Università degli studi di Milano–Bicocca
Scuola di Economia e Statistica

Corso di Laurea magistrale in
Scienze Statistiche ed Economiche

TOPIC MODELING: AN ANALYSIS OF AIRBNB REVIEWS IN MILAN THROUGH LDA AND SUPERVISED LDA

Relatrice: Prof.ssa Sonia Migliorati
Correlatore: Prof. Roberto Ascari

Tesi di laurea di:
Lorenzo Giudici
Matricola N. 850832

Anno Accademico 2023/2024

Contents

Abstract

This study aims to extract valuable insights, such as topics of interest and dissatisfaction attributes, from a corpus of 132,120 Airbnb reviews related to the city of Milan using various Natural Language Processing (NLP) techniques. By analyzing user feedback, this research seeks to uncover critical aspects of the guest experience that influence satisfaction and dissatisfaction. The application of Latent Dirichlet Allocation (LDA), a widely used topic modeling method, identified 10 topics that can be grouped into four distinct categories: location, unit, management, and evaluation. Additionally, supervised LDA (sLDA) was employed on the same corpus, with a continuous sentiment index obtained through a dictionary-based approach serving as the response variable. The results indicate that topics related to the relationship with the host are strongly associated with user satisfaction, highlighting that in the peer-to-peer accommodation market, greater importance is placed on this aspect compared to traditional hotels. Finally, applying LDA to the 4,887 negative reviews revealed that dissatisfaction is primarily linked to three topics: check-in issues, sleep problems, and inadequacy of the apartment. The findings provide valuable insights for hosts and service providers, enabling them to enhance service quality and improve the guest experience on Airbnb.

Introduction

Since the early 2000s, the widespread use of the internet has led to the proliferation of new types of data, including images and text. Among these, textual data has become particularly abundant, arising from various online interactions such as social media posts, blog entries and, notably, user reviews. This surge in data availability has given rise to the field of *Natural Language Processing* (NLP), which focuses on the automatic processing and analysis of human language by machines.

Reviews play a crucial role across various industries, including the hospitality sector, where consumer feedback directly influences the reputation and success of businesses. Recently, a new sector within the hospitality industry has emerged and rapidly expanded: peer-to-peer accommodation. This model enables individuals to rent out their homes or other properties to travelers, often through digital platforms, creating a more personalized and flexible alternative to traditional hotels. One of the most prominent platforms in this space is Airbnb. Reviews are a vital component of Airbnb's success, as they foster trust between users, help potential guests make informed decisions, and allow hosts to improve their offerings based on feedback.

In this work, a large dataset of Airbnb reviews related to the city of Milan will be analyzed using various NLP techniques to extract valuable information for both customers and service providers. The first chapter presents a brief history of NLP and provides a detailed discussion of *sentiment analysis* (SA) and *topic modeling* (TM) methods, such as Latent Dirichlet Allocation (LDA) and supervised Latent Dirichlet Allocation (sLDA). The second chapter explains the significance of reviews in the peer-to-peer accommodation sector and presents the results of the analysis. The results will be also compared with other studies that have focused on different cities, providing a broader perspective on user feedback across various contexts.

Chapter 1

From NLP to Sentiment Analysis and Topic Modeling

1.1 Natural Language Processing

1.1.1 An introduction to NLP

The expression *Natural Language Processing* (NLP) refers to the set of methods for making human language accessible to computers. A comprehensive definition is given by Liddy (2001).

Definition 1.1. *Natural Language Processing* (NLP) is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts, at one or more levels of linguistic analysis. Its purpose is to achieve human-like language processing for various tasks or applications.

Several elements of Definition 1.1 can be further detailed:

- The imprecise notion of *range of computational techniques* is necessary because multiple methods can be employed for different types of language analysis.
- *Natural occurring texts* can be either written or spoken, as long as they are expressed in a human language and not specifically constructed for the purpose of the analysis.
- The concept of *levels of linguistic analysis* refers to the fact that language processing occurs at multiple levels, and different NLP systems utilize different levels of linguistic analysis.

- *Human-like language processing* indicates that, since it strives for human-like performance, NLP should be considered a discipline within *Artificial Intelligence* (AI).
- Finally, *a range of tasks or applications* points out that NLP is generally not viewed as a goal in itself but rather as a means to accomplish specific tasks.

Although NLP can be considered a branch of AI, it is an interdisciplinary field which draws on many other intellectual traditions, such as *Linguistics*, *Computer Science* and *Machine Learning* (Eisenstein (2018)).

Nowadays, NLP is deeply embedded in our daily lives due to the large amount of text and information generated every day. It can be applied in various areas like Machine Translation, Email Spam Detection, Information Extraction, Summarization and Question Answering. Khurana et al. (2023) briefly discuss each of these areas with the relevant work done in those directions.

1.1.2 Origin and Evolution of NLP

The initial application of computer technology to natural language processing was Machine Translation (MT). In its early stages, this field was based on the assumption that language differences were mainly due to vocabulary and word order variations. As a result, early systems relied on dictionary lookups for translating words and then adjusted the word order to match the rules of the target language (Liddy (2001))

Although research on MT began in the 1940s, it wasn't until two decades later that the use of computers for literary and linguistic studies started to gain traction. Green Jr et al. (1961) introduced one of the earliest *Question-Answering* (Q&A) systems, known as the BASEBALL system. This system was designed to answer questions posed in ordinary English about baseball games, specifically regarding their results, such as "Who won the game between the Yankees and the Red Sox on July 10?". After looking up words and phrases in a dictionary, the system extracted the requested information by matching it with a structured database.

According to Campesato (2021), the several major stages of the evolution of NLP are the following:

- 1950s-1980s: rule-based systems
- 1990s-2000s: corpus-based statistics

- 2000s-2014: machine learning
- 2014-2020: deep learning

Early NLP, extending up to the 1980s, encompassed several decades and primarily focused on rule-based approaches that relied heavily on conditional logic. Both theoretical and practical advancements were notable during this period. Theoretical work concentrated on representing meaning and developing tractable solutions that the then-existing theories of grammar were not able to produce (Liddy (2001)). Moreover, many prototype systems were built, such as ELIZA (Weizenbaum (1966)), which simulated the conversation between a psychologist and a patient, or LUNAR (Woods (1970)), designed to interpret and answer questions about a database of geological information related to the Apollo moon missions. Despite their limitations, these systems showed that NLP was feasible for computers.

The subsequent phase (1990s-2000s) in NLP marked a shift toward primarily statistical analysis of documents collections, while the third phase (2000s-2014) saw the rise of *Machine Learning* (ML), incorporating algorithms such as Decision Trees and Markov Chains (Campesato (2021)). This transition was driven by the increased availability of large volumes of electronic text and more powerful computers with enhanced speed and memory. Statistical approaches became effective in addressing various fundamental problems in computational linguistics, such as part-of-speech (POS) identification and word sense disambiguation, and thus became standard in NLP (Liddy (2001)). Both statistical and ML approaches involve training a model on a training set and then applying it to make predictions on a test set of data.

Finally, the most recent phase of the evolution of *Natural Language Processing* is the result of the combination between NLP and neural networks architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and long short term memory (LSTMs). These neural networks utilize word embeddings - vector-based representations of words - as input and are trained using backpropagation. For instance, the use of CNNs can be observed in tackling problems associated with NLP tasks like sentence classification, sentiment analysis and machine translation (Khurana et al. (2023)).

1.1.3 NLP techniques and components

NLP can be divided into two main components: *Natural Language Understanding* (NLU) and *Natural Language Generation* (NLG), which respectively focus on understanding and generating text (Khurana et al. (2023)). In detail, NLU aims to comprehend human language by determining the context of a text string or document while NLG is the process of producing meaningful sentences from some internal representation. NLU addresses some important NLP tasks that are relevant to this work and will be discussed in detail later, i.e. *sentiment analysis* and *topic modeling*. *Sentiment analysis* (SA) seeks to determine whether the sentiment of a document is positive, neutral or negative whereas *topic modeling* (TM) refers to the determination of the main topics discussed in a document. According to Campesato (2021), these two tasks can be seen as specific instances of text classification: in the first case the labels refers to the sentiment of the text (positive, negative or neutral) while in the latter the classes are the major topics of the document. However, some *sentiment analysis* techniques take into account also the strength of the polarity and some *topic models* are based on the assumption that a document can exhibit more than a single topic.

In general, depending on which task you have to perform, different NLP techniques have been developed. Nevertheless, there are some common steps which are usually applied regardless of the specific purpose of the analysis.

Text pre-processing

Data pre-processing plays a vital role in cleaning up unwanted words, characters or punctuation that are not useful for machine interpretation. Consequently, NLP defines several techniques that can be selected based on the specific use case. The most important ones are listed below (Tabassum & Patil (2020)):

1. Change to Lowercase
2. Sentence Segmentation
3. Tokenization
4. Parts-of-speech Tagging
5. Stopwords removal
6. Stemming

7. Lemmatization

Before performing any type of NLP task, it is useful to convert all the text to *lowercase*. This prevents the computer from treating words that differ only in capitalization (e.g., "Italy" and "italy") as separate elements.

Sentence segmentation refers to the process of breaking a text document into individual sentences. This technique is also known as *sentence boundary detection*, because it requires identifying word boundaries such as full stops or other terminal punctuation marks. Once a text is segmented into sentences, *tokenization* follows, which entails splitting sentences into individual words or *tokens*. For example, the sentence "Rome is the capital of Italy" is tokenized into ("Rome", "is", "the", "capital", "of", "Italy"). The splitting is done when a space or a separating punctuation mark, such as comma, colon or semicolon, is found.

Parts-of-speech Tagging (or POS Tagging) is performed when each word is classified (or tagged) according to its grammatical category, such as noun, verb or adjective. This process aids in understanding the role each word plays in a sentence.

Moreover, in any text there are lots of words, the so-called *stopwords*, which are frequently used but don't add significant meaning to the content. These are mainly articles ("the", "a", "an"), prepositions ("on", "in", "at"), conjunctions ("and", "or", "but"), pronouns ("he", "she", "we") and also auxiliary verbs ("is", "are", "been"). Considering again the sentence "Rome is the capital of Italy", it is easy to understand that the words "Rome", "capital" and "Italy" are sufficient to grasp the meaning of the phrase, while "is", "the" and "of" are unnecessary. Therefore, it's a good practice to remove all these *stopwords* from the text.

Lastly, it's often beneficial to reduce words to their base form, which can be done through either *stemming* or *lemmatization*. The first is an aggressive word shortening technique that chops off suffixes. For example, the stemmed form of both "running" and "runner" is "run". Stemming algorithms are typically fast but can occasionally produce nonsensical or incorrect words ("caring" may be reduced to "car"). In constrast, lemmatization is a more refined process that removes or replaces the word's suffix to revert it to its base form, known as *lemma*, which is always a meaningful word. For instance, applying this technique to the word "ran" will yield the base form "run". Lemmatization takes into account the word's meaning and grammatical role, making it slower than stemming but more accurate.

It's crucial to understand that the order in which this steps are implemented is highly significant. Performing the same pre-processing steps in a different sequence can lead to varying results or even introduce errors into the analysis.

Feature Extraction

Feature Extraction refers to the representation of the text in vector forms that can be used by ML algorithms. Various tecnhiques are commonly used, each with its own strengths and suited for different applications. Below are some of the most commonly used methods. (Campesato (2021)):

1. Bag-of-words Model (BoW)
2. N-grams
3. tf-idf
4. Word Embeddings

The *Bag of Words* (BoW) model treats a text as an unordered multiset of words. Using a dictionary of unique words from the entire corpus, BoW creates an array where each element represents the number of occurrences of a word in the document. This straightforward approach allows for a simple way to quantify word frequency.

Example 1.1. *Suppose that a corpus is composed of two documents, Doc1 and Doc2, defined as:*

- *Doc1 = "Rome is the capital of Italy"*
- *Doc2 = "All roads lead to Rome"*

The vocabulary of the unique words that appear in the corpus is

["Rome", "is", "the", "capital", "of", "Italy", "All", "roads", "lead", "to"]

Given the vocabulary, the two documents can be then represented as:

- *Doc1 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]*
- *Doc2 = [1, 0, 0, 0, 0, 0, 1, 1, 1, 1]*

where the i-th element of the vector stands for the frequency of the i-th word in the vocabulary

However, the BoW model disregards word order, and prioritizing frequently occurring words is not always effective in classification or clustering tasks (Tabassum & Patil (2020)). Additionally, this method fails to capture the semantic relationships between words.

To address this limitation, an extension of the BoW model called the *N-gram* technique can be applied. This method considers sequences of N adjacent words rather than individual ones, allowing it to capture local context and improve the understanding of word relationships within the text.

Example 1.2. *Considering the same two documents of Example 1.1 and considering a 2-gram (or bi-gram), the new vocabulary is:*

[*("Rome", "is"), ("is", "the"), ("the", "capital"), ("capital", "of"), ("of", "Italy"), ("All", "roads"), ("roads", "lead"), ("lead", "to"), ("to", "Rome")*]

Each document can be so represented as

- *Doc1* = [1, 1, 1, 1, 1, 0, 0, 0, 0]
- *Doc2* = [0, 0, 0, 0, 0, 1, 1, 1, 1]

where the i*-th element of the vector stands for the* i*-th pair of words in the vocabulary*

Although *N-grams* capture some relationships between words and can be used to predict possible missing words by calculating probabilities, , they may fail to account for long-range dependencies. Additionally, using higher values of N can result in very sparse representations.

As an additional improvement to the BoW algorithm, the tf-idf scheme can be considered. This method determines the elements of the vector representing each document by considering both the frequency of a given word within the document (*term frequency*, tf) and the number of documents in the entire corpus that contain that specific word (*inverse document frequency*, idf).

In detail, the *term frequency* (tf) of a word is calculated as the number of times that the word appears in the document, eventually dividing by the length of the document as a normalization technique.

Example 1.3. *Considering the same two documents of Example 1.1, the term frequencies (tf) of "is" and "Rome" are:*

- *tf("is")* = $\frac{1}{6}$ *(in Doc1)*

- *tf("is") = 0 (in Doc2)*
- *tf("Rome") =* $\frac{1}{6}$ *(in Doc1)*
- *tf("Rome") =* $\frac{1}{5}$ *(in Doc2)*

On the other hand, the *inverse document frequency* (idf) measures the importance of a word across the entire corpus. It is computed as the logarithm of the ratio of the total number of documents to the number of documents containing that word. A word that appears in fewer documents is considered more significant for identifying the content of a specific document.

Example 1.4. *Considering the same two documents of Example 1.1, the inverse document frequencies (idf) values of "is" and "Rome" are*

- *idf("is")* $= \log(\frac{2}{1}) = \log(2) = 0.69$
- *idf("Rome")* $= \log(\frac{2}{2}) = \log(1) = 0$

The word "Rome" appears in all the documents of this small corpus and so it has no importance. On the contrary, "is" has a positive idf value.

The tf-idf value is simply the product of the tf value and the idf value, representing the relative importance of the word within a document relative to the entire corpus. While this scheme is highly useful for determining word significance, it still overlooks semantic context and can be less effective for tasks like matching phrases across documents.

Example 1.5. *The tf-idf values of the words "is" and "Rome" in the corpus are:*

- *tf-idf("is")* $= \frac{1}{6} \times \log(2) = 0.12$ *(in Doc1)*
- *tf-idf("is")* $= 0$ *(in Doc2)*
- *tf-idf("Rome")* $= \frac{1}{6} \times \log(1) = 0$ *(in Doc1)*
- *tf-idf("Rome")* $= \frac{1}{5} \times \log(1) = 0$ *(in Doc2)*

The two documents can be then represented as vectors in which the elements are not just word frequencies (as in Example 1.1), but rather the tf-idf values:

- *Doc1 =* [***0***, ***0.12***, *..., ..., ..., ..., 0, 0, 0, 0*]

- *Doc2* = [***o***, *o*, *o*, *o*, *o*, *o*, ..., ..., ..., ...]

Although the BoW and the tf-idf scheme are very useful and applied techniques, they often create very large and sparse vector for words. A more effective approach are the so-called *word embeddings*, which provide real-valued word representations in a multi-dimensional space. As well as reducing dimensions, they can capture the semantic meaning of words in the sense that words with similar meanings are located close to each other in the vector space. In practice, each word in a text is converted into its corresponding vector using a pre-trained embedding model such as Word2Vec (Mikolov et al. (2013)) or GloVe (Pennington et al. (2014)). These vector representations serve as input features for many NLP tasks and methods. Despite their advantages, *word embeddings* still have some limitations, such as conflating multiple senses of a word into a single vector. For a comprehensive review of this field and its challenges, see Bakarov (2018).

1.2 Sentiment Analysis

1.2.1 An introduction to Sentiment Analysis

Sentiment Analysis (SA), also known as opinion mining, is a field of NLP that focuses on extracting and analyzing emotions and opinions contained in a text. Its goal is to determine the emotional attitude of the author and assess whether the input text expresses a positive, negative, or neutral sentiment.

Although research in linguistics and NLP has a long history, studies on opinions and sentiments only began in the early 2000s. According to Liu (2022), this shift occurred due to the vast range of applications, especially in the commercial domain, combined with the growing availability of opinionated data on the web, which fueled a strong interest in SA research. Opinions now play a central role in almost all human activities: businesses and organizations want to understand what customers think about their products, and individual consumers often seek out other users' opinions before making a purchase. Today, such information is easily accessible through social media data (e.g., reviews, forum discussions, blogs), making traditional surveys less necessary for companies and direct inquiries to friends redundant for consumers. However, what is essential is the availability of specific tools that can automatically detect sentiments from the vast amount of data available

SA can be explored at three different levels (Liu (2022)):

- document level
- sentence level
- entity and aspect level

At the document level, the task is to classify whether an entire opinion document expresses a positive or negative sentiment. The underlying assumption is that each document reflects opinions on a single entity (e.g., a specific product). This type of analysis is not widely used, as longer texts often contain multiple opinions.

At the sentence level, each sentence is analyzed individually to determine whether it expresses a positive, negative, or neutral sentiment. The results can then be aggregated to estimate the overall polarity of the document. Nevertheless, a single sentence can also convey multiple sentiments. Thus, both document-level

and sentence-level analyses have limitations, and in some cases, a more granular approach is required.

Aspect-level SA, also referred to as feature-level analysis, does not focus on language constructs like paragraphs or sentences, but rather on the opinion itself. The key idea is that each opinion consists of both a *sentiment* (positive or negative) and a *target* (the object of the opinion). In many applications, opinion targets are described by entities and their various aspects. For example, an entity could be a restaurant, with aspects such as food quality, service, or location. The goal of this level of analysis is to identify *what* people like and dislike, that is, to detect sentiments associated with different aspects of the same entity. For instance, in the sentence *"The laptops performance is impressive, but its battery life is short"*, two aspects (performance and battery) are evaluated for the same entity (laptop). In the first case, the sentiment is positive, while in the second, the customer's opinion is negative. As one might expect, identifying the different aspects within a text is not straightforward, making aspect-level sentiment analysis significantly more challenging than document-level or sentence-level analysis.

1.2.2 Methods for Sentiment Analysis

Regardless of the level at which it is performed, there are two main categories of *sentiment analysis* methods (Taboada (2016)):

- rule-based methods
- machine learning methods

Rule-based SA approaches, often referred to as lexicon-based or dictionary-based methods, rely on a predefined lexicon, where each *token* (usually a word) s assigned a score indicating positivity, negativity, or neutrality. The overall polarity of a given text is determined by aggregating the scores of individual tokens. Since no training data is required, this approach is considered unsupervised. The main disadvantage, however, is that the context or domain-specific meaning of words is not considered. For instance, a word like "small" may convey a positive meaning in one sentence and a negative one in another, but this variation is not captured by rule-based methods (Wankhade et al. (2022)).

On the other hand, the machine learning approach involves training a classifier on a labeled dataset, and then using this classifier to determine the sentiment of a new corpus. This is a supervised technique that offers clear advantages but also

has some drawbacks. While training a classifier can be relatively straightforward (given a set of documents labeled as positive or negative), it may result in poor accuracy if the training set is too specific and not generalizable to new text sources. Consequently, applying sentiment analysis to a new domain often requires training a new model on new data (if available), which can be time-consuming (Taboada (2016)).

In the following sections each of these areas will be discussed, starting from ML methods, and relevant work in both approaches will be presented. It is worth noting, however, that the distinction between rule-based and machine learning methods is not always clear-cut, and several hybrid approaches have been proposed in recent literature (Wankhade et al. (2022)).

Machine-Learning Approaches

As mentioned earlier, SA can be considered a standard problem in text classification, where the labels typically consist of two (positive and negative) or three categories (including neutral). Consequently, each of the main ML classification techniques can theoretically be applied. Syntactic and linguistic factors are utilized to construct the classifier, which is then used to predict the sentiment of instances with unknown classes (Wankhade et al. (2022)).

The first method to consider is the Naive Bayes (NB) classifier, which employs Bayes' theorem to predict the probability of a given set of features belonging to a specific label. It is sometimes referred to as a soft classification technique, because it provides not only the estimated true class but also the probabilities for all possible classes. This method is generally applied when the size of the training data is small. An improved version of the NB classifier was proposed and tested on restaurant reviews by Kang et al. (2012).

Support Vectors Machines (SVM) represent a non-probabilistic supervised learning technique that seeks to determine the hyper-plane that best separates data into distinct classes. Key studies using SVM for sentiment analysis include those byLi & Li (2013), Borg & Boldt (2020) and Wang et al. (2020).

Logistic Regression (LR) is sometimes employed in binary classification problems. It identifies which input features are most effective for distinguishing between positive and negative classes and uses Maximum Likelihood estimation to calculate the optimal parameters. Hamdan et al. (2015) applied LR in a sentiment analysis context.

Moving to Decision Trees (DT), this technique recursively divides data into different subsets, each assigned a specific label. DTs are often used in opinion mining, alongside Random Forests (RF), which are essentially combinations of multiple decision trees. Yan-Yan et al. (2010) applied this technique in the camera domain, proposing a strategy to integrate sentence-level features and comparing their results with those of NB and SVM classifiers.

Finally, there are semi-supervised approaches that utilize training datasets composed of both labeled and unlabeled instances. For example, Janjua et al. (2021) combined preprocessing and classification techniques.

Rule-Based Approaches

Despite the appeal of all these ML techniques, lexicon-based models in SA are proven to be most robust across different domains without the need to modify the dictionaries (Taboada et al. (2011)). Moreover, porting dictionaries to a new language or domain is not an onerous task (Brooke et al. (2009)). Therefore, lexicon-based approaches are sometimes preferred because they demonstrate the best performance in terms of the trade-off between accuracy and computational complexity.

The lexicon-based approach consists of three different steps (Taboada (2016)):

- Identifying relevant words and phrases
- Identifying relevant sentences
- Aggregating the individual words or phrases extracted

The most significant words conveying subjectivity in a text are adjectives, and considerable effort has been devoted to extract semantic orientation from them. However, researchers have increasingly recognized that sentiment is also conveyed through other parts of speech, such as nouns, verbs, adverbs, and phrases that containing these words (Benamara et al. (2012)). Generally, sentiment dictionaries list positive and negative words, specifying the polarity for each. Additionally, many lexicon-based approaches include information about the strength of polarity, indicating how positive or negative a word is. Notable sentiment dictionaries include SentiWordNet, which contains 38.000 words (Baccianella et al. (2010)) and the Macquarie Semantic Orientation Lexicon, which has nearly 76.000 words (Mohammad et al. (2009)). It is important to note that the intensity of a sentiment can be augmented or diminished by specific adjectives or adverbs, such as "most",

"really" or "arguably". Furthermore, certain words are used to negate sentiment, such as "not", "never" and "without", while entire phrases, like "it's amazing that", can also convey sentiments. Considering these aspects is essential for enhancing the performance of sentiment analysis systems (Taboada (2016)).

Not all parts of a text contribute equally to the overall opinion expressed. For example, a movie review might include sections related to other films by the same director or featuring the same actors, which may have little bearing on the author's opinion of the movie in question. Topic-detection methods can be employed to identify these irrelevant sections (Taboada (2016)). Another challenge is distinguishing less relevant aspects or objective information in a review, such as plot summaries or product descriptions. Various methods have been proposed to differentiate between subjective and objective sentences or paragraphs (Wiebe & Riloff (2005), Taboada et al. (2009)). Additionally, some text segments summarize or encapsulate the overall opinion, and adjectives located in different parts of a review can carry varying weights. According to the theory that evaluation tends to occur at the boundary points of discourse, some authors have developed sentiment analysis methods that assign greater weight to words appearing towards the end of the text (Taboada & Grieve (2004)).

After extracting words and phrases from a text, potentially using a pruning method on sentences, the final step is to aggregate the different semantic orientations. The most commonly used method is averaging (Turney (2002)). However, this technique often falls short in cases where discourse structure significantly influences argument construction. Specifically, the use of concession relations ("even if...", "although...") and condition relations ("unless...", "as long as...") can alter the polarity of sentiment words. Recent years have seen new proposals and significant advancements in this area (Feng (2015), Joty et al. (2015)).

1.2.3 A rule-based sentence-level method

The aim of this section is to present the sentiment analysis utilized method utilized by the R package *sentimentr* (Rinker (2021)). This rule-based method calculates text polarity at the sentence level in the English language and will be applied in the next chapter to the Airbnb reviews domain. The choice of this specific technique is informed by the work of Ding et al. (2021), who successfully employed this method in a similar context.

Consider a document D composed of different paragraphs p_i, $i = 1, 2, ..., N$. Each paragraph is composed of J_i sentences, $p_i = \{s_1, s_2, ..., s_j, ..., s_{J_i}\}$, and each sentence is composed of K_{ij} words, $s_{i,j} = \{w_1, w_2, ..., w_k, ..., w_{K_{ij}}\}$. The notation $w_{i,j,k}$ indicates the $k-th$ word of the $j-th$ sentence in the $i-th$ paragraph.

As mentioned previously, this analysis operates at the sentence level, meaning each sentence $s_{i,j}$ is considered and subsequently transformed into an ordered bag of words. Punctuation is removed, with the exception of pause punctuation (commas, colons and semicolons), which are classified as pause words (or comma words, cw). The words $w_{i,j,k}$ are then searched and compared against a pre-defined dictionary of polarized words. Positive words ($w^+_{i,j,k}$) are tagged with a +1, while negative words ($w^-_{i,j,k}$) receive a -1. Each polarized words obtained, denoted as $pw_{i,j,k}$, is aggregated with other words to create a polar cluster, $c_{i,j,l}$, where $l = 1, ..., L_{i,j}$ and $L_{i,j}$ represents the number of polarized word in the sentence $s_{i,j}$. This cluster is formed considering a fixed number of items before (n_b) and after (n_a) the polarized word $pw_{i,j,k}$. In mathematical terms:

$$c_{i,j,l} = \{w_{i,j,k-n_b}, ..., pw_{i,j,k}, ..., w_{i,j,k+n_a}\} \tag{1.1}$$

Pause locations (cw) are taken into account when calculating the limits of the cluster, as these marks indicate a change in thought, suggesting that there is no connection between the words before and after them. Additionally, each cluster $c_{i,j,l}$ is a subset of the sentence $s_{i,j}$, meaning that words from other sentences should not be included. Consequently, the polarized cluster can be further restricted, leading to the following definition:

$$c_{i,j,l} = \{w_{i,j,k-n^*_b}, ..., pw_{i,j,k}, ..., w_{i,j,k+n^*_a}\} \tag{1.2}$$

where

$$n^*_b = \min\{n_b, k-1, \min\{u \in \{1, 2, ..., n_b - 1\} \mid w_{i,j,k-u} = cw_{i,j,k-u}\}\} \tag{1.3}$$

and

$$n^*_a = \min\{n_a, K_{i,j} - k, \min\{u \in \{1, 2, ..., n_a - 1\} \mid w_{i,j,k+u} = cw_{i,j,k+u}\}\} \tag{1.4}$$

Some examples can help to clarify how the polarized cluster is formed.

Example 1.6. *Consider the sentence* $s_{i,j}$

"We had a nice holiday in Hotel Michelangelo last August"

In this case, there are no punctuation marks or pause words present. According to a predefined polarized dictionary, the word "nice" *is classified as a polarized positive word. Assuming* n_b=n_a=4*, the polarized cluster* $c_{i,j,1}$ *(the only cluster in this sentence, as there is only one polarized word) is formed as follows:*

$c_{i,j,1} = \{$"we", "had", "a", "**nice**", "holiday", "in", "hotel", "michelangelo"$\}$

It's important to note that only $n_b^* = 3$ *words before the polarized word, which is the* 4*-th word in the sentence (*$pw_{i,j,4}$*) were considered, because*

$$\min\{n_b, k-1\} = \min\{4, 3\} = 3$$

Thus, the cluster correctly captures the relevant context surrounding the polarized word "nice".

Example 1.7. *Consider the sentence* $s_{i,j}$

"Hotel Michelangelo is very nice, the host unpleasant"

In this case, two polarized words are identified: "nice"*, which is positive, and* "unpleasant"*, which is negative. Consequently, two different polarized clusters,* $c_{i,j,1}$ *and* $c_{i,j,2}$ *can be formed. Additionally, the punctuation mark "," is classified as a pause word (*cw*). The two clusters are:*

$c_{i,j,1} = \{$"hotel", "michelangelo", "is", "very", "**nice**", ",", "$\}$

$c_{i,j,2} = \{$",", "the", "host", "**unpleasant**"$\}$

The pause word "," limits the upper bound of the first cluster and the lower bound of the second one.

Specifically, for the first cluster

$$n_a^* = \min\{4, 9-5, 1\} = 1$$

where $n_a = 4$ *is the default value,* $K_{i,j} = 9$ *is the number of elements in the sentence,* $k = 5$ *is the position of the polarized word in the sentence and* 1 *is the index of the fist element after* $pw_{i,j,5}$ *which is a comma word.*

On the other side,

$$n_b^* = \min\{4, 9-1, 3\} = 3$$

where $n_b = 4$ *is the default value,* $k = 9$ *is the position of the polarized word in the sentence and* 3 *is the index of the first element before* $pw_{i,j,9}$ *which is a comma word.*

Regarding the second cluster $c_{i,j,2}$*, the upper bound is calculated as*

$$n_a^* = \min\{4, 9 - 9\} = 0$$

Since the polarized word "unpleasant" *is the last word in the sentence, no words can be considered after it.*

Each word in each polarized context cluster $c_{i,j,l}$ is tagged as neutral ($w^0_{i,j,k}$), negator ($w^n_{i,j,k}$), amplifier ($w^a_{i,j,k}$) or downtoner ($w^d_{i,j,k}$). The polarized "core" word of the cluster, $pw_{i,j,k}$, is weighted according to its specific value on the dictionary and re-weighted based on the function and number of valence shifters within the cluster. An amplifier ($w^a_{i,j,k}$) increases the polarity of $pw_{i,j,k}$, while a downtoner ($w^d_{i,j,k}$), reduces it.

Example 1.8. *Consider the following sentences*

"Hotel Michelangelo is nice"

"Hotel Michelangelo is very nice"

"Hotel Michelangelo is quite nice"

In all the sentences the polarized word "nice" *is identified as positive. However, the intensity of the sentiment varies due to the presence of valence shifters:*

- *in the second sentence, the word* "very" *is an amplifier that increases the positiveness of* "nice"
- *in the third sentence, the word* "quite" *is a downtoner that reduces the positiveness of* "nice"

Neutral words ($w^0_{i,j,k}$) have no effect; they only influence the word count when forming the cluster. Negation words ($w^n_{i,j,k}$) are handled as follows: if an odd number of them is found, an amplifier is converted to a downtoner and vice versa. This is based on the principle that two negatives cancel out to form a positive, while three negatives result in a negative, and so forth.

Example 1.9. *Consider the following sentences*

"Hotel Michelangelo is nice"

"Hotel Michelangelo is not nice"

"We can't say that Hotel Michelangelo is not nice"

In all the sentences the polarized word is "nice"*, which is clearly positive. In the second sentence, the word* "not" *acts as a negator, causing the polarity to shift to negative. In the*

third sentence, however, there are two negators: "not" *as part of* "can't" *and* "not" *before the polarized word. As a result, the polarity remains positive due to the two negations cancelling each other out.*

Finally, adversative conjunctions (e.g. "but", "however", "although") are also used to adjust the weight of the context cluster. If an adversative conjunction appears before a polarized word, it up-weights the cluster; otherwise, it down-weights it.

Example 1.10. *Consider the following sentences*

"The hotel was nice but far away from the city centre"

"Although the distance the hotel was nice"

The polarized word considered is "nice"*, which is positive. After creating the polarized cluster* $c_{i,j,l}$*, it is observed that in the first sentence an adversative conjunction appears after the polarized word, downtoning the sentiment polarity. In this case, the presence of* "but" *reduces the perceived positivity of the hotel. Conversely, in the second sentence, the adversative conjunction* "although" *appears before the polarized word, leading to an up-weighting of the polarity.*

A polarity index for each sentence (δ), which is unbounded, is obtained as the ratio between the sum of the weighted context clusters, ($c'_{i,j}$) and the square root of the word count in that sentence ($\sqrt{K_{i,j}}$), i.e.

$$\delta = \frac{c'_{i,j}}{\sqrt{K_{i,j}}} \tag{1.5}$$

In detail, the numerator of the equation 1.5 is:

$$c'_{i,j} = \sum_{l=1}^{L_{i,j}} \left((1 + w_{amp} + w_{deamp}) \cdot pw^{p}_{i,j,l} \cdot (-1)^{2+w_{neg}} \right) \tag{1.6}$$

where

- $pw^{p}_{i,j,l}$ is the weight associated to the l-th polarity word, which is the "core" of the l-th cluster
- w_{neg} is calculated as the number of negator words ($w^{n}_{i,j,k}$). If this is an odd number, then the power is equal to -1, reversing the polarity

- w_{amp} and w_{deamp} are calculated according to the number of amplifiers ($w^{a}_{i,j,k}$) and downtoners ($w^{d}_{i,j,k}$), also taking into account the presence of adversative conjunctions.

An important point of this method is that it does not simply classify a sentence as positive, negative, or neutral but rather returns a numeric and continuous index of polarity. A positive index indicates an overall positive sentiment in the text, while a negative index reflects a negative sentiment. A value closer to zero indicates weaker polarity.

The polarity obtained naturally depends on the dictionary used. A version proposed by the author combines Jockers (2017)'s version (in the *syuzhet* package with an augmented Hu & Liu (2004) dictionary (in the *lexicon* package). However, users are encouraged to expand this dictionary by adding specific terms or expressions relevant to their area of application.

1.3 Topic Modeling

1.3.1 An introduction to Topic Modeling

Topic models are statistical techniques used to uncover latent themes within a collection of documents. Given a large set of text data, the availability of automatic tools that comprehend underlying topics is invaluable, as they enable the compression of the corpus into a concise summary. *Topic Modeling* (TM) can thus be considered a methodology that present a substantial volume of data in a low dimension while elucidating hidden concepts and latent variables.

Topic models can be classified into two categories (Kherwa & Bansal (2019)):

- *probabilistic* models
- *non-probabilistic* models

Non-probabilistic approaches emerged in the early 1990s, performing dimension reduction from an algebraic perspective. In Deerwester et al. (1990), *Latent Semantic Indexing* (LSI) was introduced, demonstrating how to use Latent Semantic Analysis to automatically index and retrieve documents from extensive databases. Based on the BoW assumption, this technique employs singular value decomposition (SVD) to the tf-idf matrix, identifying a linear subspace that captures most of the variance in the collection, thus reducing the dimensionality on the document side of the matrix while retaining the meaningfulness of the words. In this manner, LSI generated the first topic sets from documents (Churchill & Singh (2022)). Another *non-probabilistic* technique for *topic modeling* is the *Non-Negative Matrix Factorization* (NNMF), which factorizes a matrix of non-negative values into two new matrices such that their product equals the original matrix. When applied to the document-term matrix of a corpus, this method produces a topic-word matrix and a topic-document matrix. The latter matrix can be interpreted as a topic distribution over documents, thereby facilitating an understanding of the topics associated with each document (Churchill & Singh (2022)).

Conversely, *probabilistic topic models* were developed to enhance algebraic models like LSI by incorporating probabilistic elements through generative model approaches (Kherwa & Bansal (2019)). *Probabilistic LSI* (pLSI), proposed by Hofmann (1999), models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables representing topics. A primary limitation of pLSI is the absence of a generative

model at the document level. To address this, *Latent Dirichlet Allocation* (LDA) was proposed, likely the most recognized approach in TM. This method, which has served as a springboard for many other *topic models*, will be discussed in detail in the subsequent section.

All the methods described thus far are *unsupervised topic models*, as they operate with unlabeled documents, aiming solely to identify hidden structures without prior information regarding the topics. If each document is associated with a response variable, a model can be estimated to learn the relationship between document topics and the variable itself. Consequently, given a new unlabeled document, this model can infer its topic structure and facilitate predictions. Such models are termed *supervised topic models*. The most notable of these is an extension of LDA called *supervised* LDA (sLDA) and will be investigated later.

1.3.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a hierarchical Bayesian model consisting of three levels, designed for collections of discrete data, such as text corpora. This model was first introduced by Blei et al. (2003). The fundamental premise of LDA is that documents are random mixtures over K latent topics, with each topic represented as a distribution over words. It is assumed that each document exhibits these topics in varying proportions (Blei & Lafferty (2009)).

The *three levels* of the model are as follows:

- **document level**: each document is viewed as a distribution over topics
- **topic level**: each topic is characterized by a distribution over words
- **word level**: words in documents are generated by topics

Moreover, the model is *Bayesian* because its parameters are trated as random variables with specific probability distributions (known as prior distributions). LDA is also classified as a *generative* model, as it delineates the process by which documents may have been generated from hidden variables. Specifically, each term in a document is presumed to arise from a specific process: a topic is first selected, and subsequently, a word is drawn conditionally from that topic.

Since only words are observed in reality, the objective is to infer the hidden structure of the model, namely the topics themselves and the extent to which each document embodies these topics. This succinct description of each element

within the collection can be beneficial for fundamental tasks such as classification, novelty detection, and summarization.

It is important to note that, although LDA was initially developed and is predominantly applied to model text corpora, it can also be utilized with various types of data, such as in bioinformatics or finance.

The generative model

Before explaining the generative process in detail, some basic notation and terminology are introduced:

- the corpus $\mathcal{D}$ is a collection of $D \in \mathbb{N}^+$ documents, each composed of $N_d \in \mathbb{N}^+$ elements, where $d = 1, ..., D$. Each element essentially represents a pair *topic-word*
- the set of topics $\mathcal{K}$ consists of $K \in \mathbb{N}^+$ elements
- the dictionary $\mathcal{V}$ is a set of $V \in \mathbb{N}^+$ unique words
- $t_{d,n} = k \in [1, K]$ denotes the topic of the n-th element of the d-th document. Using vector notation, $t_{d,n}$ can be represented as a K-dimensional vector, where the k-th element is one and all others are zero.
- $w_{d,n} = v \in [1, V]$ stands for the n-th word of the d-th document. Using vector notation, $w_{d,n}$ can be represented as a V-dimensional vector, where the v-th element is one and all others are zero.
- the random vector $\theta_d \in \mathcal{S}_K$, where $d = 1, .., D$ and $\mathcal{S}_K$ is the K-simplex, indicates the mixing proportions of topics in the d-th document. Specifically, $\theta_{d,k}$ denotes the proportion of topic k in the d-th document, with the constraint $\forall d \in \mathcal{D}, \sum_k \theta_{d,k} = 1$ (the proportions of topics in a document sum to one)
- the random vector $\phi_k \in \mathcal{S}_V$, where $k = 1, ..., K$ and $\mathcal{S}_V$ is the V-simplex, indicates the distribution of words in the k-th topic. Specifically, $\phi_{k,v}$ represents the probability of observing the v-th word given topic k, with the constraint $\forall k \in \mathcal{K}, \sum_v \phi_{k,v} = 1$ (the proportions of words in a topic sum to one)
- $\alpha \in \mathbb{R}_+^K$ and $\beta \in \mathbb{R}_+^V$ are the hyperparameters of the distributions of topics in documents and words in vocabulary, respectively.

Latent Dirichlet Allocation (LDA) relies on three primary assumptions that, while not entirely realistic, facilitate the development of a tractable model and have demonstrated effectiveness in practice:

- *Bag of words*: words are exchangeable in documents
- *Bag of documents*: documents are exchangeable in the corpus
- *Fixed number of topics*: the number of topics is a fixed parameter, that must be chosen *a priori*

The three main steps of this generative model, which is represented in Figure 1.1, are as follows:

1. for each topic $k \in \mathcal{K}$, the distribution of words over the vocabulary $\mathcal{V}$, ϕ_k, is drawn from a Dirichlet distribution with parameter β,

$$\phi_k \sim \mathrm{Dir}(\beta)$$

2. for each document $d \in \mathcal{D}$, a topic distribution θ_d is drawn from a Dirichlet distribution with parameter α,

$$\theta_d \sim \mathrm{Dir}(\alpha)$$

3. for each element $n = 1, ..., N_d$ of the d-th document

   (a) a topic $t_{d,n}$ is drawn from a multinomial distribution with parameter θ_d,

   $$t_{d,n} \sim \mathrm{Mult}(\theta_d)$$

   (b) a word $w_{d,n}$ is drawn from a multinomial distribution with parameter $\phi_{t_{d,n}}$,

   $$w_{d,n} \sim \mathrm{Mult}(\phi_{t_{d,n}})$$

In the original formulation of LDA (Blei et al. (2003)), each word probability for each topic $\phi_{k,v}$ was assumed to be a fixed quantity. This simplyfing assumption can lead to challenges when a new document contains words that were not present in the *training set* - the collection of documents used to estimate the model parameters. In such cases, zero probabilities are assigned to unseen words, which can hinder the model's performance. To address this issue, a smoothing technique

is employed, which involves adding a second Dirichlet prior. This adjustment helps to redistribute some probability mass to unseen words, effectively allowing the model to assign non-zero probabilities to new words based on the learned distributions, thereby enhancing its robustness and generalizability.

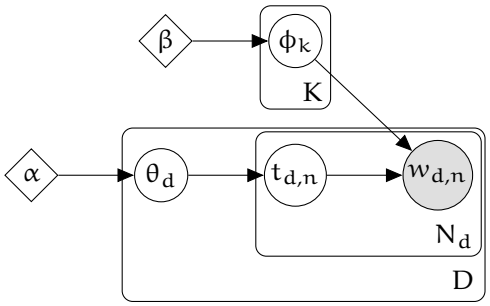


Figure 1.1: A graphical model representation of LDA. Nodes shaped as diamonds, α and β, represent hyperparameters. Nodes shaped as circles, θ_d, ϕ_k and $t_{d,n}$ represent parameters with distributions. Colored nodes, $w_{d,n}$, represent observed data

The posterior distribution and parameters estimate

As it was said before, only words are observed variables. In other terms, what is observed are $w_{d,n}$, for each document $d = 1, ..., D$ and for $n = 1, ..., N_d$. The remaining variables are hidden, or latent, and together they form the underlying topical structure of the collection. These hidden variables include the topics ϕ_k, $k = 1, ..., K$, i.e. the distributions of words associated with each of the k topics, the per-document topic proportions, θ_d, $d = 1, ..., D$, and the per-word topics assignments $t_{d,n}$, $d = 1, ..., D$, $n = 1, ..., N_d$. Given the hyperparameters α and β, the *joint distribution* of all the hidden and observed variables is:

$$p(w, t, \theta, \phi \mid \alpha, \beta) = \prod_{d=1}^{D} p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} p(t_{d,n} \mid \theta_d) p(w_{d,n} \mid \phi_{t_{d,n}}) \right) \prod_{k=1}^{K} p(\phi_k \mid \beta) \tag{1.7}$$

where:

- $p(\theta_d \mid \alpha)$ is the Dirichlet distribution for the topic proportions in document d
- $p(t_{d,n} \mid \theta_d)$ is the categorical distribution assigning a topic for the n-th element of document d

- $p(w_{d,n} \mid \phi_{t_{d,n}})$ is the probability of the n-th word in document d, given the distribution of the assigned topic
- $p(\phi_k \mid \beta)$ is the Dirichlet distribution for the word distribution in topic k

The marginal distribution of documents, i.e. the observed probability of documents given the hyperparameters, is obtained by marginalizing out the latent variables θ, ϕ and t in (1.7):

$$p(w \mid \alpha, \beta) = \int \left(\prod_{d=1}^{D} \left(\int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{t_{d,n}} p(t_{d,n} \mid \theta_d) p(w_{d,n} \mid \phi_{t_{d,n}}) \right) d\theta_d \right) \right) \times \prod_{k=1}^{K} p(\phi_k \mid \beta) d\phi_k \tag{1.8}$$

The key inferential problem that must be solved is that of computing the posterior distribution of the hidden variables, given the observed words w and the hyperparameters α and β, which is

$$p(t, \theta, \phi \mid w, \alpha, \beta) = \frac{p(w, t, \theta, \phi \mid \alpha, \beta)}{p(w \mid \alpha, \beta)} \tag{1.9}$$

As reported in Blei & Lafferty (2009), this distribution can be thought as the "reversal" of the generative process described earlier. Given the observed corpus, the posterior represents the distribution of the hidden variables which generated it. This distribution is intractable to compute due to the integral involved in (1.8). However, a wide range of approximation techniques can be found in literature.

Approximation techniques for the posterior distribution

Blei et al. (2003) proposed a mean field variational inference algorithm (VI). The idea is to approximate the intractable posterior distribution with a simpler one containing free *variational parameters*. These parameters are then fit so that the approximation is close to the true posterior. The mean field variational distribution for LDA is

$$q(t, \theta, \phi \mid \lambda, \xi, \gamma) = \prod_{d=1}^{D} \left(q(\theta_d \mid \gamma_d) \prod_{n=1}^{N_d} q(t_{d,n} \mid \xi_{d,n}) \right) \prod_{k=1}^{K} q(\phi_k \mid \lambda_k) \tag{1.10}$$

where each variable is independent of the others, with each associated with a different variational parameter. The next step is to set up an optimization problem

that determines the values of λ, ξ and γ. The idea is to minimize the Kullback-Leibler (KL) divergence between the variational distribution $q(t, \theta, \phi \mid \lambda, \xi, \gamma)$ and the true posterior $p(t, \theta, \phi \mid w, \alpha, \beta)$, i.e.

$$(\lambda^*, \xi^*, \gamma^*) = \arg\min_{(\lambda,\xi,\gamma)} D(q(t, \theta, \phi \mid \lambda, \xi, \gamma) \,||\, p(t, \theta, \phi \mid w, \alpha, \beta)) \tag{1.11}$$

It can be shown that this problem is equivalent to maximizing a specific objective function, often referred to as Evidence Lower Bound (ELBO), which is

$$\mathcal{L}(\lambda, \xi, \gamma) = \mathbb{E}_q [\log p(w, t, \theta, \phi \mid \alpha, \beta)] - \mathbb{E}_q [\log q(t, \theta, \phi \mid \lambda, \xi, \gamma)] \tag{1.12}$$

The goal is then to maximize this lower bound:

$$(\lambda^*, \xi^*, \gamma^*) = \arg\max_{(\lambda,\xi,\gamma)} \mathcal{L}(\lambda, \xi, \gamma) \tag{1.13}$$

This procedure yields the following update equations for the variational parameters:

$$\xi_{d,n} \propto \exp\left(\mathbb{E}_q [\log \theta_d] + \mathbb{E}_q \left[\log \phi_{1:K, w_{d,n}}\right]\right) \tag{1.14}$$

$$\gamma_d = \alpha + \sum_{n=1}^{N_d} \xi_{d,n} \tag{1.15}$$

$$\lambda_k = \beta + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \xi_{d,n,k} \cdot w_{d,n} \tag{1.16}$$

After choosing some initial values, these three equations are applied iteratively until convergence. The final estimates of the parameters γ_d, $\xi_{d,n}$ and λ_k are used as approximations of the posterior distribution of topics for each document, the assignments of topics to words and the distributions of words for each topic, respectively.

A slight modification of this method, which is called collapsed variational inference (CVI), was proposed by Teh et al. (2006). This approach aims to improve the quality of the approximation by eliminating some latent variables through marginalization. In particular, this means "collapsing" (or integrating out) the topic distributions and working directly with a reduced representation of the model. Compared to "simple" VI, CVI can be more efficient in terms of accuracy, as it reduces the number of latent variables approximated, but is also more

complex to implement, due to the need to exactly integrate out some of the variables.

Nevertheless, the most popular method to estimate the posterior distribution of a latent variable model is probably Gibbs Sampling (GS). It is a Markov Chain Monte Carlo (MCMC) technique that generates samples from the joint posterior distribution, by iteratively sampling each variable from its conditional distribution (given the current values of the other variables). In the context of LDA, this approach is also known as Collapsed Gibbs Sampling (CGS) because it involves marginalizing out the topic distributions from the sampling process. In detail, these are the steps of the CGS (Heinrich (2005)):

1. an initial topic $t_{d,n}$ is randomly assigned to each word $w_{d,n}$ in the corpus. The following count values are calculated:
   - $n_{d,k}$ = number of words in document d assigned to topic k
   - $n_{k,w}$ = number of times word w is assigned to topic k in the entire corpus
2. for each word $w_{d,n}$ in document d:
   (a) the conditional probability of assigning each topic k to that word given the current assignments, i.e. $p(t_{d,n} = k \mid t_{-d,n}, w_{d,n}, \alpha, \beta), \forall k = 1, ..., K$, is computed. This probability is proportional to both the number of times topic k was assigned to other words in the document d and the number of times the word w is assigned to topic k in the entire corpus. In mathematical terms,

$$p(t_{d,n} = k \mid t_{-d,n}, w_{d,n}, \alpha, \beta) \propto \frac{\left(n_{d,k}^{-d,n} + \alpha\right)}{\sum_{k'} \left(n_{d,k'}^{-d,n} + \alpha\right)} \cdot \frac{n_{k,w_{d,n}}^{-d,n} + \beta}{\sum_{w'} n_{k,w'}^{-d,n} + \beta} \tag{1.17}$$

   where:
   - $n_{d,k}^{-d,n}$ = number of words in document d assigned to topic k, excluding the current word $w_{d,n}$
   - $n_{k,w_{d,n}}^{-d,n}$ = number of times word $w_{d,n}$ is assigned to topic k in the entire corpus, excluding the current word $w_{d,n}$

   (b) the word $w_{d,n}$ is assigned to the topic k with the highest conditional probability. The count values for the k-th topic, i.e. $n_{d,k}$ and $n_{k,w}$, are then updated.

3. Steps 2(a) and 2(b) are repeated until convergence, i.e. until topic assignments don't significantly change throughout the iterations.

4. After convergence, the final count values can be used to calculate the model parameters estimates, i.e.:

   - topic distribution per document (θ_d)

$$\theta_{d,k} = \frac{n_{d,k} + \alpha}{\sum_{k'} \left(n_{d,k'} + \alpha\right)}, k = 1, ..., K \tag{1.18}$$

   - word distribution per topic (ϕ_k)

$$\phi_{k,w} = \frac{n_{k,w} + \beta}{\sum_{w'} \left(n_{k,w'} + \beta\right)}, w = 1, ..., V \tag{1.19}$$

Despite its speed, which makes it suitable and efficient for large datasets, the VI approach can lead to inaccurate posterior estimates. The CVI approach, on the other hand, typically improves the accuracy, by reducing the bias of the independence assumptions and lowering the dimensionality of the model. However, the integration step can be computationally demanding. CGS is probably the most used method: indeed, it does not require strong assumptions on the form of the posterior distribution and is relatively straightforward to implement, with simple updates for topic assignments. The main disadvantages are that CGS can take longer to converge for very large datasets and it's a stochastic approach, which means multiple iterations may be needed to avoid dependency on initial assignments.

On the choice of the number of topics

The three assumptions on which LDA is based, presented at the beginning of this section, include a *fixed number of topics*. However, it's not straightforward to choose this value, as no information about the number of concepts behind a corpus of documents is given in advance. Various methods have been proposed in the literature (Arun et al. (2010), Cao et al. (2009), Griffiths (2004)). In this section, the method proposed by Deveaud et al. (2014), which will be used in the next chapter, is presented.

LDA's topics, which are the distributions over the vocabulary, are often represented and labeled using the n words with the highest probabilities. Considering an operator which returns the top-n arguments that obtain the

n largest values for a given function, it can be specifically used to obtain the set of the most probable n words for a topic k, $\mathbb{W}_k$

$$\mathbb{W}_k = \arg\max_{w}[n]\phi_{k,w} \tag{1.20}$$

The simple and heuristic way proposed to estimate the number of topics, K, consists in maximizing the information divergence between all pairs of LDA topics, k_i, k_j, with $k = 1, ..., K$. In other words, the optimal number of topics is the one for which LDA modeled the most scattered topics. In mathematical terms:

$$\hat{K} = \arg\max_{K} \frac{1}{K(K-1)} \sum_{(k,k')\in\mathbb{T}_k} \mathcal{D}(k||k') \tag{1.21}$$

where K is the number of topics given as a fixed parameter to LDA and $\mathbb{T}_k$ is the set of K topics modeled by LDA. The authors propose to use the Jensen-Shannon (JS) divergence, which is a symmetrized version of the more famous Kullback-Leibler (KL) divergence. The divergence between two topics, k and k', can be so defined as:

$$\mathcal{D}(k||k') = \frac{1}{2} \sum_{w\in W_k\cap W'_k} \phi_{k,w} \log\left(\frac{\phi_{k,w}}{\phi_{k',w}}\right) + \frac{1}{2} \sum_{w\in W_k\cap W'_k} \phi_{k',w} \log\left(\frac{\phi_{k',w}}{\phi_{k,w}}\right) \tag{1.22}$$

From a practical point of view, after estimating multiple LDA with different values of K (for example, $K = 2, ..., 20$), the value $\hat{K}$ that maximizes the divergence and its associated topic model is chosen.

Despite its simplicity, it's worth remembering that estimating LDA multiple times is often time-consuming. Moreover, the final topics obtained should always be interpreted. If the set of the most probable words for a topic, $\mathbb{W}_k$, contains words that are difficult to associate, it's good practice to consider a lower value of K.

1.3.3 Supervised Latent Dirichlet Allocation (sLDA)

Supervised Latent Dirichlet Allocation (sLDA) is an extension of LDA which incorporates a supervised learning component, enabling it not only to identify topics but also to predict a response variable associated with each document.

The response can can take various forms: an unconstrained real-value variable, a positive real-valued variable, a categorical variable with ordered or unordered

class labels or an integer-valued variable. Some examples include the number of stars associated with a hotel review, the number of people who read an online article or the section of the online newspaper (politics, sport,...).

sLDA was first presented in Mcauliffe & Blei (2007).

The generative model

Hereinafter, the case of a real-valued response variable will be considered. The three assumptions on which LDA is based (*bag of words*, *bag of documents* and *fixed number of topics*) remain valid. Moreover, a relationship between the topic distribution in the document and the response variable is assumed.

For the sake of simplicity, the same notation of LDA will be used. Additionally, for each document $d \in \mathfrak{D}$ the response variable will be denoted as $y_d \in \mathbb{R}$.

The generative model is composed of the following steps, some of which are identical to those in the LDA case:

1. for each topic $k \in \mathcal{K}$, the distribution of words over the vocabulary V, ϕ_k, is drawn from a Dirichlet distribution with parameter β,

$$\phi_k \sim \mathrm{Dir}(\beta)$$

2. for each document $d \in \mathcal{D}$, a topic distribution θ_d is drawn from a Dirichlet distribution with parameter α,

$$\theta_d \sim \mathrm{Dir}(\alpha)$$

3. for each element $n = 1, ..., N_d$ of the d-th document

   (a) a topic $t_{d,n}$ is drawn from a multinomial distribution with parameter θ_d,

   $$t_{d,n} \sim \mathrm{Mult}(\theta_d)$$

   (b) a word $w_{d,n}$ is drawn from a multinomial distribution with parameter $\phi_{t_{d,n}}$,

   $$w_{d,n} \sim \mathrm{Mult}(\phi_{t_{d,n}})$$

4. for each document $d \in \mathcal{D}$, the response variable y_d is generated from a normal distribution with parameters $\eta^T \bar{t}_d$ and σ^2,

$$y_d \sim \mathcal{N}\left(\eta^T \bar{t}_d, \sigma^2\right)$$

where:

- $\eta = (\eta_1, ..., \eta_K)$
- $\bar{t}_d = (\bar{t}_{d,1}, ..., \bar{t}_{d,K}) = \frac{1}{N_d} \sum_{n=1}^{N_d} t_{d,n}$

In other words, the response variable is modelled as:

$$y_d = \eta_1 \bar{t}_{d,1} + ... + \eta_K \bar{t}_{d,K} + \epsilon_d \tag{1.23}$$

where η is the vector of regression coefficients of a normal linear model whose covariates are the unobserved empirical frequencies of the topics in the d-th document. Since these covariates sum to one, it is unnecessary to include an intercept.

In the model proposed by Mcauliffe & Blei (2007), the word probabilities for each topic ϕ_k were assumed to be fixed quantities. However, it is possible to perform smoothing and adding a second Dirichlet prior, as was done in unsupervised LDA. The model obtained is represented in Figure 1.2.

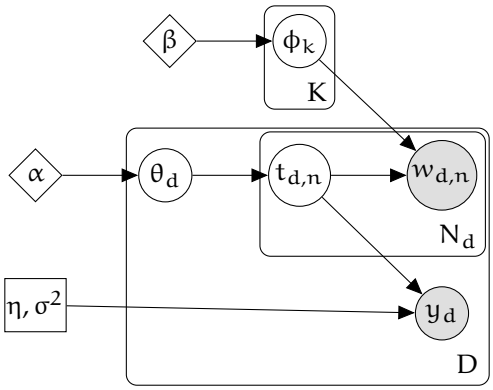


Figure 1.2: A graphical model representation of LDA. Nodes shaped as diamonds, α and β, represent hyperparameters. η and σ^2 are the parameters of the linear model. Nodes shaped as circles, θ_d, ϕ_k and $t_{d,n}$ represent parameters with distributions. Colored nodes, $w_{d,n}$ and y_d, represent observed data

The posterior distribution and parameters estimate

In supervised LDA both the words and the response variable are observed. In other terms, what is observed are $w_{d,n}$, for each document $d = 1, ..., D$ and for $n = 1, ..., N_d$, as well as y_d, for $d = 1, ..., D$. Similarly to "classic" LDA, all other variables are hidden. Given the hyperparameters α and β, the *joint distribution* of all the hidden and observed variables is:

$$p(w, t, \theta, \phi, y \,|\alpha, \beta, \eta, \sigma^2) = \prod_{d=1}^{D} \left[p(y_d \mid t_d, \eta, \sigma^2) \cdot p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} p(t_{d,n} \mid \theta_d) p(w_{d,n} \mid \phi_{t_{d,n}}) \right) \right] \times \prod_{k=1}^{K} p(\phi_k \mid \beta) \tag{1.24}$$

This distribution is quite similar to the *joint distribution* of LDA (1.7). The additional term $p(y_d \mid t_d, \eta, \sigma^2)$ represents the probability of the response variable y_d, given the latent topics of document d, t_d, the regression parameters η and the variance σ^2

By marginalizing out the latent variables θ, ϕ and t, the marginal distribution of the words and the observed variable can be obtained, which is:

$$p(w, y, \mid \alpha, \beta, \eta, \sigma^2) = \int \left[\prod_{d=1}^{D} \left(\int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{t_{d,n}} p(t_{d,n} \mid \theta_d) p(w_{d,n} \mid \phi_{t_{d,n}}) \right) p(y_d \mid \mathbf{t}_d, \eta, \sigma^2) \, d\theta_d \right) \right] \times \prod_{k=1}^{K} p(\phi_k \mid \beta) \, d\phi_k \tag{1.25}$$

The posterior distribution, which is intractable and should consequently be approximated using specific techniques, can be then expressed as:

$$p(t, \theta, \phi \mid w, y, \alpha, \beta, \eta, \sigma^2) = \frac{p(w, t, \theta, \phi, y \mid \alpha, \beta, \eta, \sigma^2)}{p(w, y, \mid \alpha, \beta, \eta, \sigma^2)} \tag{1.26}$$

Approximation techniques for the posterior distribution

Mcauliffe & Blei (2007) proposed an approximation method that combines variational inference (VI) with an expectation-maximization (EM) algorithm.

In detail, the process can be split in two different steps, which are repeated iteratively until convergence is reached:

1. **Variational E-step**:

   In this phase, the goal is to approximate the distribution of the hidden variables θ, t and ϕ, i.e. the posterior distribution $p(t, \theta, \phi \mid w, y, \alpha, \beta, \eta, \sigma^2)$, by employing variational methods. In particular, a variational distribution is introduced:

   $$q(t, \theta, \phi \mid \lambda, \xi, \gamma) = \prod_{d=1}^{D} \left(q(\theta_d \mid \gamma_d) \prod_{n=1}^{N_d} q(t_{d,n} \mid \xi_{d,n}) \right) \prod_{k=1}^{K} q(\phi_k \mid \lambda_k) \quad (1.27)$$

   The aim is to find the optimal values of the so-called *variational parameters*. These optimal values are obtained through the minimization of the KL divergence between the true posterior and the variational distribution or, analogously, through the maximization of the ELBO, which is:

   $$\mathcal{L}(\lambda, \xi, \gamma) = \mathbb{E}_q \left[\log p(w, t, \theta, \phi, y \mid \alpha, \beta, \eta, \sigma^2) \right] - \mathbb{E}_q \left[\log q(t, \theta, \phi \mid \lambda, \xi, \gamma) \right] \quad (1.28)$$

   The estimates are obtained through an iterative procedure quite similar to the one explained in the case of unsupervised LDA. In detail, the update equations for the variational parameters in the case of sLDA are:

   $$\xi_{d,n} \propto \exp \left(\mathbb{E}_q \left[\log \theta_d \right] + \mathbb{E}_q \left[\log \phi_{1:K, w_{d,n}} \right] + \frac{y_d}{N\sigma^2} \eta - \frac{\left[2 \left(\eta^T \xi_{d,-n} \right) \eta + (\eta \circ \eta) \right]}{2N^2\sigma^2} \right) \quad (1.29)$$

   $$\gamma_d = \alpha + \sum_{n=1}^{N_d} \xi_{d,n} \quad (1.30)$$

   $$\lambda_k = \beta + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \xi_{d,n,k} \cdot w_{d,n} \quad (1.31)$$

   where, in 1.29,

   $$\xi_{d,-n} = \sum_{j \neq n} \xi_{d,j} \quad (1.32)$$

   The update equation of $\xi_{d,n}$ is the only one that differs from the case of unsupervised LDA, as it includes an additional term resulting from the presence of the log probability of the response variable given the current assignments in the ELBO.

2. **M-step**

   The goal of this phase is to update the parameters of the supervised model, i.e. η and σ^2, by maximizing the expected log probability of the response variable (which is one of the terms of the ELBO):

$$\left(w, \sigma^2\right) = \arg\max_{w,\sigma^2} \sum_{d=1}^{D} \mathbb{E}_q \left[\log p(y_d \mid t_d, w, \sigma^2)\right] \tag{1.33}$$

   where

$$t_d = \sum_{n=1}^{N_d} \xi_{d,n} \tag{1.34}$$

   is the representation of the d-th document based on topics, which is used to predict the response variable y_d. The optimization is usually performed numerically, using two appropriate update equations for the two parameters.

A slight modification of this method involves using Gibbs sampling instead of Variational Inference in the E-step to approximate the posterior distribution. In detail:

1. **E-step**

   The conditional probability of assigning each topic k to a specific word given the current assignments and the supervised model parameters, i.e. $p(t_{d,n} = k \mid t_{-d,n}, w_{d,n}, y_d, \alpha, \beta, \eta, \sigma^2), \forall k = 1, ..., K$, is computed. In mathematical terms, it can be expressed as:

$$p(t_{d,n} = k \mid t_{-d,n}, w_{d,n}, y_d, \alpha, \beta, \eta, \sigma^2) \propto p(t_{d,n} = k \mid t_{-d,n}) \cdot p(w_{d,n} \mid t_{d,n} = k) \cdot \\ \cdot p(y_d \mid t_d, \eta, \sigma^2) \tag{1.35}$$

   where

$$p(t_{d,n} = k \mid t_{-d,n}) = \frac{\left(n_{d,k}^{-d,n} + \alpha\right)}{\sum_{k'} \left(n_{d,k'}^{-d,n} + \alpha\right)} \tag{1.36}$$

$$p(w_{d,n} \mid t_{d,n} = k) = \frac{n_{k,w_{d,n}}^{-d,n} + \beta}{\sum_{w'} n_{k,w'}^{-d,n} + \beta} \tag{1.37}$$

$$p(y_d \mid t_d, \eta, \sigma^2) = \mathcal{N}(y_d \mid \eta^T t_d, \sigma^2) \tag{1.38}$$

Equations 1.36 and 1.37 are exactly the two factors of the conditional probability in *unsupervised* LDA (1.17). In the case of *supervised* LDA, however, an additional term representing the probability of the response variable, y_d, given the current assignments, is added. After calculating all these probabilities, each word $w_{d,n}$ is assigned to a topic k and the step is repeated with the new assignments, usually for a fixed number of iterations.

2. **M-step**

   Given the last assignments of the E-step, which are an approximation of the posterior distribution, the supervised model parameters are calculated by maximizing the log probability of the response, in the same manner as in the previous case.

Other possible techniques for approximating the posterior distribution are available, such as Gibbs Sampling, but they will not be discussed as they will not be applied in the following sections of this work.

It's worth finally noting that it's possible to extend the specification of sLDA assuming diverse response types, such as a categorical label or a non-negative integer count. In these cases it's reasonable to apply a normal linear model to a suitably transformed version of such a response. When no transformation results in approximate normality, it's possible to make use of generalized linear models (GLM).

Chapter 2

An application on online Airbnb reviews

2.1 User-generated contents (UGC) in the hospitality industry

2.1.1 UGC in the hotel industry

In the first chapter, it was explained that research in NLP has significantly increased in the last two decades, particularly in recent years. Since the end of the 20th century, the advancement of internet technology has led to the development of so-called Web 2.0 applications, which enable people to contribute to media content and interact with other users. In this context, the notion of *user-generated content* (UGC) becomes highly important (Kaplan & Haenlein (2010)):

Definition 2.1. User-Generated Content (UGC) refers to any form of content - such as text, images, videos or reviews - created and shared by users of a platform or service.

Customers reviews are often considered the most influential and important form of UGC, as they establish a communication channel among different customers, as well as between service providers and customers (Casaló et al. (2015)). Online reviews, in particular, are recognized as an essential information source for customers in their decision-making processes: according to Ady et al. (2015), nearly 95% of travelers read online reviews before purchasing travel services. Service providers, in turn, actively seek insights from reviews to better understand customers needs and expectations (Phillips et al. (2015)).

Despite their critical importance, fully leveraging the breadth of information contained in online reviews is not straightforward. UGC consists primarily of unstructured data, making it challenging to extract meaningful insights. However, recent advances in NLP - driven by the increased availability of such content and innovations in computing - have led to the emergence of new text analysis methods. These methods, which include *sentiment analysis* and *topic modeling*, are being applied to large collections of customers reviews to better understand dimensions of customer experience (Sutherland & Kiatkawsin (2020)).

In the context of the hospitality industry, Li et al. (2013) conducted a content analysis of 42,668 Chinese reviews in Beijing, identifying six key factors: logistics, facilities, reception services, food and beverages, cleanliness and maintenance, and value for money. A similar study of 919 English TripAdvisor reviews in New York City found out six factors: location, neighbourhood, room size, beds, staff, and breakfast. Among these, staff quality emerged as the most important factor across all types of hotels (Kim et al. (2016)). Further studies applied LDA to 266,544 English hotel reviews across 16 different countries and 104,161 English reviews of South Korean accommodations. These studies identified 30 and 14 topics of interest, respectively, indicating that the number of topics can vary depending on accommodation type and context (Guo et al. (2017), Sutherland et al. (2020)). Additionally, Xu & Li (2016) and Padma & Ahn (2020) applied latent semantic analysis and word frequency analysis to uncover the determinants of guest satisfaction and dissatisfaction, suggesting that these two constructs have sometimes to be considered separately.

2.1.2 The sharing economy in the accommodation sector

The term *sharing economy*, sometimes referred to also as *gig economy*, *on-demand economy*, *peer economy* or *collaborative economy*, was introduced by Lessig (2008) and can be summarized as follows:

Definition 2.2. Sharing economy refers to the collaborative consumption enabled by activities of sharing, exchanging, and renting resources without owning the goods

Definition 2.2 can be explained by breaking down the key concepts:

- *collaborative consumption* involves a consumption model where individuals cooperate with one another.

- cooperation consists of *sharing*, *exchanging* or *renting* resources, such as goods and services. Goods can be *shared* when multiple people use them (for example, students sharing an apartment) or *rented* when someone borrows them for a limited time. It is also possible to *exchange* goods or services between people (as seen on skill-sharing platforms).
- *without owning the goods* highlights that users can access resources without necessarily owning them. For example, users pay for temporary access while ownership remains with someone else.

Some well-known examples of sharing economy include:

- **ride-sharing services**, such as Uber, which allow individuals to share rides. Instead of owning a car, users can request and pay a ride from a driver
- **car-sharing services**, such as Turo, where individuals rent out their personal vehicles to others in a peer-to-peer model
- **clothing rental services**, such as Rent the Runway, which allow users to rent designer clothing and accessories for a fraction of the purchase price - ideal for special occasions or trying out new styles without buying
- **tool-sharing services**, such as NeighborGoods, a platform where individuals can borrow or rent tools and equipment from their neighbours, reducing the need for everyone to own expensive items they rarely use

One of the sectors most transformed by the sharing economy is the hospitality industry. In recent years, **home-sharing services**, where individuals rent out their homes, apartments or even single rooms, have gained significant importance. The most famous platform for home-sharing is certainly Airbnb. Launched in 2007, Airbnb connects hosts with travelers seeking accommodation, and it now operates globally.

Airbnbs importance within the hospitality industry can be observed through its size and growth. As of 2017, Airbnb was valued at nearly twice the value of Hilton Worldwide Holdings, reaching 31 billion USD and operating across over 191 countries (Sutherland & Kiatkawsin (2020)). Research has shown that a 1% increase in Airbnb listings is associated with a 0.05% decrease in quarterly hotel revenues (Zervas et al. (2017)), and another study found that a 1% increase in Airbnb listings led to a 0.02% decrease in revenue per available room (RevPAR) among hotels in the same market (Dogru et al. (2019)).

In Italy, the economic impact of Airbnb in 2016 was estimated at 3.4 billion EUR, with 3.6 million travelers staying in an Airbnb accommodation over the preceding 12 months (Airbnb (2016)). By 2023, there were about 608.000 listings, with the highest concentrations in Tuscany (12.9%), Sicily (11.4%), and Lombardy (11.1%) (JFC (2024)). In the city of Milan alone, 456,000 people stayed in Airbnb accommodations in 2016, contributing an estimated 48 million EUR to lodging and 229 million EUR to local businesses such as restaurants and museums (Airbnb (2016)). This underscores the broader positive impact of the sharing economy on various sectors beyond hospitality.

Airbnb, like other sharing economy platforms, is often cited as an example of *disruptive innovation* (Kiatkawsin et al. (2020)), a concept introduced by Christenson (1997). *Disruptive innovation* refers to innovations that quickly transform existing markets, often by introducing simpler, more convenient, or more accessible products or services than those provided by industry leaders. While Airbnbs innovative nature lies in offering competitively priced options, it also revolutionized how tourists view, select, and experience accommodations (Guttentag (2015)). Moreover, Airbnb provided individuals with the opportunity to list properties without significant investment or prior experience (Gunter (2018)), leading many to opt for short-term rentals. This shift has contributed to housing shortages in certain cities (Overstreet (2023)).

In JFC (2024), an index was developed to evaluate the sustainability of Airbnb's presence in different regions. This index is calculated as the ratio between the number of residents and the quantity of Airbnb accommodations within a given area. In certain Italian regions, such as Sardinia, Tuscany and Liguria, and in cities such as Florence and Venice, the presence of Airbnb is shown to be somewhat unsustainable.

2.1.3 UGC in the peer-to-peer accommodations

With the growing significance of the sharing economy in the accommodation market, both researchers and industry practitioners have increasingly focused on identifying factors that influence consumer behavior and their purchasing decision-making processes. By pinpointing the topics most relevant to Airbnb guests, valuable insights can be gained regarding the determinants of guest satisfaction (Sutherland & Kiatkawsin (2020)). Some studies, such as Priporas et al. (2017), utilized questionnaires to assess the importance of specific factors.

Similar to the "traditional" hospitality industry, it is useful to extract information from the abundant UGC available on the web. However, despite similarities between the Airbnb experience and traditional hotels - such as overlapping categories of important attribute (Zhang et al. (2020)) - the peer-to-peer accommodation market also has unique characteristics. For instance, guests may stay in a wider variety of unit types, including entire apartments, private rooms or even shared spaces (Mody et al. (2017)). Additionally, the relationship and communication with the host are often given significant importance (Moon et al. (2019)).

Given the relative novelty of the phenomenon, research on UGC in the accommodation sector of the sharing economy is still in its early stages. Notably, recent studies have employed *sentiment analysis* and *topic modeling* methods, particularly LDA, to identify the determinants of guest experiences in Airbnb accommodations.

In Sutherland & Kiatkawsin (2020), LDA was applied to a large dataset of over one million reviews related to New York City. The optimal number of topics was determined to be 43, which were then classified into four categories: evaluation, location, accommodation unit and listing management. The large number of topics reflects the diversity of a city as large as New York. For instance, some location-related topics were specifically tied to areas like Manhattan and Central Park.

A comparative analysis between two cities was performed by Kiatkawsin et al. (2020). Online reviews from Hong Kong and Singapore, two long-term rival tourist destinations, were analyzed using LDA. The optimal number of topics varied between the two cities: 12 topics for Hong Kong and 5 for Singapore. However, all topics from both destinations could be logically assigned to the previously mentioned categories.

Both studies highlight the distinctiveness of topics related to listing management, which includes the host-guest relationship and communication, and are unique to peer-to-peer accommodations. With regard to the evaluation category, topics such as "affective evaluation", "evaluation of host" or "overall evaluation" - often associated with guest satisfaction - were identified. However, it is important to note that while satisfaction is typically linked with positive sentiment, topic modeling merely identifies discussion topics without determining whether a review is globally positive or negative. Moreover, Herzberg's two-factor theory (Herzberg (1966)) posits that satisfaction and dissatisfaction are not opposites

but two independent continuums. This means that the factors contributing to customer satisfaction are not the same as those causing dissatisfaction. The absence of a satisfaction factor does not necessarily indicate a negative experience.

In a study by Ding et al. (2021), a rule-based sentence-level *sentiment analysis* method was used to extract the sentiment of Airbnb reviews from 12 different cities. LDA was then applied separately to positive and negative reviews, identifying topics associated with satisfaction and others specific to dissatisfaction. For example, a topic called "guest conflicts" emerged in negative reviews. Additionally, *supervised* LDA (sLDA) was used with a continuous sentiment index to identify topics most closely associated with positive reviews. The results confirmed the heterogeneity of satisfaction and dissatisfaction factors, providing Airbnb practitioners with valuable insights into what aspects should be prioritized to enhance guest satisfaction without overemphasizing less-valued elements.

2.2 Analysis of Airbnb reviews in Milan

2.2.1 Goal, research framework, data collection and preprocessing

The aim of this analysis is to identify topics of interest, as well as satisfaction and dissatisfaction attributes from a collection of Airbnb reviews of listings in the Italian city of Milan.

To the best of our knowledge, no existing work in the literature analyses Airbnb reviews specifically related to Italy. The choice of Milan, the second most populated Italian city, is due to two main reasons:

1. **Milan is one of the main tourist destinations in Italy**. As the economic capital and a cultural and tourist center, Milan attracts millions of visitors every year. International annual events such as *Fashion Week*, *Salone del Mobile* or the *Italian Grand Prix* in the nearby Monza, along with one-time occasions such as *Expo 2015* or the upcoming *Winter Olympics*, have boosted tourism and the demand for accommodations, including short-term rentals on platforms like Airbnb. Studying user reviews in such a dynamic city can provide valuable insights into how the short-term rental market responds to these specific demands and trends.

2. **Milan is at the center of the debate on short-term rentals and gentrification**. In recent years, the increase in short-term rentals in cities like Milan has raised concerns about gentrification and rising living costs, particularly in central areas (Amore et al. (2020)). Analyzing reviews may offer insights into tourists' experiences and the potential effects of Airbnb on the city's urban and social fabric.

A single city, rather than a region or a country, was chosen with the aim of identifying some context-specific attributes. In addition, the limited set of location-based keywords may help achieve a more focused analysis of the topics of interest (Sutherland & Kiatkawsin (2020)).

Data were collected via Inside Airbnb (insideairbnb.com), an open-source and independent organization that provides aggregated publicly available data from Airbnb on dozens of locations worldwide for research purposes. Reviews cover the period from April 2010 to June 2024.

The workflow of the present study includes the following steps (see also Figure 2.1):

1. Collecting the Airbnb review dataset.
2. Preprocessing the text to reduce unnecessary noise.
3. Extracting topics using LDA.
4. Conducting sentiment analysis to identify positive and negative reviews.
5. Extracting topics from negative reviews using LDA.
6. Identifying topics highly associated with Airbnb user satisfaction and dissatisfaction by referring to the statistical results of sLDA.

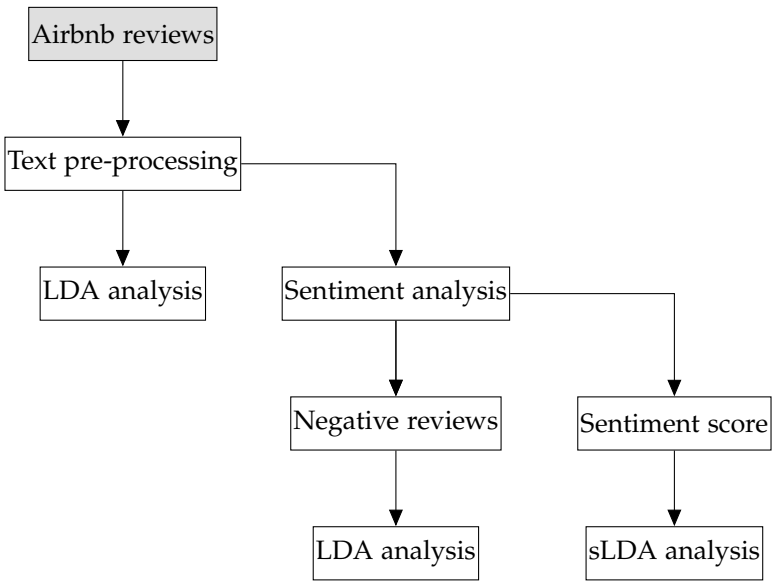


Figure 2.1: Research framework

All data handling, preprocessing, modeling, analysis and visualization were conducted via the R programming language.

In the preprocessing phase, only reviews written in the English language were identified and taken into account using Google's Compact Language Detector 3 (*cld3*) package. Reviews with fewer than 50 words were excluded, because LDA and sLDA are built on co-occurence of words and consequently short ones may produce an undesirable outcome (Ding et al. (2021)). The packages *stringr* and *textclean* were used to eliminate programming cues (such as "\n" or "
"), convert special characters like accented letters (e.g., "è" to "e") and standardize

terms. For example, British and American English variations were unified (e.g., "center" to "centre", "elevator" to "lift", "cozy" to "cosy") and different notations for the same concept were converted into single tokens (e.g., "check-in", "check in", and "checking in" to "checkin"; "ac", "air condition" or "air conditioner" to "aircondition"). Punctuation, numbers and a list of stopwords, including both standard and context-specific terms (such as "milan", "apartment" and "stay", which are not informative due to their high frequency in the corpus) were removed, and the text was converted to lowercase. After performing lemmatization, using the *textstem* package, a document-term matrix was created using the *tm* package. Terms that appeared in fewer than 1% of documents were not considered (Ding et al. (2021)).

The final dataset contains 132,120 reviews related to 13,592 listings, which are either apartments or private rooms, with a majority being entire homes. Shared rooms and hotels were not included as they represented less than 1% of the initial corpus (Table 2.1). As for their temporal distribution, the number of reviews per year is generally increasing. There are only 46 reviews from 2011, which increase to 12,545 in 2019. The drop in 2020 and 2021 is likely due to the pandemic, as the reduction in travel is reflected in a decrease in Airbnb UGC. However, starting in 2022, the number of reviews increased significantly, indicating that the importance of Airbnb accommodation is growing rapidly (Figure 2.2).

Entire home	**Private room**
115,955	16,165
87.76 %	12.24 %

Table 2.1: Airbnb listings and room types

Latent Dirichlet Allocation (LDA) was performed using the *topicmodels* package, while the *ldatuning* and *LDAvis* package were used to identify the optimal number of topics and to represent them. Sentiment analysis and supervised Latent Dirichlet Allocation (sLDA) were performed using the *sentimentr* and *lda* packages, respectively.

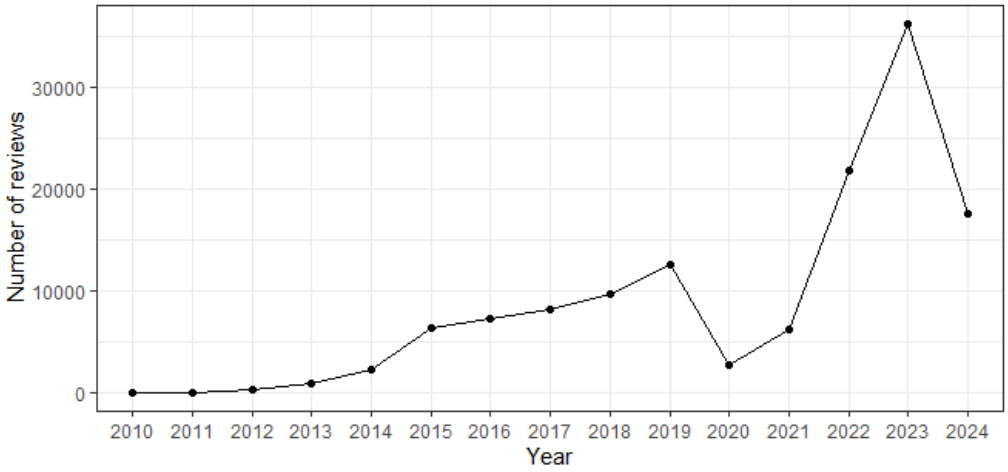


Figure 2.2: Number of reviews per year

2.2.2 LDA on the whole corpus

Technical implementation

The optimal number of topics was chosen based on both the Deveaud criterion (see 1.3.2) and empirical considerations. Indeed, models with a high number of topics are often shown to be less interpretable (Ding et al. (2021)).

First, a 10% sample of the reviews was used to select the Gibbs sampling parameters, i.e. number of iterations, burn-in period and thinning interval. Various models with different numbers of topics were estimated, and the posterior likelihood values of the corpus, conditional on the topics assignments for each iteration, were observed. Graphical analysis indicated that a burn-in period of 500 iterations was necessary for the posterior likelihood to stabilize. Additionally, 2700 iterations and a thinning interval of 200 were selected, resulting in 11 samples used to estimate the final model parameters. These choices were with consideration of the computational complexity involved in estimating LDA on a very large dataset.

Each number of topics between 2 and 20 was tested via grid search. The Deveaud criterion values for each number of topics are showed in Figure 2.3. The highest value of the index was obtained with 16 topics. However, models with 6, 10 and 13 topics were also considered, as they correspond to local maxima of the Deveaud index. Analyzing the most probable words of each topic revealed that models with 13 and 16 topics were less interpretable, as some topics appeared mixed and referred to similar words. Consequently, the model with 10 topics was selected.

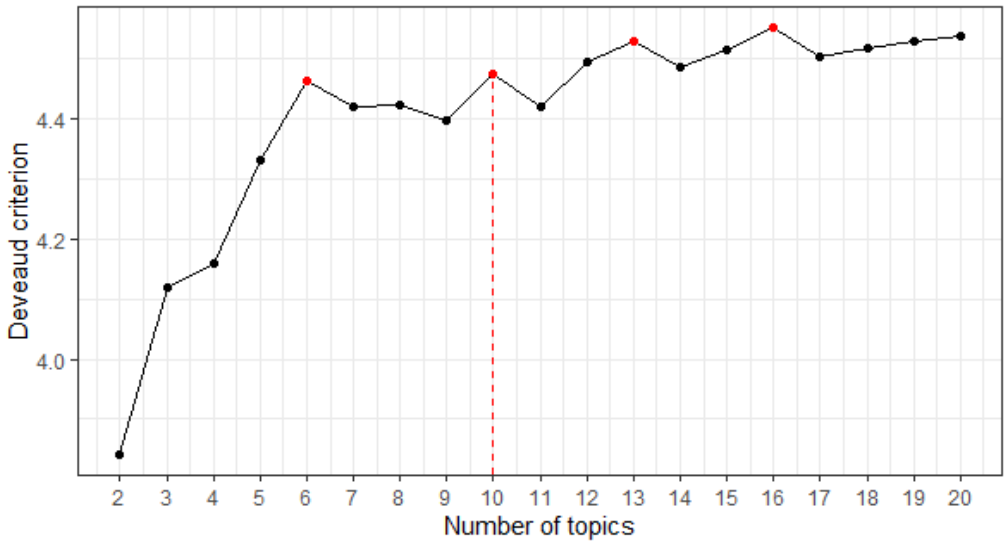


Figure 2.3: Deveaud criterion values for 2 to 20 topics - LDA on the whole corpus

Results and interpretation

Topic labeling was performed by examining the most relevant top words and reading the ten reviews that had the highest proportion of that specific topic. The 10 topics obtained are summarized in Table 2.2. As in Sutherland & Kiatkawsin (2020) and Kiatkawsin et al. (2020), each of the topic can be asssigned to one of four categories. Specifically, topics 4 and 8, respectively referred to as *Proximity to transport* and *Neighbourhood* pertain to the **Location** of the listing. Topics 1, *Issues*, 5, *Unit amenities*, and 9, *Unit evaluation* are related to the **Accommodation Unit**. Topic 3, *Check-in/out*, topic 6, *Host evaluation* and topic 8, *Host Welcoming* belong to the **Listing Management** category. Finally, topics 2 and 10, i.e. *Experience* and *Feeling*, can be assigned to the **Overall Evaluation** category.

Topic models can be visualized using the R *LDAvis* package, which allows for understanding the proportional prevalence of each topic and the relationships between them (Sievert & Shirley (2014)). As shown in Figure 2.4, the 10 topics exhibit similar proportions in the corpus, with circles being of equal size. There is some overlap between topics 6, 8 and 10, indicating that Airbnb users often commented on *Host evaluation*, *Host welcoming* and *Feeling* simultaneously. In particular, topic 10 includes words such as "again", "back" and "come", which are associated with a revisit intention. This suggests that a user is likely to return if he had a positive experience with the host. Additionally, there is a weak link between the two topics which regard Location, as they share some common words.

Topic	**Top 30 relevant words**
1: Issues	night, work, noise, door, build, water, good, aircondition, shower, floor, issue, quite, overall, sleep, problem, hot, thing, window, use, however, open, outside, lift, light, get, one, little, may, small, inside
2: Experience	like, airbnb, go, house, best, make, one, want, experience, time, first, better, even, every, way, find, see, day, look, thing, sure, person, book, say, take, ever, need, much, feel, know
3: Check-in/out	checkin, get, time, day, arrive, leave, late, even, host, checkout, early, instruction, clean, night, able, find, hour, luggage, morning, clear, wait, flight, let, flexible, arrival, message, next, allow, key, book
4: Proximity to transport	minute, metro, station, walk, central, train, close, tram, duomo, away, near, stop, easy, take, bus, supermarket, location, get, convenient, around, line, short, airport, little, be, main, ride, right, directly, front
5: Unit amenities	room, bed, comfort, kitchen, bathroom, two, space, big, small, bedroom, good, nice, live, one, people, clean, use, large, coffee, enough, machine, spacious, towel, sofa, wash, cook, balcony, little, three, private
6: Host evaluation	host, clean, recommend, location, need, helpful, responsive, super, quick, perfect, friendly, always, question, extremely, exact, respond, comfort, look, help, available, definitely, answer, describe, amenity, picture, spacious, communication, andrea, photo, short
7: Neighbourhood	restaurant, walk, area, shop, location, distance, street, bar, quiet, park, within, around, neighbourhood, right, duomo, close, cafe, perfect, store, navigli, near, safe, locate, attraction, corner, grocery, spot, plenty, main, car
8: Host welcoming	give, host, make, welcome, provide, recommend, visit, thank, recommendation, lovely, wonderful, meet, breakfast, local, warm, even, eat, comfort, fantastic, go, tip, marco, detail, information, show, food, touch, helpful, offer, excellent
9: Unit evaluation	nice, well, easy, city, centre, good, clean, close, locate, transport, recommend, equip, need, public, access, neighbourhood, communication, cosy, quiet, enjoy, reach, near, area, pleasant, studio, far, convenient, connect, furnish, foot
10: Feeling	again, beautiful, feel, thank, home, definitely, amaze, back, come, perfect, love, lovely, next, absolutely, much, like, wonderful, super, visit, location, time, enjoy, view, return, make, design, right, cosy, terrace, safe

Table 2.2: Relevant words for each topic - LDA on the whole corpus

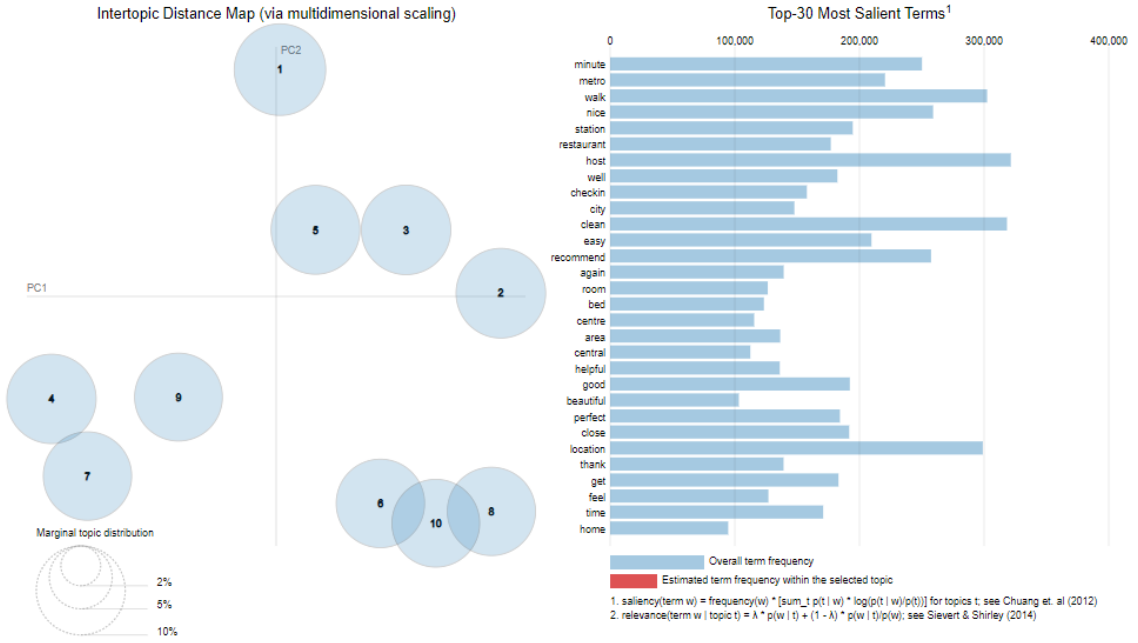
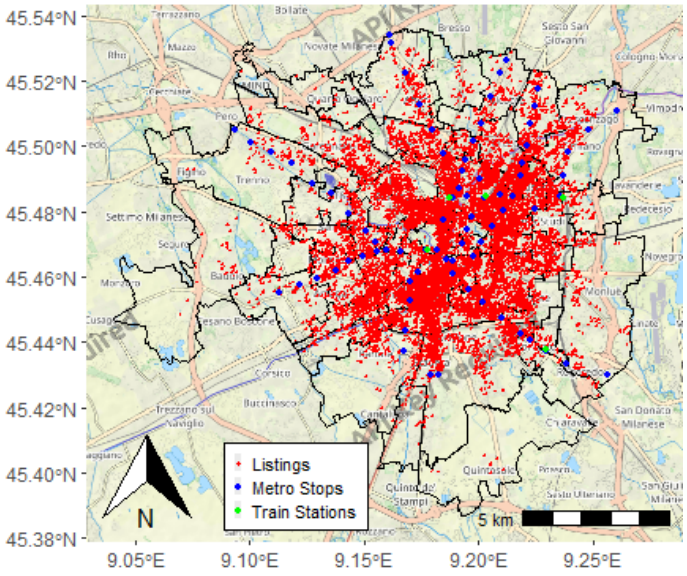


Figure 2.4: Visualization of topics - LDA on the whole corpus

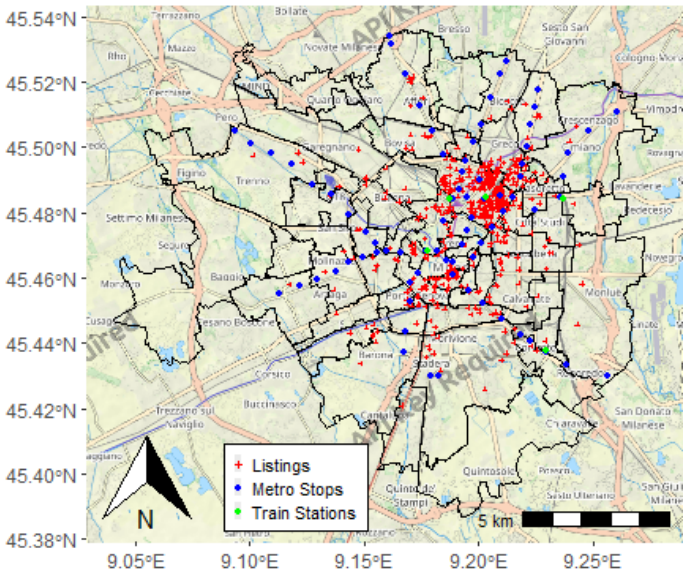
Furthermore, topics with location-specific patterns can be validated with a geographical representation and by assessing whether city-specific words are linked to them. For example, topic 4, *Proximity to Transport*, is likely to be most associated with listings located in the surroundings of metro stops or train stations. Figure 2.5 shows that apartments with with at least 6 reviews where topic 4 has the highest proportion are primarily situated close to one of the main train stations, especially *Stazione Centrale*, or near one of the several metro stops. On the other side, topic 7, *Neighbourhood*, actually contains terms such as "duomo" and "navigli", which are specific to Milan. The correct association of these words is likely a direct consequence of analyzing data from a single city, allowing for a more precise and contextually relevant topic identification.

2.2.3 Sentiment Analysis and LDA on negative reviews

"In a manner similar to the approach used by Ding et al. (2021), *sentiment analysis* was conducted to identify positive and negative reviews. Specifically, the sentiment analysis method described in the previous chapter (see 1.2.3), which provides a continuous sentiment index, was applied to the corpus. The dictionary used is the one proposed by Hu & Liu (2004). This choice was made after testing various dictionaries on a sample, reviewing the results, and evaluating the coherence of the sentiment index. The dictionary chosen was the one that



(a)



(b)

Figure 2.5: Geographical representation of listings, metro stops and train stations. In **(a)** all the listings are represented, in **(b)** only listings with more than 5 reviews with a prevalence of topic 4 (*Proximity to transport*)

produced the best results. As summarized in Table 2.3, the vast majority of the reviews are classified as positive, as the continuous sentiment score obtained is greater than zero. Conversely, there are only 4887 negative reviews, representing less than 4%. There is also a very small number of reviews associated with a null sentiment index, which are therefore classified as neutral.

Positive reviews	**Neutral reviews**	**Negative reviews**
127,189	44	4,887
96,27%	0,03%	3,70 %

Table 2.3: Results of sentiment analysis

As previously mentioned, Herzberg's theory (Herzberg (1966)) posits that satisfaction and dissatisfaction are two independent, continuous attributes and factors that lead to a customer satisfaction can be different from those that cause dissatisfaction. In this context, it is particularly useful to perform an analysis on negative reviews only. Indeed, since they are underrepresented in the collection, topics related to dissatisfaction might not have emerged in the overall *topic modeling* analysis from the previous section.

The choice of the optimal number of topics was once again made using the Deveaud criterion (1.3.2), which clearly indicate that this value is equal to three (Figure 2.6).

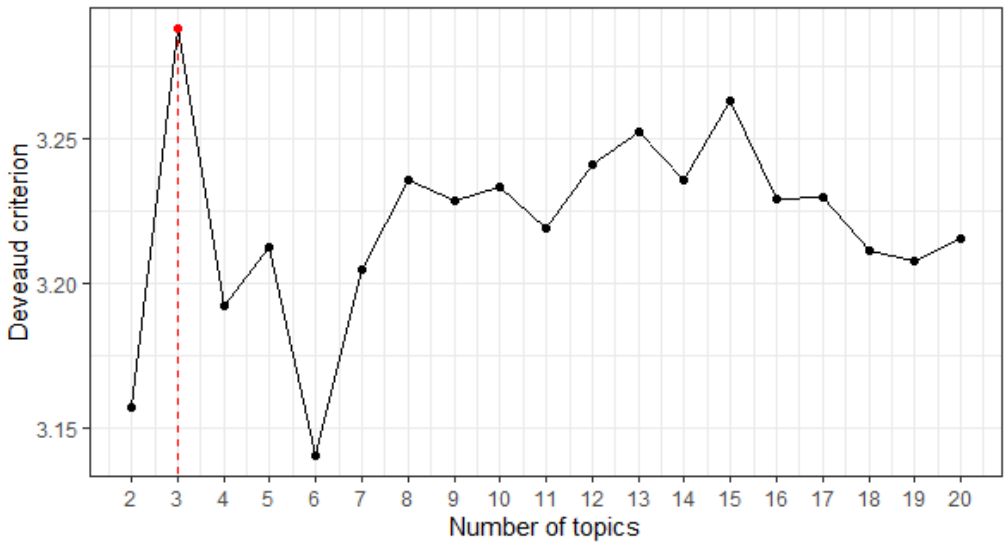


Figure 2.6: Deveaud criterion values for 2 to 20 topics - LDA on negative reviews

Topic labelling was performed by examining the 30 most relevant top-words and the 10 reviews with the highest topic proportion. As summarized in Table

2.4, topic 1, *Check-in and management*, relates to users' experiences with issues upon arrival or, more generally, problems with the host. Many users, for example, complain that the listing owner was not understanding about their delay, due to public transport problems, or that he was not there upon their arrival to welcome them and hand over the keys. Topic 2, *Dirtiness and inadequate amenities*, is associated with reviews describing problems in the listing, such as insufficient cleanliness, poorly equipped kitchens or a lack of towels. Topic 3, *Sleep problems*, is linked to various issues that often prevented guests from sleeping. The main of them include noise (caused by neighbours, people on the street, nearby construction sites or trains and trams), lack or malfunction of the air conditioning in summer (which forced guests to keep the windows opened, with further noise issues) or excessive cold in winter.

Topic	**Top 30 relevant words**
1: Check-in and management	host, get, checkin, time, day, airbnb, leave, say, go, come, ask, arrive, take, key, experience, first, even, find, hour, tell, late, give, book, pay, back, try, message, want, call, make
2: Dirtiness and inadequate amenities	clean, bed, room, shower, one, bathroom, use, water, like, kitchen, feel, make, floor, dirty, thing, smell, small, break, two, look, bedroom, towel, provide, bed, toilet, sofa, old, wash, picture, live
3: Sleep problems	night, location, work, good, noise, minute, sleep, nice, close, walk, aircondition, however, door, problem, open, build, window, need, hot, station, cold, area, metro, quite, issue, around, heat, street, outside, wifi

Table 2.4: Relevant words for each topic - LDA on negative reviews

2.2.4 Supervised LDA

Technical implementation

Supervised Latent Dirichlet Allocation was implemented with the following starting values for the hyperparameters: the Dirichlet hyperparameter for the topic proportions (α) and for the topics multinomials (β) were set to 1 and 0.1, respectively (as in Ding et al. (2021)). The coefficients of the regression model (η) were all initialized at 0 while the variance (σ^2) was set to 0.25.

The model was estimated through an expectation-maximization algorithm, which uses Gibbs sampling in the expectation phase to approximate the posterior distribution. A total of 50 expectation-maximization iterations with 100 Gibbs sampling sweeps per iteration was tested and chosen on a sample of 10% of reviews. As with *unsupervised* LDA, the high dimensionality of the data was taken into account in this choice to avoid excessively long computation times.

A suitable topic number was determined by conducting test runs with different values, ranging from 5 to 11. Each estimation took approximately 4 hours. Since quantitative criteria, such as the perplexity, tend to decrease as the number of topics increases (Ding et al. (2021)), and models with a high number of topics are often difficult to interpret, a human judgement approach was used to evaluate the coherence and interpretability of the model. In particular, a model with 9 topics was found to be the most appropriate.

Results and interpretation

The 9 topics obtained are quite similar to those of *unsupervised* LDA. As shown in Table 2.5, these are *Proximity to transport*, *Feeling and experience*, *Issues*, *Unit amenities*, *Host welcoming*, *Unit evaluation*, *Neighbourhood*, *Check-in/out* and *Host evaluation*. All except the second share their top relevant words with a topic from LDA (Table 2.2) and were therefore labeled with the same names. In contrast, the topic *Feeling and experience* appears to be the result of merging topics 2, *Experience*, and 10, *Feeling*, from classical LDA, as it contains words from both of them, such as "like", "airbnb", "experience" from the former or "again", "feel", "thank" from the latter. However, a model with 10 topics fails to distinguish them but rather results in two mixed and not uninterpretable topics.

As previously mentioned, sLDA focuses on evaluating the relationships between different topics and the review polarity score, which indicates the users' satisfaction level. The response variable is modeled through a regression model with the empirical frequencies of topics in the documents serving as covariates (see Equation 1.23). Table 2.6 and Figure 2.7 report the coefficients of each topic along with their significance.

Host evaluation and *Unit evaluation* are the two topics with the highest coefficients, 0.80 and 0.73, respectively. This indicates that the host's availability and kindness (some of the top-words for this topic are "quick", "helpful", "responsive"

Topic	**Top 30 relevant words**
1: Proximity to transport	station, minute, walk, metro, train, central, duomo, stop, tram, away, near, bus, close, easy, centre, take, line, supermarket, get, city, airport, convenient, distance, around, within, be, restaurant, good, locate, main
2: Feeling and experience	home, back, feel, come, best, thank, amaze, again, like, love, beautiful, make, house, definitely, time, ever, one, airbnb, experience, welcome, super, perfect, go, much, hope, always, kind, wonderful, host, every
3: Issues	noise, door, work, night, sleep, shower, hot, aircondition, window, floor, issue, bed, good, water, one, however, open, street, problem, room, hear, light, small, get, cold, overall, quite, build, thing, heat,
4: Unit amenities	kitchen, bed, bathroom, room, machine, bedroom, coffee, two, wash, comfort, towel, live, cook, small, shower, large, space, use, sofa, big, good, work, one, well, water, equip, nice, dry, spacious, wifi
5: Host welcoming	make, give, welcome, recommendation, provide, recommend, local, wonderful, host, thank, experience, home, tip, beautiful, fantastic, arrival, visit, feel, city, touch, meet, eat, anyone, marco, helpful, best, breakfast, offer, lovely, suggestion
6: Unit evaluation	centre, transport, city, public, nice, good, easy, close, well, near, recommend, locate, park, metro, house, equip, quiet, restaurant, cosy, reach, friendly, tram, area, neighbourhood, thank, helpful, supermarket, around, host, foot
7: Neighbourhood	walk, restaurant, shop, bar, duomo, minute, distance, street, area, within, cafe, quiet, away, navigli, right, canal, neighbourhood, perfect, near, beautiful, close, store, plenty, metro, park, around, local, grocery, love, brera
8: Check-in/out	checkin, late, arrive, early, leave, time, get, hour, checkout, flight, let, even, wait, key, luggage, day, arrival, meet, night, help, book, host, ask, airbnb, allow, go, one, first, message, bag
9: Host evaluation	checkin, question, easy, quick, recommend, host, helpful, responsive, respond, super, instruction, definitely, always, clear, answer, again, exact, perfect, clean, checkout, communication, thank, friendly, location, flexible, extremely, central, smooth, describe, comfort

Table 2.5: Relevant words for each topic - supervised LDA

and "friendly"), along with an overall evaluation of the unit ("good", "close", "locate", "equip", "cosy"), best explain a positive sentiment in a review.

Next, *Host welcoming*, *Feeling and experience*, topics related to location (i.e. *Neighbourhood* and *Proximity to transport*) and *Unit amenities* are also associated with user satisfaction. Their coefficients range from 0.40 and 0.60.

Topic 3, *Issues*, which relates to problems and negative experiences during the stay ("noise", "work", "problem"), is the only one with a negative coefficient (-0.05). The presence of this topic in a review tends to decrease the sentiment score, thereby increasing user dissatisfaction.

Finally, the parameter for topic 8 (*Check-in/out*) is positive but close to zero (0.06). A topic associated with words like "checkin", "late", "arrive" and "early" doesn't seem to be strongly related to the sentiment of the review. This is likely because this topic is mostly present in reviews that describe negative experiences during the check-in and check-out phases.

With respect to the significance of the coefficients, it can be easily verified that all of them have high t-statistic values and are significant at all commonly used significance levels (Table 2.6).

Topic	Coefficient	Standard Error	t-value	$P> \|t\|$
1: Proximity to transport	0.474698	0.003676	124.290	<2e-16 ***
2: Feeling and experience	0.559163	0.003517	158.988	<2e-16 ***
3: Issues	-0.053177	0.003426	-15.521	<2e-16 ***
4: Unit amenities	0.409139	0.003903	104.818	<2e-16 ***
5: Host welcoming	0.608844	0.003656	166.548	<2e-16 ***
6: Unit evaluation	0.727525	0.003804	207.586	<2e-16 ***
7: Neighbourhood	0.550175	0.003676	149.648	<2e-16 ***
8: Check-in/out	0.061767	0.003825	16.147	<2e-16 ***
9: Host evaluation	0.804191	0.003874	207.586	<2e-16 ***

Table 2.6: Coefficients, standard errors and significance of supervised LDA model

2.2.5 Discussion

In this analysis, various NLP techniques were applied to a corpus of Milan Airbnb reviews, with the aim of extracting topics of interest, as well as identifying

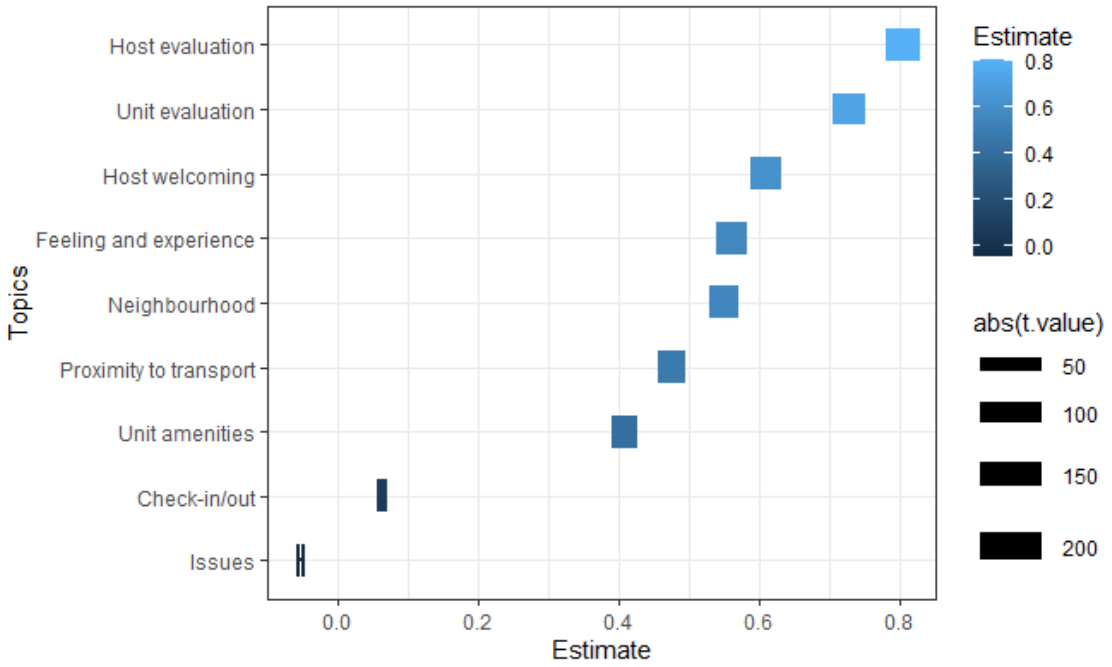


Figure 2.7: Graphical representation of coefficients and t-values of supervised LDA model

satisfaction and dissatisfaction attributes. While other recent studies applied these methods (LDA, sentiment analysis and sLDA) in similar contexts, this is the first work to focus specifically on data pertaining to Italy.

The LDA analysis revealed that the reviews are related to 10 distinct topics. Other studies focusing on different cities have reported varying numbers of topics: 43 topics were identified in New York (Sutherland & Kiatkawsin (2020)), 12 in Hong Kong, and only 5 in Singapore (Kiatkawsin et al. (2020)). Despite these differences, topics obtained in our analysis are quite similar to those of New York, Hong Kong and Singapore as they can be clustered into the same four groups. These groups suggest that guests' interests in an Airbnb stay revolve around the location of the listing, the apartment's features, host management, and having a positive overall experience.

Similar topics resulted from the sLDA analysis, which aimed to associate them with the sentiment expressed in the review. An evaluation of the host - encompassing availability, helpfulness and responsiveness - was found to be the most significant driver of satisfaction. This support the findings of some studies, such as Moon et al. (2019), which emphasize that in peer-to-peer accommodations, such as Airbnb, the importance of communication and the relationship with the

host outweighs that in traditional hotels. Additionally, topics relating location and the unit accommodation also have a positive impact on user satisfaction.

The positivity of all the coefficients, except for the one associated with issues during the stay, is due to the fact that the vast majority of reviews have a positive sentiment index. In Ding et al. (2021), a comparable analysis was applied to a corpus containing a similar number of positive and negative reviews. In that case, there was an equal number of positive and negative coefficients, reflecting an equal number of satisfaction and dissatisfaction attributes.

The LDA analysis of the 4,887 negative reviews revealed three topics related to user dissatisfaction: negative host management, particularly during the check-in phase, dirtiness and inadequacy of the unit, and a series of problems that caused sleep disturbances.

Based on these results, some valuable implications for Airbnb managers can be derived. To ensure a positive stay for guests, it's crucial for the host to be friendly and available. Providing detailed instructions, ensuring late check-ins, and offering a warm welcoming with some recommendations and tips are highly appreciated by guests. These factors are often more closely linked to the intention to revisit than the positive aspects of the accommodation unit itself. However, it's important also to ensure adequate cleanliness and basic amenities, such as an air conditioning system, which is crucial during the summer season, or soundproof window frames to prevent external noise. Additionally, the bathroom is found to be very important, and some negative reviews have mentioned issues with the shower or water temperature.

Finally, as anticipated, using data from a single city restricts the range of specific words, such as location-based keywords, that can be readily associated with a topic. For instance, "navigli" and "duomo" were accurately categorized into a geographical topic, highlighting attractions in the neighborhood of the listing.

Conclusion

In this study, the field of *Natural Language Processing* (NLP) was explored and analyzed in detail. The first chapter introduced NLP, including its definition, historical advancements, and key techniques. Despite its relatively recent development, NLP has evolved rapidly over the past decades, driven by the increasing availability of text data and computational resources.

Two central tasks in NLP, *sentiment analysis* and *topic modelling*, were examined. An overview about different methods in these two fields was presented, followed by a detailed explanation of specific methodologies.

Sentiment analysis can be conducted using rule-based methods, which assign sentiment scores to individual words based on a predefined dictionary and aggregate these scores to determine the overall sentiment of a text. One methodology detailed in the study associates each document in a corpus with a continuous sentiment index.

In contrast, Latent Dirichlet Allocation (LDA), introduced by Blei et al. (2003), is a prominent technique for *topic modeling*. LDA aims to uncover latent topics within a corpus of text documents. Subsequent extensions of LDA, such as supervised Latent Dirichlet Allocation (sLDA) proposed by Mcauliffe & Blei (2007), were also presented. These hierarchical Bayesian models can be estimated through variational inference or Gibbs sampling.

The second chapter applied *sentiment analysis* and *topic modeling* to extract insights from a collection of Airbnb reviews in Milan. As one of the most prominent platforms in the peer-to-peer accommodation sector, Airbnb has gained significant importance in the lodging industry. Analyzing user-generated content (UGC) from this platform provides valuable information for both service providers and users.

The results of the analysis revealed that, as in other cities worldwide, travelers staying in Airbnb listings place significant importance on their relationship with the host, which emerged as the most crucial factor for guest satisfaction. Other

key factors influencing the choice of an apartment include its location and the quality of the accommodation itself. Dissatisfaction, on the other hand, was often linked to issues with check-in, inadequate amenities, and sleep disturbances.

This study represents the first application of *topic modeling* to extract insights from UGC within the sharing economy of the accommodation sector in Italy and confirms the effectiveness of NLP techniques for analyzing textual data.

However, restricting the research to a single city limits the generalizability of the findings. Additionally, relying on reviews from just one platform may pose a limitation. Future research is encouraged to expand the scope by incorporating data from diverse regions and exploring reviews from other platforms. Furthermore, the sentiment analysis employed in this study could be enhanced by enriching the sentiment dictionary with context-specific words and expressions that better capture the nuances of the accommodation sector. This would lead to more accurate results, especially in identifying sentiments related to specific aspects of a stay. Lastly, integrating other forms of UGC, such as images often uploaded alongside reviews, could offer a richer perspective on customer experiences and preferences, thereby improving the breadth and depth of the analysis.

Bibliography

Ady, M., Quadri-Felitti, D. et al. (2015). Consumer research identifies how to present travel review content for more bookings. *Hotels News Resource* **95**.

Airbnb (2016). Fattore sharing: limpatto economico di airbnb in italia. PDF document.

Amore, A., de Bernardi, C. & Arvanitis, P. (2020). The impacts of airbnb in athens, lisbon and milan: a rent gap theory perspective. *Current Issues in Tourism* **25**, 3329–3342.

Arun, R., Suresh, V., Veni Madhavan, C. & Narasimha Murthy, M. (2010). On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Advances in Knowledge Discovery and Data Mining: 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part I 14*. Springer.

Baccianella, S., Esuli, A., Sebastiani, F. et al. (2010). Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, vol. 10. Valletta.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536* .

Benamara, F., Chardon, B., Mathieu, Y., Popescu, V. & Asher, N. (2012). How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*.

Blei, D. M. & Lafferty, J. D. (2009). Topic models. In *Text mining*. Chapman and Hall/CRC, pp. 101–124.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research* **3**, 993–1022.

Borg, A. & Boldt, M. (2020). Using vader sentiment and svm for predicting customer response sentiment. *Expert Systems with Applications* **162**, 113746.

Brooke, J., Tofiloski, M. & Taboada, M. (2009). Cross-linguistic sentiment analysis: From english to spanish. In *Proceedings of the international conference RANLP-2009*.

Campesato, O. (2021). *Natural language processing fundamentals for developers*. Mercury Learning and Information.

Cao, J., Xia, T., Li, J., Zhang, Y. & Tang, S. (2009). A density-based method for adaptive lda model selection. *Neurocomputing* **72**, 1775–1781.

Casaló, L. V., Flavián, C., Guinalíu, M. & Ekinci, Y. (2015). Avoiding the dark side of positive online consumer reviews: Enhancing reviews' usefulness for high risk-averse travelers. *Journal of Business Research* **68**, 1829–1835.

Christenson, C. (1997). The innovators dilemma. *Harvard Business School Press, Cambridge, Mass* .

Churchill, R. & Singh, L. (2022). The evolution of topic modeling. *ACM Computing Surveys* **54**, 1–35.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science* **41**, 391–407.

Deveaud, R., SanJuan, E. & Bellot, P. (2014). Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique* **17**, 61–84.

Ding, K., Choo, W. C., Ng, K. Y., Ng, S. I. & Song, P. (2021). Exploring sources of satisfaction and dissatisfaction in airbnb accommodation using unsupervised and supervised topic modeling. *Frontiers in psychology* **12**, 659481.

Dogru, T., Mody, M. & Suess, C. (2019). Adding evidence to the debate: Quantifying airbnb's disruptive impact on ten key hotel markets. *Tourism Management* **72**, 27–38.

Eisenstein, J. (2018). Natural language processing. *Jacob Eisenstein* .

Feng, W. V. (2015). *RST-style discourse parsing and its applications in discourse analysis*. University of Toronto (Canada).

Green Jr, B. F., Wolf, A. K., Chomsky, C. & Laughery, K. (1961). Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*.

Griffiths, T. L. (2004). Finding scientific topics. *PNAS* .

Gunter, U. (2018). What makes an airbnb host a superhost? empirical evidence from san francisco and the bay area. *Tourism Management* **66**, 26–37.

Guo, Y., Barnes, S. J. & Jia, Q. (2017). Mining meaning from online ratings and reviews: Tourist satisfaction analysis using latent dirichlet allocation. *Tourism management* **59**, 467–483.

Guttentag, D. (2015). Airbnb: disruptive innovation and the rise of an informal tourism accommodation sector. *Current issues in Tourism* **18**, 1192–1217.

Hamdan, H., Bellot, P. & Bechet, F. (2015). Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*.

Heinrich, G. (2005). Parameter estimation for text analysis. Tech. rep., Citeseer.

Herzberg, F. I. (1966). *Work and the nature of man*. Cleveland, OH: World Publishing.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

Hu, M. & Liu, B. (2004). Mining opinion features in customer reviews. In *AAAI*, vol. 4.

Janjua, F., Masood, A., Abbas, H., Rashid, I. & Khan, M. M. Z. M. (2021). Textual analysis of traitor-based dataset through semi supervised machine learning. *Future Generation Computer Systems* **125**, 652–660.

JFC (2024). Airbnb in italia - i numeri del fenomeno.

Jockers, M. L. (2017). *Syuzhet: Extract Sentiment and Plot Arcs from Text*.

Joty, S., Carenini, G. & Ng, R. T. (2015). Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics* **41**, 385–435.

Kang, H., Yoo, S. J. & Han, D. (2012). Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications* **39**, 6000–6010.

Kaplan, A. M. & Haenlein, M. (2010). Users of the world, unite! the challenges and opportunities of social media. *Business horizons* **53**, 59–68.

Kherwa, P. & Bansal, P. (2019). Topic modeling: a comprehensive review. *EAI Endorsed transactions on scalable information systems* **7**.

Khurana, D., Koli, A., Khatter, K. & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications* **82**, 3713–3744.

Kiatkawsin, K., Sutherland, I. & Kim, J.-Y. (2020). A comparative automated text analysis of airbnb reviews in hong kong and singapore using latent dirichlet allocation. *Sustainability* **12**, 6673.

Kim, B., Kim, S. & Heo, C. Y. (2016). Analysis of satisfiers and dissatisfiers in online hotel reviews on social media. *International journal of contemporary hospitality management* **28**, 1915–1936.

Lessig, L. (2008). *Remix: Making art and commerce thrive in the hybrid economy*. Bloomsbury Academic.

Li, H., Ye, Q. & Law, R. (2013). Determinants of customer satisfaction in the hotel industry: An application of online review analysis. *Asia Pacific journal of tourism research* **18**, 784–802.

Li, Y.-M. & Li, T.-Y. (2013). Deriving market intelligence from microblogs. *Decision Support Systems* **55**, 206–217.

Liddy, E. D. (2001). Natural language processing. *Encyclopedia of Library and Information Science* .

Liu, B. (2022). *Sentiment analysis and opinion mining*. Springer Nature.

Mcauliffe, J. & Blei, D. (2007). Supervised topic models. *Advances in neural information processing systems* **20**.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space.

Mody, M. A., Suess, C. & Lehto, X. (2017). The accommodation experiencescape: a comparative assessment of hotels and airbnb. *International Journal of Contemporary Hospitality Management* **29**, 2377–2404.

Mohammad, S., Dunne, C. & Dorr, B. (2009). Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 conference on empirical methods in natural language processing*.

Moon, H., Miao, L., Hanks, L. & Line, N. D. (2019). Peer-to-peer interactions: Perspectives of airbnb guests and hosts. *International Journal of Hospitality Management* **77**, 405–414.

Overstreet, K. (2023). Is airbnb contributing to the housing crisis? *ArchDaily* Accessed: September 5, 2024.

Padma, P. & Ahn, J. (2020). Guest satisfaction & dissatisfaction in luxury hotels: An application of big data. *International journal of hospitality management* **84**, 102318.

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Phillips, P., Zigan, K., Silva, M. M. S. & Schegg, R. (2015). The interactive effects of online reviews on the determinants of swiss hotel performance: A neural network analysis. *Tourism Management* **50**, 130–141.

Priporas, C.-V., Stylos, N., Vedanthachari, L. N. & Santiwatana, P. (2017). Service quality, satisfaction, and customer loyalty in airbnb accommodation in thailand. *International Journal of Tourism Research* **19**, 693–704.

Rinker, T. W. (2021). *sentimentr: Calculate Text Polarity Sentiment*. Buffalo, New York. Version 2.9.0.

Sievert, C. & Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*.

Sutherland, I. & Kiatkawsin, K. (2020). Determinants of guest experience in airbnb: A topic modeling approach using lda. sustainability, 12 (8), 3402.

Sutherland, I., Sim, Y., Lee, S. K., Byun, J. & Kiatkawsin, K. (2020). Topic modeling of online accommodation reviews via latent dirichlet allocation. *Sustainability* **12**, 1821.

Tabassum, A. & Patil, R. R. (2020). A survey on text pre-processing & feature extraction techniques in natural language processing. *International Research Journal of Engineering and Technology (IRJET)* **7**, 4864–4867.

Taboada, M. (2016). Sentiment analysis: An overview from linguistics. *Annual Review of Linguistics* **2**, 325–347.

Taboada, M., Brooke, J. & Stede, M. (2009). Genre-based paragraph classification for sentiment analysis. In *Proceedings of the SIGDIAL 2009 Conference*.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K. & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics* **37**, 267–307.

Taboada, M. & Grieve, J. (2004). Analyzing appraisal automatically. In *Proceedings of AAAI spring symposium on exploring attitude and affect in text (AAAI technical re# port SS# 04# 07), Stanford University, CA, pp. 158q161. AAAI Press*.

Teh, Y., Newman, D. & Welling, M. (2006). A collapsed variational bayesian inference algorithm for latent dirichlet allocation. *Advances in neural information processing systems* **19**.

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032* .

Wang, Z., Ho, S.-B. & Cambria, E. (2020). Multi-level fine-scaled sentiment sensing with ambivalence handling. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **28**, 683–697.

Wankhade, M., Rao, A. C. S. & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review* **55**, 5731–5780.

Weizenbaum, J. (1966). Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM* **9**, 36–45.

Wiebe, J. & Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. In *International conference on intelligent text processing and computational linguistics*. Springer.

Woods, W. A. (1970). Transition network grammars for natural language analysis. *Communications of the ACM* **13**, 591–606.

Xu, X. & Li, Y. (2016). The antecedents of customer satisfaction and dissatisfaction toward various types of hotels: A text mining approach. *International journal of hospitality management* **55**, 57–69.

Yan-Yan, Z., Bing, Q. & Ting, L. (2010). Integrating intra-and inter-document evidences for improving sentence sentiment classification. *Acta Automatica Sinica* **36**, 1417–1425.

Zervas, G., Proserpio, D. & Byers, J. W. (2017). The rise of the sharing economy: Estimating the impact of airbnb on the hotel industry. *Journal of marketing research* **54**, 687–705.

Zhang, G., Cui, R., Cheng, M., Zhang, Q. & Li, Z. (2020). A comparison of key attributes between peer-to-peer accommodations and hotels using online reviews. *Current Issues in Tourism* **23**, 530–537.

Appendix A

R Code

Listing 2.1: Library loading

```r
# caricamento librerie
library(tidyverse)
library(tictoc)
library(cld3)
library(tm)
library(stringr)
library(textclean)
library(textstem)
library(topicmodels)
library(ldatuning)
library(LDAvis)
library(tidytext)
library(ggplot2)
library(sf)
library(ggspatial)
library(raster)
library(sentimentr)
library(lda)
```

Listing 2.2: Importing reviews and removing those not in English

```r
reviews <- read.csv("reviews_milano.csv")
#TOTALE RECENSIONI 830436

#analisi della lingua delle recensioni
tic()
language <- detect_language(reviews$comments)
toc()


```

```r
#osservo la frequenza delle lingua
table(language)

length(which(language=="en"))/dim(reviews)[1]
#53.52% delle recensioni sono in lingua inglese

ind <- which(language=="en")
#444488 recensioni in inglese

reviews <- reviews[ind,]
#444488 recensioni rimaste
```

Listing 2.3: First pre-processing phase

```r
preprocessing1 <- function(reviews){

  txt <- reviews$comments #salvo un oggetto contenente solo le recensioni

  #rimozione caratteri HTML
  txt <- str_replace_all(txt, "<[^>]+>", " ")

  #rimozione \n e \r
  txt <- str_replace_all(txt, "\\s*\\n\\s*", " ")
  txt <- str_replace_all(txt, "\\s*\\r\\s*", " ")

  #rimuovo spazi multipli
  txt <- str_replace_all(txt, "\\s+", " ")

  #rimuovo altri simboli speciali HTML
  txt <- str_replace_all(txt, "&[a-zA-Z]+;", "")

  #converto caratteri non ascii in caratteri ascii
  txt <- str_replace_all(txt, "é", "e")
  txt <- str_replace_all(txt, "è", "e")
  txt <- str_replace_all(txt, "ê", "e")
  txt <- str_replace_all(txt, "á", "a")
  txt <- str_replace_all(txt, "à", "a")
  txt <- str_replace_all(txt, "ó", "o")
  txt <- str_replace_all(txt, "ò", "o")
  txt <- str_replace_all(txt, "ì", "i")
  txt <- str_replace_all(txt, "í", "i")
  txt <- str_replace_all(txt, "ú", "u")
  txt <- str_replace_all(txt, "ù", "u")
  txt <- str_replace_all(txt, "ç", "c")
```

```
31
32    #rimuovo caratteri non europei
33    txt <- str_replace_all(txt, "[^\\x00-\\x7F]", "")
34
35    #elimino eventuali spazi in eccesso
36    txt <- stripWhitespace(txt)
37
38    #rimuovo spazi bianchi all'inizio e alla fine
39    txt <- gsub("^[[:space:]]+", "", txt) # remove whitespace at beginning
40    txt <- gsub("[[:space:]]+$", "", txt) # remove whitespace at end
41
42    reviews$comments <- txt
43    return(reviews)
44  }
45
46
47  tic()
48  reviews <- preprocessing1(reviews)
49  toc()
```

Listing 2.4: Development of the dictionary for lemmatization

```
1   dictionary <- lexicon::hash_lemmas
2   dictionary$lemma[dictionary$token=="well"] <- "well"
3   dictionary$lemma[dictionary$token=="better"] <- "better"
4   dictionary$lemma[dictionary$token=="best"] <- "best"
5   dictionary$lemma[dictionary$token=="maria"] <- "maria"
6   dictionary$lemma[dictionary$token=="second"] <- "two"
7   dictionary$lemma[dictionary$token=="third"] <- "three"
8   dictionary$lemma[dictionary$token=="fourth"] <- "four"
9   dictionary$lemma[dictionary$token=="fifth"] <- "five"
10  dictionary$lemma[dictionary$token=="sixth"] <- "six"
11  dictionary$lemma[dictionary$token=="seventh"] <- "seven"
12  dictionary$lemma[dictionary$token=="eighth"] <- "eight"
13  dictionary$lemma[dictionary$token=="listing"] <- "listing"
14  dictionary$lemma[dictionary$token=="evening"] <- "evening"
15
16  #aggiungo righe al vocabolario per trasformare avverbi in aggettivi e altro
17  my.dictionary <- data.frame(token="beautifully", lemma="beautiful")
18  my.dictionary <- rbind(my.dictionary, c("centrally", "central"))
19  my.dictionary <- rbind(my.dictionary, c("conveniently", "convenient"))
20  my.dictionary <- rbind(my.dictionary, c("comfortable", "comfort"))
21  my.dictionary <- rbind(my.dictionary, c("easily", "easy"))
22  my.dictionary <- rbind(my.dictionary, c("enojoyable", "enjoy"))
```

```
my.dictionary <- rbind(my.dictionary, c("exactly", "exact"))
my.dictionary <- rbind(my.dictionary, c("nearby", "near"))
my.dictionary <- rbind(my.dictionary, c("nicely", "nice"))
my.dictionary <- rbind(my.dictionary, c("noisy", "noise"))
my.dictionary <- rbind(my.dictionary, c("perfectly", "perfect"))
my.dictionary <- rbind(my.dictionary, c("quickly", "quick"))
my.dictionary <- rbind(my.dictionary, c("recommendation", "recommend"))
my.dictionary <- rbind(my.dictionary, c("stylish", "style"))
my.dictionary <- rbind(my.dictionary, c("suggestion", "suggest"))
my.dictionary <- rbind(my.dictionary, c("walkable", "walk"))

dictionary <- rbind(dictionary, my.dictionary)

write.csv(dictionary, "dictionary.csv", row.names = F)
```

Listing 2.5: Second pre-processing phase

```
#funzione che conta il numero di parole in una recensione
count_words <- function(s) {
  strsplit(s, "\\s+")[[1]] %>% length()
}

#calcolo lunghezza delle recensioni
lunghezza_recensioni <- sapply(reviews$comments, count_words)
summary(lunghezza_recensioni)

#elimino tutte le recensioni con meno di 50 parole
ind_50 <- which(lunghezza_recensioni <= 50)
reviews <- reviews[-ind_50,]

#elimino recensioni duplicate
ind_dup <- which(duplicated(reviews$comments))
recensioni_dup <- unique(reviews$comments[ind_dup])
ind_dup_extended <- which(reviews$comments %in% recensioni_dup)
reviews <- reviews[-ind_dup_extended, ]

#join con tabella relativa alle strutture
listings <- read.csv("listings_milano.csv")
listings <- listings %>%
  rename(listing_id=id)

reviews_new <- left_join(reviews, listings) %>%
  select(review_id=id, listing_id, date, comments, neighbourhood_cleansed,
         latitude, longitude, room_type)
```

```r

#analizzo la frequenza delle recensioni per ogni type room
table(reviews_new$room_type) %>% prop.table()

#elimino hotel room e shared room
reviews_new <- reviews_new %>%
  filter(room_type %in% c("Entire home/apt", "Private room"))
reviews <- reviews_new

preprocessing2 <- function(reviews){

  txt <- reviews$comments

  #trasformo doppi e tripli puntini di sospensione, virgole e altri segni di
  #punteggiatura in spazi. Cosi facendo evito che due parole separate solo da tali
  #segni rimangano in seguito "attaccate" tra loro
  txt <- gsub("\\.\\.\\.", " ", txt)
  txt <- gsub("\\.\\.", " ", txt)
  txt <- gsub(",", " ", txt)
  txt <- gsub("-", " ", txt)
  txt <- gsub("/", " ", txt)
  txt <- gsub('\"', " ", txt)

  #rimuovo spazi bianchi in eccesso
  txt <- stripWhitespace(txt)

  #trasformo tutto in minuscolo
  txt <- tolower(txt)

  #converto alcune espressioni specifiche
  txt <- gsub("check in", "checkin", txt)
  txt <- gsub("checking in", "checkin", txt)
  txt <- gsub("checked in", "checkin", txt)
  txt <- gsub("check out", "checkout", txt)
  txt <- gsub("checking out", "checkout", txt)
  txt <- gsub("checked out", "checkout", txt)
  txt <- gsub("wi fi", "wifi", txt)
  txt <- gsub("as well as", " ", txt)
  txt <- gsub("dish washer", "dishwasher", txt)
  txt <- gsub("air conditioning", "aircondition", txt)
  txt <- gsub("air conditioner\\b", "aircondition", txt)
  txt <- gsub("air conditioned\\b", "aircondition", txt)
  txt <- gsub("air condition\\b", "aircondition", txt)
```

```
71  txt <- gsub("conditioner", "aircondition", txt)
72  txt <- gsub("\\bac\\b", "aircondition", txt)
73  txt <- gsub("a\\.c\\.", "aircondition", txt)
74
75  #rimuovo le stopwords classiche della lingua inglese
76  #(con l'esclusione di again)
77  stopwords <- stopwords("en")
78  stopwords <- setdiff(stopwords, "again")
79  txt <- removeWords(txt, stopwords)
80
81  #converto alcune parole specifiche (rilevate da analisi esplorativa)
82  #alcune sono relative alla differenza tra inglese britannico e americano
83  txt <- gsub("\\bmin\\b", "minute", txt)
84  txt <- gsub("\\bmins\\b", "minute", txt)
85  txt <- gsub("\\bcozy\\b", "cosy", txt)
86  txt <- gsub("\\bdome\\b", "duomo", txt)
87  txt <- gsub("\\bcathedral\\b", "duomo", txt)
88  txt <- gsub("centrale", "central", txt)
89  txt <- gsub("stazione", "station", txt)
90  txt <- gsub("appartement", "apartment", txt)
91  txt <- gsub("appartment", "apartment", txt)
92  txt <- gsub("\\bapart\\b", "apartment", txt)
93  txt <- gsub("\\bapart\\.", "apartment", txt)
94  txt <- gsub("\\bappart\\b", "apartment", txt)
95  txt <- gsub("\\bappart\\.\\b", "apartment", txt)
96  txt <- gsub("\\bflat\\b", "apartment", txt)
97  txt <- gsub("\\bapt\\b", "apartment", txt)
98  txt <- gsub("\\bapt\\.", "apartment", txt)
99  txt <- gsub("neighborhood", "neighbourhood", txt)
100 txt <- gsub("\\bneighbor\\b", "neighbour", txt)
101 txt <- gsub("transportation", "transport", txt)
102 txt <- gsub("\\belevator\\b", "lift", txt)
103 txt <- gsub("vacation", "holiday", txt)
104 txt <- gsub("\\bcenter\\b", "centre", txt)
105 txt <- gsub("milano", "milan", txt)
106 txt <- gsub("mailand", "milan", txt)
107 txt <- gsub("\\bcomfy\\b", "comfortable", txt)
108 txt <- gsub("\\bcouch\\b", "sofa", txt)
109 txt <- gsub("\\bgrazie\\b", "thank", txt)
110 txt <- gsub("\\bok\\b", "okay", txt)
111 txt <- gsub("airbnbs", "airbnb", txt)
112 txt <- gsub("\\bsubway\\b", "metro", txt)
113 txt <- gsub("\\bunderground\\b", "metro", txt)
```

```
  txt <- gsub("\\btraveller\\b", "traveler", txt)

  #converto l'apostrofo in spazio per separare, ad esempio,
  #le s del genitivo sassone dai nomi
  txt <- gsub("'","  ", txt)

  #rimuovo punteggiatura e numeri
  txt <- removePunctuation(txt,preserve_intra_word_dashes = FALSE)
  txt <- removeNumbers(txt)

  #converto nuovamente min e mins in minutes (dopo aver eliminato i numeri)
  txt <- gsub("\\bmin\\b", "minute", txt)
  txt <- gsub("\\bmins\\b", "minute", txt)

  #rimuovo altri caratteri NON alfanumerici (e NON spazi) rimasti (come le emoji)
  txt <- gsub('[^[:alnum:]\\s]', ' ', txt)

  #rimuovo eventuali parole rimaste giudicate non utili per l'analisi
  words_to_eliminate <- c("etc", "milan", "also", "can", "will", "really",
                          "many", "lot", "lots", "just", "highly", "bit",
                          "either", "else", "us")
  txt <- removeWords(txt, words_to_eliminate)

  #LEMMATIZATION
  dictionary <- read.csv("dictionary.csv") #creato in precedenza
  txt <- lemmatize_strings(txt, dictionary = dictionary)

  #rimuovo altre parole dopo la LEMMATIZATION
  words_to_eliminate <- c("apartment", "great", "place", "stay", "everything")
  txt <- removeWords(txt, words_to_eliminate)

  #rimuovo spazi bianchi in eccesso
  txt <- stripWhitespace(txt)

  #rimuovo spazi bianchi all'inizio e alla fine
  txt <- gsub("^[[:space:]]+", "", txt) # remove whitespace at beginning of
      documents
  txt <- gsub("[[:space:]]+$", "", txt) # remove whitespace at end of documents

  reviews$comments_preproc <- txt
  return(reviews)
}

```

```
tic()
reviews <- preprocessing2(reviews)
toc()
```

Listing 2.6: Creating the document term matrix

```
txt <- reviews$comments_preproc

corpus <- Corpus(VectorSource(txt))
dtm <- DocumentTermMatrix(corpus, control = list(wordLengths=c(2, Inf)))

#rimuovo i termini sparsi
dtm <- removeSparseTerms(dtm, 0.99)
```

Listing 2.7: LDA - choice of the number of topics

```
tic()
ftn <- FindTopicsNumber(dtm,
                        topics=2:30,
                        metrics=c("Deveaud2014"),
                        control=list(iter=2700, burnin=500, thin=200,
                                     seed=123),
                        return_models = TRUE)
toc()

#considero solo topics da 2 a 20
ftn_ridotto <- ftn[11:29, c(1,3)]

val.y <- ftn_ridotto$Deveaud2014[ftn_ridotto$topics==10]

plot_deveaud <- ggplot(data=ftn_ridotto, aes(x=topics, y=Deveaud2014)) +
  theme_bw() +
  geom_point() +
  geom_line() +
  geom_point(data=ftn_ridotto[ftn_ridotto$topics %in% c(6,10,13,16),], color="red")
       +
  scale_x_continuous(breaks = seq(0,20,1)) +
  #scale_y_continuous(breaks = seq(3,5,0.1)) +
  #geom_vline(xintercept = 10, linetype="dashed", color="red") +
  geom_segment(x = 10, xend = 10, y = 2.5, yend = val.y,
               color = "red", linetype="dashed") +
  labs(x="Number of topics", y="Deveaud criterion")

print(plot_deveaud)
```

Listing 2.8: LDA - topic interpretation

```r
#modello migliore con 10 topics
lda_model <- ftn$LDA_model[[21]]


ap_topics <- tidy(lda_model, matrix = "beta")

top_terms <- ap_topics  %>% # take the topics data frame and..
  group_by(topic) %>% # treat each topic as a different group
  top_n(30, beta) %>% # get the top most informative words
  ungroup() %>% # ungroup
  arrange(topic, -beta) #

wide_top_terms <- top_terms %>%
  select(topic, term) %>%
  group_by(topic) %>%
  mutate(term_id = row_number()) %>%   # Aggiungi un identificatore per i termini
  pivot_wider(names_from = topic, values_from = term, names_sort = TRUE) %>%
  select(-term_id)  # Rimuovi l'identificatore temporaneo

documents <- tidy(lda_model, matrix = "gamma") %>%
  arrange(document, topic)

documents_wide <- documents %>% spread(topic, gamma)
documents_wide <- documents_wide[order(as.integer(documents_wide$document)),]

#aggiungo topic massimo
documents_wide$max <- apply(documents_wide[2:11], 1, which.max)

#visualizzazione del modello
topicmodels2LDAvis <- function(x, ...){
  post <- topicmodels::posterior(x)
  if (ncol(post[["topics"]]) < 3) stop("The model must contain > 2 topics")
  mat <- x@wordassignments
  LDAvis::createJSON(
    phi = post[["terms"]],
    theta = post[["topics"]],
    vocab = colnames(post[["terms"]]),
    doc.length = slam::row_sums(mat, na.rm = TRUE),
    term.frequency = slam::col_sums(mat, na.rm = TRUE)
  )
}
serVis(topicmodels2LDAvis(lda_model))
```

Listing 2.9: Geographical analysis

```
 #importo zone di milano
OMI<-st_read("MI_Zone_OMI corretto_region.shp",quiet=TRUE)
OMI= st_set_crs(OMI,3003)  # Gauss-Boaga

#importo fermate della metro
MM<-st_read("MM_FERMATE.shp",quiet=TRUE)
MM= st_set_crs(MM,3003)  # Gauss-Boaga

#fermate della metro solo nel comune di Milano
MMinMI <- st_intersection(OMI, MM)

#mappa di Milano
map1 <- ggplot() +
    annotation_map_tile("thunderforestoutdoors",  zoomin=0) +
    geom_sf(data=st_boundary(OMI))

#dataset con tutte le strutture
listings <- reviews %>%
  select(listing_id, neighbourhood_cleansed, latitude, longitude) %>%
  unique()

listings <- sf::st_as_sf(listings, coords = c("longitude", "latitude"), crs=4326)
listings <- st_transform(listings, crs=3003)

#dataset con le strutture con topic prevalente "mezzi pubblici"
ind <- which(documents_wide$max==8)
listings_small <- reviews
listings_small$transport <- 0
listings_small$transport[ind] <- 1

count_transport <- listings_small %>%
  group_by(listing_id) %>%
  count(wt=transport) %>%
  arrange(desc(n))

ind <- count_transport %>%
  filter(n>5) %>%
  .$listing_id

listings_small <- listings_small %>%
  filter(listing_id %in% ind)

```

```r
listings_small <- listings_small %>%
  select(listing_id, neighbourhood_cleansed, latitude, longitude) %>%
  unique()

listings_small <- sf::st_as_sf(listings_small, coords = c("longitude", "latitude"),
    crs=4326)
listings_small <- st_transform(listings_small, crs=3003)

#dataset con le principali stazioni di Milano
stations <- geo(street = c("Piazza Duca d'Aosta",
                           "Piazza Sigmund Freud",
                           "Piazzale Luigi Cadorna",
                           "Via Cassinis, 3",
                           "Piazza Enrico Bottini"),
                city = rep("Milan", 5))
stations <- sf::st_as_sf(stations, coords = c("long","lat"),crs=4326)
stations <- st_transform(stations, crs = 3003)

#mappa di tutte le strutture con stazioni e fermate della metro
map2 <- map1 +
  geom_sf(data=listings, mapping=aes(color="Listings"), size = 0.05, shape=3) +
  geom_sf(data=MMinMI, mapping = aes(color="Metro Stops"), size = 1) +
  geom_sf(data=stations, mapping = aes(color="Train Stations"), size = 1) +
  ggspatial::annotation_north_arrow(which_north = "true") +
  ggspatial::annotation_scale(location="br") +
  scale_color_manual(values = c("Metro Stops" = "blue",
                                "Listings" = "red",
                                "Train Stations"="green")) +
  labs(color = "Legend") +
  theme(legend.position="inside", legend.position.inside = c(0.35,0.1),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.1, "cm"),
        legend.title = element_blank(),
        legend.key.spacing.y = unit(0.1, "cm"),
        #legend.title = element_text(size=8, face="bold"),
        legend.background = element_rect(fill = "white", color = "black", linewidth
            = 0.1))
print(map2)

#mappa con solo le strutture selezionate in precedenza
map3 <- map1 +
  geom_sf(data=listings_small, mapping=aes(color="Listings"), size = 0.5, shape=3)
      +
```

```
  geom_sf(data=MMinMI, mapping = aes(color="Metro Stops"), size = 1) +
  geom_sf(data=stations, mapping = aes(color="Train Stations"), size = 1) +
  ggspatial::annotation_north_arrow(which_north = "true") +
  ggspatial::annotation_scale(location="br") +
  scale_color_manual(values = c("Metro Stops" = "blue",
                                "Listings" = "red",
                                "Train Stations"="green")) +
  labs(color = "Legend") +
  theme(legend.position="inside", legend.position.inside = c(0.35,0.1),
        legend.text = element_text(size = 8),
        legend.key.size = unit(0.1, "cm"),
        legend.title = element_blank(),
        legend.key.spacing.y = unit(0.1, "cm"),
        #legend.title = element_text(size=8, face="bold"),
        legend.background = element_rect(fill = "white", color = "black", linewidth
            = 0.1))
print(map3)
```

Listing 2.10: Sentiment analysis

```
#uso recensioni preprocessate solo con la prima funzione
txt_sa <- reviews$comments

#converto in minuscolo
txt_sa <- tolower(txt_sa)

tic()
sentences <- get_sentences(txt_sa)
sentiment_score <- sentiment_by(sentences, polarity_dt=lexicon::hash_sentiment_
    huliu)
toc()

#aggiungo il sentiment score a reviews
reviews$sentiment_score <- sentiment_score$ave_sentiment

ind_pos <- which(reviews$sentiment_score>0)
reviews_pos <- reviews[ind_pos,]

ind_neg <- which(reviews$sentiment_score<0)
reviews_neg <- reviews[ind_neg,]

ind_nul <- which(reviews$sentiment_score==0)
#ci sono anche 44 recensioni nulle
```

Listing 2.11: LDA on negative reviews - choice of the number of topics

```
txt <- reviews_neg$comments_preproc

corpus <- Corpus(VectorSource(txt))
dtm <- DocumentTermMatrix(corpus, control = list(wordLengths=c(2, Inf)))

#rimuovo i termini sparsi
dtm <- removeSparseTerms(dtm, 0.99)

#SCELTA NUMERO TOPICS
tic()
ftn <- FindTopicsNumber(dtm,
                        topics=2:30,
                        metrics=c("Deveaud2014"),
                        control=list(iter=2700, burnin=500, thin=200,
                                     seed=123),
                        return_models = TRUE)
toc()

#considero solo topics da 2 a 20
ftn_ridotto <- ftn[11:29, c(1,3)]

val.y <- ftn_ridotto$Deveaud2014[ftn_ridotto$topics==3]

plot_deveaud <- ggplot(data=ftn_ridotto, aes(x=topics, y=Deveaud2014)) +
  theme_bw() +
  geom_point() +
  geom_line() +
  geom_point(data=ftn_ridotto[ftn_ridotto$topics==3,], color="red") +
  scale_x_continuous(breaks = seq(0,20,1)) +
  #scale_y_continuous(breaks = seq(3,5,0.1)) +
  #geom_vline(xintercept = 10, linetype="dashed", color="red") +
  geom_segment(x = 3, xend = 3, y = 3, yend = val.y,
               color = "red", linetype="dashed") +
  labs(x="Number of topics", y="Deveaud criterion")

print(plot_deveaud)
```

Listing 2.12: LDA on negative reviews - topic interpretation

```
lda_model <- ftn$LDA_model[[28]]
lda_model

```

```r
ap_topics <- tidy(lda_model, matrix = "beta")

top_terms <- ap_topics  %>% # take the topics data frame and..
  group_by(topic) %>% # treat each topic as a different group
  top_n(30, beta) %>% # get the top most informative words
  ungroup() %>% # ungroup
  arrange(topic, -beta) #

wide_top_terms <- top_terms %>%
  select(topic, term) %>%
  group_by(topic) %>%
  mutate(term_id = row_number()) %>%   # Aggiungi un identificatore per i termini
  pivot_wider(names_from = topic, values_from = term, names_sort = TRUE) %>%
  select(-term_id)  # Rimuovi l'identificatore temporaneo

documents <- tidy(lda_model, matrix = "gamma") %>%
  arrange(document, topic)

documents_wide <- documents %>% spread(topic, gamma)
documents_wide <- documents_wide[order(as.integer(documents_wide$document)),]

serVis(topicmodels2LDAvis(lda_model)
```

Listing 2.13: sLDA

```r
#conversione document term matrix nel formato richiesto da slda
dtm_m <- as.matrix(dtm)

documents <- lapply(1:nrow(dtm_m), function(i) {
  word_indices <- which(dtm_m[i, ] > 0) - 1  # 0-indexed word identifiers
  word_counts <- dtm_m[i, word_indices + 1]  # Frequencies of those words
  matrix_data <- rbind(word_indices, word_counts)
  # Ensure the matrix is of integer type
  matrix_data <- matrix(as.integer(matrix_data), nrow = 2)
  return(matrix_data)
})

vocab <- colnames(dtm)  # Assuming dtm has column names as vocabulary

response <- reviews$sentiment_score

#iperparametri (li considero fissati)
alpha <- 1 #iperparametro dirichlet per le topic proportions
eta <- 0.1 #iperparametro dirichlet per le topic multinomials
```

```
sigma2 <- 0.25

#numero iterazione (testate su un campione)
num.e.iterations <- 100
num.m.iterations <- 50

num_topics <- 5:11
slda_list <- list()

for(i in 1:length(num_topics)){
  print(paste("Modello con", num_topics[i], "topics", sep=" "))
  set.seed(123)
  tic()
  slda_model <- slda.em(documents=documents,
                        K=num_topics[i],
                        vocab=vocab,
                        num.e.iterations=num.e.iterations,
                        num.m.iterations=num.m.iterations,
                        alpha=alpha,
                        eta=eta,
                        annotations=response,
                        params=rep(0, num_topics[i]), #inizializzo coefficienti a 0
                        variance=sigma2,
                        logistic = FALSE,
                        #lambda = 10, #serve solo se regularise=TRUE
                        regularise = FALSE,
                        method = "sLDA",
                        trace = 0L,
                        MaxNWts=3000,
                        initial = NULL)
  toc()
  slda_list[[i]] <- slda_model
}

#scelgo modello con 9 topics
slda_model <- slda_list[[5]]

#visualizzo parole più probabili per ciascun topic
Topics <- apply(top.topic.words(slda_model$topics, 30, by.score=TRUE),
                2, paste, collapse=" ")

Topics <- c("Proximity to transport", "Feeling and experience", "Issues",
            "Unit amenities", "Host welcoming", "Unit evaluation",
```

```
            "Neighbourhood", "Check-in/out", "Host evaluation")

#creo matrice con le percentuali di topic per ogni documento
document_topics <- slda_model$document_sums
tot <- apply(document_topics, 2, sum)

document_topics <- rbind(document_topics, tot)
rownames(document_topics) <- c(paste("Topic", 1:9, sep=" "), "Sum")

for(i in 1:ncol(document_topics)){
  for(j in 1:(nrow(document_topics)-1)){
    document_topics[j,i] <- document_topics[j,i]/document_topics[nrow(document_
        topics),i]
  }
}

#grafico dei coefficienti
coefs <- data.frame(coef(summary(slda_model$model)))

theme_set(theme_bw())

coefs <- cbind(coefs, Topics=factor(Topics, Topics[order(coefs$Estimate)]))

coefs <- coefs[order(coefs$Estimate),]

ggplot(coefs, aes(x=Topics, y=Estimate, colour=Estimate, size=abs(t.value))) +
  geom_errorbar(width=0.5, aes(ymin=Estimate-Std..Error,ymax=Estimate+Std..Error))
      +
  coord_flip()
```

Aknowledgements

Al termine di questo elaborato, molte sono le persone che vorrei ringraziare.

Innanzitutto grazie alla mia relatrice, la prof.ssa Migliorati, che mi ha seguito con disponibilità e gentilezza durante tutta la fase di stesura, dandomi preziosi consigli. Grazie anche al correlatore, prof. Ascari: in un momento di difficoltà si è reso subito disponibile e confrontarmi con lui mi è stato sicuramente d'aiuto per portare a termine questo lavoro.

Rimanendo all'interno del mondo universitario, vorrei ringraziare tutti gli amici che ho incontrato lungo questo percorso. Grazie innanzitutto a tutto il gruppo STAT (Antonio, Camilla, Caterina, Chiara, Flavio, Luna, Rebecca, Riccardo e Sara): abbiamo condiviso ore di studio, progetti di gruppo, scleri quando non riuscivamo a prevedere la mancia dei taxisti a New York ma anche aperitivi, partite a Padel e gite a Gardaland. Grazie a Roberto: ci eravamo già incontrati ai tempi della triennale ma i molti viaggi in treno fatti assieme hanno sicuramente aiutato a conoscerci meglio. E infine grazie a tutti gli altri, anche quelli con cui ho semplicemente seguito un corso. Dopo una laurea triennale vissuta prevalentemente "a distanza", è stato bello poter condividere questi due anni con voi.

Grazie alla mia famiglia: ai miei genitori, Franca e Luigi, e a mio fratello Davide. Spenderò poche parole per loro non per poca importanza, ma perchè sono sicuro che già sanno quanto sono importanti per me, nonostante qualche litigata ogni tanto.

Tanti sono gli amici da ricordare, sia quelli storici che quelli incontrati nel corso di questi due anni.

Grazie innanzitutto ad Angelo, Luca, Giorgio, Filippo, Brando e Stefano: anche se con alcuni ci siamo un po' staccati nell'ultimo periodo, tutti i bei momenti vissuti assieme ci legheranno sempre, in un modo o nell'altro. Grazie ad Elena, compagna di (lunghe) camminate e chiacchierate (e non solo naturalmente): grazie per esserci sempre stata, per avermi sempre saputo ascoltare e perchè so

che posso davvero contare su di te. E, per chiudere gli amici storici, grazie a tutto il gruppo educatori, compresi gli ex: sebbene molti di voi non li conoscevo prima che iniziassimo a "lavorare" insieme in oratorio, sono nate delle amicizie vere, che vanno ben al di là del collaborare insieme. Sono certo che anche con voi, qualunque cosa succederà in futuro, tutto quello che abbiamo condiviso ci terrà in qualche modo legati per sempre.

Colgo l'occasione per ringraziare anche tutti i ragazzi che ho incontrato in questi anni in oratorio: non so se vi ho lasciato qualcosa, ma so che voi avete lasciato tanto a me.

Concludo menzionando le persone conosciute nel corso degli ultimi due anni. Vorrei ringraziare innanzitutto Francesco: in poco più di un anno, nonostante non ci vediamo neanche così spesso, sento di aver trovato un amico. Grazie poi a tutti quelli con cui ho condiviso l'esperienza della GMG: anche con voi sono bastate due settimane trascorse insieme per creare un bel gruppo. Menzione speciale a quelli del gruppo studio: il trovarsi assieme a studiare e condividere le pause e pranzi ha sicuramente reso la preparazione di questa tesi più piacevole.

A tutti voi e anche a tutti quelli che non ho menzionato esplicitamente GRAZIE: per essere in qualche modo parte della mia vita e perchè sono certo che condividerete con me la gioia di questo traguardo.