ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Master Thesis Data Science & Marketing Analytics

Context-Aware Recommender System (CARS) with Textual Features based on Factorization Machines

Ayesha Lynch
Student number: 434520

Supervisor: Prof. Dr. Philip Hans Franses
Second Assessor: Dr. Michiel van Crombrugge

Date final version: August 3, 2020

Abstract: In this thesis, I attempt to improve the accuracy of a User-Based Recommender System by adding text features as contextual information based on Factorization Machines. By extracting sentiment polarity, emotional states and hidden topics from review text data, the basic user-item-rating matrix is enriched with contextual features which are used to extract underlying rating patterns. With a training and testing prediction accuracy of respectively 74.1% and 70,3%, the Context-Aware Recommender System based on Factorization Machines outperforms the other Recommender Systems.

Keywords: Recommender Systems, CARS, Factorization Machines, Sentiment Analysis, LDA, R

Acknowledgement

This thesis would not have been possible without all the mental support I have gotten from everyone close to me.

To all family members and friends who have been extremely understanding and supporting throughout my journey as a student at the Erasmus University: thank you for your encouragement, love and patience.

This thesis is dedicated to my parents. Without their unconditional love, support, hard work and care, I would not have been able to come this far. I am forever grateful to you, and thankful for being your daughter. Now it is my turn to take care of you.

Contents

1. Introduction

Social media makes it possible for anyone to share their opinion: this has resulted in a huge information overload for both businesses and consumers, but it also facilitates product innovation (Castillo et al., 2017) and increases the power of word-of-mouth communication amongst them (D'Addio et al., 2017; Chevalier & Mayzlin, 2006). In addition to product descriptions or image- and video content as a source of prior-purchase information, over 80% of customers consider reviews before completing a purchase online (Brightlocal, 2019; Statista, 2020). Usually, this type of customer-generated text is rather uncontrolled and poorly structured. To extract meaningful insights, data analysts apply text mining techniques which have become increasingly important considering our intensive online activity. Various methods such as Sentiment Analysis (SA) and Topic Modeling allow us to uncover valuable insights from text, which is crucial to understanding specific customers' opinion about specific products as well as making sensible business decisions. For massive companies like Amazon, Netflix or Google, this can be a time-consuming, computationally expensive and dynamic process because of the vast amount of data and changing customer preferences.

Since opinions are shared on a large scale nowadays, online review data can provide information on how to optimize a customer's personalized experience. A popular way of doing this is with Recommender Systems (RSs). From an e-commerce perspective, RSs are tools that partly tackle information overload by showing customers their personalized product suggestions (Ricci et al., 2015). This is mainly based on different characteristics of the customer or other users who are showing similar website activity and interests for the same items. Most RSs are user- or item-based and recommend new products based on their ratings. This fails to acknowledge possible performance gains when looking at textual features from reviews, since ratings are merely numerical whilst review text contains explicit user opinion and product information. A recurring problem with recommending products based on ratings, is that an RS cannot know what exactly distinguishes for example a three-starred from a four-starred product. This information can be found in written reviews by customers. The situation in which a product was bought and reviewed is also ignored when solely relying on numerical ratings: the situation is referred to as the rating context and encompasses all non-user or item specific characteristics of a rating event.

In conclusion, text data is at least equally important to traditional structured and numerical data and should not be neglected by any company. E-commerce companies already use text data to improve for example their e-mail marketing strategy, website content and brand selection or user personalization. Despite the attention in recent studies, textual features in RSs are not as common as user- and item-rating-based systems in practice. In this thesis, I aim to examine whether a Recommender System can be improved by adding (latent) textual features obtained by different text mining techniques.

Pieces of text in a business-customer setting are often rather small, which influences the success of some methods to extract text features. The unstructured nature of text data makes data cleaning very important. The way the data is cleaned, impacts the results and insights one derives. To fully exploit text data's potential can be challenging but is important for a business' survival in the long-term. Recently, Amazon launched a Dutch version of their marketplace which is still in its introduction phase in the Dutch e-commerce market. Therefore, I chose to work with Amazon data to contribute to their development. Amazon faces fierce competition from bol.com, Mediamarkt and Coolblue. By extracting different textual features and analyzing them, I aim to help Amazon's understanding of the customer's attitude towards specific products, but primarily to improve their customers' experience by proposing a context-aware recommendation system for the Dutch Amazon website. In this thesis the following research question is answered:

To what extent can text features help understanding customers' opinions and improve a User-Based Recommender System using Amazon review data?

The remainder of this thesis is organized as follows: Section 2 discusses all theoretical concepts in Text Analytics (TA), topic modeling and Recommender Systems (RSs). Section 3 presents related work on the different methods in this research are and compares papers to the work in this thesis. In Section 4, the data used in this research with the necessary cleaning steps are presented. After this, Section 5 explains in detail how the methods applied to the data work. Section 6 presents the results and Section 7 ends with a conclusion and several limitations of this thesis.

2. Theory

2.1 Sentiment Analysis

Sentiment Analysis (SA) refers to the analytical process of extracting the author's underlying emotions, opinions and feelings from textual data (Kwartler, 2017) and is an important stage of opinion mining in TA. SA is a popular technique that has been brought to fruition over the past two decades with the increased amount of online generated text data on social media, websites and in customer service. Determining a writer's emotional intent can be achieved in various ways. When words in a text are classified as positive, negative or neutral, it is referred to as *sentiment polarity*. Each word contributes to the final sentiment judgement of the sentence or document that is being reviewed by providing a score. For example, the sentence: 'My lunch tastes great!' would be scored as 'My (0) lunch (0) tastes (0) great (1)' = 1. Neutral words get zero points, positive and negative words get scored with one and negative one respectively. Extracting specific emotions like happy, scared or satisfied, is referred to as *emotional states scoring*. A popularized framework of emotional states was created by Robert Plutchik around the 1980s (Kwartler, 2017), which is widely known as the 'wheel of emotions'. Plutchik states the following eight emotions as the basis of emotional evolution for human and animals: (1) anger; (2) fear; (3) sadness; (4) disgust; (5) surprise; (6) anticipation; (7) trust; (8) joy. Based on either pre-defined lists of words (also called lexicons) with each word having an attached emotional state or complex machine learning algorithms, emotional states scoring extracts sentiment on a more detailed level than assigning polarity scores.

SA is a text classification algorithm based on different Natural Language Processing (NLP) methods to extract subjective information (Kumar et al., 2016). Some NLP text operations are stemming, tokenization and Part-Of-Speech (POS) tagging, which is a grammar-based identification of words and lexicons (Liu, 2010). In contrast to POS, a Bag-of-Words (BOW) model relies on the representation of a bunch of related terms, neglecting word ordering and grammar rules. Two leading SA methods are the (1) Rule-based approach and the (2) Automated-based approach. In the Rule-based approach, pre-defined emotions or opinion statements and lexicons are used to label the words in text data as explained earlier. The algorithm takes the text that needs to be labeled and scans through the pre-defined list of positive and negative words. When reviewing all words, it eventually labels the sentence or document as positive, negative, neutral or another class (for example the emotions in Plutchik's wheel). This algorithm is based on the notion that the average semantic orientation of the words serves as an indicator of whether the entire text is positive or negative in the case of polarity.

Automated-based SA depends on machine learning pattern recognition algorithms instead of lexicons to perform sentiment classification. The main goal is to perform sentiment class prediction with the created model, which is not the case for Rule-based approaches. The first step is to extract features from the text using BOW or bags-of n-grams (bigrams, trigrams, skip n-grams etc.) which are different groups of words that appear often together. The extracted features go through a specific classifier such as Naïve Bayes or kNN which returns a sentiment class label (positive, negative etc.). Another way to extract group of words is by applying topic modeling machine learning techniques that find hidden topics within text and returns the group of words that are most frequently used under each topic. Topic modeling is discussed more thoroughly later in this section.

The example sentences: 'I do not like this' and 'I like this very much' contain words that are in this context referred to as *valence shifters*. This type of words affects the polarity of a piece of text and results in more accurate scoring when taken into account. In the example, "not" is a negation word which infers the opposite polarity of "like". The word pair "not like" is therefore considered as negative. In the other sentence, "very much" amplifies the meaning of the word "like" which would normally get a score of 1 based on the lexicon. With the amplifier, the score would be higher than 1 since the positive expression is stronger in this case.

There are also various text problems that are nearly impossible to overcome. Two of the most important challenges that analysts face are sarcasm (Maynard & Greenwood, 2014) and social media language (Kwartler, 2017). Imagine for example a newly released smartphone that has been received very badly by the target audience. People use sarcasm as a form of annoyance or disappointment: in a personal conversation, one can easily confirm a customer's intention of their word choice by simply asking it. A computer algorithm does not have this possibility. If a customer gave the smartphone a rating of 2 stars and left a review saying: 'Well, this phone is so great. It has everything you wish for: overpriced and an old-fashioned design. Excellent! Never been so happy like I am right now', the problem becomes very clear. The review text contains several positive words which will result in a positive polarity when going through analysis. This contradicts the 2-stars rating that the customer gave to the smartphone, resulting in a negative influence on the accuracy of the findings. Sarcasm is currently a hot topic in the research area of Sentiment Analysis and remains a drawback.

Next to sarcasm, there are words and phrases that are common in online language but are not officially real words. For example: 'The design is mehh, but the battery life is great AF!!'. The two terms 'mehh' and 'AF' are expressions which the algorithm will recognize as neutral words since lexicons do not know them. This problem can partially be tackled by creating domain-specific lexicons and manually adding known terms to the list. This is a time-consuming task which requires excellent understanding of the language on social media. Spelling errors and other inconsistencies in word usage like emojis, will remain challenging: even though there exist multiple ways to detect and solve text errors, it becomes more difficult when working with large amounts of text data.

2.2 Topic Modeling

Topic modeling refers to the analytical process of uncovering hidden topics in text data. It is an unsupervised machine learning technique, which means that there is no target variable being predicted, but instead it tries to find word structure and recurring patterns in the data. Understanding underlying topics from, for example reviews, yields useful textual information about products and brands. Several studies show successful applications of topic modeling for general purposes such as topic discovery, document classification (which is supervised learning) and topic trend analysis. A document is defined as any unit of observation in textual data: this can be a single sentence, review, product description, tweet etc.. Documents in turn belong to a corpus, which is the whole text body. Topic modeling relies on the assumption that each document is a mixture of latent topics with each topic being represented by a random mixture of words (Griffiths et al., 2007). It relies on the BOW assumption, meaning that the order of words within a sentence is ignored and word groups are clustered. Based on word frequency, co-occurrence and distance, topic modeling algorithms uncover which words are most similar to each other and clusters them together: this allows an analyst to quickly see what certain topic the words represent. Human interpretation is therefore an important part of topic modeling: the algorithm finds clusters, but domain knowledge and context define the topics. Using this information, the model also provides insights into which topics are present within each document. In topic modeling, each document is represented by a probability distribution over the topics and the topics are represented by a probability distribution over words (Newman et al., 2006).

2.2.1 LDA

Latent Dirichlet Allocation (LDA) is a generative probability-based topic modeling algorithm first introduced by Blei et al. (2003) and is the most widely used topic modeling technique. As the name already suggests, the probability distribution relies on the Dirichlet distribution. LDA has properties of both Principal Component Analysis and Factor Analysis in the sense that it reduces the dimension of the data but distinguishes itself by its strong interpretative quality: the goal of LDA is to uncover hidden topics and their related words which are directly informative. In a textual context, each document in a corpus is a representation of a finite number of topics and each topic is inferred by at least one word (Blei et al., 2003). LDA tries to find out how a document is built up and written by the author. It belongs to the soft-clustering methods because of the following two properties:

1. A word can belong to multiple topics, with different term probabilities

2. A document can be about multiple topics, with different topic probabilities

The LDA model assumes that the prior distribution of topics is 'lost' and needs to be specified again. For example, suppose that a writer already knows that its informative piece of text should be 40% about Topic Modeling and 60% about Data Mining. The writer picks words according to the probability that they belong to either one of the topics: the writer knows that the word 'text' has a higher probability of belonging to the Topic Modeling part of his document and uses it less than the word 'prediction' which belongs more to Data Mining. Similar to this writing process based on simple probabilities, LDA tries to uncover the writing pattern based on a Dirichlet distribution of the probabilities (Onan et al., 2016).

Another widely used topic modeling technique is Non-negative Matrix Factorization (NMF), which has been frequently compared to LDA in different studies (Xu et al., 2003). NMF relies on the idea of decomposing a large matrix V into two lower-dimensional matrices: all elements of the matrices have non-negative values. This is the exact opposite in the case of PCA, where values can be negative. Like LDA, NMF tries to find a finite number of topics but instead of probabilities it returns the factor loadings of each topic (factor) on the words and documents. Both NMF and LDA have their own strengths and weaknesses, but both have good performance across different datasets according to various experiments (Chen et al., 2016; Stevens et al., 2012). Based on topic coherence, stability and human judgement of the topics, LDA often has a little advantage over NMF. In addition, NMF contradicts its own name since least-squares based solution can return negative values and may find a local optimum instead of a global optimum. This means that the algorithm may not find the best solution, but instead a solution that is the best given a certain neighborhood.

2.3 Recommender systems

Recommender Systems (RSs) are a combination of software tools, information filtering and machine learning algorithms that aim to suggest relevant items or services to users who have not seen them before, because of the lack of personal experience (Aggarwal, 2016; Mahmood & Ricci, 2009; Ricci et al., 2011). RSs are one of the most successful applications of machine learning and are becoming an inherent part of modern e-commerce. By predicting user's preferences and finding similar items, Recommender Systems have made a great impact on the shopping experience and sales numbers of numerous information-based companies such as Google and Netflix. The main purpose of a Recommender System (RS) is to tackle the information overload in a large space of options and offering high-level personalized shopping experience. The era of information overload began when the spread of data became so rapid that, in the end, more information was produced than we can actually consume (Di Noia et al., 2012). The design and core technique of a RS depends on the type of decision-making problem it has to tackle: the definition of an item takes on various forms, such as movies, songs, products or webpages (Ricci et al., 2011). Amazon for example personalizes their webpages based on each individual user: there is a part of the website that is the same for every visitor, but Amazon's members of whom personal characteristics and buying behavior are known, see a (partly) personalized version. A less personalized version of a RS is a top-n list of recommended items on the homepage to draw a customer's attention: this type of recommendation is not personal but depends on item popularity and applies to every website visitor. The complexity of RSs in general increases with the number of parameters involved in building a RS. The two basic types of information needed to create a RS are:

1. Characteristics of the items and users

2. Preferences indication in a user-item interaction form (e.g. ratings, purchases etc.). These preferences can be both explicit and implicit.

Extended RSs use various types of item- and/or user features, depending on the recommendation problem. For example, when the recommended items are movies, in addition to the ratings given by users, the genre, director, length of the movie or lead actor(s) can be used as additional information in a RS. Based on this, there are two types of leading techniques in Recommender Systems: Content-based (CB) methods and Collaborative Filtering (CF) methods.

2.3.1 Content-based Recommender Systems

Content-based (CB) Recommender Systems are designed to recommend items or services that are similar to a user's past preferences (Lops et al., 2011). As the name suggests, the recommendations are based on the items' content, like the genre and director in the movie example. In CB recommendation settings, the idea is that the user's own preference on certain items and the item's descriptive set of characteristics are sufficient to make a suggestion (Aggarwal, 2016). Therefore, the ratings of other users have little to no influence on the recommendations that are produced in CB recommendation systems (in contrast to CF which will be explained later). RSs of this type are common in situations in which there is abundant information about the item's characteristics and the emergency of new products is very likely. This is because with new products, there are no preferences known yet: item descriptions must be used in this case. For example, when a new movie is released, no regular cinema visitor has seen it yet, but the cinema's website is still able to recommend the movie to a page visitor based on past ticket purchases to films of the same actor, genre or director. The main challenge is to extract the relevant attributes from these text descriptions and transform them from unstructured data to informational characteristics. Data pre-processing and Text Analytics techniques are therefore an important part of CB recommendation. The item-similarity on which the suggestions are based on, is measured by a similarity value between the user's known items and the new items. Popular approaches to measure similarity in text are term frequency/inverse frequency and the cosine similarity (Singhal, 2001; Di Noia et al., 2012). Content-based Recommender Systems ignore other users' preferences, which can be both an advantage and disadvantage: when new items are added to the suggestions space, RSs face a cold-start problem (Aggarwal, 2016). This means that there is no information about user's preferences available. In this case, CB recommendations have a huge advantage because they are based on historical preferences and item descriptions of the same user. A disadvantage is that the recommendations produced by CB RSs can be obvious and non-divers, therefore adding little to no value to the user. Looking for similarity between users can tackle this by creating more diverse recommendations. RSs that include this are referred to as Collaborative Filtering (CF) methods.

2.3.2 Collaborative Filtering

Collaborative Filtering (CF) is the most widely used and commercially adapted type of RS. The recommendation method in CF is based on the following assumptions (Zhao & Shang, 2010):

1. Users have similar preferences
2. Preferences are stable
3. Choice is predictable based on past preferences

In its core, CF relies on the idea that users who have expressed similar preferences in the past, will continue to do so (Goldberg et al., 1992). As the name suggest, CF utilizes interactions between users (collaborative) to filter out relevant items to a target user (filtering). By connecting for example rating patterns of one user to a bunch of like-minded users, recommendations are created based on both historical personal preferences and those of users that are similar to the target user. There are two types of Collaborative Filtering methods: Neighborhood-based CF and Model-based CF.

2.3.3 Neighborhood-based Collaborative Filtering

The conventional types of Collaborative Filtering are Neighborhood-based methods. Neighborhood-based CF methods (also known as Memory-based) are again dividable into two types: Item-based and User-based. Both methods cluster groups of either users or items, based on implicit/explicit rating similarity.

User-based Collaborative Filtering (UBCF) makes recommendations to one user based on the preferences of users who are like-minded in terms of ratings. In contrast to Content-based RSs, no item attributes or user characteristics are needed, but only historical preferences. The method selects users that are most similar to the target user and creates a top-n list of similar neighbors of which it aggregates their ratings in a way that it becomes a reasonable prediction for the target user's (unknown) rating. The algorithm relies on the idea that users are similar when they show the same rating pattern to the same items. For example, if person A, B & C rated the movies: Lord of The Rings, The Shining and Forrest Gump respectively 5, 4 and 2 stars, and person D rated The Shining and Forrest Gump respectively 4 and 2 stars, the algorithm will return Lord of The Rings as recommended movie for user D because the pattern suggests a rating of 5 stars and thus predicts rating (Lord of the Rings) = 5. Even if the movies are different in genre, story line, cast, etc., the preference pattern of person D's nearest neighbors (which are A, B & C) indicates that Lord of The Rings has a high utility.

Item-based Collaborative Filtering (IBCF) works in analogous way and makes recommendations based on item similarity. The method assumes that items are similar when they received similar ratings from the same user.

Even though Neighborhood-based CF is extremely popular and straightforward, it has several limitations which deteriorate its accuracy. Covering all of them goes beyond the scope of this research, but the two most important problems are sparsity and scalability. Sparsity is an inevitable problem in RS data: missing ratings make it difficult to find item or user similarity. This may easily lead to inaccurate rating predictions and therefore bad recommendations (Koohi & Kiani, 2017; Mahara, 2016). The cold-start problem is also caused by this sparsity. Another problem is the scalability, since RSs become computationally expensive as the number of users and items grow. In commercial recommender systems, Neigborhood-based CF tend to scale badly because of the data-intensity: Model-based systems provide a solution for this.

2.3.4 Model-based Collaborative Filtering

The movie example in UBCF requires users to have rated the exact same movies in order to find a pattern and recommend to another user. In sparse user-item-rating data, this becomes a problem. Imagine three horror movies The Shining, Jigsaw and The Ring being rated very well in general, but never by the same user. Neighborhood-based RSs will have trouble finding a rating pattern and create meaningful recommendations accordingly. This problem and scalability issues are tackled by Model-based Collaborative Filtering methods. The goal of this type of RSs is to create a model for rating prediction based on machine learning and data mining techniques. The most successful and widely used Model-based CF model is Matrix Factorization. Its goal is to represent the high-dimensional user-item-rating vectors into a lower dimensional space (Takács et al., 2008). Rating patterns are inferred by latent (hidden) factors that represent underlying preferences of users. The three horror movies in an earlier example are not rated together by the same user, but have a common latent feature which is in this case the genre. Matrix Factorization extracts these latent factors to shape the relationship between users and items. The algorithm seeks a number of k dimensions to represent both items and users, which is significantly lower than the original number of dimensions in sparse matrices. By lowering the dimensions of the large matrix, it decreases the data memory and computational requirements of the model (Ekstrand et al., 2011). In addition, Matrix factorization yields flexibility in modeling real-life recommendation problems because of the ability to process large datasets and sparse matrices.

2.3.5 Context-aware Recommender Systems (CARS)

Researchers are increasingly paying attention to extending RSs with different variables obtained from metadata and contextual features. It has been proved that item and user characteristics are both valuable information for building RSs, but this metadata can lead to little or no real improvement of the prediction quality of ratings (Pilászy, 2009). Recently, the focus of new approaches in RSs are context-based improvements of CF methods. Recommender Systems that take into account contextual information are referred to as Context-aware Recommender Systems (CARS). The crucial difference between user or item attributes and context is that the attributes apply to a specific product or person whilst contextual information applies to the rating event itself, for example the day of the week or emotional state someone was in when rating an item or a service (Rendle et al., 2011). Content-based RSs and Collaborative Filtering approaches ignore the idea that a user's preferences for products in one context may be different in another context.

To illustrate the importance of context, some examples are discussed. Suppose you booked an all-inclusive vacation for yourself which includes a lot of massages, sauna time, nights out and shopping trips. Years later, you want to visit the same place again, but now you have a partner and children which means your interests and activities will be different. If your trusted travel agent has an automated RS based on your historical preferences and thus is not aware of this changed context, your vacation will not match your current preferences and situation. Another example which supermarkets face on a daily basis, is grocery shopping behaviour. Numerous big supermarket chains have loyalty programs for their customers which allows them to enjoy personalized discounts based on their historical purchase pattern. This can be enriched by looking at time-specific purchase behaviour of the customers. The popularity of products does not only differ across users or seasons, but also on the timing of purchase. In the morning, freshly baked bread is very popular: when a customer receives a discount voucher for bread early in the morning instead of in the evening, the chance of actual usage of the voucher is bigger. The examples show different types of context: on the one hand, changed personal situation is not directly observable for the travel agent. On the other hand, the time of the day is fully observable for the supermarket because it is recorded at each purchase. In the latter case, the context is known directly. Situations in which context in unobservable, no explicit information is provided. In such cases, context is extracted by utilizing latent information in an implicit manner (Adomavicius & Tuzhilin, 2011). This is exactly what Sentiment Analysis and Topic Modeling methods do in the case of text data.

Contextual information can be used in an RS at different stages in the recommender building process: (1) Pre-filtering; (2) Post-filtering and (3) Modeling. In contextual pre-filtering, all context information is used to perform data selection before creating an RS and in post-filtering this happens after an RS model is built on the whole data set (Adomavicius & Tuzhilin, 2011; Domingues et al., 2014). Context variables are used as labels to segment the data before/after an RS is modeled. In this way, contextual information is indirectly accounted for, outside of the RS algorithm. In contextual modeling, the context information is directly used in the Recommender System's algorithm and becomes a part of the rating prediction. Several methods have been developed to perform contextual modeling: some examples are the DaVI approach (Domingues et al., 2011) and Factorization Machines by Rendle (2011). These and other methods are introduced and further discussed in the next section which presents several papers related to this thesis.

3. Related Work

In their research, Kumar et al. (2016) analyzed Amazon reviews and classified each review as either positive or negative using automated-based SA. They applied the following three methods: Naïve Bayes, Logistic Regression and SentiWordNet (lexical resource) and measured the performance of each method based on the recall, precision and F-statistic. To compare the results, they obtained sample performance measures for three products. On all three products, the Naïve Bayes classification outperformed the other methods. The method and results are insightful to some extent but comparing only three products does not necessarily mean that Naïve Bayes is the better method in general. In order to get more insight in what makes the positive reviews different from the negative reviews, sentence-specific word choice could be analyzed to enrich the results. A similar research by Gokulakrishnan et al. (2012) analyzed opinions on Twitter and found that tree-based methods such as Random Forest outperformed different variants of Bayesian classifiers. Comparing these studies shows that pre-processing and sample selection have a big impact on the performance of models and the results tend to be sample specific. This is supported by other researchers who studied this effect, for example Uysal & Kunal (2014) and Chandrasekar & Qian (2016), which indicates that there is no clear best method to perform automated-based SA in this case.

Bagheri at al. (2014) relax the BOW assumption in their analysis and go beyond the standard topic modelling methods to tackle the aspect detection problem in review texts. They extract multiword features (topics) on sentence-level by considering word structure and the semantic relationship between words. The proposed model is called ADM-LDA: Aspect Detection Model based on Latent Dirichlet Allocation, which is an unsupervised model that assumes a Markov chain formation of the topics in a sentence. By comparing the ADM-LDA model to a standard LDA model, results show that the aspects found by LDA tend to be more general and less coherent. The researchers conclude that this is caused by the BOW assumption.

In their paper, Onan et al. (2016) examined the performance of LDA topic modelling as features in a sentiment text classification task. This research is a good example of testing for possible performance benefits by combining multiple methods. LDA is used as a dimension reduction technique to represent the highly dimensional text data in an efficient way. They compared the predictive performance of five classification methods (Naïve Bayes, Support Vector Machines, Logistic Regression, Radial basis function network and K-Nearest Neighbors) to five ensemble methods (Bagging, AdaBoost, Random Subspace, voting and stacking), but found no significant improvements for ensembles compared to single classification methods, nor was there a significant improvement due to the use of LDA topics. This can be explained by insufficient parameter optimization in all different methods: future research might try to improve the LDA topics representation.

A very recent paper by Qiang et al. (2020), examines topic modeling techniques especially for short texts (like review texts). Conventional topic modelling techniques based on word co-occurrences are not appropriate for short text documents because of the lack of sufficient word information within a single document. The researchers define four characteristics of short texts related to topic modeling: (1) Each short text lacks word co-occurrence; (2) Because of this, each document is likely to be about only one topic instead of a mixture of topics; (3) Words that are semantically related but rarely co-occur cannot be captured well; (4) The single-topic assumption may be too strong for certain types of short texts. Three different types of models were created to each tackle one or two of these characteristics. One of them is Dirichlet Multinomial Mixture-based (DMM) which differs from LDA on the assumption that each document is sampled by only one topic over a multinomial distribution. The results showed that the DMM-based method outperformed the other models and basic LDA as reference model. However, the authors state that a meaningful evaluation metric for topic modeling is yet to discover because human interpretation of topics does not always agree with statistical evaluation. In their paper, the authors evaluated the models based on the accuracy of text classification and topic coherence. In addition, the models performed differently on all five datasets used in the experiments. Based on this, I will compare the results of LDA and DMM-based LDA based on perplexity and topic interpretability to eventually decide what topics to include in the Recommender System.

In his well-known article, Hoffmann (2004) improves the performance of a Collaborative Filtering RS by adding latent variables. His proposed method is a model-based algorithm that introduces latent class variables to discover user communities or groups of items. It has properties of both clustering algorithms and dimension reduction techniques, but an important difference here is that the user communities and items are not partitioned into groups. Traditional collaborative filtering methods are memory-based, which can be computationally expensive and lacks an explicit statistical model construction. The model-based extension in this paper shows improvements in accuracy and more stable preference predictions.

In their research, Mahadevan & Arock (2017) improve a Collaborative Filtering RS by integrating sentiment features from review text into a movie and video Recommender System. Their proposed three-step framework first measures credibility of the reviews using helpfulness votes, spotlight reviews and reviewer's quality and filters accordingly. Then, the reviews are analyzed by performing POS-tagging and Negation tagging. Finally, predictions for item recommendations are made by doing user-based CF on the ratings and using the ratings that are inferred from the review text. Based on the Mean Absolute Error (MAE) accuracy measure, the review text-based RS with inferred ratings performs slightly better.

Krishna et al. (2013) applied Sentiment Analysis to text and used the resulting sentiment scores in their RS. Their proposed method recommends places nearby a user's current location using the sentiment of reviews of other places nearby in a Learning Automata (LA) decision making process. LA systems improve their performance by gaining knowledge during its own machine and code process. It characterizes itself as being self-learned: based on positive or negative feedback from its own actions, the system either rewards or penalizes the automation. In their study, Krishna et al. (2013) apply this LA algorithm to the calculation of the sentiment scores which is eventually the base of the location recommender system.

Wang & Blei (2011) use a combination of LDA topic modelling and traditional collaborative filtering to form recommendations about both existing and newly published scientific articles for online platform users. Their algorithm shows an improvement in performance due to considering the content of scientific articles in addition to existing user libraries. This is particularly done to tackle the well-known cold-start problem. In the case of this study, the cold-start problem occurs when new articles are uploaded and hence do not appear in any user's library yet. In the case of new articles, its content must be considered by a Recommendation System to fully integrate them with the recommendation of older articles. In e-commerce, a similar issue occurs when new products are added to an online product range. This again stresses the importance of additional information in a Recommender System.

D'Addio & Manzato (2014) construct a CF Nearest-Neighborhood RS that uses review text to create item profiles instead of using structured product metadata. They try to overcome the problem of noise, false information and highly personal opinions in reviews by applying natural language processing (NLP) tools to extract relevant information. The first RS is created based on an item matrix using metadata and compared to a feature sentiment matrix-based RS. They build the CF algorithm based on *k* nearest neighbours (kNN) which computes the correlation among items using their features vectors. Implementation is done in Python using the ItemAttributeKNN package which has a major advantage because of the built-in evaluation metrics: precision-at-*k* and MAP (Mean Average Precision). The researchers found an improvement compared to their baseline RS but overlooked the problem of items not being reviewed or only having one (extreme) review. In this thesis, I avoid this by taking only the products with at least five reviews.

Domingues et al., (2011) incorporate contextual information in a top-*n* Recommender System to improve the accuracy. They use each contextual attribute as a *virtual item* in their proposed method called DaVI (Dimensions as Virtual Items). Using three datasets, the authors performed multiple experiments with two RSs: Item-based CF and Association Rules (AR). The latter is a Recommender System type in which user-item pairs retrieved from web sessions are used to make recommendations based on a specified set of relationship rules. No data on implicit or explicit ratings are needed. The results show that DaVI slightly improves IBCF in terms of accuracy based on the All-But-One protocol (Breese et al., 1998) and Recall/ Precision metrics. The AR model performs worse with the added context dimensions. It should be noted that the context features were added one at a time: the performance worsened whenever more than one contextual dimension at a time was added to the Recommender System. The authors conclude that identifying the relevant contextual information is a hard task and could be improved further. In my thesis, I aim to tackle this by examining textual features as context.

After the Netflix Prize competition for the best Recommender System in 2006, matrix factorization received an increased amount of attention in papers within the Collaborative Filtering area. Takács et al. (2008) experimented with different types of Neighborhood-based CF and matrix factorization methods for recommendation. Using the Netflix Prize data set, they found that matrix factorization outperforms Neighborhood-based CF based on the RMSE and time complexity. The researchers also tried adding regularization terms to prevent overfitting, but the improvement was not significant. A linear combination of the different methods resulted in the best improvement and tackled the scalability issue very well.

Domingues et al. (2014) explore several text mining techniques for Context-Aware Recommendation Systems (CARS). They infer the contextual data by applying *named entity recognition* (NER) and discovering topic hierarchies. They refer to user-item characteristics as impractical and time-consuming in large data sets. NER refers to the process of extracting informative units from text data like a person, location, date or monetary value and is therefore different from POS tagging which is a grammar-based approach. The authors built four different CARS: (1) pre-filtering combined reduction approach that segments the data preliminary; (2) modelling approach called DaVI, introduced by an earlier paper (Domingues et al., 2011); (3) Weighted post-filtering which uses the context to re-order recommendations; (4) post-filtering which uses the context to filter out the non-contextual recommendations. Based on Precision and MAP metrices, results show that the NER contextual information significantly performs better than the baseline Item-based CF model without context. The contextual information from the topic hierarchies only showed improvement in the post-filtering methods.

Karatzoglou et al. (2010) introduce a CF model based on Tensor Factorization (TF), which is a generalization of Matrix Factorization allowing high-order dimensional matrices instead of the two-dimensional user-item-rating matrix. The results across three different experimental datasets show an improvement of 5 to 30% compared to Matrix Factorization, based on the Mean Average Error (MAE). In Tensor Factorization, context is added as an additional third dimension instead of as features in a two-dimensional matrix (as seen in Domingues et al. 2011) and belong to the contextual modelling category of CARS types. The method of applying TF in a contextual RS is called *Multiverse Recommendations*: no pre- or post-filtering methods are needed since the context is directly used in the RS. Examples of the type of context variables used in this research are gender, companion while watching a movie, season and level of hungriness when ordering food. Note that gender is a user characteristic and should therefore not have been used as context, but as an attribute. TF consistently outperforms Matrix Factorization in this experimental setting, using real-word data.

Shortly after the Multiverse Recommender model introduced by Karatzoglou et al. (2010), Rendle et al. (2011) created a Context-Aware Recommendation System (CARS) with Factorization Machines (FM) which results into an improvement over Multiverse Recommendation both in prediction accuracy and computation time. The researchers used the exact same data sets as Karatzoglou et al. (2010) and found equal performance quality on the densest data set, while FM outperforms Multiverse Recommendation in the sparse data sets. They conclude that the strength of FM lies in the speed and scalability due to the linear model simplicity instead of computational expensive Stochastic Gradient Descent optimization.

In his related work on Factorization Machines, Rendle extended the analysis by exploring different learning techniques using the libFM software in Python (2012). The article also summarizes research on FM and proves that by adding contextual variables to the user-item-rating matrix, FM performs matrix factorization. In this thesis, the Factorization Machines model introduced by Rendle is used for the construction of the Context-Aware Recommender Systems.

4. Data

The data used in thesis for performing SA, LDA and building Recommender Systems are scraped product reviews from Amazon.com. The raw dataset contains 10,063,255 reviews of the product category 'Cell phones and Accessories' from 2002 to 2019. Since new product releases are common in this type of products, RSs are more challenging than others. To compete with Dutch competitors like Media Markt, Coolblue and Bol.com, understanding the customer's brand and product opinions alongside a powerful Recommender System are crucial for Amazon's survival. Hence, I focus on the Cell phones and Accessories category. Given the huge size of the initial dataset and the required prior user information in building the recommender system, the dataset is reduced to a dense subset. Of each unique user and item, I keep at least five reviews. Products that have been reviewed or rated only a few times might be biased because of the lack of information. Also, users who reviewed solely one or some products might be biased since it often represents a unique situation. The dataset now contains 1,128,437 rows and the following relevant variables:

- *asin*; unique Amazon product ID → total of 48,186 unique products
- *reviewerID*; unique reviewer ID → total of 152,184 unique reviewers
- *reviewText*; full text of the product review by an individual user
- *overall*; star-rating of the product on a scale of 1 to 5
- *date*; year, month, day and time of the placed review
- *verified*; logical vector TRUE or FALSE, indicating whether this author of the review is a real person

4.1 Dataset reduction

For coding simplicity, the variable names *reviewText* and *overall* are respectively substituted by *reviewtext* and *rating*. To further subset the data, several data cleaning and filtering steps are necessary:

(1) All reviews for which *verified = FALSE* are removed from the data set to ensure that bots and users who did not actually buy the product are disregarded from the subsequent analysis. Since this causes all reviews in the dataset to be verified, the column is deleted.

(2) All reviews before January 1st, 2016 are deleted: this is because there are products present in the dataset which are not relevant anymore. Especially in electronics, product development goes very rapidly which causes products to be outdated relatively soon. After this modification of the dataset, the *date* column is deleted.

(3) The dataset is highly imbalanced in the *rating* variable. Figure 1 shows the distribution of the ratings and clearly shows an overrepresentation of the higher ratings 4 and especially 5. Although it represents real-life rating situations, the imbalances affect the accuracy of the analyses. To mitigate this, a balanced subset of the data is created which consists of all reviews that have a rating score of *rating* $\leq$ *3* and an approximal similar number of reviews which have *rating* $\geq$ *4*. In the end, the amount of observations (reviews) that have a rating score higher than 4 is reduced to fight the imbalance compared to the number of observations that have a low rating.

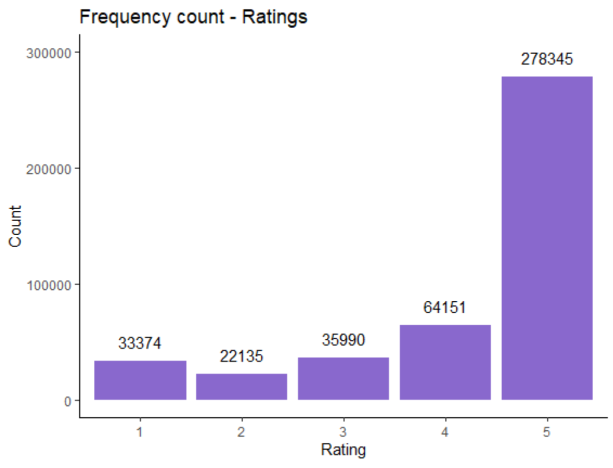


Figure 1: distribution of the rating variable which shows an overrepresentation of high ratings

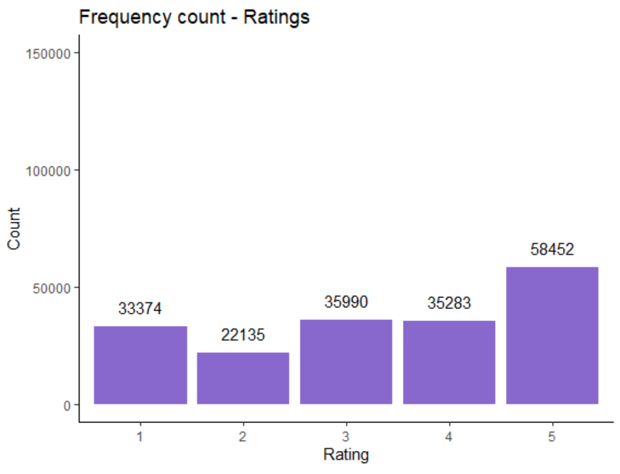


Figure 2: distribution of the rating variable after sampling on rating = 4 and rating = 5

Figure 2 shows the new distribution of the ratings after sampling. For computational purposes, the ratings are transformed to binary indicators of preference. The ratings 1, 2 & 3 have a total number of observations equal to 91,499 which are now all converted to 0. This relies on the assumption that products with a rating lower than or equal to 3 are not preferred by the user because of the low utility it represents. Ratings 4 and 5 have a total number of observations equal to 93,735 and are now converted to 1: this means that users prefer items which have a rating of at least 4 stars.

(4) Modeling Recommender Systems can be very computationally expensive[*] in R. For this reason, the dataset is further reduced by taking a sample of 10,000 unique reviewer IDs. This resulted in the following numbers:

- Total number of rows (reviews) → 20,535
- Total number of unique reviewers → 10,000 (sampled)
- Total number of unique products → 11,149

(5) The last step in the data reduction process is removing duplicated rows. For TA and topic modeling, duplicated rows are not a problem, but they are problematic when building an RS. If the algorithm encounters a duplicated row, it will not be able to tell what to do with the rating score. R provides some solution for this by, for example, allowing the algorithm to sum up the duplicated ratings. However, this is not a suitable solution since it takes away the meaning of the numerical ratings. In total, 50 duplicated rows were removed from the data.

The final dataset has 20,485 observations and the following variables:

- *asin*; unique Amazon product ID → total of 11,149 unique products
- *reviewerID*; unique reviewer ID → total of 10,000 unique reviewers
- *reviewtext*; full text of the product review by an individual user
- *rating*; binary rating of the product: 0s for all products with an original starred rating ≤ 3 and 1s for all products with an original starred rating ≥ 4.

4.2 Text pre-processing

The text in the column *reviewtext* is rather uncontrolled and requires a number of cleaning steps before going through analysis. To match the vectors of specific operations, the order of the cleaning process is important. First, I use a general substitution formula to catch common 'dirt' in text documents like multiple comma's

[*]computation time varies across versions of R, computational power of the pc/laptop and the parameters. In this thesis, resources are limited.

or exclamation marks in a row or contractions (wouldn't, can't) and replaced them respectively with one character or the full written forms.

All letters were transformed to lowercase, followed by removing stopwords. Stopwords are words that are commonly used in written language but add little to no information to the sentiment scores or topic definition and are therefore viewed as irrelevant words. Pre-defined lists of words in R are all written in lowercase, so transforming the letters is an important step to ensure that the algorithm can recognize the words.

The default 'stopwords' list in R has 1149 words that are classified as stopwards. However, when looking closely at the list, I find 22 rows that contain words which are not neutral and hence cannot be viewed as stopwords. These are kept in the analysis; this means that they are removed from the stopwords list. Ignoring them would result in unjustified removal when performing the stopwords clean-up and might affect the sentiment extraction later in the analysis. The rows that are removed from the list are presented in the Appendix – Section A.

Next, the following text features were deleted: punctuation, excess white space and numbers. The last step is stemming the review body, which means that words are reduced to their original 'stem' from which a word is derived (Kwartler, 2017). For example: in 'likely' and 'likelihood', the stem is 'like' and for the words 'battery' and 'batteries', the stem is either 'batter' or 'batteri'. As this shows, the stemmed terms can be non-existing words. Before stemming, a copy of the review text was saved and will be used later in the analysis.

4.3 Sample selection and validation

To respectively create and evaluate the Recommender Systems, the data is split into 75% training and 25% test samples. This is done twice to check the stability of the Recommender Systems' prediction performance: each RS is therefore built on two training sets and evaluated on two test sets.

5. Method

5.1 Sentiment Analysis

Sentiment Analysis will be performed using the ‘sentimentr’ and ‘qdap’ packages in R using the non-stemmed review text since these packages only recognize existing words. Various BOW and bags-of-n-grams are extracted to gain insight into which words appear most frequently together. The ‘sentimentr’ package is considered to be the best option because it takes into account valence shifters when scoring the reviews. In addition, emotional states scoring is performed with the NRC and Bing lexicons in the ‘qdap’ package which assigns words into one or more of the following ten categories: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. The algorithm considers every word it encounters while going through the review text and classifies it twice: first it is classified as either positive or negative and the second classification is one of the eight emotional states defined by Plutchik’s wheel of emotions. Words that do not contribute to either classification stadium are disregarded. When a review consists of neutral words only, all the categories will have returned values of zero. The scores are summed up to a maximum score which is equal to the number of words in the review text. For each review, the most appropriate polarity label and emotional state are determined based on the maximum score across the categories. Extracting bigrams, trigrams and topics (using LDA), will provide insight into customer’s opinion about products and related aspects. The goal of the Sentiment Analysis is to use the sentiment classification result as contextual features in the Recommender System.

5.2 LDA

In this thesis, Latent Dirichlet Allocation (LDA) is applied to the review text data to uncover hidden topics and use them as contextual features in a Recommender System. The generative process of document $\boldsymbol{n}$ in corpus $\boldsymbol{D}$ with $\boldsymbol{K}$ number of topics is as follows (Blei et al., 2003; Newman et al., 2006):

- Determine $\boldsymbol{\beta_k} \sim \boldsymbol{Dir}(\boldsymbol{\delta})$, where $\boldsymbol{\beta_k}$ is the distribution of the words for topic $\boldsymbol{k}$ and parameter $\boldsymbol{\delta}$ governing the prior Dirichlet distribution of $\boldsymbol{\beta_k}$

- Determine $\boldsymbol{\theta_n} \sim \boldsymbol{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\theta_n}$ is the topic distribution for document $\boldsymbol{n}$ and parameter $\boldsymbol{\alpha}$ governing the prior Dirichlet distribution of $\boldsymbol{\theta_n}$

For each word $\boldsymbol{w_i}$ in document $\boldsymbol{n}$:

- $\boldsymbol{k_{i,n} \sim Multinomial\ (\theta_n)}$ → select topic $\boldsymbol{k}$ for each word $\boldsymbol{i}$
- $\boldsymbol{w_{i,n} \sim Multinomial\, p(w_{i,n} \mid k_{i,n}, \beta_k)}$ → select word $\boldsymbol{i}$ conditioned on topic $\boldsymbol{k}$ based on multinomial probability

The algorithm relies on a prior Dirichlet distribution of the terms and topics in a corpus. The Dirichlet distribution is a multivariate probability distribution that creates a finite amount of numbers which all must sum up to 1 and are non-negative (the same applies to Multinomial distribution). The numbers are a vector of probabilities $\boldsymbol{p}$ which as used as prior topic or term distributions in the case of LDA. These prior distributions are used to generate the topic and term probabilities. The strength of the probability vector's distribution is governed by parameter $\boldsymbol{\alpha}$:

- when $\boldsymbol{\alpha < 1}$, the distribution $\boldsymbol{p} \sim \text{Dir}(\boldsymbol{\alpha})[0,1]$ becomes spiked around the 0 and 1
- as $\boldsymbol{\alpha}$ increases, the distribution $\boldsymbol{p} \sim \text{Dir}(\boldsymbol{\alpha})[0,1]$ moves to the center

$\boldsymbol{\alpha}$ controls the sparseness of the topic probabilities: an extremely low value of $\boldsymbol{\alpha} = 0.05$ for example, means that the probabilities generated by the Dirichlet distribution will be either extremely close to 0.0 or nearly 1.0. Translated to the LDA algorithm, it means that a word belongs either not or solely to one topic and a document is either about no topic at all, or entirely about one topic. This is the basis of Dirichlet Multinomial Mixture-based (DMM)-LDA modeling as explained by Qiang et al. (2020), mentioned earlier in the Related Work section of this research. Similar to LDA, it relies on the prior Dirichlet distribution of probability but DMM-LDA differs from LDA by the assumption that each document is sampled by only one topic.

In this thesis, the LDA and DMM-LDA model will be generated using the 'topicmodels' package in R. The input is a Document-Term-Matrix: this is an extremely sparse matrix which has reviews in the rows and terms as columns. Estimating the LDA model requires specific steps in determining the parameters, which happens as follows:

1. Select number of topics $\boldsymbol{K}$ using validation data based on the *perplexity* evaluation measure. The perplexity of a set of documents is calculated by the following formula:

   $\boldsymbol{perplexity}\ (\boldsymbol{D_{val}}) = \mathbf{exp}\ (-\frac{\mathbf{log}(p(w_{i,n}))}{\boldsymbol{Total\ number\ of\ words}}) \rightarrow$ the lower the perplexity, the better: balances fit and model complexity.
2. Select $\boldsymbol{\alpha}$ based using either Gibbs sampling or Variational Expectation Maximization (VEM). Both methods assume $\boldsymbol{\alpha_1} = \boldsymbol{\alpha_2} = \cdots = \boldsymbol{\alpha_K}$ by default, but only VEM has an estimation algorithm for $\boldsymbol{\alpha}$. Although VEM is faster, Gibbs sampling usually works better especially in short documents (Qiang et al., 2020), which is why I decided to perform Gibbs sampling and select $\boldsymbol{\alpha}$ also based on the perplexity measure.

The DMM-LDA model is built using the optimal value of $\boldsymbol{K}$ determined by the validation data set and by minimizing the perplexity with a fixed value of $\boldsymbol{\alpha} = 0.005$.

Using the term and topic probabilities, interpretation of the DMM-LDA and LDA results is possible. An important part of the interpretation part is that solely relying on the perplexity value might not suffice since human interpretability of the topics is at least equally important. Therefore, evaluation of both models will be based on the perplexity score, topic coherence and topic interpretability. The topics of the best model are used as contextual information in the Recommender System.

5.3 Recommender Systems

In this thesis, Recommender Systems are built based on the User-Based Collaborative Filtering (UBCF) algorithm and Factorization Machines (FM). The goal of a Recommender System (RS) is to suggest the item $\boldsymbol{j}$ to user $\boldsymbol{i}$ which yields the highest utility: in this case, utility is inferred by the predicted rating $\boldsymbol{r}$. The input of a RS is the user-item-rating matrix denoted as $\boldsymbol{R}$, where users are represented by the rows, items in the columns and ratings in the intersection.

5.3.1 User-Based Collaborative Filtering

In UBCF, building the Recommender System requires two steps: first, user's historical preferences need to be defined, then user-similarity must be calculated to find nearest neighbors and finally, prediction of the missing ratings (Zhao & Shang, 2010). The problem definition of CF in general, is modeled as follows (Takács et al., 2008):

- The user-item-ratings matrix $\boldsymbol{R(i,j,r)}$ is an incomplete matrix since only a subset of the matrix is observed (Aggarwal, 2016).
- The goal is to predict missing ratings $\boldsymbol{\hat{r}}$ such that the Root Mean-Squared Error (RSME) of the estimate given by:

$$\boldsymbol{RSME = \sqrt{E\{(\hat{r}-r)^2\}}}\,,$$

  is minimized.

The user-item-ratings matrix represent user's historical preferences. Recommendations are based on user similarity which can be calculated in various ways. Depending on the rating type, one metric is more suitable than the other but the difference in performance differs across datasets. In this thesis, the ratings are converted to binary values, which allows me to compare the Cosine correlation coefficient (Breese et al., 1998) and Jaccard Index (Jaccard, 1901) as similarity measures in the UBCF. Cosine similarity is calculated as follows:

$$\boldsymbol{Cos(i_x i_y) = \frac{\sum_{j\,\in\,j_{x,y}} r_{x,j} r_{y,j}}{\sqrt{\sum_{j\,\in\,j_{x,y}} r_{x,j}^2}\sqrt{\sum_{j\,\in\,j_{x,y}} r_{y,j}^2}}}$$

- $\boldsymbol{i_x}$, $\boldsymbol{i_y}$ are respectively user $\boldsymbol{x}$ and $\boldsymbol{y}$; $\boldsymbol{j\ \in\ j_{x,y}}$ indicates the items that are evaluated by both users; $\boldsymbol{r_{x,j}}$ and $\boldsymbol{r_{y,j}}$ are respectively the rating $\boldsymbol{r}$ given to item $\boldsymbol{j}$ by user $\boldsymbol{x}$ and $\boldsymbol{y}$

The Jaccard Index (JI) is calculated as:

$$\boldsymbol{JI(i_x i_y) = \frac{\sum_{j\,\in\,j_{x,y}} r_{x,j} r_{y,j}}{r_{x,j\notin j_y} + r_{y,j\notin j_x} + \sum_{j\,\in\,j_{x,y}} r_{x,j} r_{y,j}}}$$

- $\boldsymbol{r_{x,j\notin j_y}}$ are the ratings of items that are only rated by user x and not y ($\boldsymbol{j\notin j_y}$); and $\boldsymbol{r_{y,j\notin j_x}}$ are the ratings of items that are only rated by user y.

The final rating prediction is calculated by the weighted average of the target user's nearest neighbors. For the construction of the UBCF Recommender System, I use the 'recommender' package which uses a user-item-rating matrix as input. The UBCF method requires the number of nearest neighbors ($\boldsymbol{nn}$) to be set as a hyperparameter before running the model. Based on this parameter, a top-($\boldsymbol{nn}$) list of recommended items for each user can be extracted.

5.3.2 Factorization Machines

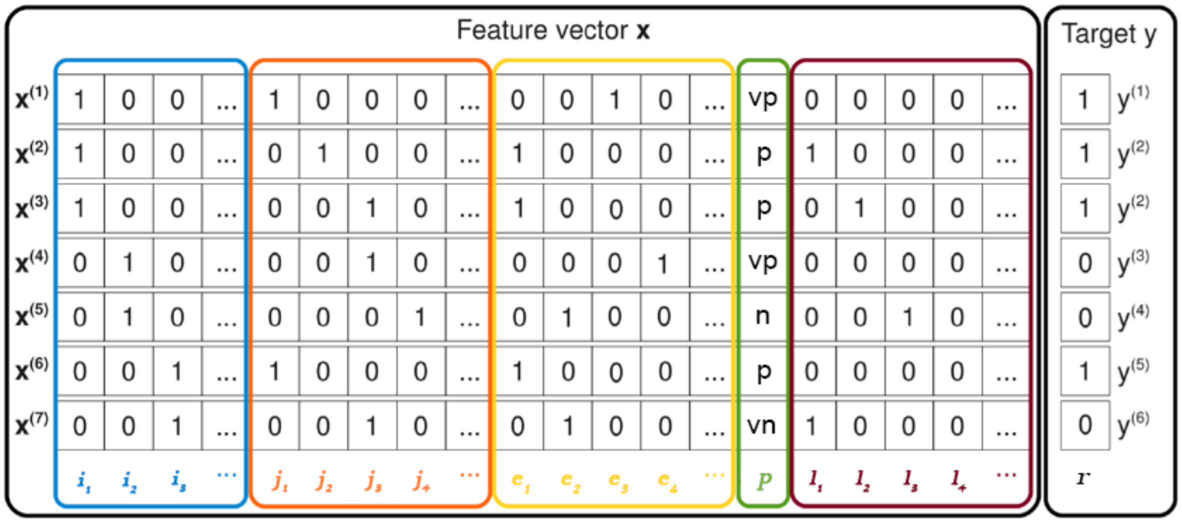
Factorization Machines (FM) are one of the most recent generalizations of matrix factorization for Recommender Systems which allows for the inclusion of contextual features into an RS (Rendle, 2012). Matrix factorization is a widely applied method for model-based CF systems. The notion behind matrix factorization is that the user-item-rating matrix $\boldsymbol{R_{i*j}}$ is decomposed and approximated by two smaller matrices in a lower $\boldsymbol{k}$- dimensional space:

$$\boldsymbol{R_{i*j}(i,j,r) \approx W_{i*k} * H_{k*j}},$$

where $\boldsymbol{W}$ has $\boldsymbol{i}$ (users) rows and $\boldsymbol{k}$ latent factors; and $\boldsymbol{H}$ has $\boldsymbol{k}$ rows and $\boldsymbol{j}$ (items). The objective is to be as close as possible to the real matrix $\boldsymbol{R_{i*j}(i,j,r)}$, thus minimizing the distance between $\boldsymbol{R_{i*j}}$ and $\boldsymbol{W * H}$ based on the squared Frobenius distance:

$$\mathbf{min}\left|\boldsymbol{R_{i*j} - WH}\right|_{\boldsymbol{F}}^{\mathbf{2}}$$

FM uses feature engineering to incorporate contextual information in a two-dimensional matrix and mimics the matrix factorization approach by doing so (Rendle, 2012). The complexity of FM is, in contrast to Multiverse Recommendation (Karatzoglou et al., 2010) linear in $\boldsymbol{k}$ which allows for faster computations even with a lot of contextual data. By factorizing the variables' relationship in a lower-dimensional space, FM can deal with sparse matrices because the interaction of each feature pair is used to compute the interaction of missing feature pairs. Figure 3 presents an example two-dimensional feature vector matrix $\boldsymbol{x}$ with contextual variables. Each observation in $\boldsymbol{x}$ includes the user, item and the context variables with the corresponding rating value $\boldsymbol{r}$.



	Feature vector **x**																				Target y	
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0	0	1	0	...	vp	0	0	0	0	...	1	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	1	0	0	0	...	p	1	0	0	0	...	1	$y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	1	0	0	0	...	p	0	1	0	0	...	1	$y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0	1	...	vp	0	0	0	0	...	0	$y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	1	0	0	...	n	0	0	1	0	...	0	$y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	1	0	0	0	...	p	0	0	0	0	...	1	$y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0	1	0	0	...	vn	1	0	0	0	...	0	$y^{(6)}$
	i_1	i_2	i_3	...	j_1	j_2	j_3	j_4	...	e_1	e_2	e_3	e_4	...	p	l_1	l_2	l_3	l_4	...	r	

Figure 3: Data representation for a CARS problem. Feature vector of the data used in this thesis: this figure is a modified version of the one introduced by Rendle (2010). Every row represents a unique vector that includes users (blue) items (orange) and the context variables: emotional states (yellow) polarity scores (green) and LDA topics (purple). The last column contains the corresponding binary rating value (***r***).

For explanatory and visual purposes, each feature type has its own color. Users (***i***) and items (***j***) are binarized and make a very sparse matrix together: every row contains a value of 1 only twice (unique user and item). It becomes sparser with the added contextual variables extracted from the review text. The yellow feature represents the binarized emotional states (***e***) obtained by sentiment analysis. The green column contains the categorized polarity scores (***p***) and the purple columns show the binarized indicators of the LDA topics (***l***). In the discussing of the results, the creation of the context variables will be explained further.

The FM model is an approximation of a linear regression model and is given by the following equation (Rendle, 2012):

$$\hat{\boldsymbol{r}}(\boldsymbol{x_i}) := \boldsymbol{w_0} + \sum_{j=1}^{p} \boldsymbol{w_j x_i} + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} \{\boldsymbol{v_j v_{j'}}\} \boldsymbol{x_j x_{j'}}$$

- $\boldsymbol{p}$ is the total number of variables
- $\hat{\boldsymbol{r}}(\boldsymbol{x_i})$ is the predicted rating for each observation $\boldsymbol{i}$
- $\boldsymbol{w_0}$ is the global bias, $\boldsymbol{w_j}$ represents the strength of the $\boldsymbol{j}$th variable and $\boldsymbol{x_i}$ is the sparse feature vector for each observation $\boldsymbol{i}$

- $\{\boldsymbol{v_j v_{j'}}\}\boldsymbol{x_j x_{j'}}$ represents the interaction between variable $\boldsymbol{j}$ and $\boldsymbol{j'}$: $\{\boldsymbol{v_j v_{j'}}\}$ are unobserved variables which represent underlying interactions between pairs, inferred from the contextual variables' behavioral pattern.

Intuitively, the equation shows that the rating prediction in FM is a combination of observed and latent interactions between items. The latent variables allow for predicting ratings of user-item pairs which are otherwise unobserved. By the inferred relationship between two items using contextual information, unknown ratings are predicted.

I use the 'libFMexe' package in R to create the Factorization Machines RSs. This is a wrapper for the libFM Pyton package which allows different learning algorithms like Stochastic Gradient Descent, Alternate Least Squares and Markov Chain Monte Carlo. The FM RSs will be constructed based on the Markov Chain Monte Carlo (MCMC) learning algorithm because it handles complicated probability distributions in sparse settings rather well. In addition, the package allows for hyperparameter tuning using cross validation to determine the following two hyperparameters:

1. The number of latent dimensions $\boldsymbol{k}$
2. The standard deviation initialization $\boldsymbol{sd} \rightarrow$ this initializes the standard deviation used for the distribution of the unobserved variables $\boldsymbol{v}$ (FM equation)

For both hyperparameters, the optimal value is chosen based on the lowest error rate (1 – accuracy rate). Two FM-based RSs are created: one with context and one without context. Both RSs are compared to each other and the UBCF Recommender System based on a few evaluation metrics.

5.4 Evaluation Metrics

Evaluation of recommender systems is challenging since there is no basic evaluation metric to assess the performance (Herlocker et al., 2004). The RSs in this thesis will be evaluated based on the prediction accuracy by comparing actual and predicted ratings with the help of a confusion matrix. This is a simple summary of the number of predicted and actual values. A more recent evaluation metric is the adjusted precision metric – precision at *k* - also seen in D'Addio & Manzato (2014) where *k* was equal to 10. Precision is an evaluation metric that is used for binary output.

In the context of recommender systems, precision at *k* returns the number of relevant items that were recommended to the user based on the item's position and taking into account the length of the whole item rank. In this thesis, I will evaluate the RSs based on the prediction accuracy since an item rank is computationally difficult to obtain when multiple brands are involved.

6. Results

In this section, the creation and interpretation of all context variables (emotional states, polarity scores and LDA topics) are discussed. After each modification of the variables, an updated version of the data set is presented to show how the context variables are prepared before entering the Recommender Systems. Then, the results of the User-Based CF, FM-based RS and FM-based RS with context variables are discussed and compared with each other. Finally, the best Recommender System will be selected based on the prediction accuracy.

6.1 Sentiment Analysis

The results of the Sentiment Analysis consist of emotional states scores and polarity scores. First, I discuss the emotional states scoring using the NRC sentiment dictionary. Each word in the review text is assessed and scored on the ten categories mentioned earlier: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. Table 1 presents the first six rows of the data set including the first columns of the emotional states scores.

Table 1: Glimpse of the data with the emotional states scoring results. Each value represents the number of counted words for every emotion.

reviewerID	*asin*	*reviewtext*	*rating*	*Anger*	*Anticipation*	*Disgust*	*Fear*	*Joy*	*Sadness*	*(…)*
1	1	bulky buttons (…)	1	1	1	1	1	1	0	
1	1707	quality feel (…)	1	0	0	0	0	0	2	
1	2317	protect likely (…)	1	0	1	0	0	0	0	
1	923	bought samsung (…)	1	0	0	0	0	0	0	
2	2	good viewing (…)	1	0	1	0	0	1	0	
2	2683	study gaudy (…)	1	0	0	0	1	0	1	

Each emotional state has its own column where the values represent the number of words that fit the corresponding emotion. For example, reviewer two is happy with item number two: there are zeros everywhere except for in the columns *Anticipation* and *Joy*. By linking this row to the original data, it showed that this person rated the product with 4 stars (which is binarized to *rating* = 1). For each observation, the emotional state that fits the review text best is saved in a new column *emotionMAX*. This is based on the maximum score in each row: for row two, the best suited emotion is *Sadness* because the highest value in that row is '2'. The maximum emotional states are used as context in the CARS based on Factorization Machines. When a row does not have a unique maximum, the first emotional state is returned as the maximum. For example, if row two would have a value of '2' under *Anger*, it would be first emotion that the algorithm encounters when searching for a maximum and therefore returns it as the best fitted emotional state. This indicates that *Anger* occurs more often under the *emotionMAX* column than justified, which introduces bias and inaccurate scoring: to mitigate this, the second type of contextual information allows for an extra criterion. For each review, the polarity score is calculated based on the BING lexicon in the 'qdap' package in R. Neutral words get zero points, positive and negative words get scored with one and negative one respectively. The final judgement of the sentence is either a negative or positive number. However, if a review consists of an equal amount of positive and negative words or solely neutral words, the algorithm returns a zero. The polarity scores are converted to a categorical variable with five levels:

- A score between -5 and 0 is converted to *negative*
- A score lower than -5 is converted *very negative*
- A score exactly equal to 0 is converted to *neutral*
- A score between 0 and 5 is converted to *positive*
- A score higher than 5 is converted to *very positive*

The new variable is stored in the *polarity* column. Table 2 shows an updated version of the data set.

Table 2: Glimpse of the data with the emotional states and polarity scoring results. The column *polarity* is based on the polarity scores from the column *sentBING*.

reviewerID	*asin*	*reviewtext*	*rating*	*Anger*	*(…)*	*sentBING*	*polarity*	*emotionMAX*
1	1	bulky buttons (…)	1	1	1	0	Neutral	Anger
1	1707	quality feel (…)	1	0	0	1	Positive	Sadness
1	2317	protect likely (…)	1	0	1	-1	Positive	Anticipation
1	923	bought samsung (…)	1	0	0	-1	Negative	Neutral
2	2	good viewing (…)	1	0	1	1	Positive	Anticipation
2	2683	study gaudy (…)	1	0	0	-2	Negative	Sadness

Figure 4 and 5 respectively show the distribution of the *polarity* variable for *rating* = 0 (meaning 1 to 3 stars) and *rating* = 1 (which is 4 & 5 stars). The results are in line with the idea that reviews should be more positive for high ratings compared to low ratings: there are twice as many reviews with positive polarity scores for the high rated products (>8000 in Figure 5) compared to the lower rated products (4000 in Figure 4). In addition, the number of negative reviews based on the polarity scores is four times higher for the low-rated products compared to the 4 and 5-starred products.

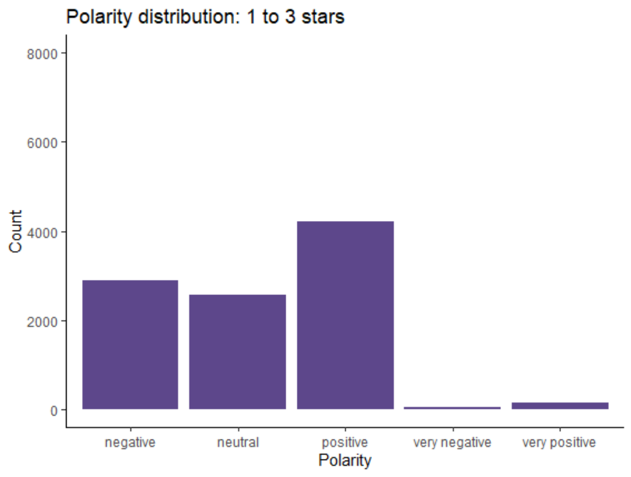


Figure 4: distribution of the polarity variable for *rating* = 0, which is the binarized value for original ratings of 1 to 3 stars

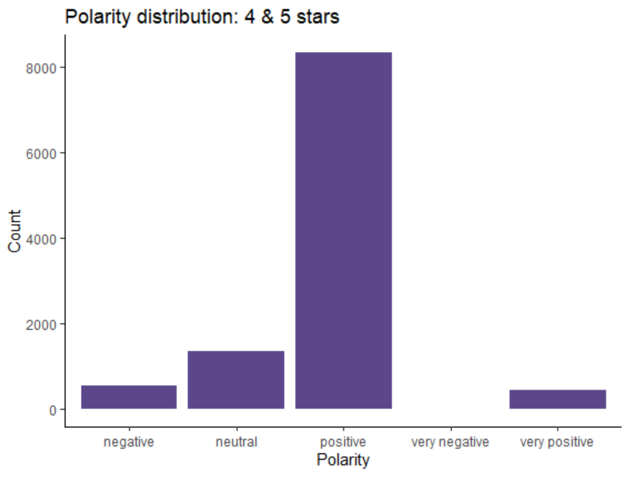


Figure 5: distribution of the polarity variable for *rating* = 1, which is the binarized value for original ratings of 4 & 5 stars

Finally, to tackle the bias towards the *Anger* emotion, I replace the value *Anger* under *emotionMAX* if at least one of the following two conditions are true:

1. if the original rating was either 4 or 5 stars and *polarity* = 'positive' or 'very positive', then *Anger* is replaced with *Joy*
2. if the columns *positive* and *negative* both have a sentiment value of '0', then *Anger* is replaced with *Neutral*

6.2 LDA

For the creation of the LDA and DMM-LDA models, the data set is split into 80% training and 20% validation set. Using cross-validation, the optimal number of topics (***k***) is selected based on the lowest perplexity score. Figure 6 presents the perplexity plot for both training and validation data for all values of ***k***[5:60]. As the number of topics increases, the perplexity decreases. At ***k*** = *45*, both lines go up again, which means that the optimal number of topics based on the perplexity score is 45. Next, the optimal value for alpha α is determined using cross-validation for all values of $\alpha[0.1{:}2]$. The results showed that the perplexity is at its lowest for $\alpha = 0.05$. As mentioned earlier, deciding the parameters with the help of cross validation does not ensure topic interpretability. To assess this, a random number of topics with their corresponding terms are discussed.

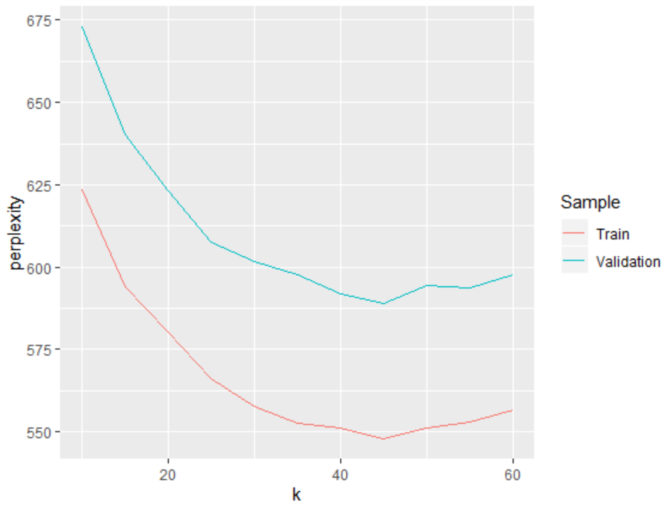


Figure 6: LDA perplexity plot. Minimum at $k = 45$.

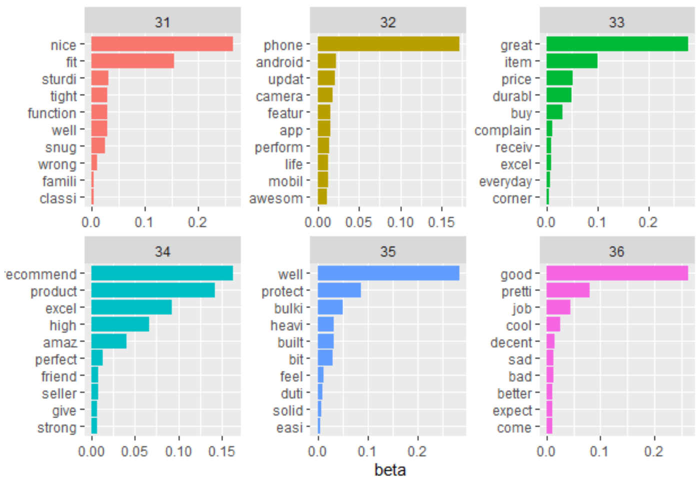


Figure 7: LDA Topics 31 to 36 with the corresponding top-10 terms

Figure 7 shows topic 31 to 36 and their respective top-10 terms that describe each topic. For example, topic 33 is about the monetary and economic value of the product and topic 35 about various aspects of the material. In the Appendix section B, all 45 topics and its terms are displayed. Since the topics are well interpretable, all 45 are kept in the analysis which means that a second cross validation is not necessary.

The DMM-LDA model (Qiang et al., 2020) is created using the optimal value of $\boldsymbol{k} = 45$ and a fixed value of $\alpha = 0.005$. This model returned the exact same topics as the LDA model, contradicting the findings of Qiang et al. (2020). There is no evidence in this thesis that short texts are more likely to be only about one topic. On the contrary, the topic descriptions in Figure 7 and Appendix section B support the findings by Qiang et al. (2020) that a topic is described by only a few words when considering short texts like reviews.

The LDA algorithm returns 45 probabilities for each review: this indicates the extent to which a review belongs to a certain topic. The maximum topic probability per review is extracted and the corresponding topic number is added to the data set as a new contextual feature. Table 3 contains an updated version of the data set.

Table 3: Glimpse of the data with the emotional states and polarity scoring results. The column *topicMAX* is based on the maximum topic probability per row.

reviewerID	*asin*	*reviewtext*	*rating*	*(…)*	*sentBING*	*polarity*	*emotionMAX*	*topicMAX*
1	1	bulky buttons (…)	1	1	0	Neutral	Anger	topic14LDA
1	1707	quality feel (…)	1	0	1	Positive	Sadness	topic39LDA
1	2317	protect likely (…)	1	0	-1	Positive	Anticipation	topic26LDA
1	923	bought samsung (…)	1	0	-1	Negative	Neutral	topic34LDA
2	2	good viewing (…)	1	0	1	Positive	Anticipation	topic7LDA
2	2683	study gaudy (…)	1	0	-2	Negative	Sadness	topic44LDA

With the added topics, hidden subjects and item aspects are uncovered. Combining the results of the Sentiment Analysis and LDA topics allows for an aspect-based interpretation of the review texts. This mimics the AMD-LDA model introduced by Bagheri et al. (2014). Observation six in Table 3 is a review by user number 2, for item number 2683. The sentiment results show that this reviewer is rather negative and/or sad about the product. However, *rating* is equal to 1, meaning that it received 4 or 5 stars. This is a contradicting observation: by taking into account the topic, it becomes clear what aspect of the product is bothering this person. Topic 44 (see Appendix B) is mainly about cellphones and Verizon, which is a telecommunication concern in the United States. In conclusion, the results show that the Sentiment Analysis and extracted LDA topics together have an important role in adding valuable information to numerical ratings. It should be noted that the results are not flawless, as seen in row 2: here, a positive polarity is accompanied by a sad emotional state and a high rating value. Only by reading the entire review text, it becomes clear that the user discussed both positive and negative aspects of the product. Interpretation of results with contradicting polarity, rating and emotional state without reading the whole review text, is impossible in such cases.

6.3 Recommender Systems

The final part of the Results section presents the UBCF Recommender System, FM-based Recommender System without context and FM-based Recommender System with context variables. Each RS is built on two sub-samples of the data set to assess the stability of the prediction performance. The data is split into 75% training and 25% test data for both sub-samples.

The UBCF RS is constructed using the following hyperparameters and specifications: $nn = 15$, $method =$ Gibbs, $similarity =$ Jaccard. Using a confusion matrix, the prediction accuracy for both test and training data of sample 1 and 2 can be calculated. Table 4 shows the confusion matrix for the training data of sample 2 as an example. The accuracy is calculated by the sum of the diagonal cells [0,0] and [1,1] which represent the correctly predicted ratings, divided by the sum of all cells. The prediction accuracy is approximately 64.1% for the training data and 63.9% for the testing data. Results of sample 1 are presented in Table 5.

Table 4: Confusion matrix – Sample 2, training data: prediction accuracy = 64.1%

CONFUSION MATRIX		*Actual values*	
		0	1
Predicted values	0	1923	3431
	1	2031	7980

The results of the UBCF are moderate and show no significant difference between the samples. This means that the results are stable. The prediction accuracy of both samples is used as a benchmark for evaluating the Factorization Machines RSs.

The Factorization Machines RSs are built with the Markov Chain Monte Carlo (MCMC) learning algorithm because it handles complicated probability distributions in sparse settings rather well. To determine the optimal number of latent dimensions ($\boldsymbol{k}$), a two-fold cross-validation with 500 iterations of MCMC on $\boldsymbol{k}$ is performed. Figure 8 shows the results of the cross-validation with the error rate as a function of $\boldsymbol{k}$. At $\boldsymbol{k} =$ 25, the error rate is the lowest. The last hyperparameter that needs to be determined is the standard deviation initialization ($\boldsymbol{sd}$). Again, using cross validation and a trade-off between the error rate and the $\boldsymbol{sd}$, results showed that the error rate is the lowest for an $\boldsymbol{sd}$ value of 0.10.

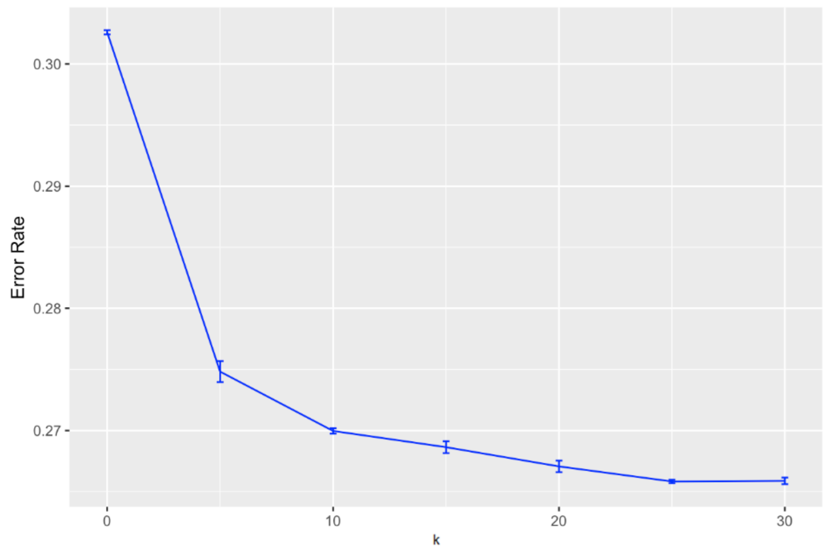


Figure 8: FM error rate plot based on a two-fold cross-validation with 500 iterations. Minimum at $k = 25$.

The final FM-based Recommender Systems are built with the hyperparameters $\boldsymbol{k}$ and $\boldsymbol{sd}$ set to 25 and 0.10. For both samples, the FM without context and FM with context outperformed the UBCF Recommender System. The results show that both RSs are stable across the samples. However, the difference between training and test accuracy is bigger for the FM with context compared to the FM without context. This means that the FM with contextual variables performs relatively worse on the test set compared to the FM without context. A possible explanation for this is the contradicting observations which occurs when the contextual variables indicate different opinion (positive polarity and a sad emotion). However, the highest accuracy rates are those of the FM with contextual variables on the training set for both samples. The prediction accuracy of the UBCF, FM and FM with context variables for sample 1 and 2 are respectively presented in Table 5 and 6. The highest prediction accuracy percentages are seen in Table 5, which shows the results of sample 1:

- On the training set, the FM with context variables performs best among all Recommender Systems with a prediction accuracy of 74.1%
- On the test set, the FM with context variables performs best among all Recommender Systems with a prediction accuracy of 70.3%

Table 5: Prediction accuracy of the three Recommender Systems on the training and test set of sample 1

RECOMMENDER SYSTEMS COMPARISON TABLE	PREDICTION ACCURACY TRAINING/TEST
UBCF	64.1%/63.0%
FM WITHOUT CONTEXT	70.8%/69.0%
FM WITH CONTEXT	**74.1%/70.3**%

Table 6: Prediction accuracy of the three Recommender Systems on the training and test set of sample 2

RECOMMENDER SYSTEMS COMPARISON TABLE	PREDICTION ACCURACY TRAINING/TEST
UBCF	64.4%/63.9%
FM WITHOUT CONTEXT	68.9%/68.8%
FM WITH CONTEXT	73.4%/67.1%

In summary, the results show that Factorization Machines improved the UBCF Recommender System on both samples. The addition of contextual variables increased the prediction accuracy on both training sets by respectively 3.3 and 4.5 percent point. The improvement on the test set is only 1.3 percent point for sample 1 and -1.7 percent point for sample 2, meaning a deterioration of the prediction accuracy. Overall, the Context-Aware Recommender System based on FM proved to be an improvement over UBCF.

7. Conclusion

This thesis tried to improve the prediction performance of a User-Based Collaborative Filtering (UBCF) Recommender System by adding features obtained from review text as contextual information. Three different contextual features are constructed and used in a Factorization Machines RS: two sentiment-based features; emotional states and polarity scores, and LDA topics which uncovered hidden subjects in the review texts. The data used in this research are scraped reviews from Amazon. Compared to a User-Based Collaborative Filtering RS, the Factorization Machines RS showed improvement on the training set for both samples, but little to no improvement on the test set. This observed difference is very small, but it is still in line with the findings of Kumar et al. (2016) and Karatzoglou et al. (2010) where both papers found their method's performance differ across samples from the same data set. On both samples, the Factorization Machines RS with context performs better than the FM RS without context, except for the test data in the second sample where the contextual variables performed worse than FM without context. Based on the results, I can now answer the research question:

To what extent can text features help understanding customers' opinions and improve a User-Based Recommender System using Amazon review data?

The emotional states, polarity scores and LDA topics proved to be extremely helpful in understanding the customer's opinion towards the product and its features. In addition to numerical ratings, exploiting the review text turned out to be more informative than the ratings since a customer's reason for a certain rating became clear. Table 5 and Table 6 show that the UBCF Recommender Systems is improved by applying Factorization Machines and by adding the textual features. However, the improvement is limited to the point where the textual features become too contradicting: this will confuse the Recommender System and produce inaccurate predictions.

8. Limitations

Conducting a perfect research is nearly impossible: this is also true for this thesis since there are a number of limitations that need to be pointed out. First, computational issues have been a major problem. Building a Recommender System is computationally and code intensive. To exploit its full potential, a strong computer is necessary. I encountered different methods, techniques and evaluation metrics which could have been a better solution but were not feasible given various computational constraints. The precision at *k* metric is a relevant addition to the prediction accuracy calculation, but was not possible to execute in this research. Secondly, although data from Amazon is very popular, it does not fully represent real-world data. Amazon modifies its research data because of business and privacy reasons, which is understandable. However, a Recommender System is best evaluated when modeled with real data and tested directly on the website. This is also another limitation to this study: The RSs are not evaluated using A/B testing or direct implementation. Statistical performance is a guideline in this case, but the actual performance will be revealed though customer interaction on the Amazon website. By recommending the right products to the right customers as predicted by an RS, the real performance can be measured using commercial metrics such as incremental sales or page views and clicks.

Next, the contextual features in this research proved to be helpful in getting a higher prediction accuracy, but the results could still be improved. Some contextual variables returned contradicting values which confuses the Recommender System. For example, a high rating is accompanied by a negative polarity and the emotional state *Anger*. Observations like this could explain the relatively big difference between the training and test data prediction accuracy percentages. As mentioned earlier, text data is uncontrolled and comes with a lot of issues like sarcasm and unknown online language. Overcoming these problems is still a major challenge and remains an important field to study further.

Future research may try to incorporate different types of contextual variables like time, location and other textual features such as bigrams and trigrams. This will help determining what contextual features are most valuable in predicting the ratings and thus improve the Recommender System.

Literature overview

Adomavicius, G., & Tuzhilin, A. (2011) Context-aware recommender systems. In *Recommender systems handbook* (pp. 217-252). Springer, Boston, MA.

Aggarwal, C. C. (2016). Recommender systems (Vol. 1). Cham: Springer International Publishing.

Bagheri, A., Saraee, M., & De Jong, F. (2014). ADM-LDA: An aspect detection model based on topic modelling using the structure of review sentences. Journal of Information Science, 40(5), 621-636.

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

Brightlocal (2019). Local Customer Service Survey. Retrieved from **www.brightlocal.com/research/local-consumer-review-survey/**

Castillo, A., Benitez, J., Llorens-Montes, F. J., & Braojos, J. (2017). Introducing Analytics Talent in the Equation on Business Value of Social Media Capability: An Empirical Investigation.

Chandrasekar, P., & Qian, K. (2016, June). The impact of data preprocessing on the performance of a naive bayes classifier. In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) (Vol. 2, pp. 618-619). IEEE.

Chen, Y., Bordes, J. B., & Filliat, D. (2016, September). An experimental comparison between NMF and LDA for active cross-situational object-word learning. In 2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob) (pp. 217-222). IEEE.

Chevalier, J. A., & Mayzlin, D. (2006). The effect of word of mouth on sales: Online book reviews. Journal of marketing research, 43(3), 345-354.

D'Addio, R. M., Domingues, M. A., & Manzato, M. G. (2017). Exploiting feature extraction techniques on users' reviews for movies recommendation. Journal of the Brazilian Computer Society, 23(1), 7

D'Addio, R. M., & Manzato, M. G. (2014, October). A collaborative filtering approach based on user's reviews. In 2014 Brazilian Conference on Intelligent Systems (pp. 204-209). IEEE.

Di Noia, T., Mirizzi, R., Ostuni, V. C., Romito, D., & Zanker, M. (2012, September). Linked open data to support content-based recommender systems. In Proceedings of the 8th international conference on semantic systems (pp. 1-8).

Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2011). Collaborative filtering recommender systems. Now Publishers Inc.

Gokulakrishnan, B., Priyanthan, P., Ragavan, T., Prasath, N., & Perera, A. (2012, December). Opinion mining and sentiment analysis on a twitter data stream. In International Conference on Advances in ICT for Emerging Regions (ICTer2012) (pp. 182-188). IEEE.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. Psychological review, 114(2), 211.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.

Koohi, H., & Kiani, K. (2017). A new method to find neighbor users that improves the performance of collaborative filtering. Expert Systems with Applications, 83, 30-39.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

Krishna, P. V., Misra, S., Joshi, D., & Obaidat, M. S. (2013, May). Learning automata-based sentiment analysis for recommender system on cloud. In 2013 International Conference on Computer, Information and Telecommunication Systems (CITS) (pp. 1-5). IEEE.

Kumar, K. S., Desai, J., & Majumdar, J. (2016, December). Opinion mining and sentiment analysis on online customer review. In 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) (pp. 1-4). IEEE.

Kwartler, T. (2017). Text mining in practice with R. John Wiley & Sons.

Liu, B. (2010). Sentiment analysis and subjectivity. Handbook of natural language processing, 2(2010), 627-666.

Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer, Boston, MA.

Mahadevan, A., & Arock, M. (2017, December). Credible user-review incorporated collaborative filtering for video recommendation system. In 2017 International Conference on Intelligent Sustainable Systems (ICISS) (pp. 375-379). IEEE.

Mahara, T. (2016). A new similarity measure based on mean measure of divergence for collaborative filtering in sparse environment. Procedia Computer Science, 89, 450-456.

Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: C. Cattuto, G. Ruffo, F. Menczer (eds.)Hypertext, pp. 73–82. ACM (2009)Google Scholar

Maynard, D. G., & Greenwood, M. A. (2014, March). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In LREC 2014 Proceedings. ELRA.

Newman, D., Chemudugunta, C., & Smyth, P. (2006, August). Statistical entity-topic models. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 680-686).

Onan, A., Korukoglu, S., & Bulut, H. (2016). LDA-based Topic Modelling in Text Sentiment Classification: An Empirical Analysis. Int. J. Comput. Linguistics Appl., 7(1), 101-119.

Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2020). Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering*.

Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In Recommender systems handbook (pp. 1-35). Springer, Boston, MA.

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: introduction and challenges. In Recommender systems handbook (pp. 1-34). Springer, Boston, MA.

Singhal, A. (2001). Modern information retrieval: A brief overview. In IEEE Data Engineering Bulletin

Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012, July). Exploring topic coherence over many models and many topics. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (pp. 952-961). Association for Computational Linguistics.

Tatemura, J. (2000, January). Virtual reviewers for collaborative exploration of movie reviews. In Proceedings of the 5th international conference on Intelligent user interfaces (pp. 272-275).

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008, October). Matrix factorization and neighbor based algorithms for the netflix prize problem. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 267-274).

Wang, C., & Blei, D. M. (2011, August). Collaborative topic modeling for recommending scientific articles. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 448-456).

Xu, W., Liu, X., & Gong, Y. (2003, July). Document clustering based on non-negative matrix factorization. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 267-273).

Zhao, Z. D., & Shang, M. S. (2010, January). User-based collaborative-filtering recommendation algorithms on hadoop. In 2010 third international conference on knowledge discovery and data mining (pp. 478-481). IEEE.

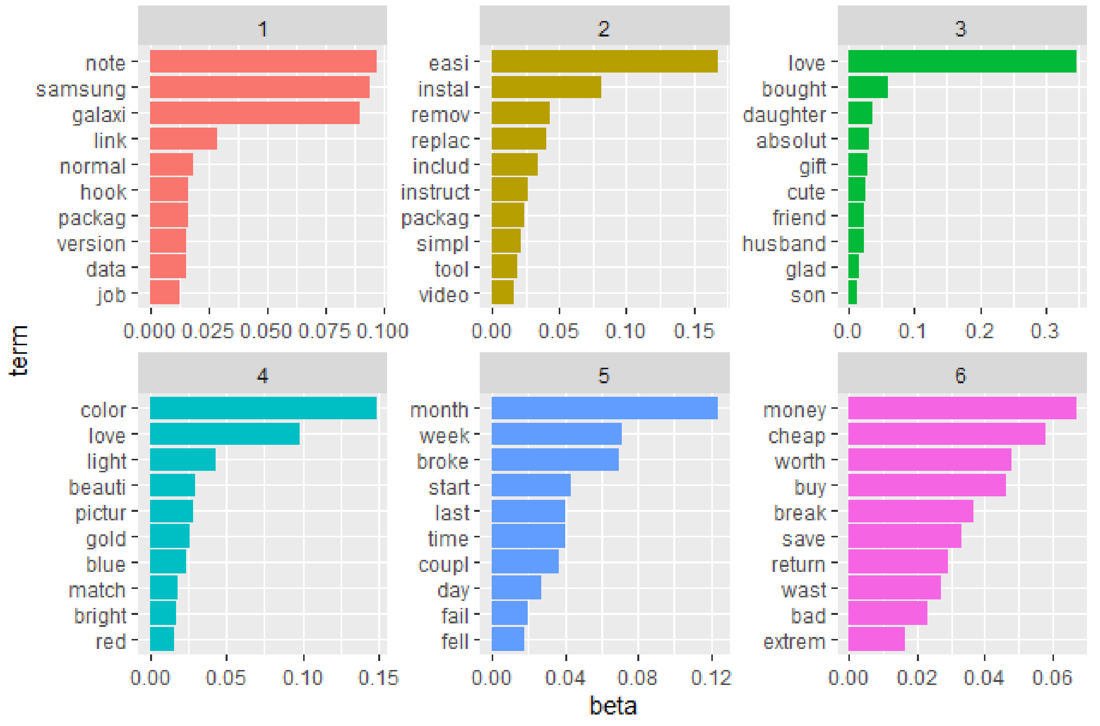
Appendix

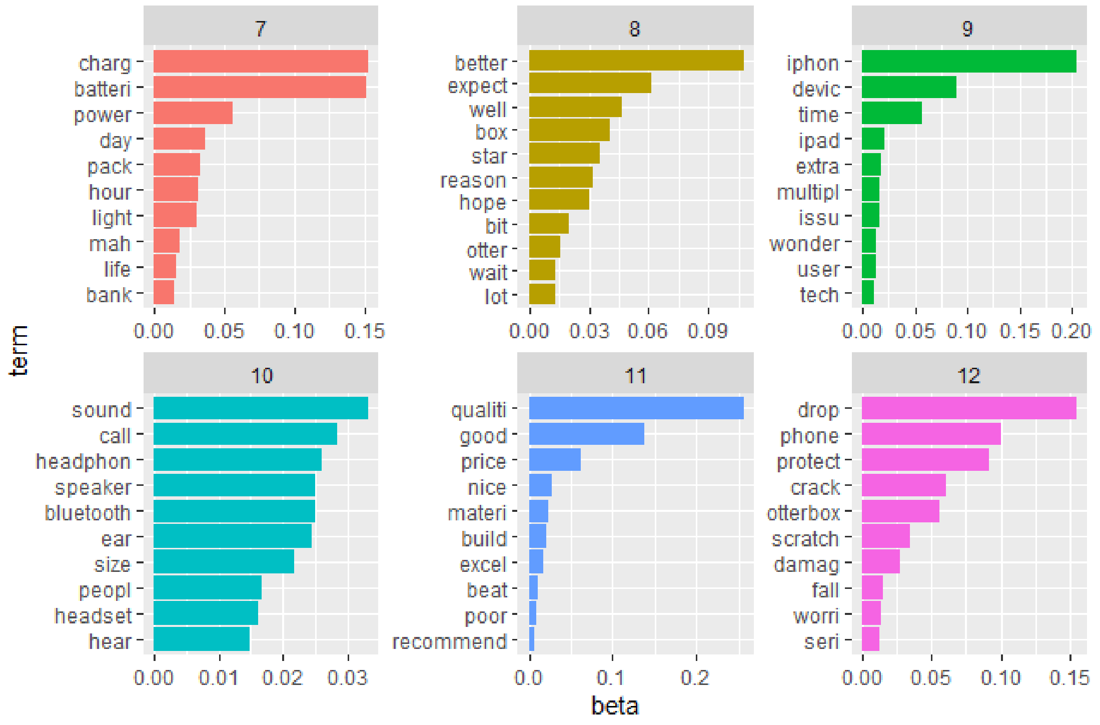
A – Removed stopwords from the 'stopwords'-list in R

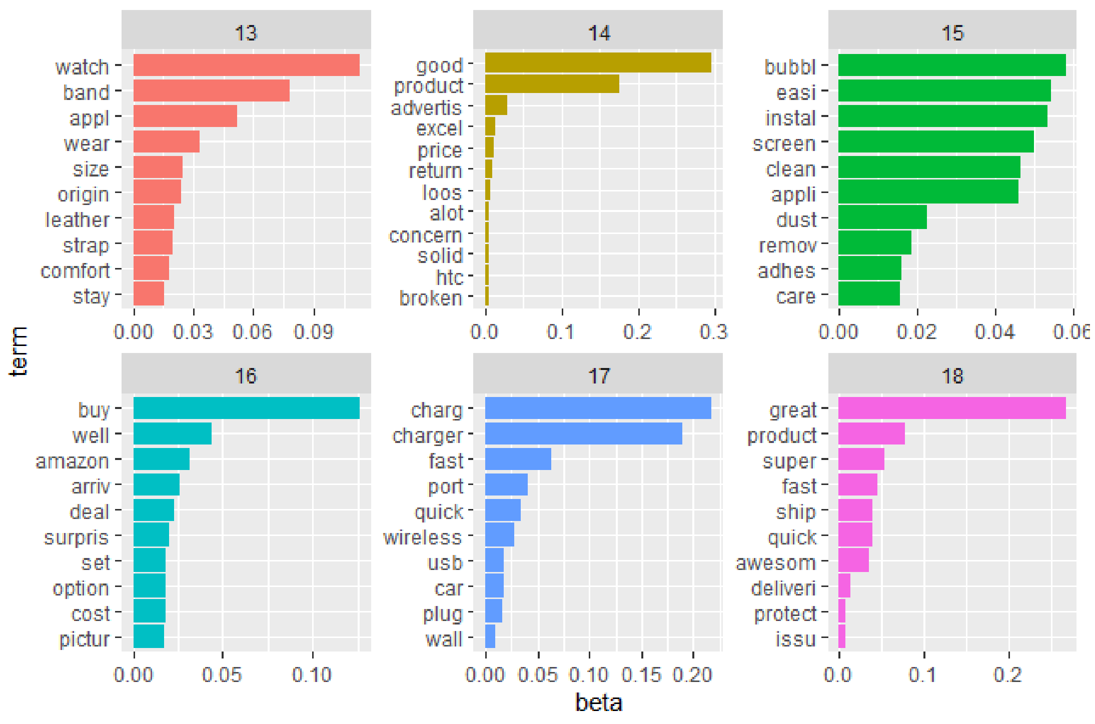
These words are removed because they all relate (in)directly to a specific opinion or emotion.

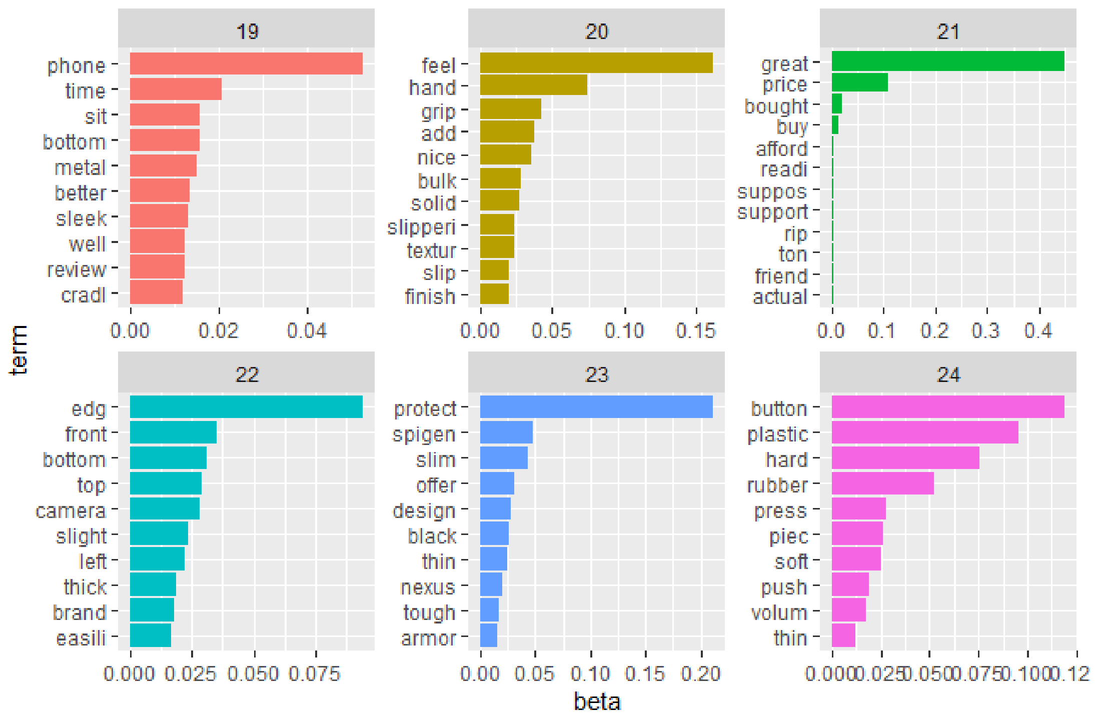
WORD	LEXICON
Appreciate	SMART
Awfully	SMART
Best	SMART
Better	SMART
Least	SMART
Likely	SMART
Unfortunately	SMART
Unlikely	SMART
Well	SMART
Best	onix
Better	onix
Good	onix
Great	onix
Greater	onix
Greatest	onix
Important	onix
Interested	onix
Interesting	onix
Least	onix
Likely	onix
Problem	onix
Well	onix

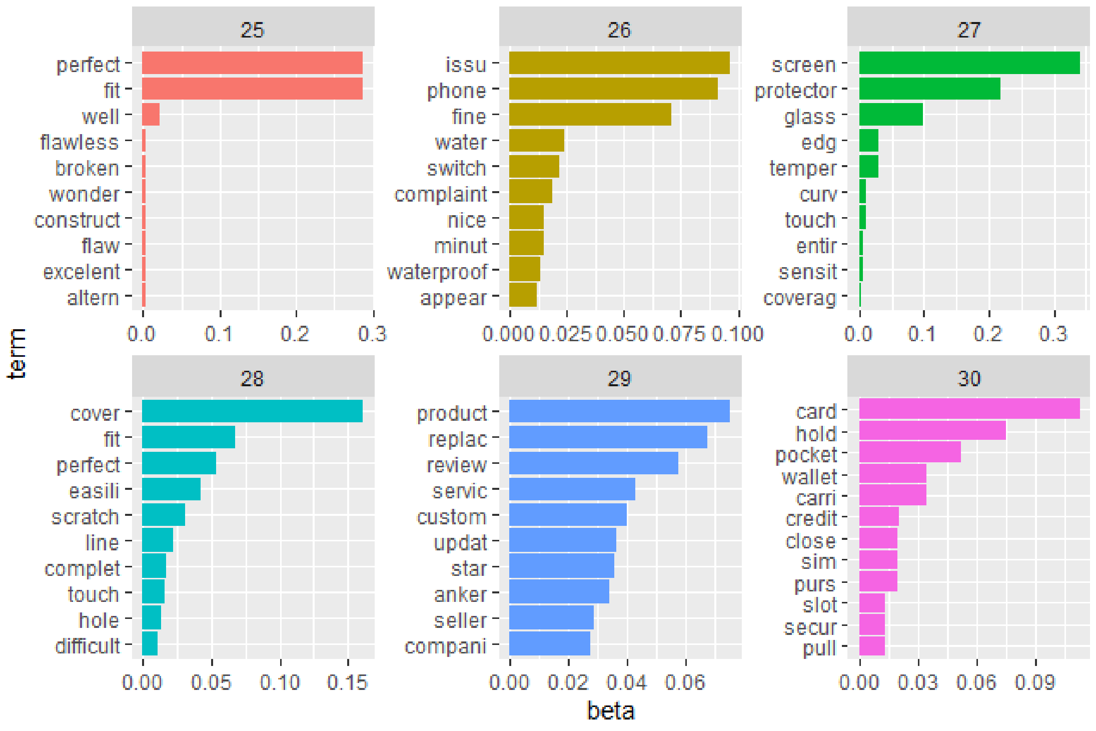
B – LDA topics 1 to 45

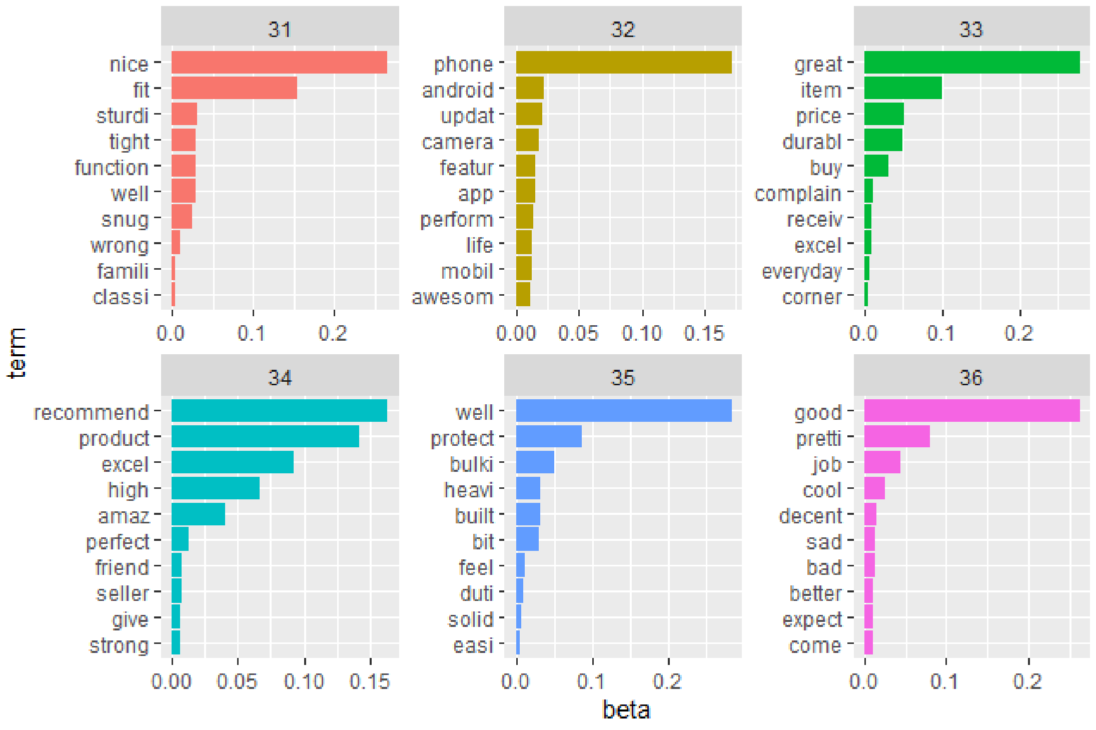


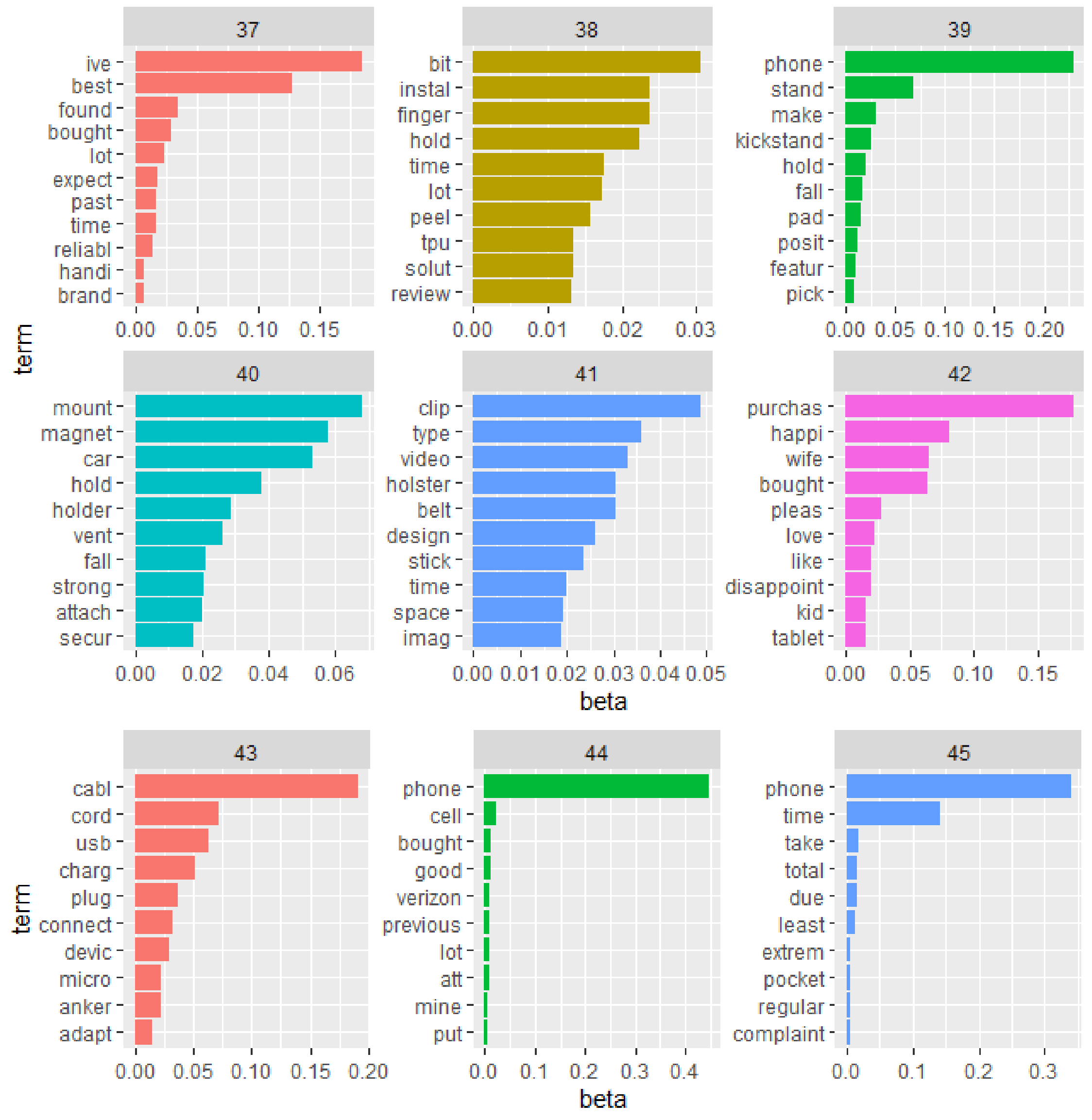



13
watch
band
appl
wear
size
origin
leather
strap
comfort
stay
0.00 0.03 0.06 0.09
14
good
product
advertis
excel
price
return
loos
alot
concern
solid
htc
broken
0.0 0.1 0.2 0.3
15
bubbl
easi
instal
screen
clean
appli
dust
remov
adhes
care
0.00 0.02 0.04 0.06
term
16
buy
well
amazon
arriv
deal
surpris
set
option
cost
pictur
0.00 0.05 0.10
17
charg
charger
fast
port
quick
wireless
usb
car
plug
wall
0.00 0.05 0.10 0.15 0.20
18
great
product
super
fast
ship
quick
awesom
deliveri
protect
issu
0.0 0.1 0.2
beta


19
phone
time
sit
bottom
metal
better
sleek
well
review
cradl
0.00 0.02 0.04
20
feel
hand
grip
add
nice
bulk
solid
slipperi
textur
slip
finish
0.00 0.05 0.10 0.15
21
great
price
bought
buy
afford
readi
suppos
support
rip
ton
friend
actual
0.0 0.1 0.2 0.3 0.4
term
22
edg
front
bottom
top
camera
slight
left
thick
brand
easili
0.000 0.025 0.050 0.075
23
protect
spigen
slim
offer
design
black
thin
nexus
tough
armor
0.00 0.05 0.10 0.15 0.20
24
button
plastic
hard
rubber
press
piec
soft
push
volum
thin
0.000 0.025 0.050 0.075 0.100 0.12
beta


25
perfect
fit
well
flawless
broken
wonder
construct
flaw
excelent
altern
26
issu
phone
fine
water
switch
complaint
nice
minut
waterproof
appear
27
screen
protector
glass
edg
temper
curv
touch
entir
sensit
coverag
28
cover
fit
perfect
easili
scratch
line
complet
touch
hole
difficult
29
product
replac
review
servic
custom
updat
star
anker
seller
compani
30
card
hold
pocket
wallet
carri
credit
close
sim
purs
slot
secur
pull
term
beta


31
nice
fit
sturdi
tight
function
well
snug
wrong
famili
classi
32
phone
android
updat
camera
featur
app
perform
life
mobil
awesom
33
great
item
price
durabl
buy
complain
receiv
excel
everyday
corner
34
recommend
product
excel
high
amaz
perfect
friend
seller
give
strong
35
well
protect
bulki
heavi
built
bit
feel
duti
solid
easi
36
good
pretti
job
cool
decent
sad
bad
better
expect
come
term
beta


37
ive
best
found
bought
lot
expect
past
time
reliabl
handi
brand
38
bit
instal
finger
hold
time
lot
peel
tpu
solut
review
39
phone
stand
make
kickstand
hold
fall
pad
posit
featur
pick
40
mount
magnet
car
hold
holder
vent
fall
strong
attach
secur
41
clip
type
video
holster
belt
design
stick
time
space
imag
42
purchas
happi
wife
bought
pleas
love
like
disappoint
kid
tablet
43
cabl
cord
usb
charg
plug
connect
devic
micro
anker
adapt
44
phone
cell
bought
good
verizon
previous
lot
att
mine
put
45
phone
time
take
total
due
least
extrem
pocket
regular
complaint
term
beta