A Feature-Oriented Sentiment Rating for Mobile App Reviews

Washington Luiz
Federal University of São João del Rei - Brazi
wluiz@ufsj.edu.br

Felipe Viegas
Federal University of Minas Gerais - Brazil
frviegas@dcc.ufmg.br

Rafael Alencar
Federal University of São João del Rei - Brazil
rafael.alencar@ufsj.edu.br

Fernando Mourão
Seek AI Labs - Brazil
fernando.mourao@catho.com

Thiago Salles
Federal University of Minas Gerais - Brazil
tsalles@dcc.ufmg.br

Darlinton Carvalho
Federal University of São João del Rei - Brazil
darlinton@ufsj.edu.br

Marcos André Gonçalves
Federal University of Minas Gerais - Brazil
mgoncalv@dcc.ufmg.br

Leonardo Rocha
Federal University of São João del Rei - Brazil
lcrocha@ufsj.edu.br

ABSTRACT

In this paper, we propose a general framework that allows developers to filter, summarize and analyze user reviews written about applications on App Stores. Our framework extracts automatically relevant features from reviews of apps (e.g., information about functionalities, bugs, requirements, etc) and analyzes the sentiment associated with each of them. Our framework has three main building blocks, namely, (i) topic modeling, (ii) sentiment analysis and (iii) summarization interface. The topic modeling block aims at finding semantic topics from textual comments, extracting the target features based on the most relevant words of each discovered topic. The sentiment analysis block detects the sentiment associated with each discovered feature. The summarization interface provides to developers an intuitive visualization of the features (i.e., topics) and their associated sentiment, providing richer information than a 'star rating' strategy. Our evaluation shows that the topic modeling block is able to organize information provided by users in subcategories that facilitate the understanding of which features more positively/negatively impact the overall evaluation of the application. Regarding user satisfaction, we can observe that, in spite of the star rating being a good measure of evaluation, the Sentiment Analysis technique is more accurate in capturing the sentiment transmitted by the user by means of a comment.

CCS CONCEPTS

• **Information systems** → *Document collection models*; • **User/Machine Systems** → *Human information processing*;

KEYWORDS

Topic Model;Sentiment Analysis;Analysis of online reviews

ACM Reference Format:
Washington Luiz, Felipe Viegas, Rafael Alencar, Fernando Mourão, Thiago Salles, Darlinton Carvalho, Marcos André Gonçalves, and Leonardo Rocha. 2018. A Feature-Oriented Sentiment Rating for Mobile App Reviews. In *Proceedings of The Web Conference 2018 (WWW 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186168

WWW 2018, April 23–27, 2018, Lyons, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186168

1 INTRODUCTION

The Internet has experienced a notable expansion and popularization in recent years. New components and services are being incorporated at a very fast pace, increasing the number of new users making use of these services. Every day new applications are created that generate and use a higher amount of data of the most diverse types. This large amount of data available on the Web has generated in recent years a differentiated, challenging and intriguing scenario for various applications: there is more data that can be effectively analyzed. As stated in [2], "too much information is as bad as none". Thus, organizing and finding the appropriate information resources to meet the needs of users has become a great challenge.

There are several scenarios where new techniques are being proposed in order to effectively extract relevant information. One example that illustrates these scenarios has to do with e-commerce applications such as Amazon and Netflix. In such case, given a large collection of distinct items and products, several strategies were proposed (the so-called Recommendation Systems) to increase end-users' satisfaction, by automatically selecting those items potentially useful to them [6]. Another challenging application is related to the increasingly popular online social networks, where techniques such as sentiment analysis and graph mining are fundamental for identifying interesting and non-trivial patterns that could be explored to enhance end-users' experience, brand analysis, and others [31]. Finally, a third and rather recent application scenario has to do with the so-called *App Stores*, such as Google Store and Apple Store, that provide more than 2 million applications (each) so that

users can search, buy, share and develop applications for mobile devices. As one could expect, a lot of useful data is produced for both platforms, in the form of textual comments and star ratings. Recent studies have shown that such information can be effectively explored to leverage application development, with a clearer and more user-centered definition of requirements and improvements [16], which corresponds to the scenario of focus of the present work.

Despite the app store star ratings strategy being an important source of information for developers to have an evaluation of their applications, a detailed analysis of the specific app features (e.g., functionalities, bugs, requirements) can only be obtained using the content of the reviews made by the users themselves. Recent studies report that revisions made about applications may indeed contain i relevant information to developers [29]. However, a manual review of all these available revisions is impracticable for several reasons: (1) the large volume of generated data (e.g., a recent study shows that iOS users submit 22 revisions per day for each application on average [28]); (2) the quality of reviews, ranging from innovation tips and ideas to insults; and (3) reviews containing mixed opinions about different characteristics, making it difficult to differentiate positive or negative opinions about each of the mentioned characteristics. Clearly, new proposals for methods that are able to automatically extract this information are more than desirable.

In this paper, we propose a general framework that allows the developer to filter, summarize and analyze reviews written by users about applications in distribution platforms such as *App Stores*. More specifically, our framework is able to automatically extract relevant features from reviews and analyze the sentiment associated with each one of them. The ultimate goal of this proposal is to allow developers to improve their developed applications in a user-centric fashion, based on an automatic analysis of all reviews made by its users. This general framework has three main building blocks, namely, (i) a topic modeling strategy, (ii) a sentiment analysis and (iii) a summarization interface. Briefly speaking, the topic modeling strategy aims at finding semantic topics from textual comments. It has to do with the ability to represent the set of comments as a bag-of-words matrix and decompose the properly represented dataset matrix (a.k.a., design matrix) into matrices that capture the latent relationships between terms and comments. It is also responsible for extracting the target features by the most relevant words of each discovered topic. Several strategies do exist for this matter, such as non-negative matrix factorization, LDA, and its probabilistic counterpart, to name a few. The sentiment analysis strategy has to do with detecting the sentiment (i.e., positive or negative) associated with each discovered feature. Again, several strategies do exist for this task, such as SACI and VADER. Finally, the summarization interface provides to the developer an intuitive visualization of the features (i.e., topics) and their associated sentiment, thus giving richer information than what can be obtained with star ratings. Thus, the desiderata for our proposed framework is to offer, for each of its main building blocks, methods that are able to perform their tasks without losing any useful information to the developers, allowing fully automatic operation.

In order to evaluate our proposal, we instantiate the proposed framework and provide an extensive analysis considering seven collections available in [16], composed by user reviews and its star ratings extracted from Google Play Store. Our experimental evaluation consists in instantiating gradually, each of the blocks that compose our framework, in order to highlight the associated advantages (in terms of obtained information). Our results demonstrate that star ratings, though coarse-grained, are an important strategy to be considered by developers in application evaluation process. As a second step, we instantiate the first block of our framework adopting the topic modeling strategy NMF [23] to infer different topics for each application. Then, we group the comments of each topic, identifying the main words related to them. Finally, we calculate the average star rating associated with each group of comments. With this strategy, we mitigate the main issue of star ratings namely, their inability to characterize opinions about different fine-grained app features provided by the users. We instantiate the second building block of our framework by adopting the SACI [34] sentiment analysis strategy in order to identify the sentiment associated with each topic and to compare it with the average star rating calculated in the previous step. We demonstrate that the ratings identified by the sentiment associated with topics are more trustworthy to the users' comments than star ratings. Finally, we present a visual interface proposal that summarizes all the gathered information, highlighting the impact of each identified feature on the global application evaluation.

In sum, the main contributions of this paper are: (1) A generic framework that allows the automation of the analysis process of reviews, identifying the main topics (i.e., functionalities, requirements, bugs, etc.) related to each application, as well as their related user satisfaction in terms of users' comment polarity (i.e., positive or negative sentiment strength); (2) A visual summarization interface containing application name and rating information, the extracted features and their sentiment strength. We also present information about how each feature is impacting the application rating. This information can also be re-arranged by the relevance of a feature or by the sentiment strength (positive or negative scores). The visual interface aims to facilitate the understanding of all content that the framework has processed in a given domain; and (3) An extensive experimental evaluation instantiating each building block of our framework and highlighting the benefits associated with each one of them regarding the concise information provided to developers.

2 BACKGROUND AND RELATED WORK

In this section, we review relevant works on topic models and sentiment analysis. We discuss some work on this matter, as well as some recent ones, presenting their strengths, weaknesses and how we can adapt all of them to our proposal. The interest of those strategies stems from the consolidation that revisions made by users about applications may contain information relevant to developers [29].

2.1 Topic Modeling

Topic modeling addresses the problem of discovering relationships between documents (D) and topics (Z), as well as, relationships between the terms (W) that compose the documents and topics, thus enabling the textual documents to be organized and analyzed according to the discovered topics [19]. The topic modeling process can be divided into three main steps: data representation, latent topic decomposition, and topic extraction.

In data representation, textual documents are usually represented by adopting a fixed-length vector representation based on simple

term occurrence information, encoded by the so-called TF-IDF score [37] (and the variants thereof). The TF-IDF score of a term t captures the frequency of t in a document d as well as the rarity of t within the entire document collection. Another important aspect in data representation is the steps of text preprocessing, allowing great improvement in the following steps. The traditional step of preprocessing in Information Retrieval includes the removal of special character, numbers, plural and stopwords.

The latent topic decomposition methods can be divided into two main models: the probabilistic models and the non-probabilistic models. In the probabilistic models, the data is modeled based on two concepts: each document follows a topic distribution $\theta^{(d)}$; and each topic follows a term distribution $\phi^{(z)}$. Let $\theta^{(d)}=P_d(z)$ be the topic distribution over topics z observed in a document d. Also, let $\phi^{(z)}=P(w|z)$ be the probability distribution over terms w considering documents belonging to the abstract topic z. $\theta^{(d)}$ and $\phi^{(z)}$ reflect to what extent a term is important to a topic and to what extent a topic is important to a document, respectively. Both concepts are key for the so-called probabilistic latent semantic indexing (pLSI) [19] algorithm, that takes into account the co-occurrence probability of terms and documents $P(d,w)$ as a combination of multinomial distributions conditionally independent. More specifically, let $P(z|d)$ be the conditional probability of observing a topic z given a document d, based on $\theta^{(d)}$, and $P(w|z)$ be the conditional probability of observing a term w given a topic z. Thus, $P(d,w)=P(d)P(w|d)$ and, marginalizing, $P(w|d)=\sum_z P(w|z)P(z|d)$. Note that pLSI does not take into account how $P(w|z)$ (ϕ) is generated (thus, ultimately, also $P(z)$), being harder to generalize to real-world scenarios. On the other hand, in [5] the authors propose the so-called latent Dirichlet allocation (LDA), which generalized pLSI in terms of how $P(w|z)$ is estimated: its assumes a Dirichlet distribution—a continuous multivariate distribution that reflects the fact that documents usually contribute to just a few topics and topics usually contribute to a small set of words. Despite being one of the most used strategies for latent topic decomposition, LDA has its own challenges, such as data sparsity and generation of incoherent topics. To address the limitations of LDA, a lot variants, such as, BTM[9], LTM [8] and FS [16], were proposed to refine the information for the LDA method.

The non-probabilistic topic modeling comprises strategies such as matrix factorization. In this case, a dataset with n documents and m different terms is encoded as a design matrix $A\in\mathbb{R}^{n\times m}$ and the goal is to decompose A into sub-matrices that preserve some desired property or constraint. A well-known matrix factorization procedure that is amenable to topic modeling is the singular value decomposition (SVD) [15]. In this case, the design matrix A is factorized into three matrices, namely, $U_{n,k}$, $\Sigma_{k,k}$ and $V_{m,k}$, such that $A=U\Sigma V^t$. The matrix U captures the relationship between terms and topics, where v captures the relationship between documents and topics. And finally, the matrix σ captures the relationship between topics. In these produced sub-matrices, similar terms and documents are usually are located nearby regions, according to some distance metric. Finally, another widely used strategy for matrix decomposition that is also applicable to topic modeling is the non-negative matrix factorization (NMF) [23]. Under this strategy, the design matrix A is decomposed into two sub-matrices $H\in\mathbb{R}^{n\times k}$ and $W\in\mathbb{R}^{k\times m}$, such that $A\approx H\times W$. In this notation, k denotes the number of latent factors (i.e., topics), H encodes the relationship between documents and topics, and W encodes the relationship between terms and topics. The restriction enforced by NMF is that all three matrices do not have any negative element. Notice that, unlike NMF, SVD factors contains both positive and negative entries. When dealing with properly represented textual data, the design matrix usually contains non-negative term scores, such as TF-IDF, with well-defined semantics (e.g., term frequency and rarity). It is natural to expect the extracted factors to be non-negative so that such semantics can be somehow preserved. We thus consider NMF as our matrix factorization strategy of choice. As a final note, as with the probabilistic strategies, the non-probabilistic ones can also generate incoherent topics, which is not desirable.

To address the problem of extracting relevant/coherent topics from previously identified topics many strategies have been proposed like in [10] where the authors used an interactive non-negative matrix factorization to allow the human manipulate (i.e filter, merge, create, exclude) topics. In [12] the authors propose the use of LSI followed by a clustering algorithm in order to identify relevant semantic topics to automate the process. Recently, in [4] a new technique, based on NMF, is proposed: the semantic topic combination (SToC). The main goal is to group semantically similar topics. Briefly speaking, after the NMF factorization of the design matrix A, a weighted tri-partite graph is built on top of matrices H and W (obtained from NMF factorization, $A\approx H\times W$), where the weights come from these sub-matrices. Such graph captures, by definition, the relationship between documents and terms, as well as of terms and topics. A topic transition graph is then built through random walk on this tri-partite graph. Such random walk procedure is able to uncover indirect relationships between topics, ultimately grouping similar ones together, by means of the Bhattacharyya [3] distance metric. At each interaction, the two topics with the smallest distance are merged. This process continues until all topics are finally merged.

In [10], the authors proposed a framework called Modeling topics with user behavior based on non-negative matrix interactive (UTOPIAN). UTOPIAN is a semi-supervised strategy and allows users to interact with the generation of topics. To do this, the method uses the SS-NMF [7] (a supervised version of the NMF) to build the topics, and then, the framework provides an interface which allows the users to exclude or join topic terms and create their own topic. However, these interactions demands that the user to have knowledge about the respective domain of the data and it is costly since all the operations are manually.

2.2 Sentiment Analysis

Sentiment Analysis (SA) consists of the task of automatically detect the polarity (positive or negative) of sentiment embedded in a message or textual [11, 30, 32, 42]. Basically, there are two types of methods for sentence-level analysis: machine learning based and lexicon based. While one advantage of machine learning based methods is their ability to adapt, since these methods create trained models for specific purposes and context, their drawback is the requirement of labeled data in which is costly, even prohibitive, for some tasks.

Consequently, there have been a number of recent efforts that propose new sentence-level methods based on lexicons. Lexicon based methods create lexicons and combined different natural language processing techniques to determine a sentence polatiry [21]. There are three common strategies used to create a large lexicon. The first consists in determining the orientation of subjective terms. Esuli *et al.* proposes a method that explores the definitions of a given term in online dictionaries [14]. The second corresponds to the corpus-based approaches that use seed words and patterns in unlabeled corpora to induce domain-specific lexicons. An effective corpus-based approach is to construct lexical graphs using word co-occurrences and then to perform some form label propagation over these graphs [41]. Recent works have learned transformations of word embeddings in order to induce sentiment lexicons [36]. The SENTPRO [17] is a corpus-based based method which extends [41] by incorporating word embeddings to construct lexical graphs in order to perform label propagation. In addition, the strategy uses a bootstrap method to obtain confidence sentiment scores. This method was able to overcome the state-of-the-art methods of inducing sentiment lexicons.

Finally, the third way consists of expanding a given human curated lexicon (dictionary-based approaches). VADER [21] is a robust lexicon-based method for the English language since it implements several rules (grammatical and syntactical aspects) of the English language to infer the polarity of a text. In addition to the method, the authors in [21] made available vast human curated lexical dictionary in which the polarity of words ranges from −4 to +4. In recent efforts [13], the authors used the WordNet to expand a lexicon dictionary. WordNet [27] is a lexical repository of the English language commonly used in computational linguistics and natural language processing. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms, called *synsets*. Synonyms are words that denote the same concept and are interchangeable in various contexts. The set of synonyms (synsets) are interconnected through semantic concepts and lexical relations. WordNet has 117, 000 synsets connected to each other by means of a small number of "conceptual relationships". In addition, a synset contains a brief definition and, in most cases, one or more sentences that illustrate the use of the meanings of the word of the synonym set. The main idea in [13] is to start from a small number of words whose polarity is known and to use the relationships between words in the dictionary to infer the polarities of the remaining words. The words are related if they share at least a synset in the WordNet. Recently, we have observed lexicon based SACI [34], which, based on a lexicon built according to each context [38], has achieved results with superior quality to the several literature approaches, supervised an unsupervised. Thus, we adopted this strategy in the block of sentiment analysis.

To the best of our knowledge, the closest strategy that we found is the Fine-Grained Sentiment Analysis (FS) [16], in which a topic model strategy (LDA) is used to build sentimental topics. First, the strategy extracts words that co-occur often (a.k.a bi-grams) and then, it infers the sentiment strength of these bi-grams based on the sentiment score of the documents in which they occurred. In the paper, they used SentiStrength [40] to measure the sentiment-level of the documents. To generate the topics, the strategy uses the LDA method over these sentimental bi-grams. Different from this strategy where the techniques of topic modeling and sentiment analysis are dissociable, our proposal adequately delimits the role of each of these strategies to infer the relevance of the identified application characteristics.

3 PROPOSED STRATEGY

In this section, we present our proposed framework aimed at automatically extracting relevant features from reviews written by users about applications in distribution platforms such as *App Stores* and analyze the sentiment associated with each one of them. We decompose our framework into three main building blocks, namely, topic modeling, sentiment analysis, and summarization interface. There are many approaches that could be used to instantiate each block and next we describe the strategies that we consider in this paper.

3.1 Topic Modeling Content

The topic modeling block is based on the Non-negative matrix factorization (NMF) strategy. NMF is an unsupervised family of algorithms that simultaneously performs dimension reduction and clustering, with successful applications in a wide range of domains, including topic modeling. NMF [23] produces a "part-based" decomposition of latent relationships of a non-negative design matrix $A \in \mathbb{R}^{n \times m}$, where n is the number of examples (i.e., documents) and m the number of features (i.e., terms). The goal is to find a k-dimensional approximation of A ($k \ll m$) in terms of non-negative factors $H \in \mathbb{R}^{n \times k}$ and $W \in \mathbb{R}^{k \times m}$. As previously described, H encodes the relationship between documents and topics while W encodes the relationship between terms and topics. The key idea behind NMF is to approximate the column vectors of A by non-negative linear combinations of non-negative basis vectors (columns of H) and the coefficients given by the columns of W. However, defining k for NMF is not a simple task since it is widely believed that NMF is a non-convex problem with a unique solution and has no guarantee of finding the global minimum [24].

We apply NMF on a bag-of-words (**BOW**) data representation of the input reviews. Most specifically, we use the TF-IDF score, which empirically showed better results than a TF representation. Before building such matrix, we apply text preprocessing follows the traditional steps (i.e., special character, numbers, plural, URL's and stopwords). In addition, just for this step, we also removed words such as adverbs, using the VADER lexicon dictionary [21], as the vast majority of the words important for identifying topics are nouns and verbs.

After applying NMF on the input matrix, we use the SToC [4] method on the NMF outputs to redefine the number of semantic topics of a dataset, improving the cohesion of the topics built by the residual matrices of the NFM decomposition. First, SToC builds a weighted tripartite graph G_t as follows. G_t has three types of nodes, namely, D, L and F, representing documents, latent factors, and features, respectively. Each weighted edge represents the intensity of the relationship between two nodes. The weights are given by a probability distribution derived from the information of the residual matrices of the NMF matrix decomposition. The values of H represent the edges $d \leftrightarrow l$ while the values of W represent the edges $l \leftrightarrow f$, where $d \in D$, $l \in L$ and $f \in F$. The intuition behind this graph is that a document may refer to one or more topics, whereas a feature can be associated with more than one topic (latent factor). Then,

SToC is applied to convert the tripartite graph G_t into a new graph G, describing only the relationships among latent factors (nodes L). A key aspect to be observed is that each latent factor $l_j \in L$ in G_t can be indirectly reached by each latent factor $l_i \in L$, thought nodes $f \in F$ and $d \in D$. These indirect links can be converted into direct links according to

$$P(l_i \rightarrow l_j) = \sum_{k \in D^{l_i}} P(d_k|l_i) \times P(d_k|l_j) + \sum_{k \in F^{l_i}} P(f_k|l_i) \times P(f_k|l_j).$$

Each link of G summarizes this two-step path of G_t, where F^{l_i} and D^{l_i} are sets of features and documents with an input edge from l_i.

Next, the semantic sub-topics are merged in order to increase topic cohesion. The key idea here is to merge pairs of topics with a mutual probability of reaching each other in G higher than the minimum transition probability of G. To this end, we use the iterative algorithm described in [4], whose input is the graph G. Then, the pair of topics with the highest impact is selected to be merged. In this case, both topics are removed from G while the brand new topic is added to it. This process continues until the method reaches the number of τ resulting topics. The value of τ is an input parameter and, since the aforementioned strategy groups topics by pairs, we set the number k of NMF topics (i.e., latent factors) as $2 \times \tau$. Finally, for each latent topic, we use Information Gain [35] to rank terms according to their importance to the latent topics. The top R best-ranked terms from each latent topic are then selected as relevant terms.

3.2 Sentiment analysis Content

As previously mentioned, in order to instantiate the block of sentiment analysis, we adopt the SACI strategy [34]. *SACI* is composed of two components: the Sentiment Lexicon and the Sentiment Identification. The first component assigns a lexicon-semantic class for each term observed in an input collection. More specifically, we adopted a subset of six most representative polar morphological classes from [33]: Positive, Negative, Neutral, Inverter, Reductor and Amplifier. The premise is that each class plays a distinct role in the consolidation of subjective sentences [20, 25, 39]. Moreover, we assume that each term may be assigned to distinct classes according to the analyzed domain.

The second component of *SACI* performs the sentiment identification in document collections. The technique involves the construction of a transition graph among terms. We build the transition graph of terms based on directed paths extracted from the analyzed reviews as follows. First, we define as central nodes all terms, or n-grams, that represent the target topics of analysis (e.g., features identified in the first building block). Then, starting from each occurrence of each central node in a review, we extract its predecessor and successor terms belonging to the same sentence and define this ordered sequence of terms as a directed path. Thus, we define one distinct path for each occurrence of each target topic in the review collection. Finally, we transform the extracted set of paths into a graph by considering each distinct term as a node and defining one directed edge for each pair of terms adjacent to each other in at least one of the extracted paths.

Formally, let D be the set of analyzed reviews and M the set of central nodes. Also, we define a parameter l (maximum radius) that stands for the applicability radius of each sentiment transformation within each sentence. First, we extract from each review $d_{i'} \in D$ a set $F_{i'}$ of sentences. For each occurrence of a central node $m_{j'} \in M$ in a sentence $f_{k'} \in F_{i'}$, we extract a path $c_{k'}$ that is composed of the l predecessor terms of $m_{j'}$, the central node $m_{j'}$ itself, and the l successor terms of $m_{j'}$, preserving the order of occurrence of the terms[1]. Through this process, we define a set C of all paths extracted from all sentences in the whole review collection. Then, taking into account the set T of distinct terms observed in D, we define a graph G in which each term $t_i \in T$ corresponds to a node, and for each pair of terms $t_i, t_j \in T$ there is a directed edge from t_i to t_j if t_i precedes t_j in at least one path $c_{k'} \in C$. The edge weight is defined as the probability of t_i precede t_j in the entire set C. We also defined the *Entry* and *Exit* states that represent the input and output nodes of our graph, respectively. If a term t_i starts a path, there will be an edge from node *Entry* pointing to it, as well as whenever a term t_i ends a path, it will be connected to node *Exit*. Thus, each path may represent more than one sentence of $F_{i'}$. Further, each node t_i of G has an attribute that determines its sentiment class, such as defined by the input lexicon which is a *SACI*'s parameter.

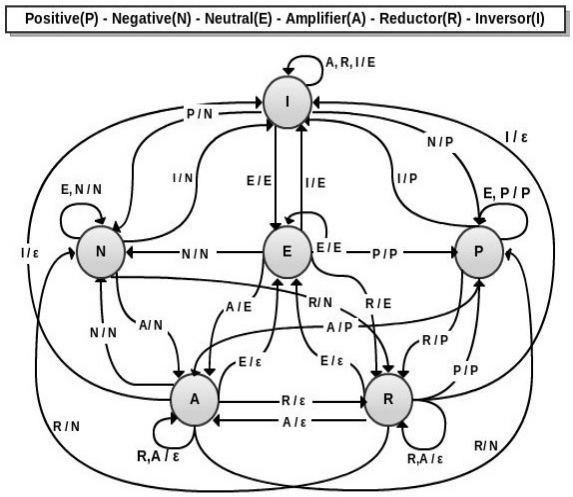


Figure 1: Transformation Rules (T_S and T_O) through the *Mealy* Machine.

Given the transition graph, first, *SACI* determines the sentiment associated with each path via a state-transition process wherein the next state depends on the current state and the input data. This process resembles systems modeled by Mealy machines [26]. We define $F=\{P, N, E, I, R, A\}$, which corresponds to the lexicon-semantic classes, $F_0=\{E\}$, since the initial sentiment of each path is set to neutral, $\Sigma=\{p, n, e, i, r, a\}$ is the lexicon-semantic class of the next term on the path in the transition graph, $\Lambda=\{p, n, e, \varepsilon\}$ is the path sentiment, where the transformation rules, T_S and T_O, are depicted in Figure 1. Each combination of current state and input results in an output symbol that represents the new path's sentiment and an output state. The path is sequentially traversed and whenever a term is reached, the path's sentiment is updated according to the rules. This process continues until the last term of the path is reached. In this case, the last output symbol different from ε corresponds to the sentiment assigned to each path $c_{k'}$ ($sentiment[c_{k'}]$).

[1]Terms shorter than three characters are ignored at this step, since they usually comprise prepositions and articles that do not help to infer the semantics of a review.

Besides a sentiment, each path $c_{k'}=\{Entry, t_1, t_2, \ldots, t_k, Exit\}$, from the path set C being analyzed, defines a probability $prob[c_{k'}]$, as shown by Equation 1, where $prob[t_j|t_i]$ corresponds to the probability of reaching t_j starting from t_i and $prob[C]$ denotes the sum of the probabilities of all paths from C.

$$\begin{aligned} prob[c_{k'}] &= \frac{1}{prob[C]} * [\log(prob[t_1|Entry] * prob[t_2|t_1] * \ldots \\ &\quad \ldots * prob[t_k|t_{k-1}] * prob[Exit|t_k])] \\ &= \frac{1}{prob[C]} * [\log(prob[t_1|Entry]) + \log(prob[t_2|t_1]) \\ &\quad \ldots + \log(prob[t_k|t_{k-1}] + \log(prob[Exit|t_k]))] \end{aligned} \quad (1)$$

Given the sentiment related to each distinct path, as well as the probability associated with it, we define the probability of occurrence related to each collective sentiment $class \in \{neutral, positive, negative\}$ in D as $prob[class] = \sum_{\{c_i \in C | sentiment[c_i] = class\}} prob[c_i]$. Thus, unlikely paths and sub-paths in the graph are less relevant for the analysis process, whereas paths with high probabilities (i.e., frequent opinions) are prioritized.

Note that, different from traditional algorithms that require a prior knowledge of each individual opinion to determine the collective one in a review collection, SACI assumes that collective analysis could be better performed when exploiting overlaps among distinct reviews of the collection. Moreover, we highlight that it is common that a single review evaluates more than one feature. The sentence level analysis performed by SACI allows that all mentioned features in the same review be analyzed separately.

3.3 Summarization Interface

Several studies [1, 18, 22] were performed to support the hypothesis that reviews contain a rich source of information about user experience, and more importantly, this information can be used as a measure of quality. In [35], the authors conducted experiments to verify the relationship between the number of stars and the content of review comments of applications from Google Play. The results showed that the information in reviews does not properly represent the star rating. In addition, the authors proposed a new method of evaluating applications by extracting information related to sentiment from the reviews. They applied Machine Learning strategies to show that the proposed sentiment-based approach is more effective at predicting the reviews than star rating.

Inspired by those works, we propose a framework capable of extracting useful information about which are the relevant topics (Section 3.1) and their respective sentiment strength (Section 3.2) given an application. Figure 2 illustrates the output of our framework for an application, showing the overall sentiment strength (as stars rating) followed by the actual score. Then, there are the topics extracted by the topic model method (first block) and their respective sentiment score (second building block). In our approach, we normalized the sentiment strength in values from 1 to 5 to maintain a correspondence with the star rating. Consider an application whose the sentiment inferred in their reviews was 80% positive and 20% negative. Normalizing 80%, the sentiment rating achieved is 4 stars. The arrows indicate whether the topics are below average (red) or above average (green) when compared to the sentiment strength considering all reviews assigned that application. These arrows allow developers to detect easily which topics are negatively impacting the overall rating.

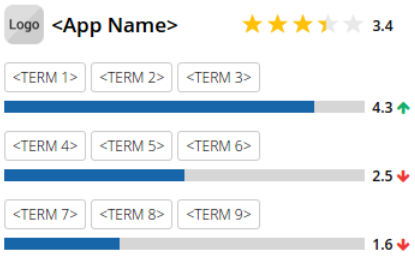


Figure 2: Example interface of the proposed framework.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of the proposed solution. First, we describe the datasets. Next, we report and analyze the obtained experimental results for each building block of our framework.

4.1 Datasets

In our experimental evaluation, we considered seven collections available in [16]. Each collection is composed of user reviews followed by their star ratings extracted from Google Play Store. For all explored collections, we performed stopword removal (using the standard SMART list). Table 1 provides a brief summary of the reference datasets, reporting the number of features (words), documents, the mean number of words per document (density) and the mean star rating assigned by users.

Table 1: Dataset characteristics

Collection	#Feat	#Doc	Density	Star Rating
Angrybirds	1.903	1.428	7.135	3.27
Dropbox	2.430	1.909	9.501	3.01
Evernote	6.307	8.273	11.002	3.76
PicsArt	1.436	2.265	2.937	4.29
Pinterest	2.174	3.168	4.478	4.61
TripAdvisor	3.152	2.816	8.532	4.57
WhatsApp	1.777	2.956	3.103	4.41

4.2 Star Rating

Star ratings are the main mechanism for developers to infer user satisfaction. Table 1, in the last column, presents the mean number of stars assigned by users to each application. Angrybirds, Dropbox and Evernote have a mean of three stars, whereas TripAdvisor, PicsArt, Pinterest and WhatsApp applications have a mean of four stars. Although simple, star ratings provide useful assessments since they presents an overall evaluation of a particular application, in a direct, objective and quick to be extracted manner. However, the developers have no further details such as "Why does Dropbox has three stars?". In order to obtain this information, developers need to inspect manually the comments left by users, which is unfeasible given the amount of data. In addition, there is no guarantee that the stars assigned by users match the comments made.

4.3 Feature Identification

In this section, we present the results related to the instantiation of the topic modeling block of our framework. Our objective is, based

on a semantic and automatic organization of the reviews, to evaluate whether the information generated complements the star rating evaluation, presenting which features are frequently mentioned in the reviews. Having identified the topics and the most important terms related to each topic, following the strategy presented in the Section 3.1, we organize the reviews in different groups according to the topics associated with them. Next, we measure the mean number of stars related to each group of reviews. Table 2 presents the results of this analysis. In the first column, we present some of the main words of each topic and in the second column the mean star rating related to the reviews of each topic.

To illustrate the usefulness of our results, let's consider the **TripAdvisor** application. It is an application that assists users in the purchase of air tickets and in the reservation of hotels, which received a mean of 4 stars. Looking at the characteristics extracted for this application, we can see that, while topics 3 (*guidebook tour*) and 4 (*trips plan research*) are evaluated on average better than the global evaluation of the application, topics 1 (*version map feature view web*) and 2 (*hotel search back results*) are evaluated on average worse than the overall evaluation. Using this information, developers are able to identify which features are evaluated positively (topics 3 and 4) and which are evaluated negatively (topics 1 and 2) and focus their efforts on improving the negatives ones. Let's now consider the **Angrybirds** application. It is is a game that, based on the comments, contains paid advertisements and apparently the game levels were not loading properly. **Evernote** is an application used for work scheduling and time optimization, so features such as *crashing launch opening* and *time long platforms* are poorly evaluated for this app. Similar analyses can be done for all applications presented in Table 2. In spite of these analyzes, an issue arises: *Is it possible to infer the mean number of stars automatically by evaluating the sentiment contained in the reviews?* In the following section, we intend to answer this question.

Table 2: Semantic topics extracted from user reviews on each evaluated collection. Each topic is described through its most representative terms extracted by our framework. Lines with headline GLOBAL refers to the entire collection, without grouping by semantic topic.

AngryBirds		
Topic	**Stars**	**SA**
GLOBAL	3.27	2.87
fix crashing click bugs	2.01	1.71
screen touch home load	2.23	1.66
level load crashed	2.08	1.61
paid adds overrated	2.75	1.60

Dropbox		
Topic	**Stars**	**SA**
GLOBAL	3.01	2.97
crash push notification connect	1.96	1.86
documents pages numbers edit	3.05	4.10
service storage cloud customer	3.28	3.80
rename updates files	2.95	3.41

Evernote		
Topic	**Stars**	**SA**
GLOBAL	3.76	3.33
crash updated launch opening	1.97	1.55
version latest upgrade	2.41	2.36
time long platforms	3.25	2.59
tool business personal productivity organizational	4.42	4.57

Whatsapp		
Topic	**Stars**	**SA**
GLOBAL	4.41	3.57
slow installation run	3.07	2.30
send fix pictures profile	3.66	1.57
stay connected keeping accessible	4.59	4.77
add video call	4.00	3.66

PicsArt		
Topic	**Stars**	**SA**
GLOBAL	4.61	4.38
simply style options	4.77	4.71
mobile pictures edit	4.65	4.64
time long slow	3.81	3.04
install find	4.03	2.95

TripAdvisor		
Topic	**Stars**	**SA**
GLOBAL	4.29	4.01
version map feature view web	3.13	3.40
hotel search back results	3.58	3.52
guide book tour	4.37	4.61
trips plan research	4.44	4.60

Pinterest		
Topic	**Stars**	**SA**
GLOBAL	4.57	3.77
time battery duration	4.16	5.00
navigate ways fast	4.56	4.58
update fix latest load	2.74	2.14
search results work	3.07	2.94

4.4 Sentiment Rating

In this section, we present the results related to the instantiation of the sentiment analysis (SA) building block of our framework. Our goal is to evaluate whether it is possible to automatically and consistently infer a star rating for the applications by evaluating the feedback associated with each of them. Thus, we applied the SA technique in the global set of reviews from each application, as well as in the sub-sets of reviews associated with each of the identified topics and converted the output probability distributions into a 5 point rating, such as described in Section 3.3. The obtained results of applying this strategy can be found in the third column of Table 2.

From the reported results, we can observe that it is possible to automatically extract a star rating evaluation from the sentiment analysis and, in general, they are quite correlated with the stars assigned by the users. However, we cannot neglect the fact that, for all evaluated applications, there are inaccuracies—both in the global evaluation and in the analysis of topics. For instance, consider the **Angrybirds** application, in which we can see that, for all extracted topics, the inferred reviews' sentiment is lower than the associated mean star rating. On the other hand, for the **Dropbox** application, the sentiment inferred from the reviews is higher than the assigned stars. Therefore, one more question needs to be evaluated: *What is more consistent with the reviews written by users, the stars assigned by them or the stars inferred from the sentiment contained in the reviews?* We answer this question next.

4.5 Analysis of Reviews

Star rating is a measure that captures the user satisfaction w.r.t. a given application. An issue raised in this work is whether star rating matches the user's sentiment expressed in the comments made for an application. As comments may refer to more than one feature, in several cases, we observe star ratings inconsistent with the evaluation of each feature in isolation. Table 3 highlights these inconsistencies in the evaluated collections. For instance, feature 1 of **Angrybirds** has 2.75 stars, but 1.6 in SA. We selected two comments with 5 and 3 stars to better understand such rating differences. In the first comment, a user has assigned 5 stars to the application, however she/he still wants the money back—a strong user dissatisfaction indicator. In the second comment, the user explicitly states dissatisfaction regarding the update, advertisements and theme music – she/he assigned 3 stars to **Angrybirds**. In both cases, it is noteworthy that the SA algorithm assigned 1.6 stars to this feature, which better summarizes the actual sentiment on the comments. For the second feature of the same application, we have a similar case, where there

Table 3: Detailed analysis of inconsistency between Start and SA ratings.

Angrybirds			
Features	**Stars**	**SA**	**Reviews**
paid adds overrated	2.75	1.60	Want money back uhm paid for this going stay free want money back **(5 stars)**
			Annoying update this new update pretty annoying too many pop what you wouldn except from paid app plus really don like this new theme music **(3 stars)**
level load crashed	2.08	1.61	New birthday levels crash app crashes with new release level **(3 stars)**
			Crashes every time into level and just can play know that probably fault but the game its great **(4 stars)**
Dropbox			
Features	**Stars**	**SA**	**Reviews**
documents pages numbers edit	3.05	4.10	Finally new update can finally view documents the new update great **(5 stars)**
			Loving dropbox also love the new look wish there was way view the full file names files and edit file names there reason that windows went away **(4 stars)**
service storage cloud customer	3.28	3.80	Dropbox the for cloud storage dropbox excellent solution for don usually save whole lot files the small free account has been adequate far but when comes transferring files other machines without usb drive access from phone has been very useful tool **(5 stars)**
Evernote			
Features	**Stars**	**SA**	**Reviews**
crashing updated launch opening	1.97	1.55	Ios updated app updated crash open can even open without crashing now after updates other **(4 stars)**
			Crashes after update keeps crash even open **(4 stars)**
time long platform	3.25	2.59	The most amazing app been using evernote but long time running the platform time lost **(5 stars)**
PicsArt			
Features	**Stars**	**SA**	**Reviews**
time long slow	3.81	3.04	Love but hate its really slow **(5 stars)**
			Good easy handle sometimes very slow **(5 stars)**
install find	4.03	2.95	Awesome awesome apps but dont install mobile **(5 stars)**
			Perfect app everything you need whatever filter you like her love clipart and doodles here you find what want the camera awesome but slowly **(4 stars)**
Pinterest			
Features	**Stars**	**SA**	**Reviews**
time battery duration	4.16	5.00	Cool awesome very good app does not take battery life and very helpfull **(3 stars)**
			Greatest app ever prepare spend all your time but not your battery life **(5 stars)**
update fix latest load	2.74	2.14	With this new update can the search page where all the categories are plz fix **(4 stars)**
			Not happy since this last update keeps force closing and moves slower before this update would have rated please fix **(3 stars)**
TripAdivisor			
Features	**Stars**	**SA**	**Reviews**
guide book tour	4.37	4.61	Super helpful use tripadvisor almost exclusively when are planning trip **(4 stars)**
trips plan research	4.44	4.60	Simple and easy use using tripadvisor plan trip italy pretty awesome see other travelers **(4 stars)**
			Very helpful love using trip advisor plan travels the reviews are extremely helpful **(4 stars)**
Whatsapp			
Features	**Stars**	**SA**	**Reviews**
slow installation run	3.07	2.3	Its nice using what its very easy get touch with friends but the problem very slow running update **(4 stars)**
			Working slow since many days **(4 stars)**
send fix pictures profile	3.66	1.57	Anyone can profile pic its dangerous **(4 stars)**
			Profile pictures dont appear now unable see others profile pictures **(5 stars)**

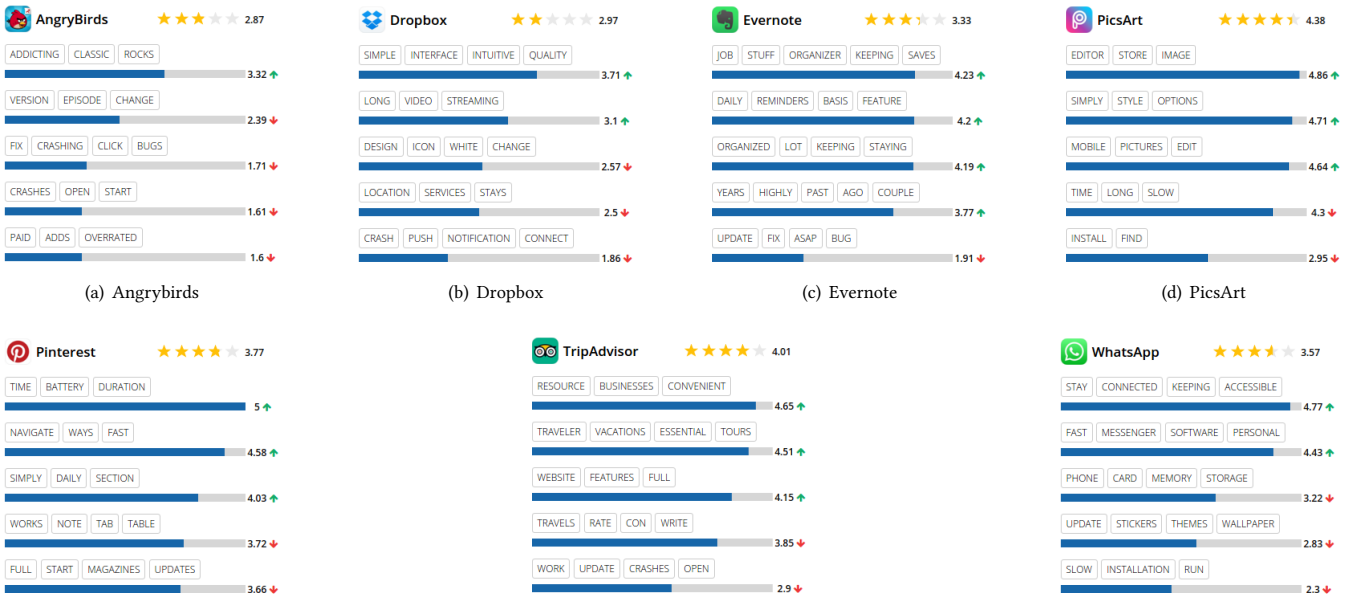
is an inconsistency between the stars and the user's sentiment. Users complain that they are not able to properly run the application due to software problems, but, due to past experiences, they have assigned 4 stars to it. Thus, the relevance of the SA strategy here is to capture user dissatisfaction about each specific feature.

Considering the **Dropbox** application, we select two features in which SA performed better than star rating. The first review in Table 3 shows that the user liked the update in which the document viewing functionality was finally added. In the second review, the user states that *"Dropbox for cloud storage is excellent"*. However, if the developers evaluate the users' feedback based only on the stars ratings in isolation, they would consider it as a low overall star rating value, incorrectly judging that the document viewing functionality is not a well accepted feature.

Similar inconsistencies were also observed in the reviews for **Evernote** features (Table 3). In this case, comments regarding startup failures have an unusual average of 4 stars. Naturally, people generally do not like to have problems when opening the application (in some cases, users even give up of using the application due to such kind of problems). On the other hand, SA was able to capture this behavior (SA equals to 1.55), doing as better job in determining a potentially bad user experience. For the second feature, we can observe an interesting point – the user explicitly expresses enjoyment with the application, with a 5-star rating, but also complains of wasting a lot of time running the platform. Here, we have a mix of sentiments hidden by a 5-star rating. By analyzing the corresponding user comment, SA was able to extract a better picture of user sentiment, inferring half of the stars as being the actual user sentiment. Once again, in our analysis, one can see that SA provides a more trustful picture of the users' opinion than the simple star rating evaluation.

For the **PicsArt** application, we have two features positively classified by the star rating, but negatively classified by SA. Observing the reviews in Table 3, we can see cases such as *"love but hate its very slow"* and *"good easy handle sometimes very slow"*. Thus, we noticed that users assigned 4-5 stars because they like the application as a whole but, there are specific characteristics that causes user dissatisfaction, such as low application response times. The same applies to *install find*, where the users find the application great but they have trouble in installing it on their mobiles. These cases end up generating an undesired bias in the star rating analysis, which justifies our proposal for the SA block.

In the comments related to **Pinterest**, we found a feature evaluated as 5 by SA (i.e., the highest possible value): it has to do with the duration of the battery while using the application. Users evaluated the application well because they could use the application for a long time without consuming much of their mobile battery. Based on this, the developers would have good feedback regarding their application, striving to maintain this well accepted feature to guarantee a greater adherence to their application. However, we noticed that the last application update caused problems to the users, due to interruptions in the application. This ultimately brought considerable user dissatisfaction without being effectively reflected by the star ratings in isolation. In the **TripAdvisor** application, the SA method inferred rates highly consistent with the star ratings. Based on the comments, we also noticed that this is a well-evaluated application because it is useful and it meets the users' expectations, such as evinced by the comments.



(a) Angrybirds (b) Dropbox (c) Evernote (d) PicsArt

(e) Pinterest (f) TripAdvisor (g) WhatsApp

Figure 3: Evaluation Interfaces for apps.

Finally, for the **WhatsApp** application, the mean star rating for the first feature is 3.07. However, observing this feature's reviews, we can see that they are mainly negative. As before, SA properly captures the negative sentiment for this feature. For the second feature, it was well evaluated in terms of star rating, but when we analyze the comments, we see that this feature is considered a bad one. This last case is interesting because users are satisfied with the services provided by the application (they attributed 5 stars), but they are unsatisfied with a specific application feature, which means that the sentiment related to this feature is much lower than the score assigned by the star rating.

In face of the foregoing discussion, we conclude that the extraction of features using the first block (topic model) is of great importance for the task of summarization obtained by means of user comments. Moreover, in spite of the fact that star ratings provide useful information, the SA technique is more accurate, guaranteeing better results by exploiting textual comments.

4.6 Evaluating the Summarization Interface

In this section, we present the output interface of our framework. As described in Section 3.3, this interface encompasses all results shown in Table 2 in a friendly manner, facilitating the interpretation of the results related to the first block (topic modeling), as well as, the second one (sentiment analysis).

Due to space limitations, we choose five topics for each application to illustrate the interface utility. We can see in Figure 3-a (Angrybirds) that all topics (features) are of great value to the developer. This application has one well-evaluated topic, related to *songs* and *themes*. On the other hand, it has several characteristics that cause the application to be poorly evaluated. For example, *bugs that cause application closure* and *advertisements in the paid version*. We would like to point out that the arrows (green and red) ease to interpret positives or negatives features on the interface. Thus, developers are able to quickly identify promising opportunities to improve the application and increase user satisfaction.

Regarding Dropbox (Figure 3-b), users positively qualify the features related to the *application interface* and the *support for loading large videos*. Using our framework, developers would know that these features do not need improvements, since they already satisfy the user needs. Hence, developers should pay more attention to the negative ones, such as *design*, *connectivity*, *bugs* etc. Following the analysis, Evernote (Figure 3-c) has the worst evaluated feature (*update failures*). Through our framework developers can easily know about this very specific issue, prioritizing their work towards improving this pain point.

The PicsArt application (Figure 3-d) has the features *photo editing* and *storage*, *interface style* and *photo editing* well-evaluated by users, as well as features that cause high inconvenience to users, such as *extended time* and *finding package installers*. The Pinterest application (Figure 3-e) has a feature (*time battery*) that reached a 100% user satisfaction (sentiment). However, as highlighted in the proposed interface, the use of the application in tablets causes an undesired increase in users' dissatisfaction. For the TripAdvisor application (Figure 3 -f), there is a disappointment about *updating the application* and the *difficulty of evaluating hotels within the application*. Finally, the WhatsApp application (Figure 3-g) has two features very well-evaluated (*accessible* and *instantaneous*), but the features related to *storage*, *theme upgrades* and *time-consuming installation* were poorly evaluated by users. In sum, the proposed interface serves the purpose of providing a detailed and accurate picture of application features in terms of user needs and satisfaction, being a strong advisor to guide application development prioritization, guaranteeing better end-user adherence.

5 CONCLUSIONS AND FUTURE WORK

In this work, we proposed a novel framework that offers to mobile application developers a more detailed and informative view of the evaluations made by users, identifying which features contribute positively or negatively to the users' evaluation. This framework has three main building blocks that extract semantic topics and relevant features (i.e. app functionalities); assign sentiment to each extracted feature and summarize all information in an intuitive interface. Analyses on seven collections of reviews from Google Play Store demonstrated that, given an application, (1) our framework allows the organization of user comments in subcategories that facilitate the understanding of which features impact the overall evaluation and (2) Sentiment Analysis is more accurate in capturing the sentiment transmitted by the user than star ratings. In sum, the main implication is that with this new framework, developers may easily detect which topics are negatively impacting the overall rating of a specific application and should be improved for the sake of a better user experience. As future work, we plan to instantiate different configurations regarding topic modeling, such as SVD, LDA, BTM and pLSI and sentiment analysis, such as VADER, so that we are able to further understand their impact in our framework. Moreover, we intend to propose other visualization interfaces, evaluating all of them through concepts related to semiotic engineering.

6 ACKNOWLEDGMENTS

This work was partially supported by CAPES, CNPq, Finep, Fapemig, MasWeb and InWeb.

REFERENCES

[1] A. I. Anam and M. Yeasin. Accessibility in smartphone applications: What do we learn from reviews? In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 35:1–35:2, New York, NY, USA, 2013. ACM.

[2] W. H. Auden. *The Complete Works of W. H. Auden: Prose*, volume 2. Princeton University Press, April 2002.

[3] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of Cal. Math. Soc.*, 35(1):99–109, 1943.

[4] P. V. Bicalho, T. de Oliveira Cunha, F. H. J. Mourão, G. L. Pappa, and W. M. Jr. Generating cohesive semantic topics from latent factors. In *BRACIS*, pages 271–276. IEEE Computer Society, 2014.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[7] Y. Chen, M. Rege, M. Dong, and J. Hua. Non-negative matrix factorization for semi-supervised data clustering. *Knowledge and Information Systems*, 17(3):355–379, 2008.

[8] Z. Chen and B. Liu. Topic modeling using topics from many domains, lifelong learning and big data. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–703–II–711. JMLR.org, 2014.

[9] X. Cheng, X. Yan, Y. Lan, and J. Guo. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941, Dec 2014.

[10] J. Choo, C. Lee, C. K. Reddy, and H. Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, Dec 2013.

[11] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *ACM WWW*, 2003.

[12] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[13] E. C. Dragut, C. Yu, P. Sistla, and W. Meng. Construction of a sentimental word dictionary. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1761–1764, New York, NY, USA, 2010. ACM.

[14] A. Esuli and F. Sebastiani. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 617–624, New York, NY, USA, 2005. ACM.

[15] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[16] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In T. Gorschek and R. R. Lutz, editors, *RE - 22nd International Requirements Engineering Conference*, pages 153–162. IEEE Computer Society, 2014.

[17] W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky. Inducing domain-specific sentiment lexicons from unlabeled corpora. *CoRR*, abs/1606.02820, 2016.

[18] S. Hedegaard and J. G. Simonsen. Extracting usability and user experience information from online user reviews. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2089–2098, New York, NY, USA, 2013. ACM.

[19] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[20] X. Hu, J. Tang, H. Gao, and H. Liu. Unsupervised sentiment analysis with emotional signals. In *Proc. of WWW '13*, pages 607–618, Republic and Canton of Geneva, Switzerland, 2013.

[21] C. J. Hutto and E. Gilbert. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM'14*, 2014.

[22] H. Korhonen, J. Arrasvuori, and K. Väänänen-Vainio-Mattila. Let users tell the story: Evaluating user experience with experience reports. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4051–4056, New York, NY, USA, 2010. ACM.

[23] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[24] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Comput.*, 19(10):2756–2779, 2007.

[25] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proc. of WWW '11*, pages 347–356, Hyderabad, India, 2011.

[26] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[27] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[28] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Publications, 2002.

[29] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 125–134. IEEE, 2013.

[30] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.

[31] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, Jan. 2008.

[32] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86, 2012.

[33] L. Polanyi and A. Zaenen. Contextual valence shifters. *Computing attitude and affect in text: Theory and applications*, pages 1–10, 2006.

[34] L. Rocha, F. Mourão, T. Silveira, R. Chaves, G. Sá, F. Teixeira, R. Vieira, and R. Ferreira. Saci: Sentiment analysis by collective inspection on social media content. *Web Semantics: Science, Services and Agents on the World Wide Web*, 34:27–39, 2015.

[35] P. Rodrigues, I. S. Silva, G. A. R. Barbosa, F. R. d. S. Coutinho, and F. Mourão. Beyond the stars: Towards a novel sentiment rating to evaluate applications in web stores of mobile apps. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 109–117, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[36] S. Rothe, S. Ebert, and H. Schütze. Ultradense word embeddings by orthogonal transformation. *CoRR*, abs/1602.07572, 2016.

[37] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 1988.

[38] G. Sá, T. Silveira, R. Chaves, F. Teixeira, F. Mourão, and L. Rocha. Legi: Context-aware lexicon consolidation by graph inspection. In *ACM SAC*, 2014.

[39] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June 2011.

[40] M. Thelwall, K. Buckley, and G. Paltoglou. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163–173, 2012.

[41] L. Velikovich, S. Blair-Goldensohn, K. Hannan, and R. McDonald. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 777–785, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[42] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMLP*, pages 347–354, USA, 2005.