UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Enlik -

Topic Modeling for Requirements Engineering: An Analysis of Ridesharing App Reviews

Master's Thesis (30 ECTS)

Supervisor(s): Tahira Iqbal
Kuldar Taveter, PhD

Tartu 2022

Topic Modeling for Requirements Engineering: An Analysis of Ridesharing App Reviews

Abstract:
Research in AI technology has become more popular today, helped by the rising of data volumes, powerful algorithms, and easier access to high-performance computing. Natural Language Processing (NLP) as a subset of AI technology plays an important role in the future of conversational AI because of its capability to interpret our natural language. On the other hand, ridesharing app industry is growing exponentially, helped by the rise of mobile device technology and the need for faster and cheaper mobility options. In the current thesis, we provide an overview of the current industrial practices in the development of NLP applications for analyzing app reviews and identify the gap in the state-of-the-art practices. To bridge the gap, this thesis proposes a method to extract information from the app reviews, with the goal to help ridesharing app developers to identify which features are most needed and which are less important. The proposed method is compared with the other similar methods and is validated with Europe's top 10 ridesharing apps, including Bolt, Uber, Blablacar, Cabify, Via, Getaround, OlaCabs, Taxi.eu, Freenow, and Yandex Go. This contribution helps the ridesharing app developers to determine the requirements for developing their apps.

Keywords:
Natural Language Processing, Data-Driven Requirements, Ridesharing, App Reviews

CERCS:
P170 Computer science, numerical analysis, systems, control

Teemade modelleerimine nõuete analüüsiks: sõidujagamisrakenduste arvustuste analüüs

Lühikokkuvõte:
Tehisintellekti tehnoloogiate alane uurimistöö on tänapäeval muutunud populaarsemaks, millele aitavad kaasa andmemahtude kasv, kõrge jõudlusegs algoritmid ja hõlpsam juurdepääs suure jõudlusega andmetöötlusele. Loomuliku keele töötlemine osana tehisintellekti tehnoloogiatest mängib olulist roll vestluspõhise tehisintellekti tulevikus, sest suudab tõlgendada meie loomulikku keelt. Teisest küljest, sõidujagamisrakenduste tööstus kasvab plahvatuslikult, millele aitavad kaasa mobiilseadmete tehnoloogia levik ning vajadus kiiremate ja odavamate liikumisvõimaluste järele. Käesolevas magistritöös anname ülevaate tööstuslikest sõidujagamisrakenduste arvustuste analüüsi loomuliku keele rakendustest ja selgitame välja praeguste praktikate puudujäägid. Puudujääkide leevendamiseks paneme ette meetodi teabe eraldamiseks rakenduste arvustustest eesmärgiga aidata sõidujagamisrakenduste arendajatel tuvastada, milliseid funktsioone on kõige rohkem vaja ja millised on vähem olulised. Töös võrreldakse ettepandud meetodit teiste vastavate meetoditega ja valideeritakse Euroopa kümne kõige populaarsema sõidujagamise rakendusega, kaasa arvatud Bolt, Uber, Blablacar, Cabify, Via, Getaround, OlaCabs, Taxi.eu, Freenow, ja Yandex Go.. See panus aitab sõidujagamisrakenduste arendajatel määrata kindlaks nõuded rakenduste arendamiseks.

Võtmesõnad:
loomuliku keele töötlus, Andmepõhised nõuded, Sõidujagamine, Rakenduste arvustused

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Requirements engineering as the initial phase of software engineering becomes critical to ensuring that the software development process will succeed. Requirements engineering consists of four core activities, including elicitation, documentation, validation, and management [PR15]. In this thesis, we will concentrate on the elicitation activity to understand the user's expectations from the ridesharing app. Requirements elicitation is discovering some features that our users need through several methods such as interviews, surveys, brainstorming sessions, or currently available user reviews in the app store. Elicitation using user reviews has a limitation. It can be very time-consuming to analyze a large number of user reviews if we're doing manual analysis.

Machine learning is a branch of artificial intelligence and computer science that plays an important role in uncovering key insights from a data science project. It can imitate the way of human learning, supported by improved data quality and algorithms to make it better. There is a lot of data in the digital world today, generated not only by people but also by computers, phones, and other smart devices. These generated data will continue to grow in the coming years, as stated in Statista[1] research department. As the volume of data surpasses the ability of humans to make sense of it and manually analyze it, we will turn increasingly to the machine learning framework that can learn from the data automatically. These powerful capabilities can be applied to wide range of fields, from skin cancer detection[MVK+20], self-driving vehicles[SDP+21], spam detection[MGK+21], and many more. In correlation with this thesis, following the exponential growth of the app reviews text dataset, app developers will require machine learning to assist in identifying relevant app requirements for further app development and continuous improvement.

Natural language processing (NLP) techniques have been used in many different subjects such as language translation[KAA20], predictive text[Ind15], sentiment analysis[KG15], chatbot[LBP+18], and many more. NLP has boosted the interest in requirement engineering (RE) research, as RE is one of the most natural language-intensive fields in the software engineering area [FZA21]. This close relationship between NLP and RE becomes the source of inspiration for researchers to seek to apply NLP techniques and tools in the step of processing requirements texts [ZAF+21]. In this past decade, NLP techniques have seen a tremendous breakthrough, especially after the development of game-changing technologies generally classified under the umbrella of deep learning [YHPC18]. It created stronger synergies between NLP and RE, especially with the recent widespread availability of natural language (NL) content relevant to RE, such as app reviews from users in some popular app stores.

[1] https://www.statista.com/statistics/871513/worldwide-data-created/

App reviews from the two biggest mobile app platforms, Google Play Store[2] and Apple App Store[3] become one of the biggest free available sources for app developers to get users' feedback about their mobile apps. The positive reviews can lead to more user attraction, but negative reviews can bring a sales loss. Besides that, it also can help app developers to cope with the future development plan through information like feature requests, bug reports, and user experience.

1.2 Problem Statement

The number of app reviews from the most popular ridesharing app, such as Uber [ond19], can be massively huge, with up to 10.1 million text reviews according to the Google Play Store page of Uber app[4] per July 2022. Unfortunately, not all app reviews are informative and useful for app developers. App reviews in the app store are also mixed with many different languages worldwide, which will be irrelevant for those countries that only speak English, and vice versa. Some reviews also contain uninformative reviews that need to be filtered out to create high-quality input data. Analyzing this massive number of reviews with its variance will be time-consuming and very difficult to process, possibly being error-prone for app developers.

1.3 Objectives

The main objective of this thesis is to create an analysis report based on the ridesharing app reviews that will be useful for the app developer in the purpose of getting to know what their customer wants with their app. The scope of this thesis project is currently focused only on the field of ridesharing apps.

To avoid the manual approach, which is time-consuming, we are applying some automation approaches based on the NLP approaches using Python programming language and its supporting libraries. Getting started with some open-source projects, we're doing the preprocessing steps, such as removing non-English reviews, filtering out inconsistent and uninformative reviews, and applying topic modeling to analyze the app reviews.

[2] https://play.google.com/store

[3] https://www.apple.com/app-store/

[4] https://play.google.com/store/apps/details?id=com.ubercab

1.4 Research Questions

Based on the problems mentioned in the previous section, we formulated the following two research questions (RQs) to guide our research.

- **RQ1**: What kind of NLP approaches are more effective for automatic app review classification?

  Many different NLP approaches are currently available. We will focus only on the approaches that are available as open-source projects and have good documentation.

  Answering RQ1 involves:

  - Comparing the performance of different NLP approaches and currently widely used Pre-Trained Models (PTM) for NLP research

- **RQ2**: What topics should the app developer focus on based on the analysis of topic proportion with its average rating and sentiment scores?

  Using topic proportion, average ratings, and sentiment scores, we can measure each topic's performance and create the "most important topics" report based on these results.

  Answering RQ2 involves:

  - Analyzing the proportion of topics across all top-10 ridesharing apps
  - Comparing the average rating and sentiment scores for each topic per each ridesharing app, in order to get the most important topics that need to be focused on

1.5 Contribution

This thesis work will contribute to creating a text analysis for ridesharing app reviews which will help the app developer to use this analysis for future development in the domain of ridesharing apps. The proposed approach used 100,000+ app reviews from Europe's top 10 ridesharing apps.

1.6 Outline of the Thesis

The thesis is structured into three main chapters with the conclusion, references, and appendices. The remainder of the thesis is structured as follows. Chapter 2, **Literature Review**, discusses some related works of literature to this thesis. Chapter 3, **Study Setup**, describes a detailed description of the analysis process, starting from how we choose the top ten ridesharing apps in Europe, data preprocessing, and building the text corpus for

the following data analysis. Chapter 4, **Implementation**, explains our proposed topic modeling techniques. Chapter 5, **Findings and Analysis**, reports on the findings and analysis for our project. Chapter 6, **Conclusion**, discusses the key results, limitations, and future work for this thesis. For the last two chapters, **References**, include related works of literature and articles, and **Appendices**, stated all the necessary information that becomes a part of this thesis project.

2 Literature Review

In the past few years, researchers have proposed many new approaches to extract features from app reviews with different methodologies automatically. However, some limitations prevent app developer teams from using the information in the app reviews. First, app stores have a large number of app reviews, which require a considerable effort to analyzing them. Second, the quality of app reviews varies widely, from helpful suggestions and ideas for app development to emotional comments. Third, one review usually contains a mix of sentiments, which makes it harder to filter positive, neutral, and negative feedback. In this chapter, we discussed some literature in the field of app review analysis that tried to tackle some of these limitations.

Guzman et al. [GM14] proposed an automated approach to identify fine-grained features in the app reviews, then give the sentiment scores from the identified features, and use LDA topic modeling techniques to group these features into more meaningful high-level features. The data source includes 32210 app reviews from seven apps as the training data and 2800 manually peer-analyzed app reviews as the test data. Using Collocation Finding Algorithm to get the collection of words, SentiStrength to get sentiment scores, and LDA topic modeling to group several features into a set of words that construct a topic. The result stated that feature extraction has high precision results for apps with short reviews (simply praise or dispraise). They also found that their approach works well for all app categories except games and high precision results for apps with short reviews.

Di Sorbo et al. [DSPA+17] identified the difficulty with manual analysis by the app developer when it has a massive amount of app review data and the unstructured nature of their content. Their data sources comprised 2622 app reviews, mined from three different app stores and relating to 12 apps. They propose SURF (Summarizer of User Reviews Feedback) to extract specific topics and then group each app review according to its topic with a report in XML format.

The study of Rekanar et al. [ROB+21] tried to find out the issue of the ineffectiveness of the digital contract tracing app to be aware of user concerns. Using 1287 app reviews from Google Play Store and Apple App Store of the Irish HSE COVID-19 contact tracing app, they analyzed the sentiment analysis to classify positive, negative, and neutral sentiment from each app review. They used manual analysis because the precision and recall value from the current automated sentiment analysis is still not good enough for their research. Another reason is that the total number of analyzed app reviews is not huge and is suitable for manual analysis. The final result is that most of the negative comments were aimed at the app's performance and usability.

The study by Binkhonain and Liping Zhao [BZ19] compared 24 related papers related to machine learning (ML) algorithms for identification and classification purposes. Of the 24 selected papers, only 20 have reported their performance measures. There are a total of 16 different ML algorithms, categorized into seven supervised learning (SL), four

unsupervised learning (USL), and five semi-supervised learning (SSL). Rule-Based (RB) has the highest precision scores, but only one study reported this algorithm. Support Vector Machines (SVM), Naive Bayes (NB), and Decision Tree (DT) have multiple reports from different studies, but their performance varies from study to study. The result showed that supervised learning (SL), such as SVM and NB, performed better than unsupervised learning (SSL), such as Latent Dirichlet Allocation (LDA) and K-means. Their study stated that SVM and NB have the best performance, while SVM became the most frequently used algorithm. ML algorithms perform better when features are created with individual words instead of phrases. The best result was also achieved when the original word was used before stemming, or lemmatization pre-process. To tackle some challenges in the implementation of ML algorithms for Requirements Engineering (RE) purposes, the author suggested the creation of shared datasets in order to take full advantage of supervised ML algorithms. Moreover, for better result in comparison and also calling for the close collaboration between RE and ML communities to address many challenges in the development of real-world applications of ML to RE.

The study of Shah et al. [Sha] combined the existing app review classification model with app feature extraction techniques to develop the tool named REVSUM to extract information from app reviews for future software maintenance and release planning activities. The author also prepared the labeled review dataset from previous studies in order to help to train the new model. It stated that having annotated app reviews in the training data will improve recall scores at the cost of a drop in precision.

The study of Jung Akim [aK20] utilized 12,566 Netflix app reviews from the Amazon app store and compared the classification result from LDA Topic Modeling and BERT Pre-Trained ML model. They found that LDA model works well for app reviews with similar keywords, while BERT model is sensitive to sentiments, especially sarcastic reviews. Both approaches have only 43% classification agreement with 6,283 unseen reviews, and BERT occasionally works better to detect sarcasm reviews.

The SAFE approach proposed by Johann et al. [JSABM17] is a simple rule-based approach to extract and match the app features from both app descriptions and app reviews. The authors did a manual analysis of app descriptions from 10 apps in the Apple App Store and identified the most frequent text patterns used to tag each app's features. For the evaluation, the author used a random sample of 5000 app reviews for each app. SAFE first apply some text preprocessing steps, including filtering three types of sentences: URL, email address, and quotations. Then, word-tokenizes each sentence and attach a POS (part-of-speech) tag to each token. For the feature matching, SAFE used the synonym sets of the words. For example, "take photo" and "capture image" is a match. Regarding the study result, SAFE significantly improved precision and recall scores compared to the previous study. Despite the encouraging results, we will need a machine learning approach for the future improvement of SAFE.

SIMBA (SIMilarity Based Approach) from Oehri et al. [OG20] is a fine-grained

approach for identifying and measuring similarity between feedback across different platforms and languages. Their dataset comprised 464,330 app reviews across four platforms: Google Play, App Store, Twitter, and Facebook. A total of 4,100 manually annotated reviews by two annotators were used for the evaluation process. SIMBA has a translation step for non-English app reviews before transforming it into the preprocessing steps. It combines currently available approaches such as NLP, lexical sentiment analysis, and supervised ML. For the classification in SIMBA, the authors used Multinomial Naive Bayes (MNB) and divided it into three categories: "feature request", "bug report", and "other." The results showed that SIMBA had a 79% average correlation with manually annotated app reviews, which indicated a strong positive correlation with human annotation. Regarding future improvement, the authors mentioned adding lexical databases to handle common slang words in app reviews and creating ML models from a more diverse dataset on different platforms.

Table 1 provides the summary of the research papers we have discussed, including their dataset, methodology, and the result. The most used domain in these research papers is the popular mobile apps in the App Store, such as Netflix for entertainment, Whatsapp for communication, and TripAdvisor for travel. By implementing different ML algorithms, these papers are trying to extract information from app reviews automatically while also measuring their performance result. Support Vector Machines (SVM) and Naive Bayes (NB) are the best two ML algorithms discussed in many studies. SVM and NB also give the best result in terms of precision and recall score.

After reviewing some related works above, we have found some research gaps that we want to explore in this thesis. Even though many different mobile app categories have been discussed, the domain ridesharing app category has not become the main focus of these researches. Short reviews that only include praise or dispraise (for example: "I love it," "I hate this app") according to Guzman et al. [GM14], gave no information for analysis, which will be handled in this thesis in the NLP preprocessing steps. Manual review by Rekanar et al. [ROB$^+$21] is time-consuming work that leads to a small number of analyzed reviews and has the possibility of losing some important reviews that have not been analyzed. This manual approach leads to our decision to use automatic approaches in order to analyze a large number of app reviews. In correlation to our first research question (RQ1), which leads to finding some open-source NLP projects that can help us analyze a massive number of text reviews using automation ML frameworks. Meanwhile, our second research question (RQ2) will try to analyze the topics from the app review not only from its sentiment score but also using its rating score, which is missing from the study of Guzman[GM14]. Finally, we can fill the resulting gap for the domain of the ridesharing app category using the coherence of app review with its rating and sentiment scores.

Table 1. Overview of Literature Review Including its Dataset, Methodology, and Result

Paper Name	Dataset	Methodology	Result
A Fine Grained Sentiment Analysis of App Reviews [GM14]	**Traning data:** total 32,210 reviews from 7 apps (Apple App Store and Google Play Store) **Test data:** 2800 manually peer-analyzed reviews	**Collocation Finding Algorithm** Collection of words that co-occur unusually often **SentiStrength** Lexical sentiment extraction tool **LDA Topic Modelling** Grouping several different features into a set of words	**Feature Extraction** Works well for all app categories except games (AngryBird), with the possible explanation that game app features can be described in different ways. High precision results for apps with short reviews (usually only include praise or dispraise), as it will be filtered by this automated approach to help app developers focus on the more informative reviews. **Sentiment Analysis** Strong positive correlation between the sentiment score and truth set sentiment score
SURF: Summarizer of User Reviews Feedback [DSPA+17]	2622 app reviews, mined from 3 different app stores and relating to 12 apps	**SURF tool for analyze large amount of app reviews:** Extract the specific topics (e.g., UI improvements, security/licensing issues, etc.); Identify the specific task (e.g., bug fixing, feature enchancement, etc.); Present the result in the form of a condensed, interactive, and structured agenda of recommended software changes	Output report in XML format; Can be filtered along a two-level hierarchy: 1. the sentences are grouped together according their topics (e.g., App, GUI, etc.) 2. sentences in each topic are grouped by intention categories, which were assigned during the intention classification step

Sentiment analysis of user feedback on the HSE's Covid-19 contact tracing app [ROB+21]	1287 user reviews from Google Play Store / Apple App Store of Irish HSE Contact Tracker app	Manual analysis because the precision and recall rates from current automated sentiment analysis are still not perfect	Classification of the issues (Pillar); Positive / Neutral / Negative sentiment; Review Segment; Most of negative comments were aimed at a performance and usability
A review of machine learning algorithms for identification and classification of non-functional requirements [BZ19]	Using Wohline's snowballing method to listing down the papers required for this research; **24 related papers**: IEEEXplore (10 papers) Springer Link (8 papers) Science Direct (3 papers) ACM Digital Library (2 papers) Semantic Scholar (1 paper)	**Snowballing approach** alternative to the traditional systematic literature review (SLR) approach; Backward snowballing (looking at the reference list of a paper); Forward snowballing (looking at the citations in which the paper is actually cited) **Extracting and Synthesizing the data** extract seven data items from each paper; using constant comparison method (CCM) and summarization for qualitative analysis and synthesizing the extracted data;	**RQ1:** 16 different ML algorithms from 24 selected papers, categorized into: 7 supervised learning, 4 unsupervised learning, 5 semi-supervised learning (SSL) **RQ2:** Revealed a general process pattern for applying ML-based approach, divided into 3 major phases: the text preparation phase, the learning phase, the evaluation phase **RQ3:** From 24 selected papers, only 20 of them have reported their performance measures **Key Findings**: ML-based approaches give more than 70% accuracy; Overall, SL perform better than USL; ML algorithms produce the best result when using original word, instead of stemming and lemmatization

Extracting information from app reviews to facilitate software development activities [Sha]	**SHAH dataset:** labeled review dataset belonging to different app categories; review datasets along with the annotation guidelines (AGs) from previous studies	**REVSUM:** Combined review classification and automatic feature extraction methods **Review Classification Model:** Indicate that the traditional classification model (using Bag of Words features) gave result that can be competed with deep learning model	Using the context information from the previous or next sentences of app review can improve the performance of classification models **App Feature Extraction:** SAFE has high recall, low precision - Supervised CRF (Conditional Random Field) model has high precision, low recall (opposite of SAFE); Having annotated app reviews in the training set enables to improve recall at the cost of the drop in precision **Competitive Analysis:** REVSUM, as useful tool for information extraction to be used for software maintenance and release planning activities
Netflix App review Topic Modeling [aK20]	12,566 app reviews from Netflix app in Amazon App Store	LDA Topic Modeling BERT Pre-Trained Model	LDA-Mallet and BERT have only 43% classification agreement with 6,283 unseen reviews. LDA topic model works well with reviews that have words that are coherent with the context. Not working very well with sarcastic reviews. BERT model occasionally works better when detecting sarcastic reviews.

SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews [JSABM17]	App descriptions and reviews of 10 apps in category "productivity" from the Apple App Store. For the evaluation, a random sample of 5000 reviews for each app	**Text Preprocessing** Filters 3 types of sentences: sentences contain URL, email address, and quotations; bullet points and multiple symbols are removed; word-tokenizes each sentence and attach POS (part-of-speech) tags to each word token. **Application of SAFE pattern** Analyze and decompose sentence; extract raw features; remove duplicate and noise;	**Feature Extraction from App Descriptions** Based on 10 apps and 197 features (20 per app), they got overall average precision of 55.9% and a recall of 43.4% **Feature Extraction from the App Reviews** Recall of 70.9% and precision of 23.9% **Matching Features in the Descriptions and the Reviews** Recall of 56% and precision of 70.4%
Same Same but Different: Finding Similar User Feedback Across Multiple Platforms and Languages [OG20]	464,330 user feedback entities from four different platforms, for a duration timespan of 60 days. 4,100 manually annotated feedback.	SIMBA (SIMilarity Based Ap- proach), a fine-grained approach for identifying and measuring similarities between feedback written across different platforms and languages. Combine some approaches such as automatic machine translation, NLP, lexical sentiment analysis, and supervised ML.	SIMBA strongly correlates to human judgment, with an average correlation of 79% for cross-platform user feedback written in English and 78% in four different languages. SIMBA works slightly better on longer feedback than on feedback samples including feedback of less that five words.

3 Study Setup

This study focused on the ridesharing app category. The main goal of our study is to automatically identify app features as mentioned in the ridesharing app review. We select the top ten ridesharing apps in Europe based on these articles from CNBC [cnb19] and Onde [ond19]. After we selected ten ridesharing apps, for each app, we used AppAnnie to find information about its total downloads, release date, and latest update based on data in November 2020, when we finished the data collection process. For all ten ridesharing apps, Table 2 shows information from Google Play Store and Table 3 shows information from Apple App Store. In AppAnnie, the total downloads and latest updates from Apple App Store were not available for free users, and we are not able to include them.

Table 2. Top 10 Ridesharing Apps in Europe in Google Play Store (data per November 2020)

No	App Name	Total Downloads	Release Date	Latest Update
1	Bolt	10,000,000+	Aug 19, 2013	Nov 12, 2020
2	Uber	500,000,000+	Jan 28, 2012	Nov 10, 2020
3	BlaBlaCar	50,000,000+	Jan 20, 2012	Nov 10, 2020
4	Cabify	10,000,000+	Jan 12, 2012	Nov 11, 2020
5	Via	500,000+	Aug 5, 2014	Nov 8, 2020
6	Getaround	1,000,000+	Nov 17, 2014	Nov 9, 2020
7	Ola Cabs	100,000,000+	Jun 2, 2012	Nov 2, 2020
8	Taxi.eu	500,000+	Jan 18, 2012	Feb 27, 2020
9	Free Now	10,000,000+	Jan 18, 2012	Nov 11, 2020
10	Yandex Go	50,000,000+	Oct 25, 2011	Nov 10, 2020

Table 3. Top 10 Ridesharing Apps in Europe in Google Play Store (data per November 2020)

No	App Name	Release Date
1	Bolt	Jul 24, 2013
2	Uber	May 21, 2020
3	BlaBlaCar	Apr 13, 2010
4	Cabify	Nov 14, 2011
5	Via	Jul 1, 2013
6	Getaround	Jan 27, 2011
7	Ola Cabs	Jul 11, 2012
8	Taxi.eu	Sep 30, 2011
9	Free Now	May 17, 2011
10	Yandex Go	Oct 26, 2011

3.1 Overview of Study Setup

For this study, we use data mining techniques, natural language processing, LDA Topic Modeling LDA and BERTopic. Figure 1 shows an overview of the study setup.

First, we collect the app reviews from ten ridesharing apps and extract text comments and rating scores from each review. The collection of the app review process will be discussed in chapter 3.2. Then, the preprocess data step will be discussed in chapter 3.3 when we preprocess the text data using some NLP techniques. The following steps apply the preprocessed reviews into Topic Modeling and BERTopic methods to extract a list of topics discussed in chapters 4.1 and 4.2. This list of topics from both methods will be compared to get the final topics discussed in chapter 5.1. In chapters 5.2 and 5.3, we will calculate the proportion of each topic across all ten ridesharing apps using these final topics. Furthermore, average rating and sentiment score will be calculated for each final topic to create a final analysis about specific topics that ridesharing app developers need to prioritize. In the following, we explain the main steps of our study setup.

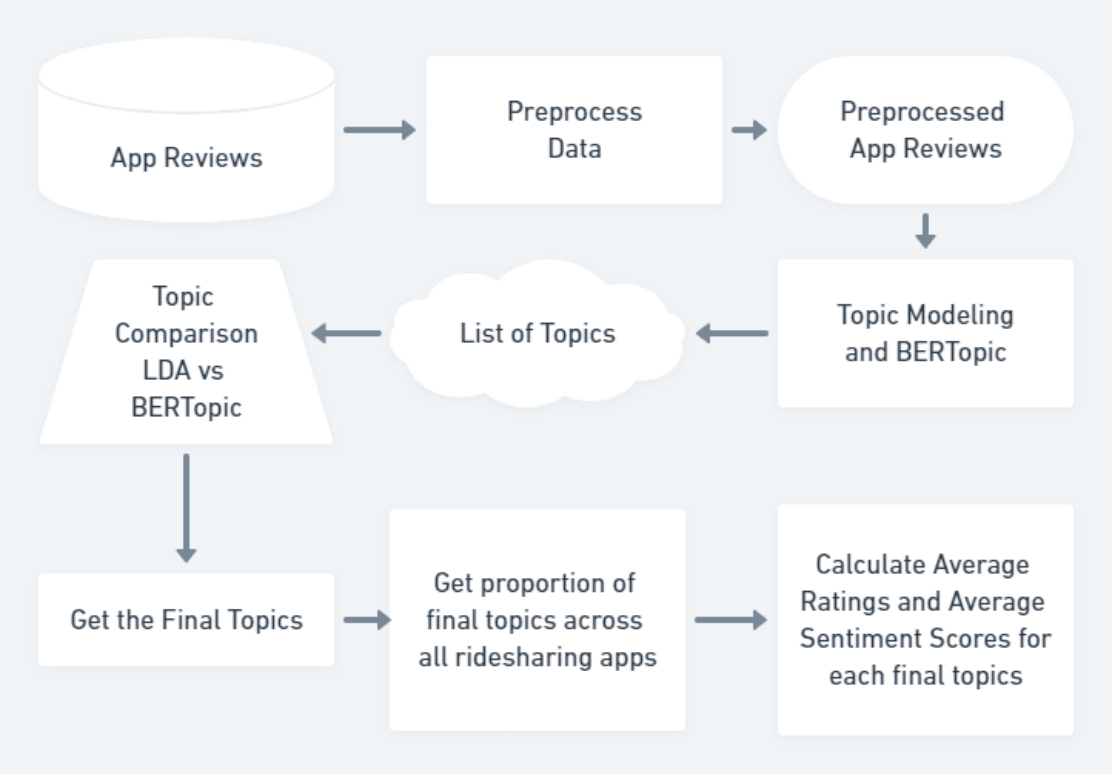

Figure 1. Overview of the study setup

3.2 Data Collection Process

In the first step, we collect app reviews from Google Play Store and Apple App Store. For Google Play Store, we used google-play-scraper [5] developed by JoMingyu. For Apple App Store, we used app-store-scraper [6] developed by *cowboy-bebug*. Both libraries provide APIs in Python for app store data crawling. We stored the mining result in csv files. A reproducible jupyter notebook for mining the app review is available at this link[7].

Table 4 shows the total number of app reviews and average ratings from each rideshar-ing app in both app stores and combines both numbers for general information for each app.

Table 4. Total Reviews and Average Ratings from Google Play Store and Apple App Store

No	App Name	OS	Total Reviews	Average Ratings	TotalReviews Combined	AverageRatings Combined
1	Bolt	Android	51907	3.96	55061	3.91
		iOS	3154	3.02		
2	Uber	Android	10000	3.3	20342	3.13
		iOS	10342	2.96		
3	BlaBlaCar	Android	19452	4.3	44480	4.21
		iOS	23308	4.13		
4	Cabify	Android	3261	2.6	10645	3.7
		iOS	7384	4.19		
5	Via	Android	1873	3.6	4265	3.63
		iOS	2392	3.65		
6	Getaround	Android	731	3.46	3219	3.3
		iOS	2488	3.25		
7	Ola Cabs	Android	10000	1.54	10922	1.9
		iOS	922	2.19		
8	Taxi.eu	Android	211	2.8	775	3.32
		iOS	564	3.52		
9	Free Now	Android	11078	3.24	25428	3.52
		iOS	14350	3.73		
10	Yandex Go	Android	7053	3.26	7224	3.25

[5] https://github.com/JoMingyu/google-play-scraper

[6] https://github.com/cowboy-bebug/app-store-scraper

[7] https://github.com/enliktjioe/master-thesis-2021/tree/main/review_mining

3.3 NLP Preprocessing

Data preprocessing is needed to get a better quality of the dataset. To preprocess data means to transform it into a well-format dataset that we can use more effectively in a data science project. [Gan19]. There are different types of text preprocessing techniques, including

1. Lowercasing, mapping all words into the same lowercase form
2. Stemming, transforming words into their root form, for example, 'playing' becomes 'play'. Examples are listed in Figure 2
3. Lemmatization, similar to stimming, maps a word into its root form, but in a more proper way. Examples are listed in Figure 3
4. Stopword Removal, removing commonly used words in a language such as "a", "the", "is", "are", and so on.

	original_word	stemmed_words
0	connect	connect
1	connected	connect
2	connection	connect
3	connections	connect
4	connects	connect

	original_word	stemmed_word
0	trouble	troubl
1	troubled	troubl
2	troubles	troubl
3	troublemsome	troublemsom

Figure 2. Example of Stemming

	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble

	original_word	lemmatized_word
0	goose	goose
1	geese	goose

Figure 3. Example of Lemmatization

The study of Jiahao Weng [Wen19] tried to compare the result of text analysis with and without preprocessing steps. The authors stated how preprocessing could improve the accuracy of sentiment analysis by clearing the context of the analyzed sentence.

Before the first preprocessing steps (P1), we already did the basics of NLP preprocessing steps. These include lowercasing text, lemmatization, and stopword removal. After that, we started the planned preprocessing steps, including removing non-English reviews, filtering out inconsistent and uninformative reviews, correcting typos, and building a text corpus. Table 5 shows the total number of app reviews before and after each preprocessing step for every app. P1 is the first preprocessing step where we removed non-English app reviews. P2 is the second step where we removed inconsistent app reviews (star rating and sentiment are not matched). P3 is the third step where we remove uninformative reviews, which are usually short reviews that only include praise or dispraise (for example: "I love it," "I hate this app") according to Guzman et al. [GM14]. In the following, we explain all these preprocessing steps in more detail.

Table 5. List of the Top 10 Ridesharing Apps, Followed by the Initial Number of User-Reviews, and Number of Consistent and Informative User-Reviews

No	**App Name**	**#Reviews**	**(P1) English Reviews**	**(P2) Consistent Reviews**	**(P3) Informative Reviews**
1	Bolt	55061	40365	17930	10785
2	Uber	20342	19833	8285	6763
3	BlaBlaCar	44480	16212	8822	3156
4	Cabify	10645	1899	763	551
5	Via	4265	3962	1863	1417
6	Getaround	3219	894	365	249
7	Ola Cabs	10922	10875	3766	3573
8	Taxi.eu	775	220	72	43
9	Free Now	25428	10939	4285	2728
10	Yandex Go	7224	2888	1261	622
Total		***182361***	***108084***	***47412***	***29813***

3.3.1 Remove Non-English Reviews

We found out that some app reviews we have collected contain non-English reviews. Most of the non-English reviews come from Apple App Store, as our data scraper for Apple App Store does not have a language filter parameter. We're using the Python library name *langdetect*[Dan14] to remove around 70,000+ non-English app reviews. The languages in the non-English reviews are from European countries which not use English as the main language, such as Russia, Germany, and France.

The *langdetect* library has a scoring system for every app review text, including the language code (two letters) and its probability score. The example of *langdetect* library can be seen in Figure 4. In Figure 5 when the app review is too short, it might give multiple language probability.

```
detect_langs("Shared my promo code with my friend and didnt get
    any discount. No live chat no way to text them")

[en:0.9999972783798426]

detect_langs("pourquoi ne pas afficher toutes les voitures
    disponibles autour du point de prise en charge ? faire des promos
     c est  bien mais si pas de voitures disponibles....des erreurs
    7002 fr quentes, des indications de temps  d  arrive
    fantaisistes. Une fois le chauffeur s lectionn  on  s  aperoit
     souvent  q u il  finit une course et  a  prend du temps ce qui
    fait que le compteur  d  arrive  reste fig  sur un temps
     d  attente  acceptable  pour bloquer  l usager  mais qui peut
     facilement  tre  le double de celui indiqu  ....")

[fr:0.9999982827184632]
```

Figure 4. Examples of *langdetect* library

```
detect_langs("Very fast and affordable")
[en:0.5714280394408676, da:0.42857192445741144]

detect_langs("Best app ever")
[no:0.7142813554856114,fr:0.1428580578700273, en:0.14285644644057932]
```

Figure 5. Examples of Multiple Languages Probability

To decide if it is English or non-English, we made a Python function to get an English 'en' probability score. If 'en' scores exceed 0.1, it will be categorized as English text. If 'en' score is below or equal to 0.1 or does not contain 'en' score, it will be categorized as non-English text. Here are the example results from the function,

3.3.2 Remove Inconsistent Reviews

Inconsistent reviews happen when users give negative reviews but give high star ratings instead. Similarly, users give positive reviews but give low star ratings instead. For example, the review text is "Horrible app, not user friendly," but it was associated with five stars. To detect inconsistent reviews, we are using a comparison between star ratings

with their sentiment score. We calculated the sentiment score using SentiStrength Python library [Hun19], the wrapper for its original Java library.

SentiStrength gives scores to app reviews with a value ranging from -5 (most negative) to +5 (most positive). Sentiments with scores -5, -4, -3, and -2 are considered as negative sentiments, while scores -1, 0, and +1 as neutral sentiments, and +2, +3, +4, +5 are positive sentiments. For star rating, the possible values are between 1 and 5. Negative ratings are 1 and 2, neutral ratings are 3, and positive ratings are 4 and 5. Figure 6 shows the example of consistent app reviews. Figure 7 shows the example of inconsistent app reviews.

We kept only the consistent review in order to minimize the misinformation that can happen in our analysis. For example, an app review with a sentiment score of +4 and a rating of 5 will be considered a consistent review. Meanwhile, an app review with a sentiment score of +4 and a rating of -1 will be considered an inconsistent review. Table 6 shows the mapping of positive, neutral, and negative scores for both sentiment and rating.

```
Great app!!
Sentiment Score: 3
Star Rating: 5
isInconsistent: False


Just love it!
Sentiment Score: 3
Star Rating: 5
isInconsistent: False


Great app!
Sentiment Score: 3
Star Rating: 5
isInconsistent: False
```

Figure 6. Example of Consistent App Reviews

```
Absolute no difference between "Comfort" and "Normal" options (except for the price). No big deal.
Sentiment Score: -2
Star Rating: 4
isInconsistent: True
```

```
It make travel easier... It's quite good... And also with less fares only when have you promotions
Sentiment Score: 3
Star Rating: 1
isInconsistent: True
```

```
Pretty good. Gps doesnt always get the best route.
Sentiment Score: 0
Star Rating: 4
isInconsistent: True
```

Figure 7. Example of Inconsistent App Reviews

Table 6. Detail of Sentiment and Rating Scoring System

No	Category	Score(s)
1	Positive Sentiment	[+2, +3, +4, +5]
2	Positive Rating	[4, 5]
3	Neutral Sentiment	[-1, 0, +1]
4	Neutral Rating	[3]
5	Negative Sentiment	[-5, -4, -3, -2]
6	Negative Rating	[1, 2]

3.3.3 Remove Uninformative Reviews

Some uninformative app reviews do not give app developers any beneficial information. We used the AR-Miner framework introduced by Chen et al.[CLH+14]. As AR-Miner source code was not available publicly, we implemented the logic using the SVM (Support Vector Machine) classifier. Several examples of uninformative app reviews that we have found in our dataset:

- good and reasonable prices
- amazing simple and suitable
- a very good service provider
- great prices great people
- nice and easy to operate
- it very easy and nice

Most uninformative reviews are short app reviews without significant information to analyze.

3.3.4 Building Corpus

The final step before running statistical analysis like topic modeling [WMvS83] is to create a text corpus. Before building the text corpus, we normalize the app reviews, including removing the stop words, removing the punctuation, and lemmatizing the words. We used Gensim corpora library [Ře17] to create the text corpus and convert it into *tf_idf*[8] format, short for **term frequency–inverse document frequency**. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [RU11]. We saved the corpus file into a binary file for Python named *pickle*[9]. Figure 8 shows the example in Gensim to create *tf_idf* corpus using Jupyter

[8] `https://radimrehurek.com/gensim/models/tfidfmodel.html`
[9] `https://docs.python.org/3/library/pickle.html/`

Notebook. This binary file of *tf_idf* corpus will be used for Gensim's text classification and clustering framework.

```
tf_idf = models.TfidfModel(term_doc, smartirs='ntc')[term_doc]
[[(id2word[id], freq) for id, freq in cp] for cp in tf_idf[:1]]
```

```
[[('apology', 0.258303175195304),
  ('appointment', 0.19374341386795016),
  ('car', 0.1660609726432199),
  ('chair', 0.258303175195304),
  ('cross', 0.2304987993967995),
  ('day', 0.12187014507285714),
  ('driver', 0.07236556805706493),
  ('explanation', 0.2304987993967995),
  ('extraction', 0.258303175195304),
  ('incident', 0.2142342821987155),
  ('journey', 0.12323004598478411),
  ('leg', 0.258303175195304),
  ('location', 0.1423610134036225),
  ('man', 0.19374341386795016),
  ('month', 0.19374341386795016),
  ('occasion', 0.20269442359829498),
  ('one', 0.18642990640021098),
  ('pain', 0.258303175195304),
  ('people', 0.15244204814412854),
  ('place', 0.18024642394263304),
  ('refund', 0.14019220224671175),
  ('response', 0.15244204814412854),
  ('swore', 0.258303175195304),
  ('text', 0.258303175195304),
  ('trip', 0.10301677900469748)]]
```

```
with open(config['csv_input_local']['bolt_apple_corpus'], 'wb') as f:
    pickle.dump(tf_idf, f)
```

Figure 8. Using Gensim Corpora library to make Corpus file

4 Implementation

Topic modeling is one of the unsupervised machine learning techniques that is useful to discover the abstract "topic" from a collection of text documents. One of the most common algorithms for fitting a topic modeling is Latent Dirichlet Allocation (LDA). It will map each document in the text corpus to a set of topics that represent a group of words in the document. An example of a topic can be the set of words location, map, app, address, area which describes users' experiences when checking location using the ridesharing app. In LDA, every document is a combination of topics, and every topic is a combination of words [SR18]. Figure 9 shows the example of topic modeling result using the LDA technique.

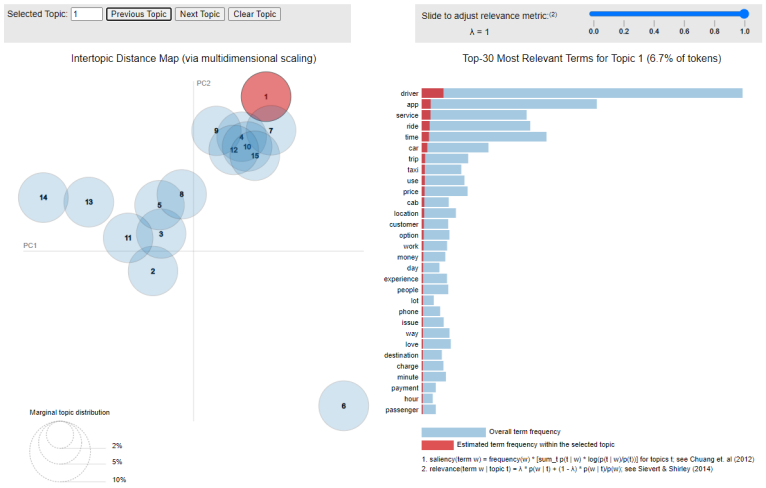

Figure 9. Example of Topics created by LDA

Each app review contains some contexts that will be useful if we can understand them. LDA is one of the most popular topic modeling techniques. We use Gensim as one of the most widely-used Python libraries for implementing LDA Topic Modeling. Other than LDA, we are also using Pre-Trained Model BERT to implement topic modeling, utilizing an open-source library named BERTopic [Gro20].

To find the optimal number of topics, we compute coherence scores using different topics ranging from 2 to 40, as shown in Figure 10. We chose 20 as the number of optimal topics as this number gave the highest value before the coherence score flattened out.

We realized that the data preprocessing step has filtered out more than 50% of the total app reviews by comparing the coherence score from three different datasets and three different topic modeling approaches. We used Bolt app reviews because it has the highest number of app reviews compared to the other nine ridesharing apps, and we thought it

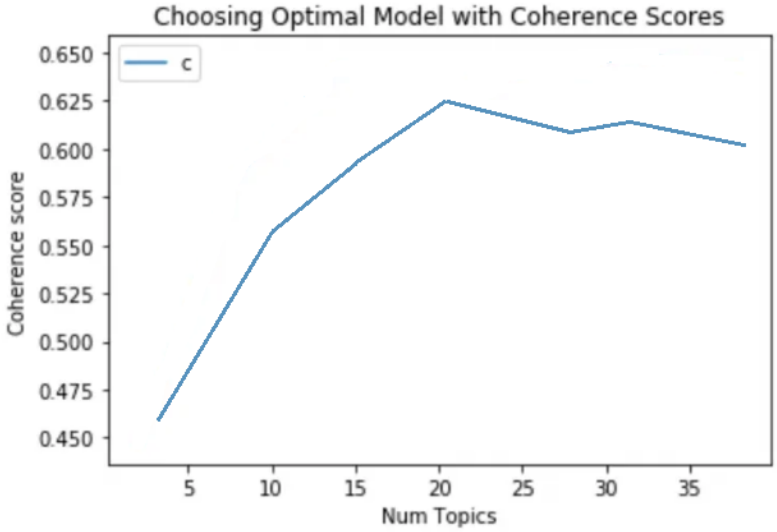

Figure 10. Coherence Score per Number of Topics

Table 7. Coherence Score for Different Topic Modeling Techniques using 20 Topics

Dataset	**#Reviews (BEFORE)**	**#Reviews (AFTER)**	**LDA Standard**	**LDA Mallet**	**BERTopic**
Bolt P1 (Remove Non-English Reviews)	55061	40365	0.3156	0.4268	**0.4410**
Bolt P2 (P1 + Remove Inconsistent Reviews)	40365	17930	0.4139	**0.4874**	0.4152
Bolt P3 (P2 + Remove Uninformative Reviews)	17930	10785	0.4051	**0.4905**	0.4518

was enough to get a coherence score comparison. Table 7 shows the total number of app reviews from Bolt app using three different datasets and its coherence score for three different topic modeling approaches. Based on coherence score comparison, we selected two out of three methods for topic modeling comparison, LDA Mallet and BERTopic, as both perform better compared to the LDA Standard model. In the following, we explain the topic result from both LDA Mallet and BERTopic.

4.1 LDA Mallet

Mallet (MAchine Learning for LanguagE Toolkit) [McC02] is a java-based open-source toolkit written by Andrew McCallum and used for many machine learning applications to text, including Topic Modeling. Gensim is a Python library that provides a wrapper for LDA. To use LDA Mallet in Gensim, we can call the module with syntax gensim.models.wrappers.LdaMallet. This module contains Gibbs sampling from Mallet that allows LDA model to get the topic result from the unseen documents. Table 8 shows all 20 topics based on the top 10 keywords extracted using LDA Mallet technique. LDA technique only puts the topic as a number, so we need to label it manually. There are several topics with the same name because it has very similar keywords, and we decide to give the same one.

Table 8. LDA Mallet Topic Modeling using Bolt P3 dataset

No	Topic Name	Keywords	Counts
1	Price	price, destination, experience, friendly_driver, change, easy_use, competitor, good_price, comfort, increase	2234
2	Praising Features	service, love, life, fast_efficient, download, rude, affordable_price, mind, availability, really_good	764
3	Time	time, taxi, drivers_alway, point,pricing, pickup, travel, driving, drop, order	762
4	Location	location, map, app, address, area, order, pick, feature, improvement, street	628
5	Driver Service	driver, app, phone, lot, man, move, booking, cancel_trip, drive, instance	577
6	Driver Service	driver, experience, client, end, rating, cancel, condition, music, professional, conversation	509
7	User Experience	work, issue, thing, traffic, friend, drive, scam, user, transport, yesterday	503
8	Driver	driver, passenger, vehicle, case, city, scooter, transportation, arrival, item, promotion	492
9	Customer Support	service, charge, email, cab, month, hour, reply, good, year, person	482
10	Payment	card, money, problem, pay, account, application, detail, error, card_payment, report	450
11	Ride Experience	customer, company, route, driver, journey, reason, refund, safety, review, screen	418
12	User Experience	trip, option, fare, rider, business, star, complain, stop, charge, travel	407

13	Ride Experience	car, driver, system, team, quality, book, bit, woman, world, attitude	399
14	Ride Experience	ride, cash, show, notification, fix, enjoy, amazing_service, ad, period, purpose	396
15	Driver Service	driver, call, place, road, pick_location, today, datum, kind, occasion, side	338
16	Customer Support	guy, people, response, number, payment, code, complaint, message, lot, situation	321
17	Driver Service	driver, day, support, promo, amount, promotion, week, recommend, customer_service, job	320
18	Feature Request	app, request, update, estimate, cost, fee, min, today, rubbish, arrival	300
19	Time	time, driver, minute, direction, distance, estimation, airport, load, great_experience, talk	288
20	Promotion	app, discount, rate, network, share, info, a lot, experience, convenience, comment	197

4.2 BERTopic

Bidirectional Encoder Representations from Transformers (BERT) is one of the best NLP pre-training models developed by Google and released as an open source project in 2018. It was pre-trained using massive amounts of text data from English Wikipedia. In its research stages, BERT achieved revolutionary results in 11 natural language understanding tasks, including sentiment analysis, sentence classification, semantic role labeling, and disambiguation of words with multiple meanings. It effectively addresses ambiguity, which is the greatest challenge in natural language understanding. BERT gives the capability to parse the human language with a relatively human-like common sense.

Meanwhile, BERTopic [Gro20] is a topic modeling tool developed by Marteen Grootendorst in 2020 that utilizes Transformers to create dense clusters allowing for easily interpretable topics while keeping important words in the topic description. Figure 11 shows the BERTopic components based on its official documentation. Table 9 shows 20 topics extracted using BERTopic, including its top 4 keywords. Similar to the previous technique, we needed to label these topics manually and got several topics with the same name.

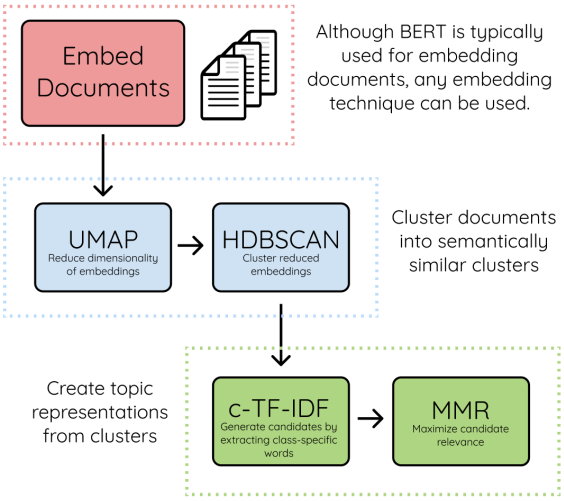

Figure 11. BERTopic Main Components

Table 9. BERTopic Topic Modeling using BOLT P3 Dataset

No	Topic	Keywords	Count
N/A	(OUTLIER)	-1_app_my_driver_on	3371
1	Driver Service	0_drivers_friendly_good_nice	1920
2	Driver Service	1_app_drivers_good_friendly	467
3	Pricing	2_prices_discounts_price_good	455
4	Praising Features	3_service_reliable_great_excellent	429
5	Ride Experience	4_ride_rides_enjoyed_enjoy	428
6	Driver Service	5_bolt_was_driver_but	362
7	Praising Features	6_efficient_reliable_app_good	360
8	Praising Features	7_experience_keep_enjoy_love	304
9	Praising Features	8_bolt_experience_good_awesome	298
10	Pricing	9_affordable_good_efficient_nice	291
11	Location	10_location_app_pick_my	286
12	Ride Experience	11_taxify_taxi_my_this	275
13	Driver Service	12_driver_he_my_code	240
14	Praising Features	13_transport_transportation_trips_easy	231
15	Time	14_minutes_time_waiting_wait	190
16	Location	15_gps_map_location_maps	189
17	Praising Features	16_uber_better_cheaper_more	182
18	Payment	17_card_payment_account_app	174
19	Driver Service	18_rude_drivers_they_worst	167
20	Promotion	19_promo_promos_code_promotions	166

5 Findings and Analysis

In this chapter, some findings and analysis were discussed to determine what is going well and what is the thing that can be improved for future improvement.

5.1 Final Results

Based on the LDA Mallet and BERTopic results above, we do the manual analysis to interpret the LDA model and select the final topics. We merge some topics with very similar meanings based on the keyword given by the model and the related app reviews. We follow these criteria before merging the topics:

- One topic consists of multiple keywords that we will use to compare with other keywords from different topics. If the keywords have similar meanings, we merge them into one topic.
- If the keywords are not clear enough, we will look into the app review related to each topic. If the app reviews explain the same context, we merge them into the same group.

All final 13 topics are listed in Table 10, followed by the top five words and a brief description of each topic.

Table 10. The List of Identified Topics from App Reviews

No	Topic Name	Top 5 Words	Brief Description
T1	Reporting Issues	issue, scam, response, problem, error	Reporting issues about the app features
T2	Customer Support	support, email, people, response, call	Customer's complaint via the customer support
T3	Driver Service	driver, problem, cancel, rude, conversation	Reviews related to driver's behavior
T4	Feature Requests	app, taxi, experience, access, improvement	Reviews related to additional feature request
T5	Location	location, address, direction, map, gps	App experience when user choose pickup and destination location
T6	Payment	card, money, refund, cash, change	Payment-related issues
T7	Praising Features	easy, convenience, safe, fast_affordable, efficient	Positive feedback given by user about the overall experiences
T8	Pricing	price, money, discount, cheap, affordable	Reviews related to app pricing
T9	Promotion	promotion, request, discount, code, service	Reviews related to in-app promotion
T10	Purpose of Ride	taxi, transport, work, family, scooter	The purpose of riding is for work, visiting family, and using a scooter.
T11	Ride Experience	ride, rate, enjoy, easy_use, passenger	Reviews related to user's ride experience
T12	Safety	safe, night service, journey, experience	Reviews related to the safety of ride experience
T13	Time	time, minute, wait, arrival, pickup	Reviews related to time spent for pick up and arrival

5.2 Proportion of Topics Across 10 Apps

Several topics frequently appear in a specific app, while other topics appear infrequently. For example, in Getaround app, users mentioned more about customer support than promotion. It could be that Getaround app did not give much promotion compared to other ridesharing apps.

Table 11 shows the proportion of topics in each ridesharing app. We put bold text in the table cell to highlight the topic contributing at least ten percent of total reviews. Topic "praising features" frequently appeared in most apps except Ola Cabs. Getaround has the highest proportion of topic customer support compared to the other nine ridesharing app. Ridesharing app users are more likely to praise the app feature (praising features) more than report the app issue (reporting issues) or request a new feature (feature request). For more performance analysis of each topic, we used the average rating and average sentiment score as the metric in the following.

Table 11. Proportion of the Topics in Different Ridesharing Apps

No	Topic Name	Bolt	Uber	Blablacar	Cabify	Via	Getaround	Ola Cabs	Taxi.eu	Free now	Yandex Go
T1	Reporting Issues	4%	4%	3%	4%	4%	7%	6%	0%	3%	6%
T2	Customer Support	4%	5%	5%	7%	5%	**25%**	3%	2%	3%	6%
T3	Driver Service	**13%**	**13%**	**22%**	**10%**	**12%**	**15%**	**12%**	2%	7%	**10%**
T4	Feature Request	6%	4%	4%	3%	5%	4%	3%	9%	5%	5%
T5	Location	4%	4%	1%	3%	7%	3%	3%	2%	4%	2%
T6	Payment	6%	6%	4%	8%	8%	5%	3%	5%	5%	6%
T7	Praising Features	**24%**	**19%**	**21%**	**16%**	**16%**	**15%**	8%	**26%**	**23%**	**19%**
T8	Pricing	3%	4%	2%	3%	3%	8%	8%	2%	3%	3%
T9	Promotion	**11%**	**12%**	**12%**	**10%**	**10%**	4%	**10%**	**16%**	**10%**	**12%**
T10	Purpose of Ride	4%	4%	2%	7%	3%	1%	6%	**21%**	**16%**	**10%**
T11	Ride Exp	**10%**	**12%**	**13%**	**10%**	**15%**	4%	**26%**	2%	7%	7%
T12	Safety	3%	4%	2%	7%	6%	3%	4%	2%	5%	6%
T13	Time	9%	9%	8%	**11%**	5%	6%	**10%**	9%	8%	9%

5.3 Average Rating and Sentiments of Each Topic

In order to measure the performance of each topic to the selected ridesharing app, we calculated its average rating and sentiment score. The average rating is the mean value from the total accumulation of star rating values per topic, and the average sentiment score is the mean value from the total accumulation of sentiment scores per topic. For example, for star rating, if topic 1 has a total of 10 reviews with accumulated star rating values equal to 35, then the average rating is 35 divided by 10, which is 3.5. Using the exact total of 10 reviews from topic 1, if the accumulated sentiment score values equal to -7, then the average sentiment score is -7 divided by 10, which is -0.7.

Table 12. Average Rating (R) and Sentiments (S) for first 5 ridesharing apps (Bolt, Uber, Blablacar, Cabify, and Via)

No	Topic Name	Bolt		Uber		Blablacar		Cabify		Via	
		R	S	R	S	R	S	R	S	R	S
T1	Reporting Issues	2.9	-0.1	2.7	-0.3	**3.3**	**0.6**	**2**	**-1.1**	3.9	1
T2	Customer Support	3.5	0.6	3.1	0	4.2	1.4	3.6	0.5	3.9	1
T3	Driver Service	3.9	1	3.7	0.7	4.5	1.8	3.5	0.6	3.9	1.1
T4	Feature Request	3.8	0.9	3.5	0.5	4.4	1.6	3.6	0.7	**4.6**	**2**
T5	Location	3	0	2.9	-0.2	4.2	1.4	2.4	-0.5	3.6	**0.6**
T6	Payment	4.2	1.4	**4.2**	**1.4**	**4.6**	**1.9**	**3.7**	**0.9**	4.2	1.5
T7	Praising Features	**4.3**	**1.5**	4.1	1.3	4.5	1.7	3.6	0.6	4.2	1.4
T8	Pricing	**2.6**	**-0.6**	**2.2**	**-1.1**	4.0	1.0	2.3	-0.9	**3.1**	0.2
T9	Promotion	3.7	0.9	3.5	0.5	4.0	1.2	3.5	0.6	4	1.2
T10	Purpose of Ride	3.2	0.2	2.7	-0.4	3.7	0.9	2.5	-0.4	3.6	0.7
T11	Ride Experience	3.7	0.8	3.3	0.4	4.2	1.5	3	0	3.9	1.1
T12	Safety	3.7	0.8	3.1	0.1	4.3	1.5	2.2	-0.9	4.1	1.2
T13	Time	2.7	-0.4	2.8	-0.3	4.1	1.3	2.2	-0.9	3.6	0.9

Table 12 shows the average rating and sentiment score for the first five ridesharing apps (Bolt, Uber, Blablacar, Cabify, and Via). Table 13 shows the same information for the last 5 ridesharing apps (Getaround, Olacabs, Taxi.eu, Freenow, and Yandex Go). In

Table 13. Average Rating (R) and Sentiments(S) for last 5 ridesharing apps (Getaround, Olacabs, Taxi.eu, Freenow, and Yandex Go)

No	Topic Name	Get around		Ola cabs		Taxi .eu		Free now		Yan dex Go	
		R	**S**	**R**	**S**	**R**	**S**	**R**	**S**	**R**	**S**
T1	Reporting Issues	3.6	0.7	1.9	-1.3	N/A	N/A	2.6	-0.5	2.8	0
T2	Customer Support	3.1	0.1	1.9	-1.3	**4.7**	**2.1**	3.1	0.1	3.5	0.6
T3	Driver Service	3.6	0.6	1.6	-1.6	3	0	3.3	0.3	3.5	0.5
T4	Feature Request	3.2	0.7	2	-1.2	3.5	1.2	3	0	3.8	0.6
T5	Location	3.6	1	1.7	-1.4	1.1	**-3**	2.2	-1	2.4	-0.8
T6	Payment	3	0.1	**2.2**	**-1**	3	0	3.8	0.8	3.9	0.8
T7	Praising Features	4.4	1.4	2	-1.2	3.1	0	**4**	**1.1**	**4.1**	**1.2**
T8	Pricing	2.2	-1.2	**1.4**	**-1.8**	1.1	-2	**1.7**	**-1.7**	2.5	-0.6
T9	Promotion	3.3	0.3	1.8	-1.5	3.3	0.3	3.4	0.4	3.2	0.2
T10	Purpose of Ride	**4.7**	**2.3**	1.9	-1.3	2.9	-0.3	2.1	-1.2	**2.3**	**-0.9**
T11	Ride Experience	4	1.1	1.5	-1.7	4.5	2	2.8	-0.3	3.3	0.3
T12	Safety	3.5	0.6	1.6	-1.6	**1**	-2	2.7	-0.4	2.9	-0.3
T13	Time	**1.7**	**-1.7**	1.7	-1.5	3	0.5	2.5	-0.7	2.5	-0.5

both tables, we used blue color to highlight the highest score and red color for the lowest score of rating or sentiment per topic.

In Bolt app, praising features became the topic with the highest score, while pricing became the lowest score topic. Considering the proportion of topic praising features in Bolt app, which takes 24% of total app reviews based on Table 11, there is a correlation between this topic with positive ratings and sentiments, as users mainly talked about their positive experience with the Bolt app feature. This condition also happened in other apps, including Uber, Blablacar, Via, Getaround, Freenow, and Yandex Go, when their ratings have scored at least four, and their sentiment score was above +1. Meanwhile, pricing is the lowest score topic in Bolt app which means some users have a problem with the app pricing system. This issue also can be found in other apps except for Blablacar, which gave the highest score for the pricing topic. That means Blablacar users liked its pricing

system as one of the plus points compared to other apps.

Besides pricing, another topic with the lowest score is time. Some apps, including Bolt, Uber, Cabify, Getaround, Olacabs, Freenow, and Yandex Go have low scores on this topic. It is related to pickup and arrival times for every ride that app developers need to prioritize. Meanwhile, Blablacar and Via have a positive score for this time topic which means most of their users have a good experience with pickup and arrival times. Olacabs became the ridesharing app with the most negative scores for all 13 topics. This app needs further improvement to give users a better ridesharing experience.

6 Conclusions

In this thesis, we filled the academic literature gap by analyzing the ridesharing app reviews domain using some open-source libraries and compared their result. We identify the list of important topics that ridesharing app developers should focus on for the next development cycle of their app. Topics that have been investigated will help app developers with a smaller and more helpful subset of app reviews which will be less time-consuming than analyzing all the user reviews.

6.1 Key Results

Based on the implementation and analysis in chapters 4 and 5, we can draw the following conclusions for our two research questions as stated below:

- BERTopic gave a high coherence score of 0.4410 for the first preprocessed dataset (P1), while LDA Mallet gave a higher coherence score of 0.4874 and 0.4905 for the second and third preprocessed dataset (P2 and P3). The improvement of coherence scores from each preprocessing step proved that these steps were needed to improve the dataset's quality before implementing topic modeling techniques.
- The proportion of topics helps us to know about the majority and minority topics mentioned by the user. Furthermore, the average rating and sentiment score give the prioritization score for each topic by giving a blue marker for the positive topic, a red marker for the negative topic, and a blank marker for the neutral topic. Praising features and driver service topics frequently appeared in most ridesharing apps and gave positive scores. Meanwhile, pricing, time, and reporting issues lead to negative scores. These negative score topics are needed to be prioritized by ridesharing app developers when they want to update their apps.

6.2 Limitations

The limitations of this thesis can be divided into three key points: data, software, and hardware, which are explained in the following:

- *Data limitations:* the data used in this thesis was from November 2020, which might not be relevant to the current situation of ridesharing app user experiences. There is also a deficient number of app reviews for an app like Taxi.eu, which caused the distribution of final topics and calculation of ratings and sentiment to become unavailable.
- *Software limitations:* the topic modeling technique required manual labeling for each extracted topic since it only gave the list of keywords that represent the topic.

SentiStrength [Hun19], the open-source library we used to calculate the sentiment score, required some dependencies that required extra setup for some operating systems. The second preprocessing step to remove inconsistent app reviews had filtered out more than fifty percent of total reviews, which need to be improved in future development.

- *Hardware limitations:* this study analyzes many app reviews, which is time-consuming when using a less-powered computer such as our laptop. We needed a more robust and faster computer to work faster and efficiently. We tried the University of Tartu's High-Performance Computing cloud server to do our analysis. However, later we discovered that we could not install some software dependencies on this server. Our insufficient hardware made some of our text analyses slow and nearly impossible to complete on time. For example, the calculation of sentiment scores using the SentiStrength library took around 1 hour to complete for one ridesharing app.

6.3 Future Work

For further studies, the limitations related to data, software, and hardware, as we have mentioned before, will need to be improved following the recent situation and technological advancement.

This study focused only on the top 10 ridesharing apps in Europe. At the same time, there are some ridesharing apps available only in certain regions, such as Lyft in US, Gojek, and Grab in Southeast Asia. It would be possible to continue similar analyses from another region's perspective.

With the rising of NLP research and the development of pre-trained models like BERT, we would like to explore more pre-trained models in the field of NLP that are possibly able to give a better quality of feature and topic extraction. In summary, with the future development of NLP machine learning models, this study can be optimized in order to help app developers to prioritize the future changes in their app development life cycle.

References

[aK20] Jung a Kim. Netflix app review topic modeling, 2020. https://chatbotslife.com/netflix-app-review-topic-modeling-a8a15f301855.

[BZ19] Manal Binkhonain and Liping Zhao. A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, 1:100001, 2019.

[CLH+14] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, page 767–778, New York, NY, USA, 2014. Association for Computing Machinery.

[cnb19] Beyond uber: Your guide to ridesharing apps around the world. CNBC, 2019. https://www.cnbc.com/2019/11/08/top-ride-sharing-apps-in-europe-asia-south-america-africa-and-usa.html.

[Dan14] Michal Danilák. Langdetect, port of nakatani shuyo's language-detection library to python. 2014. https://github.com/Mimino666/langdetect.

[DSPA+17] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Corrado A. Visaggio, and Gerardo Canfora. Surf: Summarizer of user reviews feedback. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 55–58, 2017.

[FZA21] Alessio Ferrari, Liping Zhao, and Waad Alhoshan. Nlp for requirements engineering: Tasks, techniques, tools, and technologies. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 322–323, 2021.

[Gan19] Kavita Ganesan. All you need to know about text preprocessing for nlp and machine learning, 2019. https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html.

[GM14] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 153–162, 2014.

[Gro20] Maarten Grootendorst. Bertopic, leveraging bert and c-tf-idf to create easily interpretable topics, 2020. https://github.com/MaartenGr/BERTopic.

[Hun19] Yong Zhun Hung. Python-sentistrength, python 3 wrapper for sentistrength java library. 2019. https://github.com-/zhunhung/Python-SentiStrength.

[Ind15] Nitin Indurkhya. Emerging directions in predictive text mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(4):155–164, 2015.

[JSABM17] Timo Johann, Christoph Stanik, Alireza M. Alizadeh B., and Walid Maalej. Safe: A simple approach for feature extraction from app descriptions and app reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 21–30, 2017.

[KAA20] Nabeel Sabir Khan, Adnan Abid, and Kamran Abid. A novel natural language processing (nlp)–based machine translation model for english to pakistan sign language translation. *Cognitive Computation*, 12(4):748–765, 2020.

[KG15] Monisha Kanakaraj and Ram Mohana Reddy Guddeti. Nlp based sentiment analysis on twitter data using ensemble classifiers. In *2015 3Rd international conference on signal processing, communication and networking (ICSCN)*, pages 1–5. IEEE, 2015.

[LBP+18] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Vasundhara Rathod, and Shreya Bisen. Implementation of a chatbot system using ai and nlp. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) Volume-6, Issue-3*, 2018.

[McC02] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit, 2002. https://mallet.cs.umass.edu.

[MGK+21] Aaisha Makkar, Sahil Garg, Neeraj Kumar, M. Shamim Hossain, Ahmed Ghoneim, and Mubarak Alrashoud. An efficient spam detection technique for iot devices using machine learning. *IEEE Transactions on Industrial Informatics*, 17(2):903–912, 2021.

[MVK+20] M. Monika, N. Vignesh, Usha Kumari, M.N.V.S.S. Kumar, and Laxmi Lydia. Skin cancer detection and classification using machine learning. *Materials Today: Proceedings*, 33, 08 2020.

[OG20] Emanuel Oehri and Emitza Guzman. Same same but different: Finding similar user feedback across multiple platforms and languages. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 44–54, 2020.

[ond19] 11 top international ridesharing companies in 2019/2020. Onde, 2019. https://onde.app/blog/top-ride-sharing-companies.

[PR15] K. Pohl and C. Rupp. Requirement engineering fundamentals 2nd edition, 2015. Rocky Nook Inc.

[ROB+21] Kaavya Rekanar, Ian O'Keeffe, Sarah Buckley, Raja Abbas, Sarah Beecham, Muslim Chochlov, Brian Fitzgerald, Liam Glynn, Kevin Johnson, John Laffey, Bairbre Mcnicholas, Bashar Nuseibeh, James O'Connell, Derek O'Keeffe, Michael O Callaghan, Abdul Razzaq, Ita Richardson, Andrew Simpkin, Cristiano Storni, and Jim Buckley. Sentiment analysis of user feedback on the hse's covid-19 contact tracing app. *Irish Journal of Medical Science (1971 -)*, 191, 02 2021.

[RU11] Anand Rajaraman and Jeffrey David Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011.

[SDP+21] Abhishek Soni, Dharamvir Dharmacharya, Amrindra Pal, Vivek Kumar Srivastava, Rabindra Nath Shaw, and Ankush Ghosh. *Design of a Machine Learning-Based Self-driving Car*, pages 139–151. Springer Singapore, Singapore, 2021.

[Sha] Faiz Ali Shah. Extracting information from app reviews to facilitate software development activities. PhD Dissertation. University of Tartu.

[SR18] J. Silge and D. Robinson. Text mining with r: A tidy approach, 2018. https://www.tidytextmining.com/topicmodeling.html.

[Wen19] Jiahaow Weng. Nlp text preprocessing: A practical guide and template, 2019. https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79.

[WMvS83] B. Al W. Martin and P. van Sterkenburg. On the processing of a text corpus: From textual data to lexicographical information. Lexicography: Principles and Practice. London, U.K.: Academic, 1983.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13:55–75, 08 2018.

[ZAF+21] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza T. Batista-Navarro. Natural language processing for requirements engineering: A systematic mapping study. *ACM Comput. Surv.*, 54(3), apr 2021.

[Ře17] Radim Řehůřek. Gensim, topic modelling for human, 2017. https://github.com/RaRe-Technologies/gensim.

Appendix

I. Glossary

Terms	Definition
AR-Miner	App Review Miner
ALBERT	A Lite BERT
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag-of-Word
CNN	Convolutional Neural Network
DL	Deep Learning
LDA	Latent Dirichlet Allocation
ML	Machine Learning
NN	Neural Network
POS	Part-of-Speech
PTM	Pre-Trained Model
REVSUM	Review Summarizer
RoBERTa	Robustly Optimized BERT Pretraining Approach
RNN	Recurrent Neural Network
SAFE	Simple Approach for Feature Extraction
TM	Topic Modeling

II. Github Repository

`https://github.com/enliktjioe/master-thesis-2021`

III. Additional Results

Table 14. Topic Modeling Result using LDA Mallet for dataset BOLT P1

Dominant Topic	Keywords	# Docs	% Docs
0.000000	car, driver, passenger, city, fact, street, excuse, kind, turn, condition	14582	0.361300
1.000000	taxi, rate, day, application, app, fee, scooter, star, work, move	2622	0.065000
2.000000	ride, easy_use, business, night, scam, town, voucher, book, offer, market	1785	0.044200
3.000000	customer, discount, charge, end, month, distance, morning, pay, care, arrival	1651	0.040900
4.000000	time, minute, order, cab, pick, min, pickup, start, estimation, every_time	1518	0.037600
5.000000	driver, client, show, life, cancellation, rude, office, block, cancel_trip, communication	1458	0.036100
6.000000	location, destination, option, map, place, address, point, pick_location, screen, everytime	1449	0.035900
7.000000	issue, app, response, today, thing, update, experience, safety, yesterday, review	1447	0.035800
8.000000	price, guy, reason, transportation, pricing, platform, purpose, alternative, range, demand	1445	0.035800
9.000000	driver, work, experience, friend, area, vehicle, person, job, family, drop	1366	0.033800
10.000000	love, service, friendly_driver, nice_one, good, driving, fast_affordable, convenience, fast_efficient, affordable_price	1271	0.031500
11.000000	driver, trip, fare, direction, feedback, system, cancel_ride, man, rubbish, mall	1211	0.030000
12.000000	card, money, company, account, payment, amount, change, cash, pay, card_payment	1183	0.029300

13.000000	support, email, journey, complaint, reason, week, refund, customer_service, reply, team	1169	0.029000
14.000000	app, problem, lot, user, error, notification, network, feature, bug, trouble	1160	0.028700
15.000000	time, route, drivers_alway, drive, traffic, cost, estimate, road, travel, home	1140	0.028200
16.000000	app, phone, number, code, call, message, contact, datum, download, detail	1056	0.026200
17.000000	driver, rider, promo, rating, case, quality, attitude, moment, cancelled_trip, profile	1033	0.025600
18.000000	app, people, bit, improvement, comfort, hope, thief, driver, fix, side	926	0.022900
19.000000	service, driver, request, promotion, app, hour, transport, recommend, country, competitor	893	0.022100

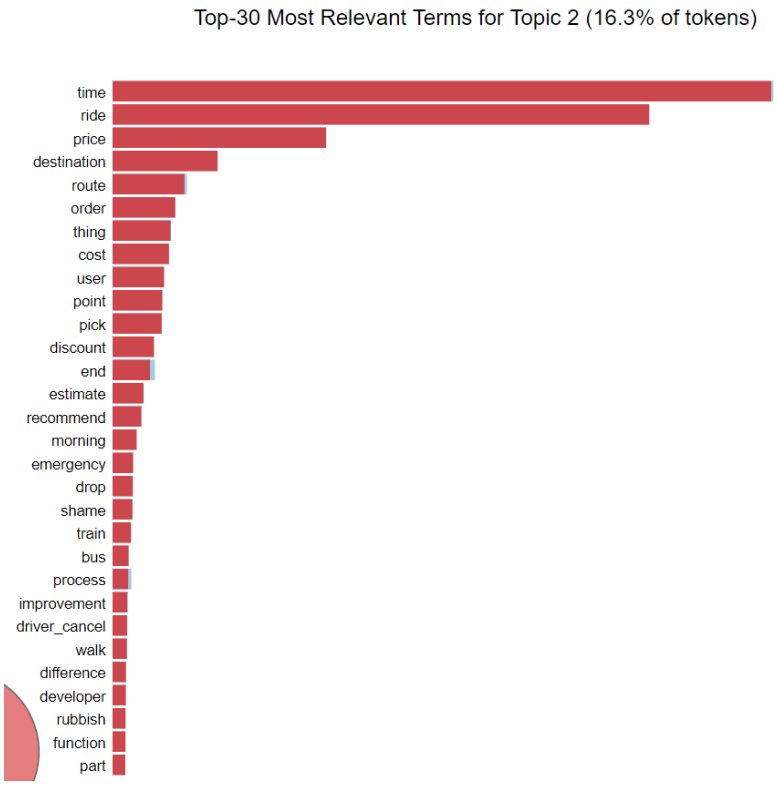

Figure 12. Extracted Keywords From Topic Ride Time

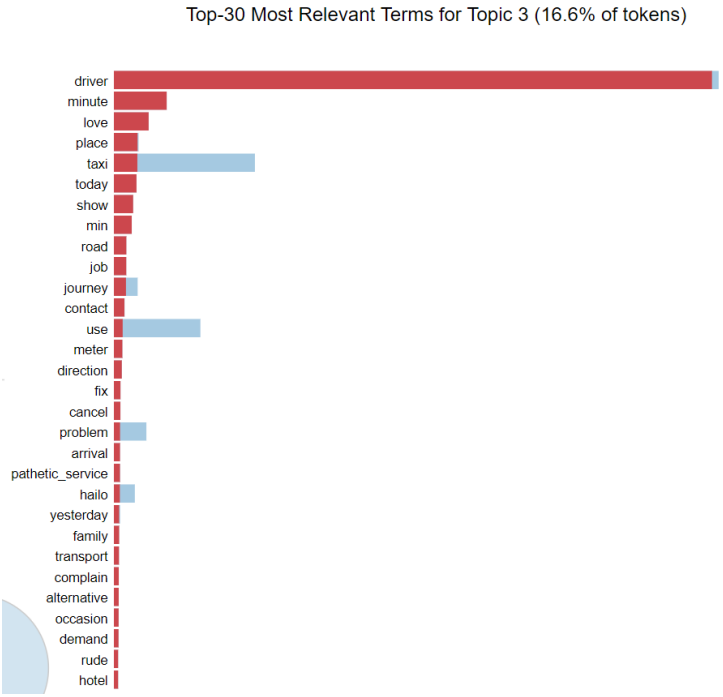

Figure 13. Extracted Keywords From Topic Driver Quality

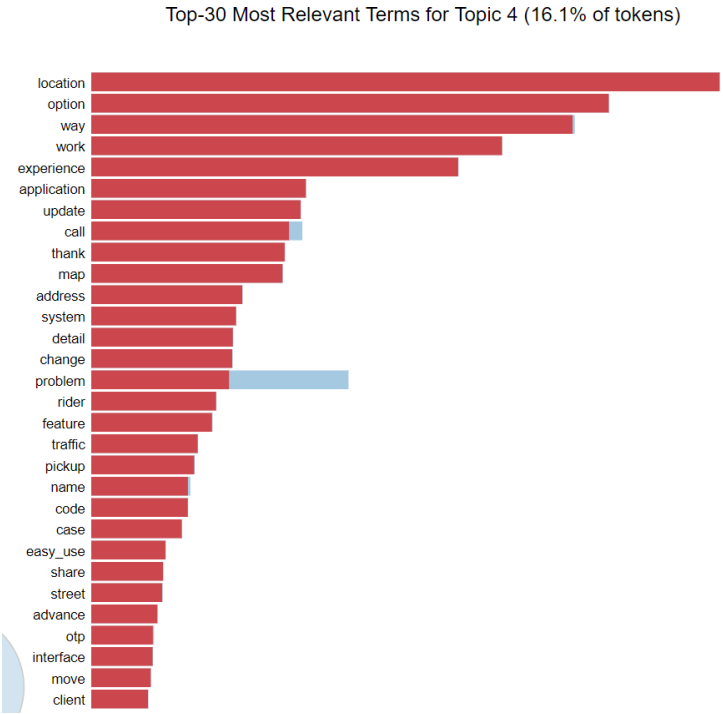

Figure 14. Extracted Keywords From Topic Location

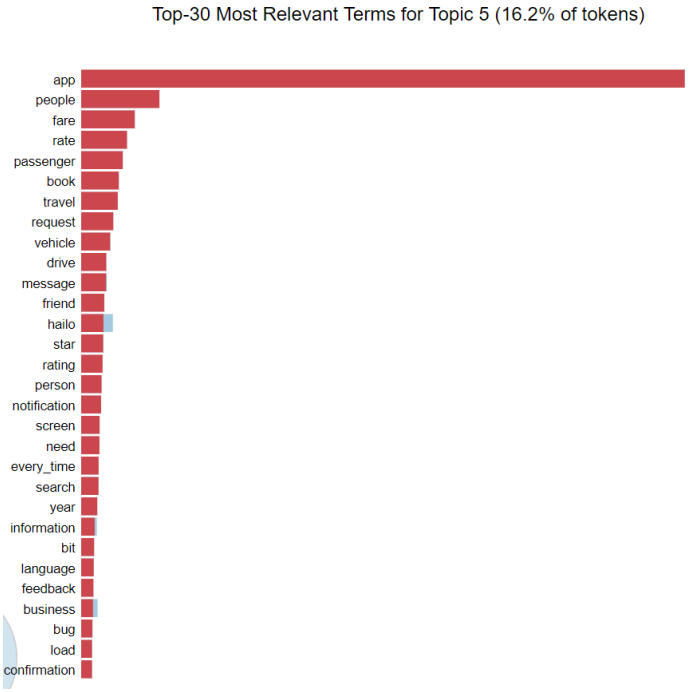

Figure 15. Extracted Keywords From Topic App Experience

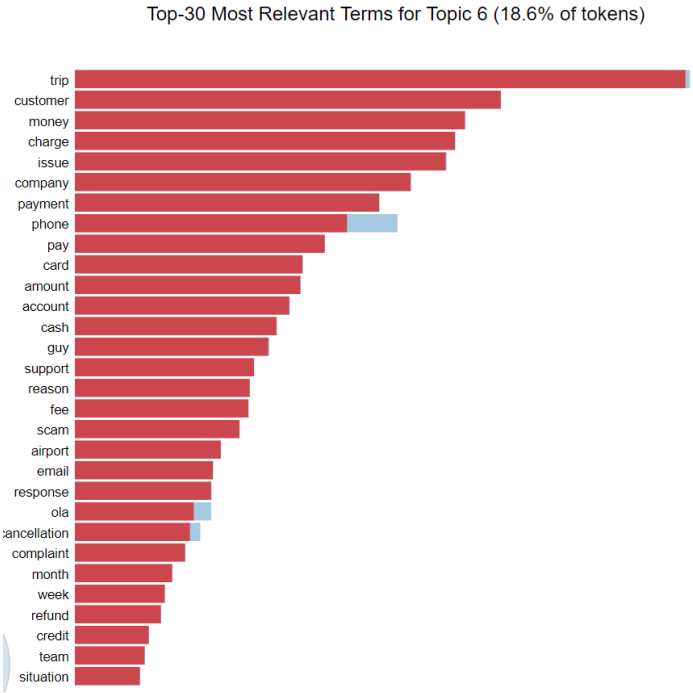

Figure 16. Extracted Keywords From Topic Trip Experience

IV. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Enlik -**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Topic Modeling for Requirements Engineering: An Analysis of Ridesharing App Reviews,

   supervised by Tahira Iqbal and Kuldar Taveter, PhD.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Enlik -
05/08/2022