1/7 \times 0.176 \approx 0.025$
the	$1/ \times 0 = 0$	garden	$1/6 \times 0.585 \approx 0.098$	are	$1/7 \times 0.585 \approx 0.083$
-	-	-	-	friends	$1/7 \times 0.585 \approx 0.083$

Table 3: Tern Frequency Inverse document frequency score for each term

4 Sentiment Analysis

The rapid expansion of internet services has empowered individuals to freely express their opinions on a wide range of topics[16]. Sentiment analysis, a great technique within the field of Natural Language Processing (NLP), is designed to achieve a singular objective: extracting sentiments from textual content[13]. To delve deeper into this, sentiment analysis is a method of text classification, providing an outcome based on predefined categories. Researchers typically focus on classifying text into two categories (positive or negative sentiment) or three categories (positive, negative, or neutral sentiment).

Textual data come in all shapes and sizes. As a result, sentiment analysis can adapt to the needs of a researcher. The analysis can be computed in many levels, some of which are the following:

1. **Document level** aims to determine the overall sentiment expressed throughout an entire document. A review, an article, a social media post are some of the textual data that can be considered a document. The primary goal is to classify the document as positive, negative, or neutral based on the predominant sentiment conveyed. Document-level sentiment analysis provides a high-level understanding of the overall sentiment polarity, which can be valuable for tasks such as summarization, sentiment classification of reviews, or assessing the general sentiment of a large corpus of text.[5]

2. **Sentence level** sentiment analysis, where we dig deeper in a document, focuses on analyzing the sentiment expressed within individual sentences of a document. Instead of classifying the sentiment of the entire document, each sentence is classified as positive, negative, or neutral based on the sentiment it conveys. This level of analysis is particularly useful for tasks such as sentiment-aware summarization, opinion mining in product reviews, or sentiment-aware chatbots.[5]

[5]Note that for certain cases, we may need to get the sentiments per chunk of text. A chunk may be more than a sentence and less than a document itself. In this case we define the chunks to be a sentence or a document and we move forward with the option we have selected.

3. **Aspect level** in which the research goes beyond documents or sentences. In this case one aims to extract sentiments from textual data (i.e. reviews) on specific features, aspects or entities mentioned in the document. For example the color, functionalities, camera lenses performance, battery life of a mobile phone, a place or location etc). This level of analysis provides fine-grained insights into the sentiment towards specific aspects or entities, facilitating more targeted and actionable insights for decision-making tasks such as product improvement, customer feedback analysis, or reputation management.

There are many and different ways to conduct sentiment analysis. The choice is up to the researcher. Of course each of the methods that are going to be presented here have some advantages and disadvantages, which the researcher should carefully consider before making a choice. The main approaches to conduct sentiment analysis are three, namely:

1. Machine learning (ml) approach
2. Rule based (rb) approach
3. Lexicon based (lb) approach

Conducting sentiment analysis with the help of ml approach, involves using ml models and algorithms to automatically classify textual data. Machine learning approach can be completed using a supervised or unsupervised methods. Some of the most common models to build in case this kind of approach is required are the following :

- Support Vector Machine (SVM).[3]
  - A SVM is a non probabilistic classifier which aims to construct a hyperplane on the space of predictor variables. By using SVM method a hyperplane is created that divided data into two or more classes. An effective division occurs when the hyperplane has the largest distance to the nearest training-data of any class. Support Vector Machines are widely used in sentiment analysis tasks due to their effectiveness in handling

high-dimensional data and ability to learn complex decision boundaries. In sentiment analysis, SVMs are utilized to classify textual data into positive, negative, or neutral sentiment categories based on the features extracted from the text. These features may include word frequencies, n-grams, or word embeddings representing the semantic meaning of the text. SVMs aim to find the hyperplane that best separates the feature space into different sentiment classes, maximizing the margin between the classes while minimizing classification errors. By learning from labeled training data, SVMs can generalize well to unseen text and effectively classify sentiment in new instances.[29]

- N-gram Sentiment Analysis.[32]

  - N-gram sentiment analysis is a technique in natural language processing (NLP) where sentiment analysis is performed based on the occurrence of n-grams in the text. N-grams are contiguous sequences of n items from a given sequence of text, typically words or characters. In this approach, sentiment analysis is conducted by considering the presence and frequency of specific n-grams associated with positive, negative, or neutral sentiment. For example, in unigram sentiment analysis (n=1), individual words such as ”good” or ”bad” are considered as features for sentiment classification. In bigram sentiment analysis (n=2), pairs of adjacent words such as ”very good” or ”not bad” are analyzed for sentiment. And in trigram sentiment analysis (n=3), sequences of three words such as ”not very good” or ”really bad movie” are examined. By training a sentiment classifier on labeled data containing n-grams and their associated sentiment labels, the model can learn to classify text based on the presence and frequency of specific n-grams associated with different sentiments. N-gram sentiment analysis allows for capturing context and nuanced sentiment expressions in text, providing a more granular understanding of sentiment compared to traditional unigram-based approaches. Although this method produces great results, there is a huge need for well structured training data. As you can probably guess, sentiment analysis models that use n-gram method, in many cases are not able to be generalized and used for other context.[1]

- Naive Bayes Analysis.[37]
  - This is a probabilistic classifier. This classifier is build on Bayes theorem, and uses it to extract sentiments from data. The way that Naive Bayes work is as following

$$P(sentiment/sentence) = \frac{P(sentence/sentiment) \cdot P(sentiment)}{P(sentence)}$$

    Naive Bayes is a popular machine learning algorithm commonly used for sentiment analysis in natural language processing (NLP) tasks. In sentiment analysis, Naive Bayes classifiers predict the sentiment of a piece of text (positive, negative, or neutral) based on the occurrence of words or features associated with different sentiments. The ”naive” assumption in Naive Bayes is that features (words) are conditionally independent given the class label (sentiment), which simplifies the calculation of probabilities.

    To perform sentiment analysis with Naive Bayes, the text data is preprocessed by tokenizing the text into words or n-grams and converting it into a numerical representation, such as a Bag-of-Words (BoW) or TF-IDF vector. The occurrence or frequency of each word or n-gram in the text is then used as input features for the Naive Bayes classifier.

    The Naive Bayes classifier calculates the probability of each sentiment label given the input features using Bayes' theorem as presented above. Specifically, it computes the posterior probability of each sentiment label given the input text using the likelihood of the features given the sentiment labels and the prior probability of each sentiment label. The sentiment label with the highest posterior probability is predicted as the sentiment of the input text. Naive Bayes classifiers are well-suited for sentiment analysis tasks due to their simplicity, efficiency, and effectiveness, particularly when working with large datasets and high-dimensional feature spaces. They are capable of handling text data with minimal computational resources and can provide accurate sentiment predictions even with limited training data. Additionally, Naive Bayes classifiers perform well in scenarios where the independence assumption holds reasonably well, such as in bag-of-words representations of text data.[41]

- Maximum Entropy Analysis.[42]
  - Sentiment analysis using Maximum Entropy is like having a smart detective that looks at all the words in a piece of text to figure out if it's positive, negative, or neutral. Imagine giving this detective a bunch of sentences from movie reviews and asking whether people liked the movie or not. Maximum entropy model carefully studies the words in each sentence, like "amazing" or "disappointing," and learns from examples to understand which words are associated with positive feelings and which ones are linked to negative ones. By considering all these words together, sentiment analysis model using maximum entropy can make a smart guess about the overall sentiment of the text, helping us understand how people feel about different things, like movies, products, or social media posts. Overall maximum entropy predicts the sentiment of text data based on the idea of "maximum uncertainty". By maximum uncertainty we imply that model does not make a strong assumption about what sentiment it has until it analyzes all the words and context of the data. It starts with a very flexible, unbiased assumption and gradually adjusts its prediction based on the evidence it finds in the text.[17]
- K-Nearest Neighbours (KNN) and weighted KNN analysis.[24]
  - Sentiment analysis with k-nearest neighbors (KNN) is like asking for advice to your closest ones (friends and family for example). By putting it simple with a straight forward scenario let's consider the following. Imagine you have a bunch of reviews for a new movie, and you want to know if they're positive or negative. With KNN, each review is like a house on a street, and each word in the review is a feature, like the color of

the house or the size of the yard. When a new review comes in, KNN looks at the words in that review and finds the closest neighbors—other reviews with similar words. Then, based on how the neighbors feel (positive or negative), KNN decides the sentiment of the new review. Weighted k-nearest neighbors (WKNN) works similarly, but it pays more attention to the opinions of closer neighbors, giving them more influence in the decision-making process. So, with KNN and WKNN, we can use the collective wisdom of similar reviews to quickly determine the sentiment of new text data, making it a helpful tool for understanding people's feelings and opinions. Overall, KNN predicts sentiment based on the K most similar sentences. On the other hand, weighted KNN assumes that some of the K selected sentences are more similar than others to the one under investigation.[40]

- Multilingual sentiment analysis.
  - Multilingual sentiment analysis is like understanding emotions in different languages. Imagine having reviews written in English, Spanish, and French, and wanting to know if they're positive or negative. With multilingual sentiment analysis, techniques to interpret sentiments across various languages are used. So, instead of relying solely on words, we might look at emojis, cultural context, or even translate the text to a common language for analysis. By considering these factors, we can accurately gauge the sentiment of text data in different languages, helping us understand people's feelings and opinions globally. Overall, this option is a go-to method when textual data are written in different languages. This method figures out what are the sentiments in a mixed-language text dataset. [30]
- Feature driven sentiment analysis.
  - Feature-driven sentiment analysis is used to examining specific aspects of text to understand sentiments. Picture reading product reviews and wanting to know not just if people liked the product overall, but also which

features they liked or disliked. With feature-driven sentiment analysis, the researcher focus on identifying and analyzing sentiments related to particular attributes or elements mentioned in the text, such as performance, design, price, or customer service. By paying attention to these specific features, we can gain deeper insights into the aspects of a product or service that drive positive or negative sentiments, helping businesses understand customer feedback and make targeted improvements.[4]

Though ml approach[2] can be handy in most situations we should keep in mind that nothing good ever comes without a price. Therefore ml approach has some advantages and disadvantages that need to be clearly stated. Some of the most notable advantages of using ml approach amongst others are adaptability, deeper understanding of data, and extensibility. By adaptability we mean that ml models can learn complex patterns in text data where a human reader could possible overlook, when it comes to deeper understanding we aim to highlight the capability of ml models to capture context and nuances in linguistic text data. Finally, extensibility is a referral to the fact that after the hard job has been completed and the models are trained then ml models are able to process large volumes of text data quickly. On the other side of the light though, ml models some times may require large amount of labeled training data (like in n-gram sentiment analysis). This can be quite a time-consuming process and above all a costly one. Apart from that in some cases the models built are really complex. This could have as a result predictions being hard to interpret due to the fact that we are unable to follow the steps in acquiring those predictions due to the model complexity. Finally, a really time-consuming process is the model selection, hyper-parameter tuning and feature engineering which are of the highest importance if one wants to build an ml model to conduct sentiment analysis.

Rule based approach[5] require the user to define certain lexical rules, as the name of this methodology states, before the analysis begins. After the rules are all set, tokenization takes place on each sentence in every document where General Inquirer data set[6] is created. Later on if a word is associated with positive sentiments, $a+1$ rating is applied to the document's overall score else if a word is associated

[6]A data set that contains the sentimental score for word collected from tokenized documents.

with negative sentiments, $a-1$ rating is applied to the document's overall score. All documents start with a score of zero initially. At the end, the sum is going to be positive, negative or zero. Like ml, rule based approach has its own advantages and disadvantages. On one hand, rb approach is a transparent approach, can be domain specific and with high interpretability. Transparency implies that since the rules are human-defined, they can be understood at its maximum level and of course they can be modified easily. As rules are create based on specific document collection, researchers can tailor these rules to a specific domain and an industry-specific sentiment analysis can be conducted easily. Lastly, the results of a sentiment classification can be explained without much since a reader of the results is aware of the rules that produced the sentiment analysis results.

On the other hand, the disadvantages of the rb approach are not a few and should not be left unmentioned. Since humans generate the rules, experience is required and even after a lot of practise, a research may come across complex language patters where the generated rb system may not be able to handle novel linguistic requirements. Apart from that, analysing and creating complex rules to handle all requirements is a time consuming process. Apart from that, while language evolves as everything else, the research is required to take precautions and if possible generate more generative rules so that patterns won't be affected and predict the correct outcome. Apart from evolution, context may vary from document to document. Irony and sarcasm are only some of the variations that could be included, and a researcher should be able to oversee such roadblocks. Composing rules to handle such issues is a demanding process.

The final method discussed is the lexicon-based approach[38]. In contrast to the other methods, the lexicon approach requires the researcher to utilize a dataset, known as a lexicon, that contains terms relevant to the language or subject under scrutiny in sentiment analysis. Depending on specific requirements, this sentiment lexicon must encompass information regarding which words and phrases (at various analytical levels) are deemed positive or negative.

Researchers have the option to construct a lexicon tailored to their particular case or subject of interest, or they can make use of pre-existing lexicons. For instance, there are established lexicons such as **LIWC** (Linguistic Inquiry and Word Count) and **GI** (General Inquirer), which classify words as positive or negative based on their context-free semantic orientation. However, it's worth noting that these lexicons have limitations; for instance, LIWC does not include acronyms, emoticons, and slang.

Alternatively, lexicons like **ANEW** (Affective Norms for English Words), **SentiWordNet**, and **SenticNet** provide valence scores that indicate the intensity of sentiment associated with words.

The lexicon-based approach, among other methods, stands out for its transparency and ease of interpretation. This approach relies on predefined lexicons, which means it's clear how sentiment for each word is determined. Researchers using this method have the flexibility to create custom lexicons tailored to their specific domain, industry, language, or preferences. Another advantage is that, since pre-built lexicons are designed to assign sentiment scores to individual words, this process doesn't demand vast amounts of labeled training data for model training. Consequently, it offers a swift and cost-effective solution for sentiment analysis. Moreover, it simplifies implementation and doesn't necessitate extensive experience in machine learning. All that's needed is preprocessing of the document.

However, it's important to acknowledge the drawbacks of the lb approach, as they are not insignificant. Lexicons often fall short in covering all possible words and phrases, particularly those that are specific to certain domains, newly coined, or undergoing language evolution. Furthermore, this approach doesn't consider the context in which words or phrases are used, which can result in perplexing outcomes and ambiguities.

Dealing with negations, such as *not good* or intensity modifiers like *very good* can be a challenging task for the LB approach. Additionally, the predefined emotion categories can sometimes limit the method, as it might not fully capture the diverse range of sentiments expressed in a text.

Lastly, because of the aforementioned disadvantages, lexicons need regular updates to accommodate changes within a subject area or the evolving language and expressions of sentiment. This can be a labor-intensive process.

As analysis in general requires a lot of concentration, so is sentiments extraction. Combining the aforementioned approaches can lead to minimizing the drawbacks of every method and expose the advantages of them. As a result, applying more than one approaches is also a possible option that sometimes can lead to better results.

5 Topic Modeling

5.1 Latent Dirichlet Allocation

Topic modeling is a process that belongs to the family of models that help us categorize the documents according to the topic these documents focus on according the words used. One of the most widely used model, which we are going to discuss about in this section is **Latent Dirichlet Allocation** (*LDA*). LDA was initially introduced by Blei et all [6]. They described LDA as "*a generative probabilistic model of discrete data such as text corpora*". When it comes to use LDA model, there are four fundamental assumptions, which will be described in detail. But first let us leave the mathematical details aside for a while and try to understand the steps taken towards the model operation, through a very simple example.

Think about all those times you are composing a document. Initially, you decided what is the main topic that you want to analyze in this document and then you start writing, based on the chosen main topic and of course based on a structure that you have settled on. Now, let's assume that you have decided to write about politics. There is a very low probability that we come across words like *pasta*, *oven*, *wine*, *salt* etc. Hence, we understand that there is certain vocabulary that is used per topic. Now, let us assume that this document was applauded by many people and expect more from you. So then you decide to include more than just one topic in your new document, other than politics, for example sports and science. Each of the three documents written by you contains different set of vocabulary (each of which is a subset of the English Vocabulary).Based on that sub-vocabularies, imagine that you have a model to randomly select 50% of Politics related vocabulary, 30% of Sports related vocabulary and the rest 20% of Science related vocabulary. The result of this action is to produce a document containing all three topics mentioned above, each with a the percentage rate mentioned above.

LDA is a model that works as described above but in reverse. Given a document, LDA tries to find out the distribution of topics in this document based on the vocabulary it contains, for example for the latter document created, it would inform us that 50% of it is in topic 1 (Politics), 30% of it is about topic 2 (Sports) and the rest 20% is about topic 3 (Science). Of course there is more than that and the variety of the model uses is big and growing constantly. But at this point we shall focus around LDA topic modelling. Given that we understood the intuition behind LDA, we shall now take a step closer and dig deeper towards the model's construction in order to fully understand how LDA model operates and what are its advantages and disadvantages.

Latent Dirichlet Allocation model is built under 4 assumptions:

- Each document is a collection of words, referred to as "*Bag of Words*"
  - The order as well as the grammatical role of the words do not contain any topic related information and therefor are not considered as a valuable information to be included in the model.
  - The fact that LDA treats documents as bags of words means that it does not inherently understand the semantics or meaning of words or documents. Later on we are going to introduce to you the method LSA, which search and categorizes documents according to the semantics of the words.
- Stop words like *am, is, of, a, the, but,* ... do not contain any information and their appearance in any of the topics is very probable (if not certain). As a result they can be removed from the document in a preprocessing step. (Note that in the preprocessing step there are also other elements to consider for a smooth working of the model like capitalization or punctuation marks but you are going to get a closer look to that in the application of LDA)
- The model requires from us to provide the number of topics K, beforehand. There are certain ways we can explore and experiment to find the correct number of topics.
- At the beginning all topic assigned in the vocabulary are assumed to be the correct one, except the word it is analysed at that particular time.

Apart form the assumptions, LDA has two main principles. The first principal is that **every document is a mixture of topics**. This tells us that the model itself is not capable to answer to the question *”Which is the topic of a document?”*, but instead it provide us a mixture of distribution of topics per document. Think of the document we constructed in our previous example where it included the mixture distribution of 3 topics (50% of politics, 30% of sports, 20% of science). In the other hand there may be certain cases where a document can be assigned to 90% of topic 1 and 10% of topic 2. After all LDA provides the distribution of topics per document. The second principal is that **every topic is a mixture of words**. This implies that every word has a strong correlation with some topics and less (or non) with some other topics. For example the words *Legislation*, *Election* and *Government* are strongly associated with ”Politics” topic whereas the words *Research*, *Hypothesis* and *Laboratory* are strongly associated with ”Science” topic. Overall, I believe that LDA can not be described in a simpler way than [34] where the following is stated:

Each word in a document is generated by first sampling a topic from the topic-distribution associated with the document and then sampling a word from the word distribution associated with the topic. Thus, given a corpus, LDA tries to find the right assignment of topic to every word such that the parameters of the generative model are maximized.

Having understood the fundamental information, we are in the position to try and explore how LDA works. The structure[7] of LDA is the following:

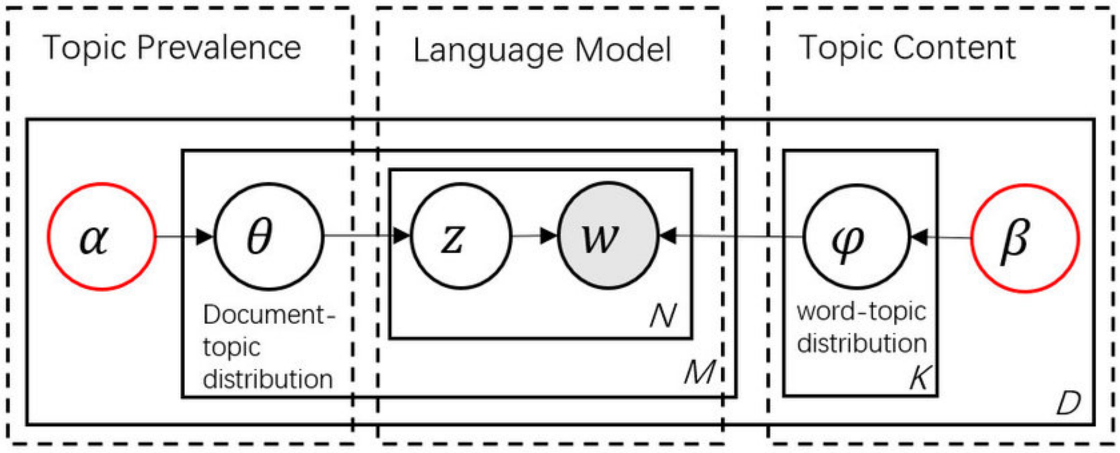

Figure 1: Full structure of Latent Dirichlet Allocation model.

where

[7]Figure shown below was used from Research Gate

α :	is the Dirichlet prior parameter for the per-document-topic proportion.
θ :	is the topic distribution of a given document
z :	is the topic assignment in word in the given document
w :	is the selected word to be examined
N :	is the collection of words from the examined document
M :	is the collection of documents that we aim to cluster
ϕ :	is the distribution of words of a selected topic
K :	is the collection of available topics
β :	is the Dirichlet prior parameter that represents per-topic-word proportion

At this point we need to fully understand how both Dirichlet prior parameters work in order to get on well with the process of LDA topic modelling. The Dirichlet prior parameter α represents document-topic density. This tells us that by setting α high, documents are assumed to be constructed of more topics and result in more specific topic distribution per document. On the other hand a high β parameter setting, indicates that topics are assumed to be made up of words with great variability and result in a more specific word distribution per topic.

As we have established a ground knowledge of the assumptions and the components of LDA, it is time to move forward to the process itself. Latent Dirichlet Allocation is basically an iterative process of topic assignment for each word in each document that we include in our analysis. A key element is the word latent, which implies as stated earlier, that all documents are constructed with the same K-topics that we decide but with a different proportion. In order to complete this iterative process, Markov Chain Monte Carlo (MCMC) methods, such as Gibbs sampling, are commonly employed to infer the latent variables (in our case topics) given the observed data (in our case documents). Let us take a step back and delve into the last sentence and more specifically explore how MCMC method help us achieve our goal, incorporating it to LDA.

MCMC methods are a family of algorithms that are able to assist us sample from complex probability distributions. With the usage of MCMC algorithms we are able to construct a Markov Chain that has the desired complex distribution as its stationary distribution. A sampling method, that is often used with the MCMC algorithm is Gibbs sampling method. In our example we, as well, are going to use Gibbs sampling method. [31]

- Markov Chain Monte Carlo (MCMC) [27]

As you may have realized MCMC consists from two parts. *Markov Chain* and *Monte Carlo*. A Markov chain is a stochastic process where the future state depends only on the current state and not on the past ones. Using mathematical language this process would be defined as

$$f(\theta^{(t+1)}|\theta^{(t)},\dots,\theta^{(1)}) = f(\theta^{(t+1)}|\theta^{(t)})$$

which implies that the distribution of θ at sequence $t+1$ given all the preceding θ values (for times $t, t-1, \dots, 1$) depends only on the value $\theta^{(t)}$ of the previous sequence t. Moreover, $f(\theta^{(t+1)}|\theta^{(t)})$ is independent of time t. In MCMC, we construct a Markov Chain such that its stationary distribution matches the target complex one we want to sample from. As for the second part, Monte Carlo methods use random sampling to obtain numerical results. In MCMC, we use Monte Carlo sampling within the Markov Chain generated in the first part. The sampling process is a sequence of iterations. MCMC algorithms iteratively sample from a sequence of configurations (by configurations I imply the different values that are given to the variables of interest). Those configurations change while the MCMC algorithms operates. Starting from a specific (or random) configuration, MCMC algorithms propose a move to the next values that the variables of interest should take. The process is terminated, or better converge to the desired stationary distribution, when a certain condition is met. This condition ensures that the probability of transitioning from one configuration to another is balanced by the probability of transitioning back. In details the aforementioned balance is expressed as followed

$$\pi(i)P(i,j) = \pi(j)P(j,i)$$

where $\pi(i)$ and $\pi(j)$ are the probabilities of being in configurations i and j, respectively, and $P(i,j)$ is the transition probability from configuration i to configuration j. When this condition is met, we say that the MCMC algorithm converged[8]. Latent Dirichlet Allocation (LDA) is a popular generative model for analyzing large collections of text data. Estimating the parameters of an LDA model typically involves inference techniques such as Variational Expectation-Maximization (VEM) and Gibbs sampling. This chapter provides a comparative analysis of these two methods, focusing on their strengths, weaknesses, and scenarios where one might be preferred over the other. When we are going to use LDA function, we are going to come to the crossroad decision where we are going to ask ourselves, what should we use upon **Gibbs** method and **VEM** method.

Before further exploring MCMC algorithm and Gibbs sampling method, two key elements for LDA method, here we present the steps taken towards setting LDA to action. It can be clearly distinguished that the methods described above are set to action in order to obtain what is requested, to cluster the documents based on their topic distribution.

1. **Initial Topic Assignment**: The LDA model begins by identifying all unique words across the entire corpus of documents. This set of unique words forms the ”vocabulary”. For each word in each document, the model randomly assigns one of the predefined topics. The number of topics is a hyperparameter that must be set before the model is run. This initial assignment of topics is arbitrary and does not carry any semantic weight. It is merely a starting point for the LDA model’s iterative refinement process. As a result of this process, each word in each document is associated with one of the topics. This initial random assignment serves as the input to the next phase of the LDA model, where the topic assignments are progressively updated based on the distribution of words

[8]Though the following details are out of scope it would be a mistake not to be mentioned. Hence, when the Markov chain is irreducible, aperiodic, and positive-recurrent, as $t \to \infty$ the distribution of $\theta^{(t)}$ converges to its stationary (equilibrium) distribution, which is independent of the initial values of the chain $\theta^{(0)}$

across topics and the distribution of topics across documents. The ultimate goal of LDA is to establish meaningful topic assignments, despite the initial randomness. The subsequent iterative process is intended to steer the model towards a configuration where the topic assignments align with the underlying thematic structure of the documents.

2. Topic Reassignment: In the next step, LDA selects a word and removes its initial random topic assignment. Then, considering all other current topic assignments for the remaining words in the vocabulary, the model assigns a new topic to the selected word. This reassignment is not random as in the initial step but is based on two conditional probabilities:

    (a) **Document-Topic Probability** (A): This is denoted as $P(\text{topic } t|\text{document } d)$. It represents the probability that the document d is of topic t, given that we are evaluating document d. This probability is calculated by counting the number of words in the document that are currently assigned to topic t. Additionally, a Dirichlet-generated multinomial distribution over topics in each document is factored into this calculation. The assumption behind this proportion is that if many words in the document are assigned to topic t, then it is more likely that the current word we are considering also belongs to topic t.

    (b) **Topic-Word Probability (B)**: This is denoted as $P(\text{word } w|\text{topic } t)$. It represents the probability that word w belongs to topic t, given that we are evaluating topic t across all documents. This probability is calculated by considering how many times word w appears in documents assigned to topic t, along with a Dirichlet-generated multinomial distribution over words for each topic. This proportion aims to capture the number of instances where word w contributes to a document being in topic t. The new topic for word w is assigned based on the product of these two probabilities:

$$P(\text{word } w \text{ is assigned to topic } t) = AB$$

3. Pick a word and delete its initial topic assignment. Then, given all other topic

assignments for the rest of the words in the vocabulary, re-assign a topic to this word.

The process of re-assignment this time is not random as initially done. In contrast it is the product of two conditional probabilities.

4. **Iterative Topic Assignment**: After the topic assignment is completed for the selected word, the LDA model repeats the same process for all other words in the vocabulary. It picks each word, removes its current topic assignment, and assigns a new topic based on the conditional probabilities described above.

5. **Convergence**: The LDA model continues to iterate over all the words, updating the topic assignments in each iteration. This process is repeated until the model reaches a state of equilibrium where the topic assignments no longer change significantly between iterations. At this point, the LDA model has converged, indicating that no further changes are needed, and the final topic assignments can be interpreted as the model's best guess for the underlying topic structure of the documents.

It is highly important to understand step 2 at a maximum level. As a result, we are going to elaborate a little bit more that step in order to fully understand the role that both Dirichlet parameters play in LDA process. As we have seen, there are two Dirichlet-generated multinomial distributions included in order to proceed to the topic assignment in each word selected. This is due to the fact that initially, this method assigns randomly topics to each of the words, where a document may end up with zero probability of a certain topic where in reality this topic should have been included in the document's mixture distribution of topics. In that case, Dirichlet probability distribution is included over K topics as it is a non-zero probability for the topic generated. This way, a topic with zero initial probability may be included in any future iterations if this should be done.

At this point we are able to go through those iterations in a more formal definition of the process. Latent Dirichlet Allocation goes through the following process to produce the clustering of the documents according to the topics.

1. Choose a sequence ofh words $w_N \sim Poison(\xi)$.

2. Choose $\theta \sim Dir(\alpha)$.

3. For each word w_i in the previously chosen sequence words w_N:

   (a) Choose a topic $z_n \sim Multinomial(\theta)$.

   (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Given the provided process just above, there is a high need to present some comments as, written in the Blei et all [6]. The dimensionality k of the Dirichlet distribution is assumed (and consequently the dimensionality of the topic variable z) as we have seen earlier is known and fixed. Apart from that, the word probabilities are represented by a $k \times V$ matrix β, where $\beta_{ij} = p(w_j = 1|z_i = 1)$, which is treated as a fixed quantity to be estimated. As for a final comment of the previously described process, the Poisson assumption is not essential for what follows; more realistic document length distributions can be used if needed. Additionally, note that N is independent of all other data-generating variables (θ and z). Therefore, N is an ancillary variable, and its randomness will be disregarded in generally.

A k-dimensional Dirichlet random variable θ can take values within the $(k-1)$-simplex (a k-vector θ lies in the $(k-1)$-simplex if $\theta \geq 0$ and $\sum_{i=1}^{k} \theta_i = 1$). The probability density of θ on this simplex is given by:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$$

where parameter α is a k-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function. Given α and β, the joint distribution of a topic mixture θ, a set of N topics $\mathbf{z}$, and a set of N words $\mathbf{w}$ is given by:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

where $p(z_n|\theta)$ is θ_i for each i that $z_n^i = 1$. The marginal distribution is obtained by integrating over θ and summing over z:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) (\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) d\theta$$

At a final step, taking the product of the marginal probabilities of a single document, we obtain the probability of a corpus.

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha)(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta))d\theta_d$$

Figure 1 presents the LDA model as a probabilistic graphical model. This diagram emphasizes the three hierarchical levels in the LDA framework. The parameters α and β are at the corpus level, sampled a single time during the corpus creation. The variables θ_d pertain to the document level, sampled once per document. Finally, the variables z_{dn} and w_{dn} are at the word level, sampled once for each word in every document.

It is important to distinguish LDA from a basic Dirichlet-multinomial clustering model. In a conventional clustering model, there is a two-tier structure where a Dirichlet distribution is sampled once for the entire corpus, a multinomial clustering variable is selected once per document, and words are chosen for the document based on the cluster variable. This model restricts a document to a single topic. Conversely, LDA has a three-tier structure, with the topic node being sampled multiple times within the document, enabling documents to be linked to multiple topics.

Figures such as Figure 1 are frequently encountered in Bayesian statistical modeling and are termed hierarchical models as noted by Gelmar et al. [20]. More specifically, they are known as conditionally independent hierarchical models according to Kass and Steffey [25]. These models are also referred to as parametric empirical Bayes models, a designation that encompasses both the model structure and the parameter estimation techniques, as described by Morris [28].

As mentioned in Section 5, we employ the empirical Bayes method to estimate parameters like α and β in basic LDA models, while also taking into account more extensive Bayesian methods. Now that we have deeply explored LDA method. It is important to take a step back and take a closer look at both VEM and MCMCM, Gibbs sampling method.

Variational Expectation-Maximization (VEM) method[35]

Variational Expectation-Maximization is an optimization-based approach used for estimating the parameters of probabilistic models such as LDA. In VEM, the objective is to maximize a lower bound on the log-likelihood of the observed data with respect to the model parameters. This is achieved through an iterative process that alternates between the Expectation (E) step and the Maximization (M) step. Despite its computational efficiency and ease of implementation, VEM has several limitations.

Some of the advantages of using VEM are the following:

- VEM tends to be computationally more efficient than Gibbs sampling, especially for large datasets. VEM often converges faster and requires fewer iterations to reach convergence compared to Gibbs sampling.
- VEM provides deterministic results, meaning that running the algorithm multiple times with the same initial conditions will produce the same output. This can be advantageous for reproducibility and debugging.
- VEM can be more scalable to large datasets because it does not require storing and updating the entire dataset in memory during each iteration, as Gibbs sampling does. Instead, VEM updates the model parameters based on statistics computed from the data, which can be more memory-efficient.
- Implementing VEM is often simpler than implementing Gibbs sampling, especially for practitioners who are less familiar with Markov Chain Monte Carlo (MCMC) methods like Gibbs sampling. VEM involves solving optimization problems, which can be more straightforward to implement using standard optimization techniques.
- VEM may exhibit more stable convergence properties than Gibbs sampling, especially in cases where the posterior distribution has complex dependencies or is multimodal. However, the convergence properties can depend on the specific problem and implementation details.

Despite these advantages we mentioned earlier, Gibbs sampling has its own strengths and may be preferred in certain scenarios. In details, VEM has the following drawbacks:

- VEM provides approximate solutions to the posterior distribution over the latent variables, which may not fully capture the true uncertainty in the data.
- VEM can be sensitive to the choice of initial parameter values, potentially leading to convergence to local optima.
- VEM optimization may converge to local optima, especially in high-dimensional parameter spaces or when the likelihood surface is complex.

Gibbs Sampling [21]

Gibbs sampling is a specific method of MCMC algorithm used when sampling from high-dimensional distributions[9]. It is particularly well-suited for models with conditional dependencies between variables by consecutively sampling from conditional distributions. Despite its computational demands, Gibbs sampling offers several advantages over VEM.

- Gibbs sampling provides exact samples from the posterior distribution over the latent variables, making it suitable for cases where precise inference is required.
- Gibbs sampling can handle non-conjugate priors and complex likelihood functions, offering greater flexibility in modeling diverse datasets.
- Gibbs sampling is less sensitive to initialization compared to VEM, reducing the risk of convergence to local optima.
- Gibbs sampling can handle missing data or incomplete datasets more effectively than VEM, allowing for more robust analysis in such scenarios.

Gibbs method operates as follows. Firstly, we begin by the initial state, which is a random assignment of values to all the variables in the model.

[9]By high-dimensional distribution it is implied a probability distribution that involves a large number of random variables or dimensions.

Then, at each iteration, one variable is selected form the model. This variable's value is updated conditioned on the current values of all the other variables as follows:

1. Sample $x_1^{(t+1)}$ from $p(x_1|x_2^{(t)},\dots,x_n^{(n)})$.
2. Sample $x_2^{(t+1)}$ from $p(x_2|x_1^{(t)},x_3^{(t)}\dots,x_n^{(n)})$.
3. $\vdots$
4. Sample $x_n^{(t+1)}$ from $p(x_n|x_1^{(t)},\dots,x_{n-1}^{(n)})$.

This update gives us an approximate sample $(x_1^{(m)},\dots,x_n^{(m)})$ that can be considered as sampled from the joint distribution for large enough m's. It is complete by considering the conditional distribution given the current values of all other variables. This process allows us to keep in hand the conditional independence structure of the model. The process of iterative sampling is repeated for each variable until the convergence is met.

In the context of this analysis, LDA, Gibbs sampling is used to infer the latent topic assignments for each word in each document, given the current state of the model and the observed data. Let,

$$n_i = (n_{i1},\dots,n_{iV})$$
$$m_d = (m_{d1},\dots,m_{dk})$$

where n_{ij} the number of occurrence of word j under topic i, m_{di} is the number of loci in d-th individual that originated from population i.

1. Update β^{t+1} with sample from $\beta_i|\mathbf{w},\mathbf{z}^{(t)} \sim D_V(\eta+\mathbf{n}_i)$
2. Update θ^{t+1} with sample from $\theta_d|\mathbf{w},\mathbf{z}^{(t)} \sim D_k(\alpha^{(t)}+\mathbf{m}_d)$
3. Update θ^{t+1} with a sample by probability

$$P(z_{dn}^i = 1|\mathbf{w},\beta^{(t+1)}) = \frac{\theta_{di}\beta_{iw_{dn}}^{(t+1)}}{\sum_{d=1}^M \theta_{di}\beta_{iw_{dn}}^{(t+1)}}$$

4. Update $\alpha^{(t+1)}$ by the following process:

    1. Sample α' from $\mathcal{N}(\alpha^{(t)}, \sigma^2_{a^{(t)}})$ for some $\sigma^2_{a^{(t)}})$
    2. Let $\alpha = \frac{p(\alpha'|\theta^{(t)},\mathbf{w},\mathbf{z}^{(t)})}{p(\alpha^{(t)}|\theta^{(t)},\mathbf{w},\mathbf{z}^{(\mathbf{t})})} \cdot \frac{\phi_{\alpha'}(\alpha^{(t)})}{\phi_{\alpha}^{(t)}(\alpha')}$
    3. Do not update $\alpha^{(t+1)}$ if $\alpha' \leq 0$. Update $\alpha^{(t+1)} = \alpha'$ if $\alpha \geq 1$, otherwise update it to α' with probability α.

In topic modeling we require the estimation of document-topic distribution θ and topic-word distribution β. Griffiths and Steyvers [23] boiled the process down to evaluating the posterior $P(\mathbf{z}|\mathbf{w}) \propto P(\mathbf{w}|\mathbf{z})P(\mathbf{z})$ which was intractable. Notice that the target posterior is marginalized over β and θ. This makes it a collapsed Gibbs sampler [33]; the posterior is collapsed with respect to β and θ.

We are going to take see this process in action within the last section where the example is included at Section 6. In that section an application is presented to actual data.Gibbs method iteratively samples new topic assignments for words based on the current topic assignments for other words in the document and the current topic distributions. This process allows LDA to approximate the posterior distribution of topic assignments given the observed data.

While VEM offers computational efficiency and ease of implementation, there are several scenarios where Gibbs sampling may be preferred for estimating LDA models. The reasons why I personally have selected to use Gibbs in the following application are the following:

- When precise inference is necessary, especially in cases where the true posterior distribution is of primary interest, Gibbs sampling provides exact samples from the posterior.

- Gibbs sampling can handle non-conjugate priors and complex likelihood functions, making it more suitable for modeling diverse datasets with intricate structures.

- Gibbs sampling is less sensitive to initialization compared to VEM, reducing the risk of convergence to local optima and providing more reliable parameter estimates.

- Gibbs sampling offers greater flexibility in handling missing data or incomplete datasets, making it preferable in scenarios where data quality is a concern.

5.2 Latent Semantic Analysis

While LDA is a complete method and it is proven to be quite helpful, as you may have already realized, it does not take into account the meaning of the words. We have already mentioned that LDA treats every document as a bag of words. As a result, it returns the distribution of topics based on the frequency of every word in the document.

In this chapter we are going to take look at the topic of Latent Semantic Analysis (LSA), a technique widely employed in natural language processing (NLP) and information retrieval (IR). Latent Semantic Analysis is a method of extracting the meaning by statistically analyzing word use patterns.[14] Better stated, LSA is a statistical model of word usage that permits comparisons of the semantic similarity between pieces of textual information. LSA was originally designed to improve the effectiveness of information-retrieval methods by performing retrieval based on the derived ”semantic” content of words in a query as opposed to performing direct word matching.[18] Overall, LSA offers a powerful framework for analyzing and extracting latent semantic relationships between terms and documents within a corpus.

At its own backbone, LSA uses principles from both linear algebra and statistics to reach our goal when set to use. Latent Semantic Analyses aims to capture the underlying structure of a corpus by representing terms and documents as vectors in high-dimensional semantic space. Having accomplish this we are able to explore the semantic similarities and relationships between terms and documents.

Since the same word often has more than one meaning (polysemy), irrelevant material will be ”connected”. The foundational concept behind LSA is the distributional hypothesis, which posits that words occurring in similar contexts tend to have similar meanings. In this model, the similarity of terms and document is determined by the overall pattern of word usage in the entire collection, so that document can be similar to each other, regardless of the precise words they contain. [11]

Latent Semantic Analysis leverages this principle by first generating a matrix of co-occurrences of each word in each document (sentences or paragraphs). Then through Singular Value Decomposition (SVD), a technique related to eigenvector decomposition and factor analysis ([9], [19]), LSA transforms this matrix into a lower-dimensional space, wherein latent semantic relationships are revealed. Rather than representing documents and terms directly as vectors comprised of individual words, LSA depicts them as continuous values along each of the k orthogonal indexing dimensions obtained from SVD analysis. As the number of factors or dimensions is significantly smaller than the count of unique terms, words exhibit interdependence. For instance, if two terms are employed in comparable contexts (documents), their vectors in the reduced-dimensional LSA representation will be similar. This methodology offers an advantage in enabling matching between two textual pieces of information, even in the absence of shared words.

The analysis conducted by Singular Value Decomposition can be understood geometrically. The outcome of SVD is a k-dimensional vector space comprising a vector for each term and document, with these vectors reflecting correlations among the terms present in the documents. Within this multi-dimensional space, the cosine or dot product between vectors indicates their inferred semantic similarity. Consequently, by establishing the vectors for two textual pieces of information, we can assess their semantic resemblance.

This approach involves a high-dimensional representation, which allows one to better represent a wide rang of semantic relations. Apart from that both terms and text objects are explicitly represented in the same space. Finally objects can be retrieved directly from query terms.

The performance of information-retrieval systems, such us this one, is often summarized in terms of two parameters: precision and recall.Recall is the proportion of all relevant documents in the collection that are retrieved by the system whereas precision is the proportion of relevant document in the set returned to the user.

This method employs a high-dimensional representation, facilitating a more comprehensive depiction of a broad spectrum of semantic relationships. Additionally, by using LSA we incorporate the explicit representation of both terms and textual objects within the same spatial context. Furthermore, this framework enables us to directly retrieve objects based on query terms.

The efficacy of information-retrieval systems, exemplified by this approach or the one described earlier (LDA), is frequently evaluated using two key parameters: precision and recall. Recall pertains to the ratio of all relevant documents in the collection that are successfully retrieved by the system, while precision refers to the ratio of relevant documents within the set returned to the user.

6 Application

In order to apply the information presented above, data is essential. This section is organized into two subsections. Initially we are going to describe the way we collected our data. Subsequently, we will employ the Latent Dirichlet Allocation (LDA) method across various cases and comment on the outcomes obtained.

6.1 Data Collection

The main goal here is to create clusters of similar documents according to the topic those documents contain. As a result, for such an application, we decided to collect the data from a web page, with web scrapping. The web page selected is

$$https://thegreatestbooks.org$$

where there is a list with the top books one should consider reading if he/she is fun of this activity. In this web page, there are 74 different genres of books to consider, and more that 15000 books to choose from.

We decided to aim to 8 of the genres, accordingly:

1. Biography (331 books)
2. Crime (323 books)
3. Mystery (319 books)
4. American History (271 books)
5. Humor (266 books)
6. Fantasy (266 books)
7. Speculative Fiction (263 books)
8. Poetry (256 books)

The rule, according to which, we ended up with those 8 genres is the amount of books included in them due to the fact that some of those have an excessive collection of books (for example Fiction is a collection with 2874 books) and others have a few of them (for example Westerns have only 15). As a result we have selected the genres that include more that 250 and less that 350 books.

The process of web scrapping is divided into 3 different parts.

6.1.1 Web Scrapping Genres Collection

Initially, as outlined earlier, we have collected the information of every genre to formulate a criterion. This criterion serves as the basis for making optimal selections, ensuring minimal variance across various genres. To complete this process the usage of the following packages was essential:

- rvest
- xml2
- dplyr
- stringr

The first package contains all the functions for establishing a connection with the designated web page and read HTML script written to construct it. We have decided to use XPath directives in order to write down the rules in the R script to locate the elements that we want to scrape (in our case, genre and number of books included in it). Ultimately, we stored the extracted genres in an *.RData* file, so that we can use the information in the next steps and there is no need to run the process again.

The extraction of genres and book counts proved to be a swift and straightforward process with low amount of interest and as a result, there is no need for further in-depth analysis. For the next two web scrapping steps, we have used the exact same R packages and as a result, we are not going to mention them again.

6.1.2 Web Scrapping Titles Collection

By visiting the source page, from which we collected the data, one will notice that there are only 25 books shown per page. As a result, initially, we defined a custom R function to identify whether the page that is currently, collecting data from is the last of the selected genre or not. This will help the process go to the next page or start the process over by selecting the next selected genre in our 8-genre list. Our custom function checks whether there is the word **Next** in the presented in the page numbering part of the HTML Script or not. When the last page is reached there the **Next** word will disappear and as a result the process is going to stop. The custom function is called *last.page.check* with only one argument, the url that we currently have visited.

To obtain the information we aim to acquire, Initially, we define two empty, dataframes. This is to break the process into two parts. The first one will scrape the information for the first four genres and the second for the last four. Initially, by using a for loop, we go through ever genre and add it to the url, parent url. Then with the usage of a while loop, in order to check whether the url presented is the last page or not, we go through each page. Last with the help of XPath we locate the list with all 25 books in the currently visited page and collect the text that contains book titles and save it to the dataframes mentioned earlier.

At the end, the final structure of the dataframes are going to have 5 columns namely:

1. Page.Number : Page number which HTML script we the books are scraped from.
2. Book.Number : The book number to the list per genre.
3. Title : The book title that we have collected.
4. Author : The authors that have written each book.
5. Genre : The main genre that each book belongs to.

The total amount of time required for this process to be completed is approximately 112.15 mins (59.68 mins for the first part and 52.48 mins for the second). The results of this process are saved to an *.RData* file as well.

6.1.3 Web Scrapping Abstracts Collection

The final step of this process is to collect the abstracts/description of books we have collected the titles from. In this process we are going to use again the same function introduced earlier, *last.page.check*. In order to create an automated process, we need to identify the rules that need to write down. By visiting the page where every book description is written, we can notice that the pattern is hidden in book ids. Each book has a certain id number that is imported to a certain url and drive us to the web page where every book abstract is written. As a result, initially, we define the process where we collect every book's id number.

Next in the new web page, we define an automated process where we collect the book description using XPath. For this process to be complete, we create four empty dataframes where we are going to store information of every book along with its descriptions. Finally, we extract the dataframes in an *.RData* file.

Each dataframe has five columns, namely:

1. Page.Number : Page number which HTML script we the books are scraped from.

2. Book.Number : The book number to the list per genre.

3. Book.Id : The id of every book.

4. Abstract : The description of each book.

5. Genre : The main genre that each book belongs to.

The total process ended after approximately 136.48 mins (33.30 mins for the first two genres, 29.48 mins for the next two genres, 28.53 mins for the next two genres and 45.17 mins for the last two genres).

6.2 LDA application

In this section, we will walk through the entire process undertaken to fit an LDA model and cluster all the documents gathered in the aforementioned steps. The packages that we used to complete this process are the following:

1. tidyverse
2. tm
3. caret
4. quanteda
5. topicmodels
6. LDAvis
7. ggplot2
8. dplyr
9. wordcloud2
10. ldatuning
11. tidytext

Let's step back and delve into the rationale behind selecting these packages. The inclusion of *tidyverse* and *dplyr* packages is driven by the need to take advantage of the state of art data science tools provided. These packages help us manipulate data, specifically for objects like dataframes or matrices. On the other hand, packages like *tm*, *quanteda* and *tidytext* help us manipulate and analyze textual data. The *caret* package contributes by providing a function to construct confusion matrices, facilitating result analysis and drawing conclusions.

The function to fit an LDA model is hidden within *topicmodels* package, so given the topic of this thesis we could not miss loading this one. Additionally, packages like *LDAvis*, *ggplot2* and *wordcloud2* are there to aid in visualizing the acquired results. Lastly, *ldatunning* package, is there to to automate the extraction of the optimal number of clusters, in our case, topics or genres.

In order to develop a comprehensive application with minimal oversight, we aim to explore as many scenarios as feasible within the constraints of time and available data. Consequently, we will analyze the following cases, evaluating each by testing alpha values of $0.9, 0.5, 0.1, 0.01, 0.001$:

1. Fit a model with the only two genres.
2. Fit a new model including the next two genres, four in total.
3. Fit a model with all eight genres included.
4. Fit a model where similar topics will be merged.

Similar to any other model fitting procedure, once data are loaded, prepossessing is a necessary step. Initially, we check if there exist any null values in the abstract column. The reason is that some books may not have description included, or for some reasons the web page where the description is included does not follow the same structure as a result our scrapping program was not able to find the information we required it to search for. After the inappropriate elements are excluded, we create a corpus out of all the abstracts, collected. With the assistance of *quanteda* package, we then extract the tokens of the elements of that corpus. The preprocessing steps that we have decided to take are the following. Firstly, we are removing punctuation as well as numbers. This is done because no major information is hidden within those elements. Secondly, we convert all the characters to lower. If this step is skipped, our model will handle differently the word ”City” and ”city”. Thirdly, we decided to remove the stop-words (of English language). Note that if this is done before converting each word to lower, than some stop-words may be skipped. The reason for that is because the vector containing all stop-words existing with lower valued characters. As a result, removing them before conversion to lower, we would end up having words like ”The” and ”A” when a sentence would start in such a way. Finally, we stemmed every element in that corpus. The process of stemming is converting ”complex” words like *scrapping* to its very basic form, *scrape*. After those steps are successfully completed, we should convert the final outcome to a document-term matrix. A matrix formed which is accepted by *topicmodels* package to be included and processed by *topicmodels::LDA* function.

6.2.1 Two genre model clustering

In this case, initially, there existed 650 abstracts for the first two genres, 12 of which are saved with no abstract. As a result, those are removed. As described earlier, We fitted five models with the aforementioned alpha values. In order to evaluate the performance of our model, we are going to take advantage of confusion matrix[10]. In the following table we present Accuracy (including 95% confident interval with p-value), Precision, Recall and F1 Score values.

α	Accuracy	95% CI	p-value	Precision	Recall	F1 Score
0.9	0.931	$(0.9085, 0.9494)$	$< 2e - 16$	0.9560	0.9102	0.9325
0.5	0.9326	$(0.9103, 0.9508)$	$< 2.2e - 16$	0.9623	0.9080	0.9344
0.1	0.9232	$(0.8997, 0.9426)$	$< 2.2e - 16$	0.9591	0.8944	0.9256
0.01	0.9357	$(0.9138, 0.9535)$	$< 2.2e - 16$	0.9654	0.9110	0.9374
0.001	0.9373	$(0.9156, 0.9548)$	$< 2.2e - 16$	0.9654	0.9137	0.9388

We can clearly see that LDA model managed most of the times to successfully label documents as topic one or topic two. The distinction in this case occurred between biography and crime genres. Given the metrics provided above, we can see that the best performance is obtained from LDA model with alpha value being 0.001. Hence, let us take a deeper view of that model's results.

In the figure shown above, beta is the per-topi-per-word probability, and term refers to the each specific word. We can clearly see the distinction amongst the groups examined at this point. The words that exist in both genre groups, in those top-20-words lists are :

1. explor (4^{th} place in both groups)
2. delv (10^{th} and 13^{th} place respectively)
3. narrat (8^{th} and 17^{th} place respectively)

Word cloud is also a useful tool we should take advantage of. We use it to represent top-20-words per genre, where the size of each word, represents *beta* value.

[10]In section *Abstract 1 - Confusion Matrix* you can find more on confusion matrices as well as the metrics we are going to take advantage of

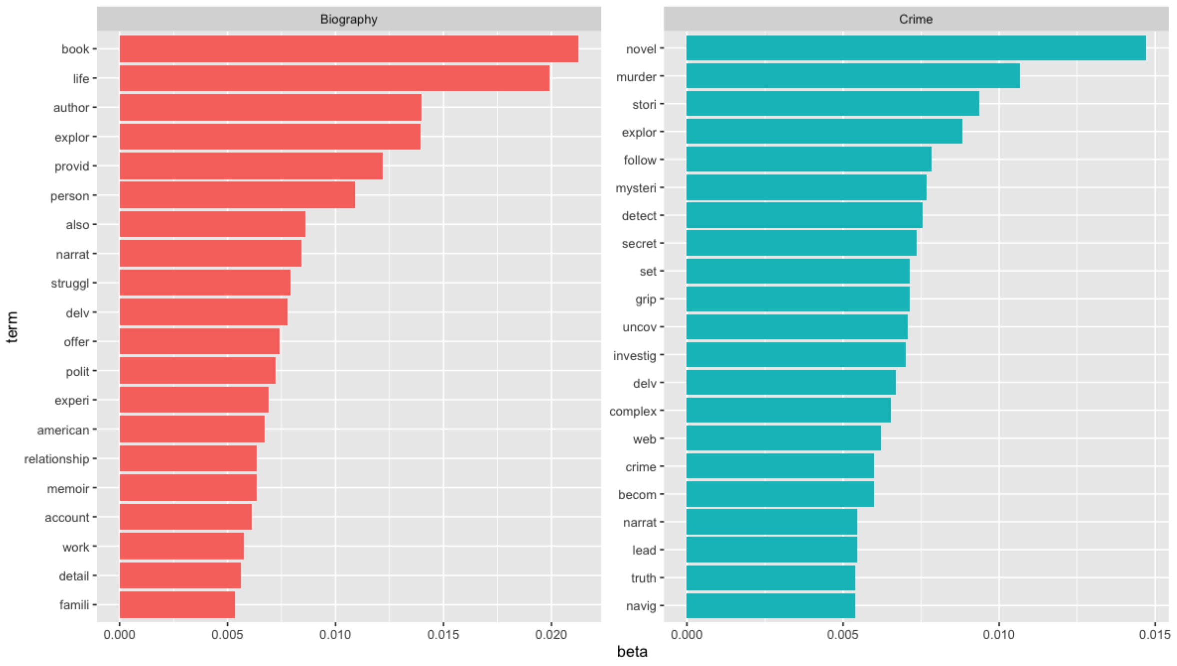

Figure 2: Top 20 terms ($\alpha = 0.001$), 2 genres

Figure 3: Word Cloud of 'Crime' genre

Figure 4: Word Cloud of 'Biography' genre

6.2.2 Four genre model clustering

Following the same pattern as the previous section, we are going to present initially the results obtained from the LDA process when four genres are included instead of two. When started, we begun with 1225 abstracts, from which 22 were without abstract to use. As a result, our analysis included 1203 book abstracts. The genres that are selected for this analysis are

1. Biography
2. Crime
3. Mystery
4. American-history

By taking a step back and thinking about the genres selected we can clearly see that there are plenty of obvious words that could exist in both 'Crime' and 'Mystery'. As a result, we do not expect the same high results.

α	Accuracy	95% CI	p-value
0.9	0.5436	$(0.515, 0.5721)$	$< 2.2e-16$
0.5	0.5594	$(0.5308, 0.5877)$	$< 2.2e-16$
0.1	0.5328	$(0.5042, 0.5613)$	$< 2.2e-16$
0.01	0.5262	$(0.4975, 0.5547)$	$< 2.2e-16$
0.001	0.4663	$(0.4378, 0.495)$	$6.075e-10$

Those results tells us that our initial observations are validated. We can clearly see that by including two topics that are similar enough, there might be some accuracy scarification. The key take away from this model fitting is that the selected genres (or at least some of them) are highly related and there is no point in distinguishing between them, at least from the terms-point-of-view. We can confirm this finding by taking a look at the terms' plots. We are going to establish our finding via presenting the results extracted from the second model, for $\alpha = 0.5$ as it has the highest accuracy amongst the proposed ones. In the following plot we can take a look at the top twenty terms appearing per topic selected for this analysis.

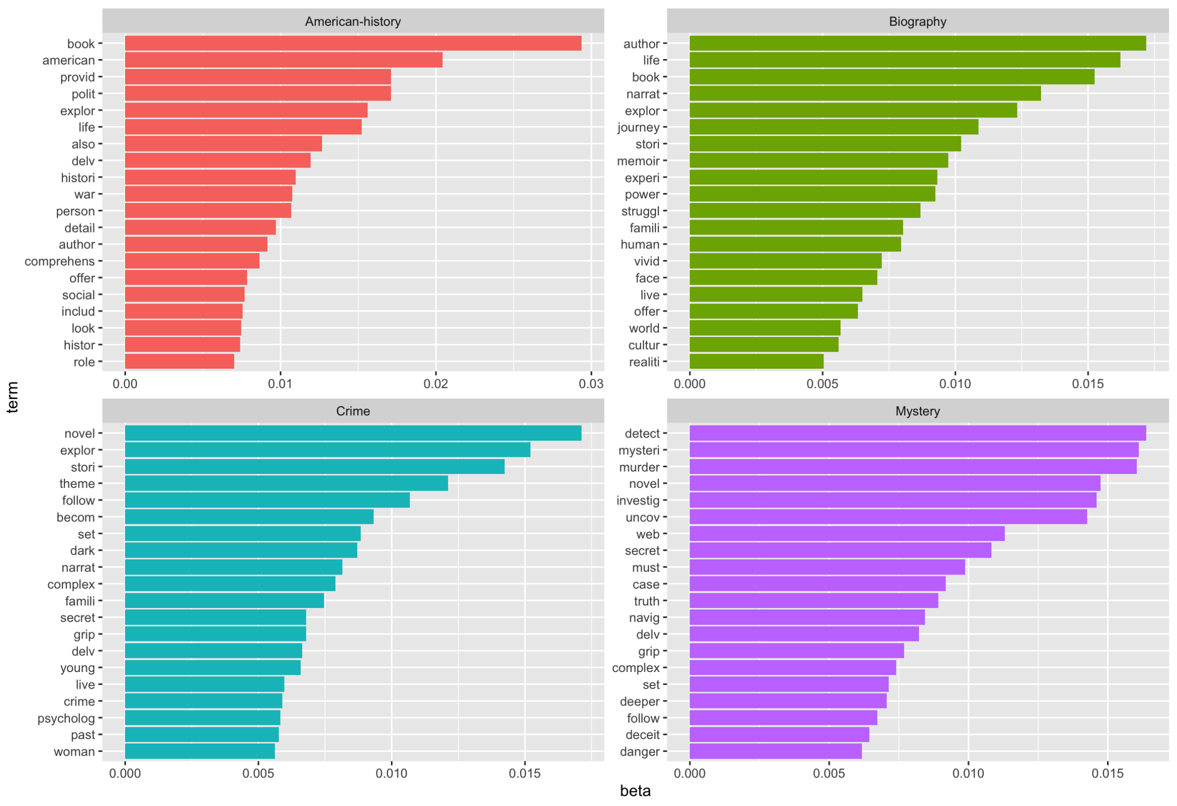

Figure 5: Top 20 terms ($\alpha = 0.5$), 4 genres

If we look carefully, and search all the terms in the preceding barplots presented (likely, we have r-programming language for that) we can see the following common words per pair of topics.

Pair	Count	Common Words
Crime - Mystery	7	novel, secret, **delv**, grip, complex, set, follow
Crime - Biography	5	**explor**, narrat, stori, famili, live
Crime - American History	2	**delv**, **explor**
Mystery - Biography	0	-
Mystery - American History	1	**delv**
Biography - American History	5	book, **explor**, life, author, offer

We can see that the common words per pair of topic has risen. Our initial guess about the common terms for genres Crime and Mystery has been established as we can see that 7 words out of top 20 are common. Furthermore, we can even notice common words within those four selected genres.

6.2.3 Eight genre model clustering

Finally, we are going to include all books scraped. This means that we are going to include all eight genres. Once again, the genres we decided to scrape are the following:

- Biography
- Crime
- Mystery
- American-history
- Humor
- Fantasy
- Metaphysical-visionary-fiction
- Poetry

As we already have encountered this case, in the previous subsection (4.2.2), we have here similar genres that probably will lead to a less accurate clustering result. Some of the cases other than Crime and Mystery are Fantasy and Metaphusical-visionary-fiction. Others would say that poetry may include parts from any of the genres. Though it is in our best interest to try and investigate what are the accuracy of the clusters that LDA will create for us, based on the books description. In this case, as well, we are going to fit multiple models, with different values for alpha.

α	Accuracy	95% CI	p-value
0.9	0.5831	$(0.564, 0.6021)$	$< 2.2e-16$
0.5	0.5981	$(0.579, 0.617)$	$< 2.2e-16$
0.1	0.5985	$(0.5794, 0.6173)$	$< 2.2e-16$
0.01	0.5498	$(0.5305, 0.569)$	$< 2.2e-16$
0.001	0.5303	$(0.5109, 0.5496)$	$< 2.2e-16$

Even though accuracy is mediocre, we can see that the it is slightly better than our previous fittings (with 4 genres). In order to dig deeper into our model's results, we are going to select $\alpha = 0.1$. In this case due to space constraints we are going to present the top 10 terms per topic.

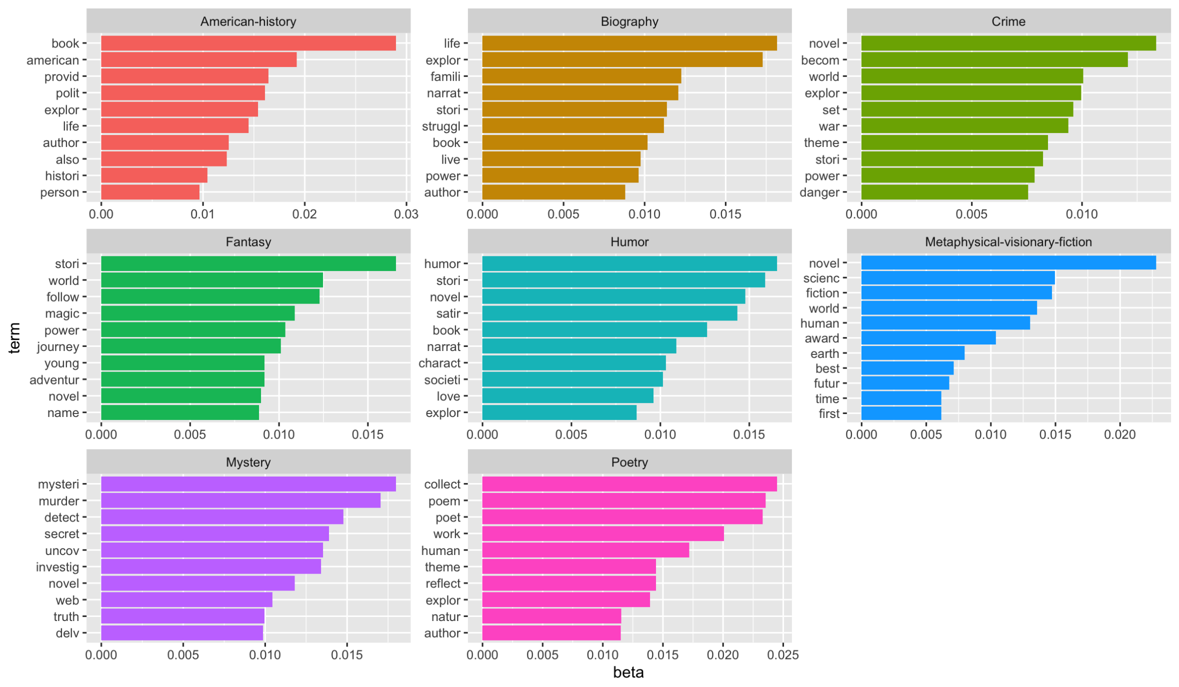

Figure 6: Top 10 terms ($\alpha = 0.5$), 8 genres

At this point, another interesting result of this model is the following confusion matrix, extracted from the results of the model used ($\alpha = 0.1$).

		Predictions							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
True	**American-history**	219	1	2	18	1	1	2	1
	biography	183	1	12	117	3	2	0	0
	crime	14	7	1	51	88	3	139	17
	humor	18	29	20	55	10	6	5	127
	metaphysical-visionary-fiction	3	63	11	16	39	96	0	13
	mystery	1	15	7	23	82	8	166	18
	poetry	6	20	536	26	4	4	1	26
	fantasy	0	213	8	12	11	16	5	8

Table 4: Confusion Matrix of LDA model ($k = 8$, $\alpha = 0.1$, $seed = 1234$)

What we can see here is that, some of the clusters created can be clearly distinguished as a single genre. On the other hand, there exist some clusters (for example 4,5,7) that can not be defined that easily as a single genre.

We can also take a moment to consider per *True* genre how LDA decided to split the books to the clusters created. for example Biography genre books have been split mainly in clusters 1 and 4, whereas Metaphysical-visionary-fiction genre books have been split more ”evenly” in multiple clusters. The reason for that is the same with the one we have already discussed. Some of the genres and the descriptions belonging in those genres are alike. This resulting into classifying certain books in the wrong genre cluster.

6.3 LSA application

In order to make sure that the reader has fully understand Latent Semantic Allocation, we are going to create a simple, yet insightful application using LSA. In this example we are going to use 50 simple sentences written by hand and apply the method LSA, in those. Let us take everything from the beginning. Initially, to follow along, we need to make sure that all the required packages are installed. For this application we will need the following packages:

1. tm

2. lsa

We will need package *tm* to go through some simple preprocessing steps. Apart from that, the main package that will assist us is *lsa* which holds the functions to conduct Latent Semantic Analysis. Initially, we need data. You can of course use your own data, but if you want to reproduce this application the data we imported, are included in the Appendix 2.

The next step is the preprocessing step. Initially, we should convert our data to a Corpus, using *Corpus(VectorSource(df$text_column))* function. At this point we are ready to go through all the preprocessing steps. In this application and because the data are create by hand so that we keep the application simple yet educative, we have to go through the following steps. Initially, we convert all the letters to lower case. When this is completed, we need to remove all the stopwords of the English language. In case other data is used you may need to go through other steps as well, like the ones presented in the previous example. When our data are ready, we convert our corpus to a term-document matrix, where the rows represent the terms and the columns represent the documents. Remember that each cell has a value when the terms shows up the connected document. Finally, we are ready to use the function *lsa::lsa* to produce the results we are looking for.

```
lsa_model <- lsa(doc_mat)
```

The outcome that is saved as *lsa_model* holds three objects.

tk: The term vector matrix **T** which constitutes the left singular vectors, being orthonormal

dk: the document vector matrix **D** which constitutes the right singular vectors, being orthonormal

sk: the diagonal matrix **S** which constitutes the singular values

Remember that LSA combines the classical vector space model — well known in textmining — with a Singular Value Decomposition (SVD), a two-mode factor analysis. A document-term matrix M is constructed with $textmatrix()$ from a given text base of n documents containing m terms. This matrix M of the size $m \times n$ is then decomposed via a singular value decomposition into the aforementioned matrices.

$$M = T \times S \times D^T$$

Initially, we can create a scree plot to figure out the number of topics presented based on semantic of the documents. In our case we have the following

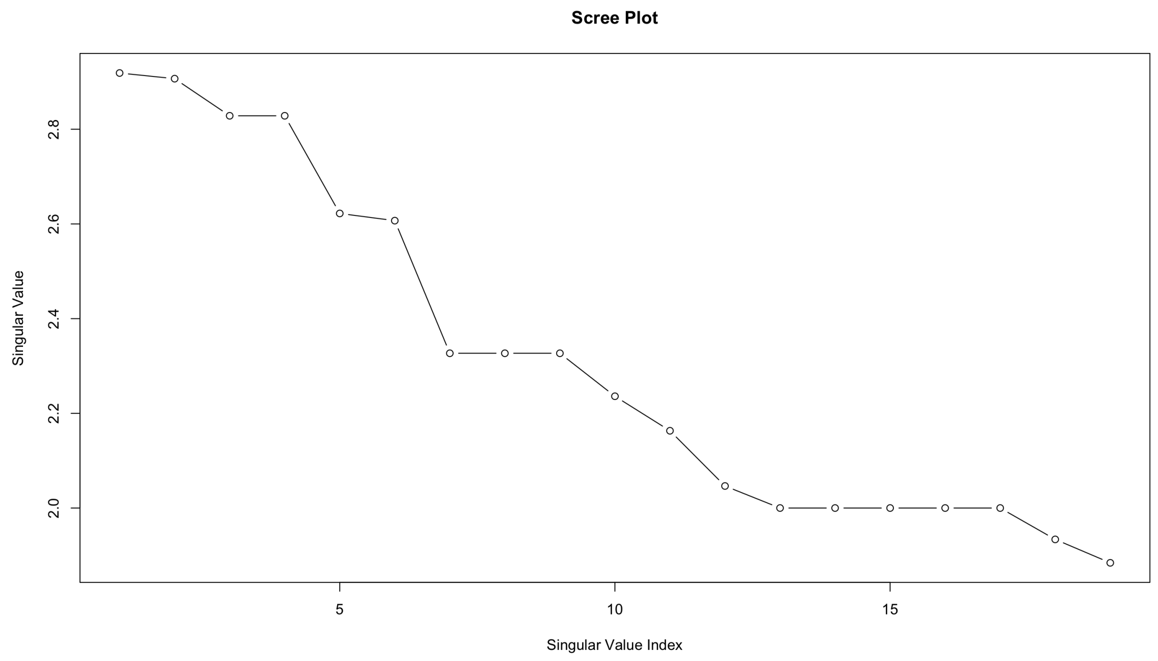

Figure 7: Scree Plot of LSA model for sample data

Where we can see that it is indicated by the scree plot that the best go-to option is 12 topics. As a result, we run the following command, to create 12 topics out of the documents in hand:

```
lsa_model <- lsa(doc_mat, dim =12)
```

the number of words included per topic can be easily viewed by the following plot.

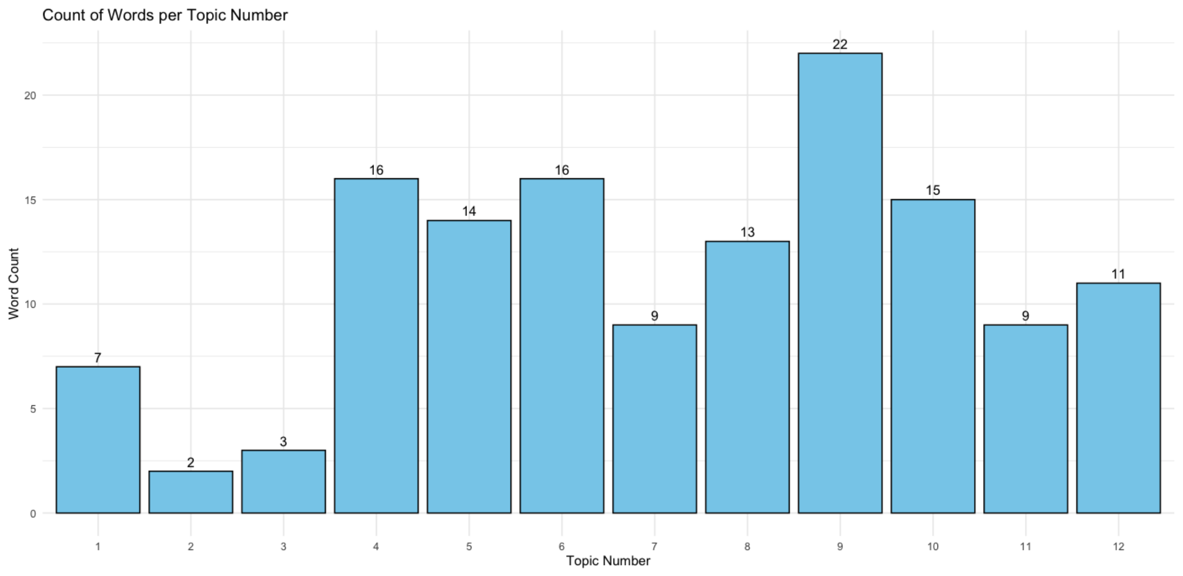

Figure 8: Histogram of number of words per topic

From this point and on there is no limit to what one can achieve given those results. The interest is high for the decomposition tables for one to try and interpret these results. As this is not the method that we want to gather our attention around, we will conclude this example by taking a quick look at the topics generated.

Topic Number	Words
1	time, actions, small, like, heals, romans, reap
2	home, fire
3	mightier, pen, sword
4	mat, chased, afraid, louder, way, will, cloud, every, lining, silver, book, cover, judge, alike, ignorance, calling
5	cat, picture, flies, fun, one, two, birds, feather, flock, together, world, kill, stone, wounds
6	day, indoors, perfect, rainy, reading, staying, worm, makes, practice, stitch, great, think, haste, waste, bliss, merrier
7	dog, early, words, always, grass, greener, side, honesty, policy
8	coffee, fresh, love, morning, smell, sets, sun, west, worth, fair, war, kettle, smoke
9	sat, bird, beauty, beholder, eye, gives, lemonade, lemons, life, angels, make, mind, sight, best, laughter, medicine, blood, thicker, water, counts, thought, spice, variety
10	thousand, speak, come, good, things, wait, nine, beggars, choosers, angels, fear, fools, rush, tread, black
11	away, mouse, ran, apple, doctor, keeps, saves, pot, rome
12	catches, chickens, count, hatch, better, better, late, never, heads, minds, place, sow

7 Selecting number of clusters

In our scenario, we may have prior knowledge about the number of clusters we intend to identify. In other instances, this information may be known or at least inferred. However, what happens when we are uncertain about the optimal number of clusters to aim for? This is where the *ldatuning* package becomes valuable. Specifically, the *ldatuning::FindTopicsNumber* function, when provided with a well-formatted document-term matrix and a range of minimum to maximum number of topics, determines the most suitable number of topics to begin with. This approach employs metrics such as Griffiths[23], Cao Juan[7], Arun[34], and Deveaud[10] to estimate the optimal number of clusters for each case.

In all our cases, prior to fitting the previously described models, we opted to use this tool to assess the accuracy of the results. Let's explore and discuss the challenges that may arise when there is insufficient information on the subject.

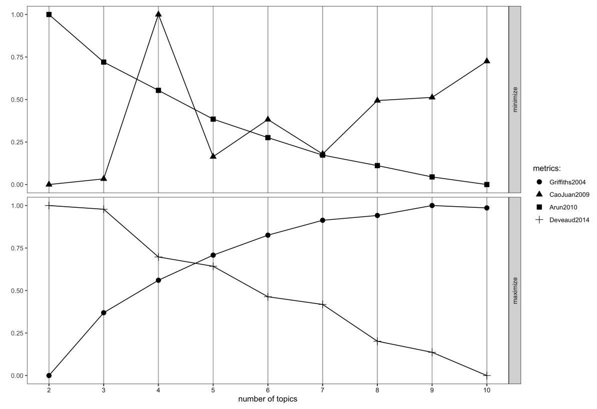

Figure 9: Two-Genre estimations

In the two genre case (10), we can see that Cao Juan minimization and Deveaud maximization methods where able to correctly predict the number of topics that one need to use in an LDA-model fitting procedure. The other two Arun and Griffiths have obtained a monotonic path (decreasing and increasing respectively) which is not suitable in our case.

8 Conclusion

To sum everything up, we have gone through different machine learning methods and gave a description of every one of them. We then introduced Latent Dirichlet Allocation in details, as it is the method we aim to work with. Finally provided LSA as an alternative accompanied with a simple example. Finally we decided that the reader may be interested composing a workflow accompanied with sentiment analysis to extract useful insights and as a result, we provided an introduction on the topic.

We have applied Latent Dirichlet Allocation (LDA), in the context of books' description from various genres, aiming to uncover underlying topics and establish meaningful clusters among them. We have demonstrated the effectiveness of LDA in extracting latent themes from textual data, thereby facilitating a deeper understanding of the inherent structures within a diverse corpus of literature.

We can see that acquiring LDA as tool for textual analysis, can provide us with a meaningful assistance towards uncovering hidden patterns and organizing large volumes of text into coherent clusters. Even in the case that we the accuracy was not as high as the first sub-application (with 4 topics) we can tell that some of the genres were closely related.

Through the process of composing this thesis, many questions have flourished and left to be explored. Looking ahead, an expansion of LDA model to data other than textual seems like an exciting research direction. Another direction would be a more dynamic LDA model. Latent Dirichlet allocation assumes that k is a constant. There have been certain research outbreaks towards that direction, though looking at k with a time correlation perspective could give back fascinating results.

Abstract 1

Evaluation Metrics

1. ROC Curve (Receiver Operating Characteristic Curve):

The ROC curve is a graphical representation of the performance of a classification model across various threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values. The area under the ROC curve (AUC-ROC) is a commonly used metric to assess the overall performance of a binary classification model. A higher AUC-ROC value indicates better discrimination between positive and negative classes, with a value of 1 representing perfect classification performance.

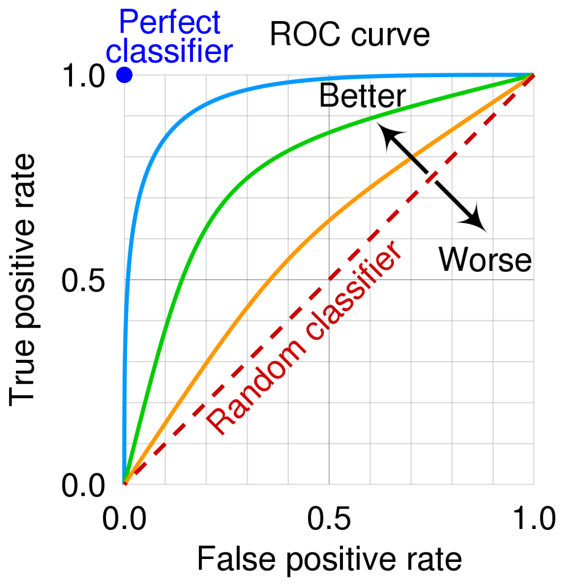

Figure 10: Two-Genre estimations

2. MSE (Mean Squared Error):

MSE is a common metric used to evaluate the performance of regression models. It calculates the average squared difference between the predicted and actual values across all data points. Mathematically, MSE is computed by taking the mean of the squared residuals (the differences between predicted and actual values). MSE penalizes large errors more heavily than smaller errors and is sensitive to outliers.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

where,

n : the number of data points

y_i : the actual value of the target variable for data point i

$\hat{y}_i$: the predicted value of the target variable for data point i

3. RMSE (Root Mean Squared Error):

RMSE is simply the square root of the MSE. It is a popular alternative to MSE as it returns error values in the same units as the target variable, making it easier to interpret. RMSE is particularly useful when the target variable's scale is important for understanding the model's performance.

$$RMSE = \sqrt{MSE}$$

4. MAE (Mean Absolute Error):

MAE is another metric commonly used to evaluate regression models. Unlike MSE, which squares the errors, MAE calculates the average absolute difference between the predicted and actual values. MAE is less sensitive to outliers compared to MSE, making it a robust metric for models where outliers may significantly impact performance.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where,

n : the number of data points

y_i : the actual value of the target variable for data point i

$\hat{y}_i$: the predicted value of the target variable for data point i

5. R-Squared (Coefficient of Determination):

R-squared is a metric used to quantify the goodness of fit of a regression model. It measures the proportion of the variance in the dependent variable that is explained by the independent variables. R-squared values range from 0 to 1, where 0 indicates that the model does not explain any variability in the target variable, and 1 indicates that the model perfectly explains the variability. However, R-squared should be interpreted with caution, as it can be artificially inflated by adding more predictors, even if they are not meaningful.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where,

y_i : the actual value of the target variable for data point i

$\hat{y}_i$: the predicted value of the target variable for data point i

$\bar{y}_i$: mean of the actual values of the target variable

6. Silhouette Score:

The silhouette score is a measure of how similar an object is to its own cluster compared to other clusters. It quantifies the cohesion within clusters and the separation between clusters. The silhouette score ranges from -1 to 1, where a score close to 1 indicates that the object is well-clustered and belongs to its cluster, a score around 0 indicates overlapping clusters, and a score close to -1 indicates that the object may be assigned to the wrong cluster. The average silhouette score across all data points is often used to evaluate the overall quality of a clustering algorithm.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where,

$a(i)$: the average distance from data point i to other points in the same cluster (intra-cluster distance)

$\hat{y}_i$: b(i) = the average distance from data point i to points in the nearest cluster that i is not a part of (inter-cluster distance)

We have to mention in this point that, silhouette score for the entire dataset is the mean silhouette score across all data points.

Confusion Matrix

Confusion matrix is a way to evaluate results obtained from a classification model. A binary classification's confusion matrix is has the following format.

		True	
		Event	**No Event**
Pred	**Event**	A	B
	No Event	C	D

Table 5: Confusion Matrix

The metrics that we are going to use for our model's evaluations are the following:

- Accuracy: Accuracy is the proportion of correctly classified instances over the total number of instances.
  - Accuracy = $\frac{A+B}{A+B+C+D}$
- Precision: Precision answers the question *"Of all the instances classified as positive how many are genuinely positive?"*
  - Precision = $\frac{A}{A+B}$
- Recall: Recall tells us of all the positive instances in the dataset how many did the classifier correctly identified.
  - Recall = $\frac{A}{A+C}$
- F1 Score: It represents a harmonic mean of precision and recall, providing a balance between two metrics.
  - F1 Score = $\frac{Precision*Recall}{Precision+Recall}$

Singular Value Decomposition

Singular Value Decomposition (SVD) is a fundamental matrix factorization technique which is used in various fields including machine learning and more specifically nlp techniques. It decomposes a matrix into a product of three other matrices, revealing important structural properties and latent relationships within the data. Let's take a closer look at SVD though to make sure everything is properly understood:

Given a matrix A of size $m \times n$, the product of the matrices mentioned earlier is constructed of:

- U: Left singular vectors matrix of size $m \times m$.
- Σ: Diagonal matrix of singular values of size $m \times n$.
- V^T (the transposed of matrix V): Right singular vectors matrix of size $n \times n$.

As a result the decomposition is represented as followed (with the aforementioned dimensions $m \times n = (m \times m) \cdot (m \times n) \cdot (n \times n)$):

$$A = U \cdot \Sigma \cdot V^T$$

The singular values in Σ are non-negative and arranged in descending order along the diagonal. These values represent the importance of each singular vector (columns in U and rows in V^T) in capturing the variability of the data. The singular vectors in U and V^T are orthonormal eigenvectors , meaning they are normalized and orthogonal to each other.

By setting SVD to use we are able to reduce the dimentionality by retaining only the top k singular values and their corresponding singular vectors. This reduction we have been talking about, in this subsection, of the original matrix A to a lower-rank approximation preserves the most significant information while we eliminate noise and less important features. Singular Value decomposition can be applied to various types of matrices, including real, complex sparse and rectangular matrices. It is particularly useful in the analysis of high-dimensional and sparse data, as well as in solving systems of linear equations and computing pseudoinverses[11].

[11]The pseudoinverse is a generalization of the inverse matrix that is used to compute the best fit

In order to compute SVD we are able to use various numerical algorithms, some of which are the Golub-Reinsch [22] algorithm, Lanczos or Krylov methods. With these algorithms we are able to efficiently compute the singular values and vectors of the matrix A.

Overall, we can sum everything up by saying that Singular Value Decomposition (SVD) is a powerful matrix factorization technique that decomposes a matrix into its constituent singular values and vectors. It enables dimensionality reduction, noise reduction, and capturing latent relationships within the data, making it a versatile tool in various areas and generally fields.

Cosine or Dot product for similarity comparison

The cosine or dot product is a mathematical operation commonly used in vector spaces, particularly prevalent in the context of information retrieval and natural language processing. It serves as a measure of similarity between two vectors by quantifying the alignment of their directions within the space.

In essence, the cosine or dot product calculates the cosine of the angle formed between two vectors, with a result ranging in the interval $cos(\phi) \in [-1,1]$. A value $\phi = 1$ indicates perfect alignment, implying maximum similarity, while a value $\phi = -1$ signifies complete misalignment, suggesting maximum dissimilarity. A value $\phi = 0$ implies orthogonality, indicating no correlation between the vectors.

Within the realm of information retrieval, this operation plays a pivotal role in assessing the semantic similarity between textual documents or terms represented as vectors. By computing the cosine or dot product between these vectors, we can extract the degree of similarity or relatedness, facilitating tasks such as document retrieval, clustering, and classification.

solution to a system of linear equations which lacks a unique solution. The pseudoinverse of a matrix A is defined as the matrix that 'solves' the least-squares problem $A \cdot x = b$. The most common use of pseudoinverse is in linear regression, where it is used to compute the best fit solution to a system of linear equations that might not have full column rank. The pseudoinverse is computed using the singular value decomposition (SVD) of the matrix A.

Information Retrieval

Information retrieval techniques have undergone significant evolution over the years, aiming to enhance the efficiency and accuracy of accessing relevant information from vast data repositories.[8] As a result we dedicate this part of our analysis to exploring information retrieval, focusing particularly on Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) as pivotal milestones in this evolution due to the needs presented above. Through a historical lens, we delve into the development, principles, and applications of LDA and LSA within the context of information retrieval, highlighting their contributions to shaping contemporary IR practices.

The advent of information retrieval can be traced back to early keyword-based search systems, where documents were indexed and retrieved based on exact matches with user-supplied query terms. However, these systems often suffered from issues such as vocabulary mismatch and lack of semantic understanding, limiting their effectiveness in capturing the nuanced relationships between documents and queries.

Latent Dirichlet Allocation (LDA), introduced by Blei, Ng, and Jordan, revolutionized the field of information retrieval by introducing a probabilistic generative model for representing documents as mixtures of latent topics. LDA assumes that each document is generated by a distribution over topics, and each topic is characterized by a distribution over words. By inferring the underlying topic structure from a corpus, LDA enables the discovery of latent semantic relationships between documents and queries, thus mitigating the limitations of keyword-based approaches.

Latent Semantic Analysis (LSA), developed by Deerwester et al., represents another milestone in the evolution of information retrieval techniques. LSA employs singular value decomposition (SVD) to transform the term-document matrix into a lower-dimensional space, where the latent semantic structure of the corpus is captured. By reducing the dimensionality of the data while preserving its semantic content, LSA facilitates semantic matching between queries and documents, thereby improving retrieval accuracy. Though in this part of our analysis you probably already have read about both LDA and LSA in details earlier.

Both LDA and LSA have found widespread applications in various domains of information retrieval, including document clustering, topic modeling, and recommendation systems. Their ability to uncover hidden semantic relationships and capture the contextual meaning of textual data has fueled advancements in personalized search, content recommendation, and information organization.

While LDA and LSA share the common goal of enhancing information retrieval through semantic analysis, they exhibit distinct characteristics and functionalities. LDA excels in modeling the thematic structure of documents and discovering latent topics, making it well-suited for tasks such as document classification and topic modeling. On the other hand, LSA's strength lies in capturing the global semantic similarity between documents, making it particularly effective for tasks like information retrieval and document summarizing.

In general, Information Retrieval (IR) is the process of obtaining relevant information from a large corpus of data, typically stored in various formats such as text, images, or multimedia.[26] It involves searching, retrieving, and presenting information to users based on their information needs or queries. IR systems are designed to efficiently locate and retrieve relevant documents or resources from a vast collection, enabling users to access the desired information effectively.

This traditional approach involves matching user queries with documents based on exact keyword matches or simple Boolean operators (AND, OR, NOT). Modern search engines employ keyword-based retrieval methods, where documents containing the queried keywords are retrieved and ranked based on relevance.

Some widely known applications of Information Retrieval[39]:

- Web Search Engines: Search engines like Google, Bing, and Yahoo utilize IR techniques to retrieve relevant web pages in response to user queries. Users can enter keywords or phrases, and the search engine returns a ranked list of web pages based on relevance.

- Digital Libraries: IR systems are used in digital libraries to facilitate access to a wide range of scholarly articles, journals, and books. Users can search for specific topics or authors, and the system retrieves relevant documents from the library's collection.

- Enterprise Search: In organizations, IR systems are employed for enterprise search to help employees locate internal documents, emails, and other resources. This aids in knowledge management, collaboration, and decision-making within the organization.
- E-commerce Product Search: Online retail platforms like Amazon and eBay use IR techniques to enable users to search for products based on keywords, descriptions, or attributes. The system retrieves relevant products from the inventory, allowing users to make informed purchase decisions.
- Legal Information Retrieval: IR systems are used in the legal domain to search for relevant case law, statutes, and legal documents. Lawyers and legal professionals can use these systems to conduct legal research and retrieve pertinent information for their cases.

In summary, Information Retrieval encompasses various methodologies for searching and retrieving relevant information from large datasets. From web search engines to enterprise search and digital libraries, IR systems play a crucial role in facilitating efficient access to information across different domains and applications.

Abstract 2

```r
sample_documents <- c(
  "The cat sat on the mat",
  "The dog chased the cat",
  "The mouse ran away from the cat and the dog",
  "The cat is afraid of the dog",
  "A rainy day is perfect for staying indoors and reading",
  "I love the smell of fresh coffee in the morning",
  "The early bird catches the worm",
  "The sun sets in the west",
  "A picture is worth a thousand words",
  "Time flies when you're having fun",
  "Actions speak louder than words",
  "The pen is mightier than the sword",
  "Where there's a will, there's a way",
  "Beauty is in the eye of the beholder",
  "When life gives you lemons, make lemonade",
  "Every cloud has a silver lining",
  "All is fair in love and war",
  "You can't judge a book by its cover",
  "Don't count your chickens before they hatch",
  "The grass is always greener on the other side",
  "Better late than never",
  "Two heads are better than one",
  "Out of sight, out of mind",
  "Practice makes perfect",
  "Honesty is the best policy",
  "Laughter is the best medicine",
  "Actions speak louder than words",
  "All good things come to those who wait",
  "An apple a day keeps the doctor away",
  "A stitch in time saves nine",
  "Beggars can't be choosers",
  "Birds of a feather flock together",
  "Blood is thicker than water",
  "The early bird catches the worm",
  "Every cloud has a silver lining",
  "Fools rush in where angels fear to tread",
  "Great minds think alike",
  "Haste makes waste",
  "Ignorance is bliss",
  "It's a small world",
  "It's the thought that counts",
  "Kill two birds with one stone",
  "The more, the merrier",
  "The pot calling the kettle black",
  "There's no place like home",
  "Time heals all wounds",
  "Variety is the spice of life",
  "When in Rome, do as the Romans do",
  "You reap what you sow",
  "Where there's smoke, there's fire"
)
doc_id <- c(1:50)

docs_df <- data.frame(doc_id,sample_documents)
```

References

[1] *Performance study of n-grams in the analysis of sentiments*, Journal of the Nigerian Society of Physical Sciences, 3 (2021), p. 477–483.

[2] M. N. AGARWAL, BASANT, *Machine Learning Approach for Sentiment Analysis*, Springer International Publishing, 2016, pp. 21–45.

[3] M. AHMAD, S. AFTAB, M. S. BASHIR, AND N. HAMEED, *Sentiment analysis using svm: A systematic literature review*, International Journal of Advanced Computer Science and Applications, 9 (2018).

[4] S. R. AHMAD, A. A. BAKAR, AND M. R. YAAKUB, *A review of feature selection techniques in sentiment analysis*, Intelligent data analysis, 23 (2019), pp. 159–189.

[5] P. BERKA, (2020).

[6] D. M. BLEI, A. Y. NG, AND M. I. JORDAN, *Latent dirichlet allocation*, Journal of machine Learning research, 3 (2003), pp. 993–1022.

[7] J. CAO, T. XIA, J. LI, Y. ZHANG, AND S. TANG, *A density-based method for adaptive lda model selection*, 72 (2009), pp. 1775–1781.

[8] G. CHOWDHURY, *Introduction to Modern Information Retrieval*, Facet Publications, Facet, 2010.

[9] J. K. CULLUM AND R. A. WILLOUGHBY, *Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory*, SIAM, 2002.

[10] R. DEVEAUD, E. SANJUAN, AND P. BELLOT, *Accurate and effective latent concept modeling for ad hoc information retrieval*, 17 (2014), pp. 61–84.

[11] S. T. DUMAIS, *Improving the retrieval of information from external sources*, Behavior Research Methods, Instruments, Computers, 23 (1991), pp. 229–236.

[12] I. EL NAQA AND M. J. MURPHY, *What Is Machine Learning?*, Springer International Publishing, Cham, 2015.

[13] S. B. A. F. Erik Cambria, Dipankar Das, (2017).

[14] N. E. Evangelopoulos, *Latent semantic analysis*, WIREs Cognitive Science, 4 (2013), pp. 683–692.

[15] S. Evert and M. Baroni, *zipfr: Word frequency distributions in r*, in Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, 2007, pp. 29–32.

[16] R. Feldman, *Techniques and applications for sentiment analysis*, Commun. ACM, 56 (2013), p. 82–89.

[17] P. Ficamos, Y. Liu, and W. Chen, *A naive bayes and maximum entropy approach to sentiment analysis: Capturing domain-specific data in weibo*, in 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), IEEE, 2017, pp. 336–339.

[18] P. W. Foltz, *Latent semantic analysis for text-based research*, Behavior Research Methods, Instruments, Computers, 28 (1996), pp. 197–202.

[19] G. E. Forsythe et al., *Computer methods for mathematical computations*, Prentice-hall, 1977.

[20] C. J. S. H. . R. D. Gelman, A., *Bayesian Data Analysis*, Chapman and Hall/CRC, 1995.

[21] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6 (1984), pp. 721–741.

[22] G. H. Golub and C. Reinsch, *Singular Value Decomposition and Least Squares Solutions*, Springer Berlin Heidelberg, 1971.

[23] T. L. Griffiths and M. Steyvers, *Finding scientific topics*, 101 (2004), pp. 5228–5235.

[24] A. R. Isnain, J. Supriyanto, and M. P. Kharisma, *Implementation of k-nearest neighbor (k-nn) algorithm for public sentiment analysis of online*

learning, IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15 (2021), pp. 121–130.

[25] R. E. KASS AND D. STEFFEY, *Approximate bayesian inference in conditionally independent hierarchical models (parametric empirical bayes models)*, Journal of the American Statistical Association, 84 (1989), pp. 717–726.

[26] C. D. MANNING, P. RAGHAVAN, AND H. SCHÜTZE, *Introduction to information retrieval*, Cambridge university press, 2008.

[27] N. METROPOLIS, A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER, AND E. TELLER, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics, 21 (1953), pp. 1087–1092.

[28] C. N. MORRIS, *Parametric empirical bayes inference: Theory and applications*, Journal of the American Statistical Association, 78 (1983), pp. 47–55.

[29] T. MULLEN AND N. COLLIER, *Sentiment analysis using support vector machines with diverse information sources*, in Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 412–418.

[30] H. NANKANI, H. DUTTA, H. SHRIVASTAVA, P. RAMA KRISHNA, D. MAHATA, AND R. R. SHAH, *Multilingual sentiment analysis*, Deep learning-based approaches for sentiment analysis, (2020), pp. 193–236.

[31] I. NTZOUFRAS, *Bayesian Modeling Using WinBUGS*, Wiley Series in Computational Statistics, Wiley, 2011.

[32] H. C. O. E. OJO, A. GELBUKH AND O. O. ADEBANJI, *Senti-n-gram: An n-gram lexicon for sentiment analysis*, Expert Systems with Applications, 103 (2018), pp. 92–105.

[33] I. PORTEOUS, D. NEWMAN, A. IHLER, A. ASUNCION, P. SMYTH, AND M. WELLING, *Fast collapsed gibbs sampling for latent dirichlet allocation*, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2008, p. 569–577.

[34] C. V. M. R. Arun, V. Suresh and M. N. Murty, *On finding the natural number of topics with latent dirichlet allocation: Some observations*, (2010), pp. 391–402.

[35] W. Romsaiyud, *Expectation-maximization algorithm for topic modeling on big data streams*, in 2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), 2016, pp. 1–7.

[36] H. S. Sichel, *On a distribution law for word frequencies*, Journal of the American Statistical Association, 70 (1975), pp. 542–547.

[37] P. P. Surya and B. Subbulakshmi, *Sentimental analysis using naive bayes classifier*, in 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 2019, pp. 1–5.

[38] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, *Lexicon-based methods for sentiment analysis*, Computational Linguistics, 37 (2011), pp. 267–307.

[39] R. Van den Branden, *Introduction to Modern Information Retrieval. Second editionG. G. Chowdhury.*, Literary and Linguistic Computing, 22 (2006), pp. 112–115.

[40] H. Wisnu, M. Afif, and Y. Ruldevyani, *Sentiment analysis on customer satisfaction of digital payment in indonesia: A comparative study using knn and naïve bayes*, in Journal of Physics: Conference Series, vol. 1444, IOP Publishing, 2020, p. 012034.

[41] M. Wongkar and A. Angdresey, *Sentiment analysis using naive bayes algorithm of the data crawler: Twitter*, in 2019 Fourth International Conference on Informatics and Computing (ICIC), 2019, pp. 1–5.

[42] H. F. X. M. Xie Xin, Ge Songlin and J. Nan, *An improved algorithm for sentiment analysis based on maximum entropy*, Soft Computing, 23 (2019), pp. 599–611.