Same Same but Different: Finding Similar User Feedback Across Multiple Platforms and Languages

Emanuel Oehri
University of Zurich
emanuel.oehri@gmail.com

Emitza Guzman
Vrije Universiteit Amsterdam
e.guzmanortega@vu.nl

***Abstract*—Users submit feedback about the software they use through application distributions platforms, i.e., app stores, and social media. Previous research has found that this type of feedback contains valuable information for software evolution, such as bug reports, or feature requests. However, popular applications receive thousands of feedback entities per day, making their manual analysis unrealistic.**

In this work, we present an approach to automatically identify similar user feedback across different languages and platforms. At the core of the approach is a word aligner that aligns words based on their semantic similarity and the similarity of their local semantic contexts. Additionally, we make use of machine translation, sentiment analysis, and text classification, to extract the sentiment polarity and content nature of user feedback written in different languages. We use the results of these components to compute a similarity score between user feedback pairs. We evaluated our approach on user feedback entities written in four different languages, and retrieved from five different mobile applications obtained from four different app stores and social networking sites. The obtained results are encouraging. Compared to human assessment, the overall performance for monolingual user feedback pairs yielded a strong correlation of 0.79. For the crosslingual feedback pairs the correlation was also strong, with a value of 0.78.

***Keywords*-user feedback; software evolution; text mining; requirements elicitation.**

I. Introduction

User feedback is crucial factor for creating and maintaining usable software [1]. Previous research demonstrated the positive effects that involving users has on both software success and user satisfaction [2]. Nevertheless, software designers are often unaware about the specific user needs and expectations before releasing their products [3]. During the development phase, stakeholders are often overlooked [4], and specific use cases are often not covered [5]. Additionally, as the world changes, so do user needs. Software systems need to evolve and adapt to the ever-changing environment and context of use, in order to best fit the new needs. It is therefore vital to consider user feedback after the software release, as it increases the overall likelihood that the software stays relevant and useful throughout its evolution.

With the growing trend of Internet use, more and more users are writing feedback about software applications, either via social media or specialized user feedback platforms. This feedback contains invaluable information, such as bug reports, feature requests and user experience descriptions [6], [7], [8] that could be used to drive the development effort and improve forthcoming releases. Nevertheless, this feedback is often scattered across different platforms, is submitted in the orders of thousands per day for popular software and its quality varies [9].

Previous work has addressed these issues by proposing automatic or semi-automatic mechanisms to filter out irrelevant feedback, group related feedback, classify the feedback into different categories relevant for software evolution, as well as prioritize feedback, e.g., [10], [11], [12], [13]. The vast majority of this work has been evaluated on feedback written in English and stemming from a single platform, with app stores receiving most of the attention.

While approaches for grouping similar feedback have been proposed, these approaches tend to use clustering and topic modeling in their solutions. Thus, they focus more on the general, topical similarities of the feedback and not on the fine-grained aspects being discussed, i.e., a specific bug or a request for new functionality related to a specific feature.

In this work we present SIMBA🐯 (SIMilarity Based Approach), a fine-grained approach for identifying and measuring similarities between feedback written across different platforms and languages. SIMBA uses machine translation, natural language processing techniques, lexical sentiment analysis and supervised machine learning for categorizing the feedback into categories relevant for software evolution. We combine the results of the different steps of our approach into a weighted function that measures the extent to which a feedback pair is similar to each other. The similarity scale used by our approach allows for the detection of feedback pairs that range from *very similar or duplicates*, *somewhat similar*, *somewhat related but not similar*, to *slightly related* and *unrelated*— depending on whether they describe the same high level topic, mention the same bug or features and describe the faced issues with the same level of detail.

SIMBA's results can help product owners, requirements engineers and developers to prioritize feedback, as feedback with a higher amount of semantical duplicates could need more urgent addressing. Additionally, it could also support the identification of inconsistent or contradicting feedback, as this typology of feedback fits into the similarity levels defined in our approach.

We performed two evaluations and assessed SIMBA on a manually annotated set of 4,100 user feedback pairs collected

2332-6441/20/$31.00 ©2020 IEEE
DOI 10.1109/RE48521.2020.00017

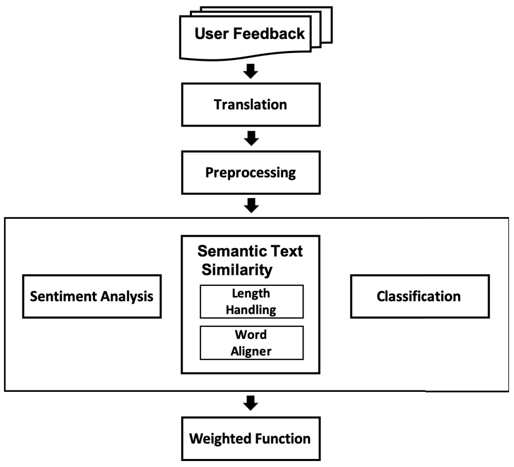

Fig. 1: SIMBA Overview.

across four different platforms (Apple App Store, Google Play App Store, Twitter and Facebook) and written in English, French, German, and Spanish. Our evaluations show that the results of SIMBA strongly correlate with human assessment.

This work contributes (1) SIMBA, an approach for detecting and measuring similarities in feedback across different channels and languages, and (2) empirical evidence that our approach has a good performance compared to human assessment.

II. The SIMBA approach

The goal of SIMBA is to automatically identify and measure similarities between feedback written across different platforms and languages. Figure 1 shows an overview of the approach. First, we *translate* all non-English feedback into English. Second, we *preprocess* the feedback text. Then, we calculate the *semantic text similarity* for all feedback *pairs* in our dataset by handling strongly differing feedback lengths and using a word-aligner. We assume that very similar texts have a similar nature, thus we *classify* the feedback into three categories: “bug report”, “feature request” and “other” by using a supervised machine learning model. We also compute the sentiment polarity of each feedback by means of a lexical *sentiment analysis* tool, as we believe that duplicate or similar feedback could have comparable sentiment polarities. Finally, we calculate a similarity score for each feedback pair in our sample by means of a *weighted function*. This weighted function combines the results from the text similarity, classification and sentiment analysis steps of our approach. The final result is a *similarity score* for each feedback pair in our data collection. A higher score implies a higher equivalence between the feedback pairs. In the following sections we describe each step.

A. Translation

We use the Google Translate API[1] to convert non-English feedback into English and apply the subsequent steps of our approach to the translated feedback. By translating the feedback into English, and then mapping the results obtained for the English version back to the feedback in the original language, we follow the same steps as previous work studying the similarity of textual artifacts written in different languages [14], [15], [16].

B. Preprocessing

We preprocess the text by: concatenating title and feedback text to a single string (if the title is available[2]), converting all text into lowercase, removing html characters common in some user feedback obtained from the web, removing URLs, all punctuation, as well as the characters ”@” and ”#” (commonly used in some of the evaluated channels as markups), and converting support account identifiers present in the user feedback to the actual software name[3].

C. Semantic Text Similarity

This step assesses the semantic similarity of two instances of user feedback by (1) applying a simple *length handling* technique to handle texts of varying lengths and by (2) *aligning* the words in a pair of feedback texts.

1) Length handling: Similar feedback can have a similar semantic meaning, while having significant differences in its length. Nevertheless, substantial differences in the length of feedback pairs are detrimental to the identification of their similarity via the *word alignment* (see Section II-C2). We mitigate such issue by carrying out a length handling process on feedback pairs presenting considerable length differences.

In line with previous work [17], our length handling process leverages the calculation of the centroid of the longer feedback text, and subsequently compresses this feedback by excluding the words which result most distant from the centroid. In our work, we use the K-means clustering implementation available in scikit-learn[4] for the calculation of the centroid. The decision of when to carry out the length handling process, as well as the number of words to be excluded, is determined by considering the length of the compared feedback pairs.

Specifically, we compress feedback that contains five or more times as many words as the other instance with which it is compared to, as well as feedback that exceeds 150 words. For the cases in which the shorter feedback contains less than 150 words, we create a compressed feedback version of only five times the number of words contained in the shorter user feedback (i.e. the instance in the pair, that is not compressed). For the cases in which both feedbacks exceed the 150 threshold, we create a representation of 150 words for each of the corresponding texts.

The threshold values were determined empirically via extensive preliminary experimentation. The chosen thresholds yield

[1]http://translate.google.com

[2]Some platforms such as app stores allow for the title and feedback text separation, whereas others, i.e., Twitter and Facebook only allow for the input of text, without a title.

[3]Tweets from application support accounts often include the names of such accounts. We converted these names to the respective software names e.g., ”spotifycares” −>”spotify” and ”dropboxsupport” −>”dropbox”.

[4]https://scikit-learn.org

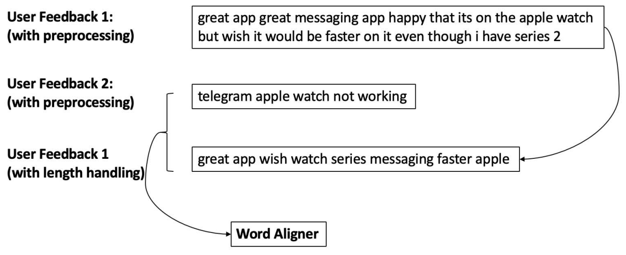
User Feedback 1: (with preprocessing)
great app great messaging app happy that its on the apple watch but wish it would be faster on it even though i have series 2
User Feedback 2: (with preprocessing)
telegram apple watch not working
User Feedback 1 (with length handling)
great app wish watch series messaging faster apple
Word Aligner

Fig. 2: Example of length handling of a user feedback pair.

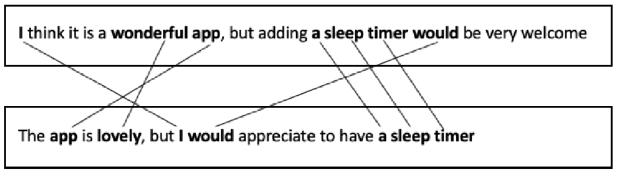
I think it is a wonderful app, but adding a sleep timer would be very welcome
The app is lovely, but I would appreciate to have a sleep timer

Fig. 3: Word aligner example of a user feedback pair.

the best results in the subsequent step of SIMBA, namely the word alignment. In addition to the optimization of the SIMBA analysis results, adopting the length handling process with appropriate thresholds led to better time performance improvements in the word aligner. Figure 2 shows an example of a user feedback pair in which the length handling technique is applied.

2) Word aligner: In natural language processing, alignment is the task of identifying pairs of semantically related units, words in our case, in a set of two texts. Figure 3 shows an example of the alignment of a feedback pair. Alignments provide valuable information regarding *how* and to what *extent* two texts are similar to each other. SIMBA uses a slightly modified version of an existing word aligner [18]. This approach aligns words in two given input sentences or short texts based on their semantic similarity and the similarity between their local semantic contexts while relying on a very small number of external resources.

The word aligner makes use of contextual evidence to determine if a word pair will be aligned. In addition, it is informed by a word semantic similarity algorithm using PPDB[5], a large automatically extracted database of lexical, phrasal and syntactic paraphrases constructed using bilingual pivoting [19] over large parallel corpora. The word aligner aligns different types of word pairs (note that we regard *nouns*, *main verbs*, *adjectives* and *adverbs* as *content words*):

- **Identical word sequences**, containing at least one noun, main verb, adjective or adverb. For example, "a sleep timer" in Figure 3.
- **Named entities**, including full and partial mentions and acronyms of the same entity. For example, "Guevara" and "Che Guevara".
- **Content words with identical or equivalent relationships**, determined by a set of handcrafted rules. For example, the relation "object that is wonderful/lovely" between the adjective "wonderful/lovely" and "app" in Figure 3.
- **Content words with similar context**, the context of a word is defined as a sequence of three words to its left and three words to its right, whereas only content words are considered. The words are aligned if the aligner identifies similarity between the context, or word similarity between the words. For example, "would" in Figure 3.
- **Stopwords**, which are not yet aligned by the identical word sequence or dependency-based module. Alignment decisions are again derived from dependencies and surrounding words. For example, "I" in Figure 3.

In SIMBA, the input of this step is the text of the feedback pairs after the length handling. For calculating a semantic similarity score as the output of the word aligner we adopt Toutanova's et al. [20] approach and compute a single calculation over all words in the two input texts. Particularly, given input feedback f_1 and f_2:

$$sts(f_1, f_2) = \frac{n_c^a(f_1) + n_c^a(f_2)}{n_c(f_1) + n_c(f_2)} \tag{1}$$

where $n_c(f_1)$ and $n_c(f_2)$ are the number of content words and $n_c^a(f_1)$ and $n_c^a(f_2)$ are the number of aligned content words in f_1 and f_2, respectively. The computed value for sts represents the word aligner coefficient, which is used as an input in our weighted function to identify similar user feedback (see Section II-F). Brychcin et al. [21] proposed to weight sts based on a words IDF[6] values. The consideration of IDF builds on the assumption that not all word alignments have the same importance for the meaning of a text. Although this is a plausible statement, our preliminary experimentation showed that this computation method did not improve the performance of our approach.

In our preliminary experimentation, we also observed that the computed word aligner coefficient is generally too low for highly similar texts. With this observation in mind, we fine-tuned the coefficient calculation. If the computed score for sts exceeds an integer by some decimal value, we consider the next higher integer as the final score—denoted by sts_{ft}. To make sure that we accurately detect unrelated user feedback pairs we mark them with a sts_{ft} of 0 if $sts < 0.5$. Additionally, to make our results comparable to the assessment done in our evaluation (see Table III) we multiplied the sts value by 4. In other words, sts_{ft} is fine-tuned so that:

$$sts_{ft}(f_1, f_2) = \begin{cases} 4 & \text{if } sts(f_1, f_2) * 4 >= 3 \\ 3 & \text{if } sts(f_1, f_2) * 4 >= 2 \\ 2 & \text{if } sts(f_1, f_2) * 4 >= 1 \\ 1 & \text{if } sts(f_1, f_2) * 4 >= 0.5 \\ 0 & \text{if } sts(f_1, f_2) * 4 < 0.5 \end{cases} \tag{2}$$

D. Sentiment Analysis

We define *sentiment* as the affect or mood expressed in a feedback text and hypothesize that similar feedback could have

[5]http://paraphrase.org

[6]The inverse document frequency (IDF) is a statistical weight used for measuring the importance of a term in a text document collection.

a comparable sentiment. Therefore, we consider sentiment as a coefficient in our weighted similarity function (see Section II-F). We use a lexical sentiment analysis tool specialized in short informal text, SentiStrength [22], for assigning a numerical score to the sentiment expressed in each feedback entity. In SentiStrength, each feedback is assigned a positive and negative sentiment score. Positive scores range from [1, 5] and negative sentiments range from [-1, -5]. The values 1 and -1 indicate the absence of any positive and negative sentiment, respectively. Similarly, 5 and -5 indicate extreme positive and negative sentiments, respectively. We compute a single sentiment score in the [- 4, 4] range for each feedback by adding both positive and negative scores. For the convenience of further computations (see Section II-F) the single sentiment score is then adjusted to fit to a [1, 9] range by adding 5 to the original values.

E. Content Classification

We hypothesize that similar feedback has a similar content nature and therefore classify the feedback into three categories: "feature request", "bug report" and "other". For content classification, we use a Multinomial Naive Bayes (MNB) model, which we empirically found had the best results among several evaluated classifiers in our previous work [13].

To train our classifier we converted our preprocessed feedback text into a vector space model using TF-IDF as a weighting scheme. We use a readily existing dataset from our previous work [13] of 1,350 manually labeled tweets for the training of the model. We apply the trained classifier to predict the corresponding categories of all feedback input into SIMBA. The output of this step is a list of all feedback entities, each associated to its respective category.

F. Weighted Function

We determine the similarity between two given user feedbacks by means of a weighted function, FS.

Let F $= \{f_1, ..., f_n\}$ be the set of user feedback for which the similarities have to be computed. We calculate the similarity between a pair of user feedbacks as follows:

$$\text{FS}(f_1, f_2) = \frac{\sum_{i=1}^{N} w_i * c_i(f_1, f_2)}{N} * 4 \tag{3}$$

where N is the number of similarity analyses considered (in our case N=3, as it denotes the three main steps of our approach: *semantic text similarity*, *sentiment analysis* and *content classification*.), c_i the similarity coefficients and w_i the weights calculated with a multiple linear regression (see Section IV-D). All of our coefficients yield a value between [0, 1], with the values closer to 1 indicating a closer similarity in the evaluated variable. The final value of FS is further normalized in the range [0,4], in order to adhere to the widely used 5-point Likert-scale, which is used to evaluate the effectiveness of SIMBA (see Section V-A). The formalization of the FS function is flexible and similarity coefficients can be easily added or modified. In Table I, the similarity coefficients and their calculations are summarized.

TABLE I: Coefficients used in weighted function for textual similarity.

Attribute	**Calculation of similarity coefficient**
Semantic text similarity	$c_1(f_1, f_2) = \frac{sts_{ft}(f_1, f_2)}{4}$
Sentiment analysis	$c_2(f_1, f_2) = \frac{(8-\lvert senti(f_1)-senti(f_2)\rvert)}{8}$
Classification	$c_3(f_1, f_2)$ = 1 if category of f_1 & f_2 is same, 0 otherwise

We discuss how we determine the weights of our similarity function in Section IV-D. The output of our approach is a similarity score between feedback pairs in the [0, 4] range.

III. SIMBA Evaluation Process

To assess the accuracy of SIMBA, we evaluated its results against human judgement. Specifically, we evaluated its accuracy by considering the similarity of (1) feedback written in English across four different platforms and (2) feedback written in four different languages on a single platform. The questions that guided our evaluation are:
Q1. Monolingual, cross-platform: How accurate is SIMBA compared to human judgement when assessing the similarity of feedback available in different platforms and written in English?
Q2. Crosslingual: How accurate is SIMBA compared to human judgement when assessing the similarity of feedback written in different languages?

We answered these questions through two distinct evaluation processes. We describe the details of our first evaluation, answering Q1, in Section IV. The details of our second evaluation, focusing on Q2, are documented in Section V.

IV. Evaluation 1: Accuracy in monolingual, cross-platform data

In our first evaluation, we focus on the similarities between feedback texts written in English across different platforms.

A. Dataset

Our dataset consists of user feedback from four different platforms: Google Play, Apple App Store, Twitter and Facebook. Google Play and the Apple App Store are mobile application distribution platforms, i.e., app stores, in which users provide dedicated reviews on the software they download. These two platforms have received considerable attention from previous research work (see Section VII). In our evaluation we also include data from two platforms that have received relatively little attention when studying user feedback about software: Twitter and Facebook. In contrast to the selected app stores, these platforms are employed by users for more generic purposes other than downloading software and reading or giving feedback about the downloaded software. We collect user feedback for five popular software applications: Spotify, Slack, Dropbox, Telegram and Waze. We opt to collect data from these five applications due to their popularity and domain diversity, and because they received feedback on all four platforms considered in our study.

TABLE II: Monolingual, cross-platform collected data.

#Google Play	#App Store	#Twitter	#Facebook	#Total
112,004	19,635	326,254	6,437	464,330

We gather 464,330 user feedback entities from four different platforms, for a duration timespan of 60 days. Table II shows the number of collected user feedback for each platform. The Google Play user feedback is gathered by using the Google Play API. For collecting the Apple App Store feedback, we leverage a publicly available script[7]. We use an open-source library[8] to access the Twitter Search API[9], and collect tweets whose content either mention[10] or reply[11] to the software company accounts of the selected applications. While the Spotify and Dropbox accounts are exclusively dedicated to giving support to end-users, the accounts of Slack, Telegram and Waze are more generic, and are also used for marketing purposes. We collect the Facebook user feedback via a publicly available script[12] that accesses the Facebook Graph API[13].

B. Creation of Labelled Dataset

We compare SIMBA's results against a sample of manually annotated *feedback pairs*. We create this dataset by using the content analysis methods described by Neuendorf [23]. For the process we create an annotation guide and sample 2,050 feedback pairs, with two annotators analyzing and labeling the sample according to the guide.

1) Annotation Guide: We made the annotation guide to systematize the creation of our labelled dataset and reduce the subjectivity in the annotation. The guide contains explanations of the different *similarity levels* between feedback pairs, supported by examples. The ordinal similarity levels range from *very similar or duplicate*, *somewhat similar*, *somewhat related but not similar*, *slightly related* and *unrelated*. Table III shows the definitions of each similarity level included in the guide.

2) User Feedback Sampling: Before selecting our 2,050 sample of feedback pairs, we perform some data cleansing steps. Also, during the sampling of the data we take into account some balancing criteria. We detail these two steps as follows.

Data Cleansing. The data retrieved from Twitter and Facebook includes tweets and posts written by the account operators (i.e. the software providers). As we are only interested in end-user feedback, we remove all such data. In addition, the Twitter data includes retweets, which are basically duplicates with the prefix "RT" at the beginning of the tweet. We remove these duplicates to avoid measuring the performance of the approach on lexically identical user feedback. Finally, we observe that for the Waze application we have a considerable number of automatically generated tweets with no user feedback information, following specific patterns, e.g., *"Driving to Queens, sharing real-time road info with wazers in my area. ETA 12:47 PM using @waze - Drive Social."*. This large number of tweets constitute noise in our data, thus, we remove them with the help of regular expressions.

TABLE III: Similarity level definitions.

Similarity level	Definition	Mapped Similarity Score
Very similar or duplicate	The two texts describe **very similar** things, are on **the same** topic and refer to the same functionalities, features, bugs, actions, events, or ideas. Some **less important information** may be missing in one of the texts, but the texts could be used as a substitute to each other, preserving their core information.	4
Somewhat similar	The two texts describe **somewhat similar** things, are on the same topic and refer to many of the same functionalities, features, bugs, actions, events, or ideas, but include **slightly different details**. One of the texts **may omit/differ on important information** present in the other text.	3
Somewhat related but not similar	The two texts describe **dissimilar things** on the same topic, but only refer to **related** functionalities, features, bugs, actions, events, or ideas. Some **important information differs/missing**.	2
Slightly related	The two texts describe **dissimilar things**, but are on the **same topic**.	1
Unrelated	The two texts **do not describe the same thing** and are not on the same topic.	0

Balancing. We select a sample of feedback pairs from the dataset so that there is a balanced number of (1) user feedback for each application, (2) platform combinations, (3) similar and dissimilar user feedback pairs, (5) varying lengths of user feedback, (6) user ratings—when available. These considerations allow for the evaluation of our approach against multiple aspects, such as the ability to find similar user feedback of varying length and ratings across different platforms, as well as the ability to differentiate between similar and dissimilar user feedback. While the consideration of the number of feedback per application, rating and platform combinations is straightforward. The balancing with respect to similarity and length requires some additional computation.

- *Similarity balancing.* When sampling pairs of user feedback the probability that they are unrelated is higher than of them being related. To ensure that feedback with varying degrees of similarity are present in the sample, we consider the cosine similarity of possible feedback pairs while sampling the data. We map the computed cosine similarity values into the five similarity levels shown in Table III. As the computed cosine value is in the [0,1] range, we define a similarity level for each 0.2 step of the cosine similarity value. For example, if a feedback pair has a cosine similarity larger or equal to 0.8 it is mapped to the *very similar or duplicate* level, whereas if a feedback pair has a cosine similarity larger or equal than 0.40 and lower than 0.60 it is mapped to the similarity level *somewhat related but not similar*.
- *Length balancing.* The cosine similarity measure has an undesired side effect. For user feedback pairs with a high cosine similarity score, the variety in text length is

[7]https://github.com/grych/AppStoreReviews
[8]http://www.tweepy.org/
[9]https://developer.twitter.com/
[10]Makes reference to application account anywhere in the tweet.
[11]Direct response to the application account.
[12]https://github.com/minimaxir/facebook-page-post-scraper
[13]https://developers.facebook.com/docs/graph-api

TABLE IV: Examples of manually extended feedback.

Sim. score	Feedback 1	Feedback 2 (original)	Feedback 2 (extended)
4	still waiting for waze to android auto	No Android auto —Waiting for Android auto	No Android auto —Waiting for Android auto. *Please bring it as soon as possible! Would make my life easier. Thanks!*
3	It is very good.	Good very good. Thanks for this app.	Good very good. Thanks for this app. *Its an excellent tool for managing your files. Recommended!*

practically non-existent. To solve this problem we follow the same approach as previous work [24] and manually extend some of the highly similar user feedback—achieving a distribution of varying lengths in our labelled dataset. For the two highest similarity levels, i.e., those with the *very similar or duplicate* and *somewhat similar* levels (see Table III) we modify 25% of the user feedback pairs. Concretely, one instance of a user feedback pair is manually extended, while preserving the original similarity level according to the definition of the class defined in the annotation guide. Table IV shows two examples of such extensions.

We also observe that our dataset contains a large number of very brief user feedback. However, we want to evaluate our approach against more extensive user feedback, which could be more informative for software evolution and requirements engineering tasks. For this purpose, we sample two sets of user feedback: *ML-SL* considering all lengths and *ML-L* excluding user feedback containing fewer than five words. For both sample sets we follow the same cleansing and balancing procedures explained above. The ML-SL results in a sample size of 1,026 feedback pairs, whereas the ML-Lin a size of 1,024.

3) Sample Annotation: In this step, two annotators independently label the similarity of the 2,050 feedback pairs using the similarity scale presented in Table III. The annotation is done through a specialized web tool. The shown feedback text is exactly the one that was collected (with no preprocessing), with the exception of the manual extensions explained in the section above. The average time to label all feedback pairs results is around 23.5 hours per annotator, which corroborates the large amount of effort to manually assess feedback similarity. To assure that the task is clear and avoid major disagreements, two trial runs with 50 feedback pairs each are executed. The disagreement rate between annotators on the final sample used in this evaluation is 28.5% and is solved through direct discussion among the annotators.

C. Metrics

The five similarity levels used by annotators in the creation of the labelled dataset are ordinal. We assign these levels a numerical value to compare our results to the scores computed by SIMBA, which are in the [0,4] range. Table III shows the mapping of the ordinal similarity levels into numerical ones.

We evaluate our approach using the Pearson product-moment correlation coefficient between the computed scores and the human labeled scores for the feedback pairs in our annotated set. This metric has been previously used to measure the effectiveness of approaches detecting text similarity on artifacts from non-software engineering domains, e.g., [14], [15], [16], [24].

D. Finding the Weights in Similarity Function

To determine the weights of each coefficient in the weighted similarity function (see Section II-F) we use a multiple linear regression: $Y = \beta_0 + \beta_1\chi_1 + \beta_2\chi_2 + \beta_3\chi_3 + \varepsilon$ where the dependent variable Y represents the array of the human labeled similarity scores for all user feedback pairs in the annotated set. The three explanatory variables in our regression equation are defined as: χ_1 denotes the array of computed semantic text similarity coefficients, χ_2 denotes the array of computed sentiment coefficients and χ_3 denotes the array of computed classification coefficients. The intercept β_0 is set to zero. We feed the regression with our input data and execute it. As a result we get values for the regression coefficients β_1, β_2 and β_3. We normalize these coefficients for the sum to be 1 and use this normalized coefficients as our weights in the weighted similarity function.

Computing the regression coefficients only once for the whole labelled set would overfit our model. We address this issue by applying a 10-fold cross validation for reporting our results. First, we randomly split our data into 10 evenly sized chunks. Then, for each of the 10 chunks we compute the weights via multiple linear regression. We then use these weights for calculating the similarity scores on the remaining nine chunks and the respective Pearson correlation. We report our results as the mean value of the Pearson correlations computed for each fold.

E. Results

We compare the results of our approach, SIMBA, against a *variation* that only includes the results of the semantic text similarity, *SIMBA$_{STS}$*. We also compare our results against a baseline that uses the cosine similarity for the computation of the similarity between feedback pairs. The same baseline metric is used in many semantic text similarity competitions, e.g., [14], [15], [16], [24]. Table VI shows an overview of our results and Table V shows examples of input feedback pairs, the similarity scores given by annotators and by SIMBA.

The correlation coefficient between the similarity score computed by SIMBA and the manually annotated set has an average of 0.79 among all samples. All of the samples result in correlation values above 0.76, indicating a strong positive correlation with human assessment. Furthermore, the results from SIMBA and its variation, SIMBA$_{STS}$, all surpass the performance of the baseline with a considerable difference.

TABLE V: Result examples from monolingual, cross-platform evaluation.

Application	Platform 1	Platform 2	Feedback 1	Feedback 2	Manual sim. score	SIMBA sim. score
Dropbox	Facebook	Facebook	I'll been trying to get in to my Dropbox account but u not letting me login with my password	I reset the password but u still not letting me in my account	3	3
Dropbox	App Store	Google Play	Works great. I love the Camera Uploads feature.	Dropbox. Love the automatic camera uploads	4	3
Spotify	App Store	Twitter	Apple Watch app please!	@SpotifyCares will there be a spotify app on apple watch?	4	2
Slack	App Store	App Store	You had one job. Slacks concept is gold. Their product is really buggy though. Our team misses messages and notifications all the time. For a messaging app, that's just not good.	Very buggy. User experience is not very good and the app is buggy.	2	1
Telegram	Facebook	Twitter	I've found a major bug with the delete system. If the other person tries to delete their message via mobile or PC, it will not delete. It will actually do nothing, and leave that message.	@telegram hi, if i delete a conversation from my end on TELEGRAM, will it be deleted from the other persons device if they are logged in?	1	1
Waze	Facebook	Facebook	When Will you fix problem with updating points!	Why don't you fix problem with updating points??	4	4

TABLE VI: Monolingual, cross-platform results.

Labelled Sample	Baseline	SIMBA$_{\text{STS}}$	SIMBA
ML-SL	0.41	0.78	0.78
ML-L	0.68	0.82	0.81
ML-COMB	0.53	0.79	0.79
ML-COMB-SPOTIFY	0.50	0.82	0.81
ML-COMB-SLACK	0.61	0.82	0.82
ML-COMB-DROPBOX	0.48	0.80	0.80
ML-COMB-TELEGRAM	0.60	0.77	0.76
ML-COMB-WAZE	0.49	0.78	0.78

This result suggests that, as expected, a simple surface overlap metric is not sufficient for the task of identifying similar user feedback.

The approach has the best results with the ML-L sample, which only includes feedback instances with a length of more than five words. Even though two instances of very short user feedback may have a similar meaning, they provide few opportunities for alignments in the semantic text similarity step. For example, for the two instances of rather similar user feedback *"Very good app"* and *"Spotify is excellent"* have a very low similarity score according to our approach due to the lack of alignments (good and excellent is not listed as a lexical paraphrase in the used database, see Section II-C2). Nevertheless, as readily pointed out, the correlation between the sample including shorter feedback (ML-SL) also has a strong positive correlation.

The difference between SIMBA and SIMBA$_{\text{STS}}$ is small and not consistent among the different labelled samples. Besides the semantic text similarity step (containing the word aligner), we also incorporate sentiment analysis and classification steps. However, the results show that the additional steps do not improve the performance of SIMBA significantly. There are only slight variations in the range [-0.01, 0.01] between SIMBA and SIMBA$_{\text{STS}}$, with neither being a clear winner. This result is also reflected in the high average weights for the semantic text similarity coefficient, 0.81, against 0.15 for the sentiment and 0.04 for the category coefficients.

When comparing the results obtained for the application-specific evaluation samples we observe that the approach yields very good results for Slack, Spotify and Dropbox

TABLE VII: Crosslingual collected data.

#English	#Spanish	#French	#German	#Total
19,635	3,433	3,006	2,178	28,252

($>$0.81 correlation for all), whereas Waze and Telegram have slightly lower, albeit comparable, results (0.78 and 0.76 correlation).

Assessing the similarity between a feedback pair is to some extent a subjective task. Therefore, a perfect correlation is unlikely. As performed in previous work [24] we measure the Pearson correlation between the two annotators for each annotated set and consider the results as the upper bound of the system. The inter-annotator correlation for each annotation set is 0.90 for ML-SL and 0.91 for ML-L. In other words, SIMBA and its variation, SIMBA$_{\text{STS}}$ have an average difference of -12% and -9.5% with the upper bound of the human assessment for the ML-LS and ML-L, respectively.

V. Evaluation II: Accuracy in crosslingual data

In our second evaluation, we focus on the similarities between feedback texts written in different languages across a single platform.

A. Dataset

For the crosslingual evaluation we collect user feedback from country specific Apple App Stores, for the same applications as in the monolingual, cross-platform evaluation (see Section IV). We collect user feedback written in English from the US App Store, German from the German App Store, user feedback written in French from the French App Store and user feedback in Spanish from the Mexican App Store. We opt to concentrate our evaluation on these specific languages, as they are popular languages in the area in which the research is conducted[14]. Thus, it is easier to get annotators that are fluent in the languages. We select these country specific stores

[14]German and French are national languages, whereas English and Spanish are offered as part of the high school curricula in Switzerland where the research took place.

as they are the countries with the largest number of available user feedback written in English, French, German and Spanish. We collect the data corresponding to a timeframe of 60 days. Table VII shows the number of collected user feedback for each language. We do not include the other platforms in our evaluation, as it is not straightforward to obtain the region from which the feedback is coming from.

B. Creation of Labelled Dataset

We follow a similar procedure as in our previous evaluation for the creation of the labelled dataset against which we compare our results. We create an annotation guide with examples in all four languages, sample 2,050 feedback pairs to be annotated, and have six annotators analyze and label the sample according to the annotation guide. We detail the main differentiating factors during the creation of the labelled dataset with respect to our previous evaluation below.

1) Sampling Balancing: For our crosslingual evaluation we follow the same approach for sampling the user feedback as in the monolingual evaluation, with a slight difference. As all user feedback is retrieved from the Apple App Store, we do not balance with respect to platform combinations. For each application and similarity level (calculated with the cosine similarity function, as described in Section IV) we sample the same number of user feedback pairs retrieved from the country-specific App Stores. Each user feedback pair contains an instance of user feedback retrieved from the US App Store and one instance retrieved from one of the other App Stores, i.e. the French, German or Mexican. As a result, we sample crosslingual user feedback pairs written in English-French, English-German and English-Spanish.

Our crosslingual data for Slack, Telegram and Dropbox is sparse, totalling less than a 100 feedback entities for each of the non-English languages for all the collection period. Therefore, we only consider Spotify and Waze in our crosslingual evaluation, as we can only create a balanced sample set with the collected data from these two applications. As in the monolingual evaluation (see Section IV), we compensate for missing user feedback pairs in higher similarity levels. We consider the length of the user feedback pairs when sampling in a similar fashion as in our previous evaluation and create two sample sets, i.e. *CL-SL* and *CL-L*. CL-SL samples feedback from all lengths, whereas CL-L only considers feedback pairs with at least 5 words.

2) Sample Annotation: The annotation is conducted by six annotators. Two annotators are in charge of the specific language combinations (e.g., English-French). The annotators are fluent in the languages in which the feedback was written. As in the monolingual evaluation, the feedback text shown to annotators is exactly the one that was collected and the disagreements are handled by annotators through direct discussion. The disagreement average between annotators is 27.67% and is solved through direct discussion among the annotators.

C. Metrics and Finding Weights

We report our results using the same metrics and the same procedure for finding weights as in our the monolingual, cross-

TABLE VIII: Crosslingual results.

Labelled Sample	**Baseline**	**SIMBA$_{STS}$**	**SIMBA**
CL-SL	0.33	0.76	0.76
CL-L	0.43	0.77	0.78
CL-COMB	0.35	0.77	0.78
CL-COMB-SPANISH	0.36	0.77	0.79
CL-COMB-GERMAN	0.34	0.77	0.78
CL-COMB-FRENCH	0.34	0.76	0.76
CL-COMB-SPOTIFY	0.36	0.78	0.79
CL-COMB-WAZE	0.34	0.76	0.77

platform evaluation (see Section IV).

D. Results

As in our monolingual evaluation, we compare the results of our approach, SIMBA, against a variation that only includes the results of the semantic text similarity step, *SIMBA$_{STS}$*. We also compare against a baseline that uses the cosine similarity for the similarity computation.

SIMBA and SIMBA$_{STS}$ greatly outperform the baseline. When comparing both variations, the complete SIMBA approach has a slightly better performance than SIMBA$_{STS}$.

The obtained results for each considered language are comparable and indicate a strong correlation with the manual assessment. While the Spanish evaluation set yields the best results, 0.79, the results for German and French are only slightly lower, 0.78 and 0.76 respectively. Therefore, we can conclude that the approach works well for all considered languages. The results of the two applications are also comparable among each other.

The results between the three different samples (CL-SL, CL-L, CL-COMB) are similar with the sample containing the shorter feedback, CL-SL, performing slightly worse. The reason could be the same as in our previous evaluation: shorter feedback pairs are more difficult to align due to limited context information and words that are common in this type of user feedback (that are usual general praises or complaints) missing in the lexical paraphrase dictionary used by the word aligner step.

The results obtained for the monolingual and crosslingual evaluation sets are comparable. We observe that the performance decreases slightly for the crosslingual evaluation sets. Particularly, there is a performance decrease of 1.42% between ML-COMB and CL-COMB. The decrease between ML-SL and CL-SL is similar, 1.39%, whereas the decrease between ML-L and CL-L is larger, 3.30%. However, in general the performance loss is not rigorous. A possible explanation for this last result is that the translations for longer user feedback may be more ambiguous, hence the performance for this sample decreases.

As in our previous evaluation, we measure the inter-rater correlation for the samples written in different languages. The inter-annotator correlation for each annotation set is 0.90 for the English-Spanish feedback and 0.91 for the English-German and 0.94 for the English-French. That is to say that SIMBA and its variation, SIMBA$_{STS}$ have an average difference of -12%, -13.5% and -18% with the upper bound of the human assessment for the for English-Spanish, English-German and English-French feedback combinations respectively.

VI. Discussion

A. Result Interpretation

The results of the two evaluations assessing SIMBA are promising. The output of SIMBA and its variation SIMBA$_{\text{STS}}$ have an average strong positive correlation of 0.79 on the monolingual feedback available in multiple platforms and of 0.78 on the crosslingual feedback. Similar results have been obtained in competitions for semantic textual similarity tasks, on other types of text e.g., [14], [15], [16], [24]. While these results cannot be compared directly, they are an indicator that our approach performs well. The upper bounds of the evaluation sets, computed on the inter-rater correlation, support this conclusion. SIMBA's performance differs on average by only 12.75% compared to the upper bounds.

Moreover, both variations of SIMBA outperform the baseline with all evaluated sample sets. This results suggests that, as expected, a simple surface overlap metric is not sufficient for the task of identifying similar user feedback on a fine-grained level.

SIMBA works slightly better on longer feedback than on feedback samples including feedback of less that five words. Even though two instances of very short user feedback may have similar meanings, they only provide few opportunities for alignments in the semantic text similarity step. As we assume that product owners, requirements engineers and developers are interested in longer, more informative user feedback, we think that the slightly less effective performance on very short texts is not much of an issue.

Besides the semantic text similarity step, we also incorporated steps for sentiment analysis and classification in SIMBA. However, the results show that the additional steps do not improve or worsen the performance of the approach significantly. A possible explanation for this, could be that the classification model is trained on feedback collected and annotated in previous work [13]. This dataset includes only tweets and they are only for three of the applications included in this work (Waze and Telegram are not included in this training dataset). This could lead to low accuracy in the results of feedback coming from different platforms or referring to different applications. Finally, the lexical sentiment analysis tool which we used is not specialized in user feedback, but has a rather generic dictionary, making some of the sentiment predictions inaccurate.

The results obtained in the monolingual and cross-lingual evaluations are comparable. Also, there is little variation in the results between languages. We can therefore conclude that the approach to translate the user feedback with the help of automatic translation and the computation of the similarity scores on the translated user feedback is promising.

In general, the results between the feedback of the different considered applications are comparable. This result indicates that SIMBA performs well on a diverse set of feedback describing a wide range of functionalities and written from an assuming heterogeneous user base.

B. Potential Applications

Product owners, requirements engineers and developers can use SIMBA's results to detect and measure similarities in feedback written in different languages across different platforms on a fine-grained level. The similarity scales provided by SIMBA allow for the detection of semantical duplicate feedback and feedback that could be mentioning the same feature or bug, albeit with differing levels of detail or concerns. This could help to prioritize feedback, as feedback with a higher number of duplicates could need more urgent addressing. Additionally, SIMBA could help identify inconsistent or contradicting feedback, which by definition falls in the similarity levels *somewhat similar* or *somewhat related but not similar* of our taxonomy. While SIMBA itself does not find the exact contradiction or inconsistency in can help filter the feedback that could be presenting this issue.

To avoid potential scalability issues, SIMBA could be used on feedback that has already been identified as broadly-related by a clustering or topic modeling algorithm. Our approach could be then applied to measure the finer-grained similarity between feedback that has readily been identified as belonging to the same topic or cluster. For instance, a topic or cluster referring to a common feature could contain feedback reporting different bugs on the feature or requesting diverging additional functionalities related to the feature. Further, there could be feedback on that same topic or cluster that reports satisfaction and dissatisfaction with the feature. While these fine differences and similarities are not captured by the clustering and modeling algorithms, they can be detected by SIMBA and they could be used to further differentiate and group the feedback elements within a given topic or cluster.

C. Future Work

We plan to improve some of the individual steps in SIMBA. In particular, we would like to add a module for the correction of mispellings to allow for better word alignment and to add additional lexical databases to the one readily used to allow for the detection of more related or similar words. Slang words, sometimes common in user feedback, are for example, not included in the current used database. We also plan to improve the accuracy of the classifier (now trained only on tweets) by creating models from a more diverse dataset, including feedback from additional platforms and stemming from further applications. Additionally, we plan to improve the lexical sentiment analysis step by using supervised machine learning models trained on data from our domain. Finally, we plan to enhance SIMBA with mechanisms for detecting conflicting feedback, which falls in the *somewhat similar* or *somewhat related but not similar* similarity level.

D. Threats to Validity

The evaluation of SIMBA strongly depends on the quality of the manually annotated set. Annotators might have misconceptions about the similarity levels and could erroneously label the set. To address this threat we created the annotated set involving more than one annotator. Additionally, we provided annotators with a guide containing detailed definitions of each similarity level and examples. This guide was adapted for each of the languages included in the experiment. Moreover, we conducted trials with the annotators to avoid misconceptions and major disagreements.

We mitigated the threat of evaluating our approach against feedback coming from specific applications, platforms or with specific ratings by stratifying our sample on this criteria. We also considered that when sampling pairs of user feedback, the probability that they are unrelated is higher than of them being related. To ensure that feedback with varying degrees of potential similarity are present in the sample, we used the cosine similarity of possible feedback pairs when sampling the data. To assure that the use of cosine similarity had no impact on the sampling of lengthier reviews, we manually extended 25% of the user feedback predicted as highly similar according to the cosine similarity—achieving a distribution of varying lengths in our labelled dataset.

The approach was evaluated on user feedback from five different popular apps, obtained from two app stores and two social networking sites. The considered apps belong to a wide range of domains, and consequently the user feedback was written in different styles and with varying vocabularies. However, we currently cannot generalize our results to other applications or platforms or to feedback written in other different languages. More extensive evaluations would need to be conducted to generalize the results.

VII. Related Work

Groen et al. [25] proposed to elicit feedback from crowds of geographically distributed users. Pagano and Bruegge [9] interviewed practitioners and found that user feedback is essential for software quality and for identifying ideas of improvement. Pagano and Maalej [6], and Hoon [26] were among the first to study user feedback in app stores. They performed exploratory studies and found that user feedback in apps stores contains valuable information for software evolution. In a similar line, other work [8], [13], [27], [28] found that user feedback on Twitter also contains valuable information for software evolution and requirements engineering. In this work we propose an approach that can help detect similarities among relevant user feedback to more easily identify recurring issues in a fine-grained manner and aid in incorporating it into the software lifecycle.

The work on user feedback mining has grown considerably in the past years. Among the most studied platforms for automatically processing user feedback are app stores. Martin et al. [29] described a survey of the most relevant work in the area. Twitter has received some attention in the last years as an additional platform for mining feedback, but to a lower degree. To our best knowledge, there is no work that has focused on mining user feedback from Facebook, even though Seyff et al. [30] previously recommended using the platform for eliciting software requirements. Previous work has proposed approaches for classifying, e.g., [7], [13], [28], [31], [32], grouping, e.g., [10], [11], [12], [13], [33], [34] and prioritizing, e.g., [10], [11], [12] user feedback, as well as for extracting software features mentioned in the feedback [35], [36], [37] and linking it to other artifacts [38].

The work that most closely relates to ours is the one focusing on grouping feedback. This previous work has proposed approaches that use topic modeling (LDA and BTM) and clustering algorithms for grouping semantically related feedback. The main difference with respect to our work is the granularity level at which the analysis is done. Clustering and topic modeling allow for a more coarse grouping, and feedback that is considered in the same group or topic shares a common theme, but not necessarily important details. In contrast, our analysis is finer-grained and allows for the detection of five similarity levels. We believe that clustering and modeling could also be used as a preprocessing step in which only the feedback in a specific cluster or topic is analyzed with SIMBA, for further scalability.

While Villarroel et al. [11] identified feedback written in multiple languages as an open challenge for automatically analyzing user feedback. There is, to our best knowledge, few research results in the domain. The only work that we are aware of in this area is the study conducted by Stanik et al. [39] in which they compared deep learning and traditional supervised machine learning approaches for classifying tweets and user feedback available in app stores written in English and Italian. In contrast, we focus on the detection of feedback similarities and conduct our evaluation on four different languages and include Facebook as an additional feedback platform in our evaluation.

VIII. Conclusions

We present SIMBA🐯, an approach to detect and measure the similarity between feedback pairs written in different languages and platforms on a fine-grained level. The approach uses a combination of automatic machine translation, natural language processing techniques, lexical sentiment analysis and supervised machine learning. We evaluated SIMBA against 4,100 manually annotated feedback pairs. Our results show that SIMBA strongly correlates to human judgement, with an average correlation of 0.79 for cross-platform user feedback written in English and 0.78 for feedback written in four different languages and stemming from a single platform. SIMBA could be used by requirements engineers, developers and product owners to detect semantically duplicate feedback on a fine-grained level and to filter feedback that could contain inconsistencies or contradicting stances with respect to a requirement or software malfunctioning.

Acknowledgments

We thank Martin Glinz for the fruitful discussions during the project that lead to SIMBA. We also thank Roberto Verdecchia for his feedback on a previous version of this paper. This work was partially supported by the European Commission and the Swiss State Secretariat for Education, Research and Innovation (SERI) within the SUPERSEDE project (ID 644018).

References

[1] S. Kujala, M. Kauppinen, L. Lehtola, and T. Kojo, "The role of user involvement in requirements quality and project success," in *Proc. of International Conference on Requirements Engineering*, 2005, pp. 75–84.

[2] S. Kujala, "User involvement: a review of the benefits and challenges," *Behaviour & information technology*, vol. 22, no. 1, pp. 1–16, 2003.

[3] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip, and C. Tran, "A case study of post-deployment user feedback triage," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering*, 2011, pp. 1–8.

[4] M. Janneck, "Challenges of software recontextualization: lessons learned," in *Extended Abstracts on Human Factors in Computing Systems (CHI)*, 2010, pp. 4613–4628.

[5] G. Lindgaard and J. Chattratichart, "Usability testing: what have we overlooked?" in *Proc. of the Conference on Human Factors in Computing Systems*, 2007, pp. 1415–1424.

[6] D. Pagano and W. Maalej, "User feedback in the appstore: an empirical study," in *Proc. of the International Requirements Engineering Conference*, 2013, pp. 125–134.

[7] E. Guzman, M. El-Halaby, and B. Bruegge, "Ensemble Methods for App Review Classification: An Approach for Software Evolution," in *Proc. of the Automated Software Enginering Conference*, 2015, pp. 771–776.

[8] E. Guzman, R. Alkadhi, and N. Seyff, "A Needle in a Haystack: What Do Twitter Users Say about Software?" in *Proc. of the International Requirements Engineering Conference*, 2016, pp. 96–105.

[9] D. Pagano and B. Bruegge, "User Involvement in Software Evolution Practice : A Case Study," in *Proc. of the International Conference on Software Engineering*, 2013, pp. 953–962.

[10] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace," in *Proc. of the International Conference on Software Engineering*, 2014, pp. 767—-778.

[11] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proc. of the International Conference on Software Engineering*, 2016, pp. 14–24.

[12] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proc. of the International Symposium on Foundations of Software Engineering*, 2016, pp. 499–510.

[13] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *Proc. of the International Requirements Engineering Conference (RE)*, 2017, pp. 11–20.

[14] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe, "Semeval-2014 task 10: Multilingual semantic textual similarity," in *Proc. of the International Workshop on Semantic Evaluation (SemEval)*, 2014, pp. 81–91.

[15] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea *et al.*, "Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability," in *Proc. of the International Workshop on Semantic Evaluation (SemEval)*, 2015, pp. 252–263.

[16] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez Agirre, R. Mihalcea, G. Rigau Claramunt, and J. Wiebe, "Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation," in *Proc. of the International Workshop on Semantic Evaluation (SemEval)*, 2016, pp. 497–511.

[17] C. Banea, R. M. Di Chen, C. Cardie, and J. Wiebe, "Simcompass: Using deep learning word embeddings to assess cross-level similarity," 2014.

[18] M. A. Sultan, S. Bethard, and T. Sumner, "Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 219–230, 2014.

[19] C. Bannard and C. Callison-Burch, "Paraphrasing with bilingual parallel corpora," in *Proc. of the Annual Meeting on Association for Computational Linguistics*, 2005, pp. 597–604.

[20] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. of the Conference of the North American chapter of the Association for Computational Linguistics on Human Language Technology*, 2003, pp. 173–180.

[21] T. Brychcín and L. Svoboda, "Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information," in *Proc. of the International Workshop on Semantic Evaluation (SemEval)*, 2016, pp. 588–594.

[22] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[23] K. Neuendorf, *The content analysis guidebook*. Thousand Oaks, CA: Sage Publications, 2002.

[24] D. Jurgens, M. T. Pilehvar, and R. Navigli, "Semeval-2014 task 3: Cross-level semantic similarity," in *Proc. of the International Workshop on Semantic Evaluation (SemEval)*, 2014, pp. 17–26.

[25] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini *et al.*, "The crowd in requirements engineering: The landscape and challenges," *IEEE software*, vol. 34, no. 2, pp. 44–52, 2017.

[26] L. Hoon, R. Vasa, J.-G. Schneider, J. Grundy, and Others, "An analysis of the mobile app review landscape: trends and implications," *Swinburne University of Technology, Tech. Rep*, 2013.

[27] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2764–2794, 2018.

[28] G. Williams and A. Mahmoud, "Mining twitter feeds for software user requirements," in *Proc. of the International Requirements Engineering Conference (RE)*, 2017, pp. 1–10.

[29] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE transactions on software engineering*, vol. 43, no. 9, pp. 817–847, 2016.

[30] N. Seyff, I. Todoran, K. Caluser, L. Singer, and M. Glinz, "Using popular social network sites to support requirements elicitation, prioritization and negotiation," *Journal of Internet Services and Applications*, vol. 6, no. 1, p. 7, 2015.

[31] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. of the International Requirements Engineering Conference*, 2015, pp. 116–125.

[32] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall, "How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution," in *Proc. of the International Conference on Software Maintenance and Evolution*, 2015, pp. 281 – 290.

[33] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in *Proc. of the International Conference on Software Engineering*, 2013, pp. 582–591.

[34] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proc. of the Working Conference on Mining Software Repositories*, 2013, pp. 41–44.

[35] E. Guzman and W. Maalej, "How do Users like this Feature? A fine grained Sentiment Analysis of App Reviews," in *Proc. of the International Conference on Requirements Engineering*, 2014, pp. 153–162.

[36] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach (t)," in *Proc. of the International Conference on Automated Software Engineering (ASE)*, 2015, pp. 749–759.

[37] X. Gu and S. Kim, ""what parts of your apps are loved by users?"," in *Proc. of the International Conference on Automated Software Engineering (ASE)*, 2015, pp. 760–770.

[38] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proc. of the International Conference on Software Engineering (ICSE)*, 2017, pp. 106–117.

[39] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *Proc. of the International Requirements Engineering Conference Workshops (REW)*, 2019, pp. 220–226.