Simulating the Impact of Annotation Guidelines and Annotated Data on Extracting App Features from App Reviews

Faiz Ali Shah, Kairit Sirts and Dietmar Pfahl
Institute of Computer Science, University of Tartu, J. Liivi 2, 50409, Tartu, Estonia

Keywords: App Feature Extraction, Supervised Machine Learning, Annotation Guidelines, Requirements Engineering.

Abstract: The quality of automatic app feature extraction from app reviews depends on various aspects, e.g. the feature extraction method, training and evaluation datasets, evaluation method etc. Annotation guidelines used to guide the annotation of training and evaluation datasets can have a considerable impact to the quality of the whole system but it is one of the aspects that is often overlooked. We conducted a study in which we explore the effects of annotation guidelines to the quality of app feature extraction. We propose several changes to the existing annotation guidelines with the goal of making the extracted app features more useful to app developers. We test the proposed changes via simulating the application of the new annotation guidelines and evaluating the performance of the supervised machine learning models trained on datasets annotated with initial and simulated annotation guidelines. While the overall performance of automatic app feature extraction remains the same as compared to the model trained on the dataset with initial annotations, the features extracted by the model trained on the dataset with simulated new annotations are less noisy and more informative to app developers.

1 INTRODUCTION

App marketplaces provide app users a channel to submit feedback in the form of a review. Users in these reviews provide valuable information such as feature requests, bug reports, user experience, and evaluation of app features (Pagano and Maalej, 2013). The analysis of opinions expressed about different features of an app in user reviews offers opportunities and insights to both app users and app developers. For app developers, it is useful to monitor the "health" of app features in the context of release planning (Maalej et al., 2016) as well as to evaluate product competitiveness and quality (Shah et al., 2016). From the users' perspective, such information helps decide which app to select from a wide range of competing apps. Both Apple's App Store and Google's Play Store receive enormous amounts of reviews every day rendering a manual analysis infeasible and demanding automated methods. One standard approach towards automation is to generate sentiment summaries of a product at feature-level involving two steps (Zhang and Liu, 2014): 1) identification of app features (also called *aspect terms* or *opinion targets* in the opinion mining literature) in user reviews, and 2) determination and aggregation of sentiments expressed on product features identified in the previous step.

Since identifying product features in user reviews is a crucial step for generating sentiment summaries, several prior studies on app review analysis have exclusively focused on this step. Previous work on aspect extraction from app reviews includes using topic modeling (Guzman and Maalej, 2014) and an approach called SAFE (Simple Approach for Feature Extraction) based on manually extracted rules (Johann et al., 2017). The performance reported with these two approaches is summarized in the first two rows of Table 1. Although these results seem to indicate that the topic modeling and the SAFE approach are complementary with regards to precision and recall, they are not directly comparable because the authors used different sets of app reviews, different annotation guidelines (AGs) and different performance evaluation methods. Johann et al. (2017) compared the performance of a version of the topic modeling approach proposed by Guzman et al. (2014) on their set of app reviews using the same AGs and evaluation procedure as in the SAFE approach. These results, shown in the 3rd row of Table 1, indicate that the performance of the topic modeling approach is much lower than reported in the original paper and

Shah, F., Sirts, K. and Pfahl, D.
Simulating the Impact of Annotation Guidelines and Annotated Data on Extracting App Features from App Reviews.
DOI: 10.5220/0007909703840396
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 384-396
ISBN: 978-989-758-379-7

Table 1: Performance obtained with different approaches to extract features from app reviews.

Ref	Approach	Prec	Rec	F1
Guzman et al. (2014)	Topic modeling (1)	0.58	0.52	0.55
Johann et al. (2017)	SAFE	0.24	0.71	0.36
Johann et al. (2017)	Topic modeling (2)	0.22	0.28	0.24
Sanger et al. (2016)	CRF	0.69	0.56	0.62

also worse than that of the SAFE approach, in particular with regards to recall. Another recent study for extracting features from app reviews, conducted on German language reviews (Sänger et al., 2016), used a supervised machine learning method of Conditional Random Fields (CRF) (Lafferty et al., 2001). Previously, CRF method has been successfully applied to extract features from LAPTOP and RESTAURANT reviews (Pontiki et al., 2016; Liu et al., 2015), which serve as standard aspect extraction benchmark datasets in sentiment analysis community. The results of Sanger et al. (2016), shown in the last row of Table 1, suggest that supervised machine learning methods might have an advantage over unsupervised and rule-based methods (such as SAFE) also for extracting features from English app reviews, given that annotated training data is available.

There are several questions previous studies fail to answer when developing systems for automatically extracting features from app reviews. An important question is related to the annotation of app features: Which word or sequence of words in a review constitutes an app feature? Training a feature extraction system using supervised machine learning methods requires a training set where all feature instances are annotated[1]. Unsupervised or rule-based systems do not need an annotated training set but they need an annotated test set to evaluate how well the system performs. Clearly, the exact annotation procedure, operationalized via AGs, potentially affects both the evaluation results and the usefulness of those results to app developers. This motivates our research question:

RQ: To what extent is automatic feature extraction from app reviews sensitive to the used AGs?

To study this research question, we use two available app review datasets that are both annotated with app features using different AGs: 1) English app review data contributed by Guzman et al. (2014), which includes annotated reviews of seven apps from App Store and Play Store (GUZMAN dataset) and 2) German app review data published by Sanger et al. (2016), which contains annotated reviews from eleven app categories of Play Store (SANGER dataset).[2]

Furthermore, we employed two undergraduate students as annotators. Each of them annotated independently 500 reviews of the seven apps contained in the GUZMAN dataset[3] following the AGs proposed by Sanger et al. (2016). Because the inter-annotator agreement between the two annotators on the newly annotated dataset is low (Dice index = 0.28), we treat the annotations of the two annotators as two different datasets.

We train and evaluate supervised CRF models on all datasets, i.e., the GUZMAN dataset, the SANGER dataset, and the datasets annotated by our student annotators using cross-category validation (CCV) settings. The CCV training regime assumes that reviews of different app categories share enough common information that a model trained on the reviews belonging to one set of apps or app categories will generalize to the apps or app categories whose reviews the model has not seen during training. In this training regime, we select to hold out one app category and train the model on the app reviews of all other categories. Finally, we test on the app reviews of the held-out app category. This procedure is repeated until all categories have been held out in turn. Then we use the average feature extraction performance of these models as a proxy for evaluating the goodness of the AGs. We simulate several changes in AGs and assess their effect using the performance of the predictive CRF modeling. Using this procedure, we are able to propose several changes to the app feature AGs that improve the quality of the annotated app reviews for both training and evaluation purposes.

2 RELATED WORK

Several methods for extracting app features from app reviews have been proposed. Guzman et al. (2014) used unsupervised LDA topic modeling (Blei et al., 2003) for automatic extraction of app features from user reviews of seven apps (three from App Store and four from Play Store). The extracted app features were evaluated against human labeled app features. The study of Gu et al. (2015) classifies review sentences into categories, such as feature evaluation, praise, feature requests, bug reports and others, and then extracts app features using 26 manually designed rules only from those sentences that belong to the feature evaluation category. Johann et al. (2017)

[1]Note that we use the words *annotated* and *labeled* as synonyms in this article.

[2]The particular apps from which the reviews are taken are not known.

[3]Reviews annotated by our students may differ from those in the GUZMAN dataset.

proposed a rule-based approach SAFE that uses Part-of-Speech (POS) and sentence patterns for extracting app features from app descriptions and user reviews. Malik et al. (Malik et al., 2018) uses syntactic relations between the features and opinion words for the extraction of app features from user reviews.

A few studies focus on summarizing reviews that are informative for developers using topic models (Guzman et al., 2015; Panichella et al., 2015; Luiz et al., 2018). Recently, supervised machine learning approaches have been used to classify app reviews into functional and non-functional requirements (Lu and Liang, 2017; Kurtanović and Maalej, 2017).

Instead of trying to directly extract app features, the study of (Vu et al., 2015) extract all potential keywords from user reviews and rank them based on the review rating and occurrence frequency. Their approach also has the ability to cluster or expand the set of extracted keywords based on their semantic similarity. The study of (Keertipati et al., 2016) extracted nouns as candidate app features from the app review sentences but they did not perform an evaluation to check whether the extracted features actually represent true app features.

All of the aforementioned studies related to app feature extraction use different techniques, review datasets and AGs; therefore, the results reported in these studies are not directly comparable to each other. Additionally, without having access to different datasets and AGs, it is difficult to assess the quality of the annotations that were used to evaluate the systems. We were able to obtain the evaluation set of Guzman et al. (2014) together with its AGs and we use this dataset as one of the annotated experimental training sets (GUZMAN dataset) in our study.

In the literature, much research (Kang and Zhou, 2017) is dedicated to automatic extraction of features from product reviews in the LAPTOP and RESTAURANT domains. The best results have been achieved using supervised learning approach such as Conditional Random Fields (CRF) and Convolutional Neural Network (CNN) (Pontiki et al., 2016; Poria et al., 2016). We know of only one study that used supervised sequence tagging model (i.e., CRF) for automatic extraction of app features from app reviews has been performed by Sanger et al. (2016) on German app reviews.

3 STUDY DESIGN

We study our Research Question *RQ* by evaluating the results of several simulation experiments on app review data with human-annotated app features. These experiments include several variables, some of which are fixed and some are systematically altered to study the Research Questions. Table 2 summarizes all design variables.

Table 2: Summary of design variables.

GIVEN DESIGN VARIABLES
1. Annotation guidelines
a) SANGER annotation guidelines (German and English)
b) GUZMAN annotation guidelines (English)
2. Annotated datasets
1) SANGER dataset (Sänger et al., 2016), annotated using SANGER guidelines
2) SHAH dataset: annotated by two annotators using translated SANGER guidelines.
a) SHAH-I: SHAH dataset labeled by annotator I.
b) SHAH-II: SHAH dataset labeled by annotator II.
3) GUZMAN dataset (Guzman and Maalej, 2014), annotated using GUZMAN guidelines
3. Modeling approach: CRF model with the following features extracted from the current word and its context of two preceding and two following words:
a) the words themselves
b) POS of the words in the sentence
c) one to four character prefixes and suffixes of the words
d) the position of the words
e) the stylistics of each word (e.g. case, digit, symbol, alphanumeric)
4. Evaluation methods
a) Type-based exact match
b) Type-based partial match
5. Training procedure
• CCV: **C**ross-**C**ategory **V**alidation on the full dataset
MANIPULATED DESIGN VARIABLES
6. Data processing flow
Step 1 (Pre-processing): remove non-consecutive app features, remove app reviews with no annotated features
Step 2 (Simulation step I): Remove *pseudo*-features
Step 3 (Simulation step II): Remove app features that do not contain a NOUN
Step 4 (Simulation step III): Remove app features that are longer than three words

To study Research Question *RQ* we use all Given Design Variables 1 to 4 and adopt CCV as training procedure to explore the effects of the Data Processing steps simulating the changes in the AGs. We assume that the quality of the training data annotations and the accuracy of the app feature extraction model are positively correlated. Thus, we use model accuracy on the test set to assess the impact of a change in the AGs. We train and evaluate CRF-based app feature extraction models on all our annotated datasets after each Data Processing steps and assess their accuracies to approximate how each step affects the quality of the annotations.

In the following, we describe all design variables in detail.

3.1 Annotation Guidelines

Our experimental datasets (described more thoroughly in the next subsection) are annotated using two distinct set of AGs, i.e., GUZMAN and SANGER AGs.

The GUZMAN AGs[4] were developed by Guzman et al. (2014) to annotate their evaluation set. These AGs define an app feature as a description of specific app functionality visible to the user (such as *uploading files* or *sending emails*), a specific screen of the app, a general quality of the app (such as *time needed to load* or *size of storage*) or a specific technical characteristic (e.g. a network protocol or HTML5). The AGs encourage to annotate the exact words used in the text but do not enforce it. The guidelines explicitly allow for annotating app features consisting of non-consecutive words.

The SANGER AGs[5] were developed by Sanger et al. (2016) to annotate app features, subjective phrases, and relationships between them. We translated these guidelines from German into English. SANGER AGs define as an app feature "anything that is part of the application or in some form connected with the app". This includes existing and requested app features, bugs and errors as well as entities referring to non-functional features such as usability, design, price, license, permissions, advertisements and updates. The guidelines explicitly instruct to annotate the mentions of the app itself as a feature. Instructions also ask to annotate implicit features represented by a single verb such as *runs*. Annotators are encouraged to keep the annotated features as short as possible although a particular length limit is not set. The SANGER guidelines specifically require not to include function words into annotated app features, which probably also influences the length of the annotated app features. Although no explicit mention about annotating consecutive vs non-consecutive words as features is made, all example features only consist of consecutive words.

Although both AGs can be used to label the same information—features in app reviews—they have differences which may influence how well the data annotated with these guidelines can be used to train a model for automatic feature extraction.

1) Using feature annotations not comprised of exact words used in the review text will make any automatic use of these annotations very difficult. Although this practice is discouraged in the GUZMAN guidelines, it is not explicitly prohibited.
2) Annotating non-consecutive app features, allowed in GUZMAN guidelines, restricts the types of models that can be used. In particular, sequence tagging models like Conditional Random Fields and Recurrent Neural Networks, which produce state-of-the-art results on the feature extraction task (Liu et al., 2015), can only process features consisting of consecutive words.
3) The instruction in SANGER guidelines to annotate mentions and references to the app itself is most probably motivated by the particular task of Sanger et al. (2016) to learn to extract both app features and their relations to subjective phrases. In the context of plain app feature extraction these features can be considered *pseudo*-features, as they not give any useful information to the app developers.
4) Similarly, the instruction in SANGER guidelines to annotate standalone abstract verbs such as *runs* is probably motivated by the joint task of learning both app features and subjective phrases. In the context of app feature extraction these aspects will likely cause problems because they are difficult to distinguish from other generic verbs not labeled as app features. Also, these very generic app features are likely of very little value to the developers.

3.2 Annotated Datasets

We have at our disposal four annotated app review datasets: GUZMAN dataset, SANGER dataset and two versions of SHAH datasets. We present some characteristics of these datasets in Table A of Appendix[6]. Note that we do not show data per individual app but aggregated per app category. Each review dataset is characterized using the following information:

a) the total number of reviews;
b) the total number of sentences in all reviews;
c) the total number of annotated app features in (tokens);
d) the number of distinct app features (types);
e) the number of app features consisting of a single word only;
f) the number of app features consisting of at least two words;
g) the type-token ratio of annotated app features (the number of feature types divided by the number of feature tokens).

The GUZMAN dataset[7] was used as an evaluation set in the study performed by Guzman et al. (2014).

[4] https://mast.informatik.uni-hamburg.de/wp-content/uploads/2014/03/Coding_Guide.pdf

[5] Available from http://www.romanklinger.de/scare/

[6] https://figshare.com/s/2bad5b1507ac0f6a8a43

[7] The dataset was obtained from the authors of Guzman et al. (2014)

It contains annotated app reviews in English language from six different app categories. Most app categories contain reviews from one app only: AngryBirds from Games category, TripAdvisor from Travel category, PicsArt from Photography category, Pinterest from Social category and Whatsapp from Communication category. The only exception is the Productivity category which contains reviews from two apps: Dropbox and Evernote. According to Guzman et al. (2014), 400 reviews were annotated for each app. However, from Table it can be seen that the GUZMAN dataset includes less than 400 reviews in each category. This is because the GUZMAN dataset[8] only contains reviews with at least one annotated app feature.

The SANGER dataset[9] was developed and used by Sanger et al. (2016). It contains reviews in German language. The SANGER dataset has the same number of reviews in each app category. The reviews in each category come from 10 to 15 different apps but the origin of each particular review is unknown to us. In addition to app features, this dataset is also annotated with subjective phrases and relations between features and subjective phrases. However, we only use the annotated app features and ignore all other annotations.

In addition to the two datasets available from other researchers, we created the SHAH dataset. The app reviews included in this dataset were selected by randomly sampling 500 reviews per app from the total pool of reviews assembled by Guzman et al. (2014) for their study. The app categories and apps in each category are the same as in the GUZMAN dataset. Similar to Guzman et al. (2014), since each app has its own user rating distribution, we used a stratified sampling procedure to sample the reviews using the distribution over ratings as stratum. For measuring the inter-annotator agreement, we adopted the Dice coefficient (Pavlopoulos and Androutsopoulos, 2014), which ranges between 0 and 1 where 1 means total agreement and 0 total disagreement. The Dice coefficient value between the two annotators was 0.28 which denotes a low agreement between the annotators. Because of that, we decided to treat the annotations of both annotators as different datasets resulting in two annotated SHAH datasets: SHAH-I and SHAH-II containing the annotations of the first and the second annotator respectively.

Based on the statistics, several differences between the datasets are visible. Firstly, the GUZMAN dataset differs strongly from the other datasets by having multi-word features twice as often as single-word features. In contrast, the SHAH-II and SANGER datasets have more single-word features than multi-word features and in the SHAH-I dataset the numbers are balanced. Several reasons may account for these differences, including how each annotator interpreted the AGs given to them, but we believe that this difference might also characterize the differences in the AGs themselves.

The second main difference manifests itself in the average number of app features per review. This quantity is largest for the GUZMAN dataset and smallest for both SHAH datasets with the SANGER dataset falling in-between. We attribute this difference to fact that the GUZMAN dataset only consists of reviews that contain at least one annotated app feature while the other datasets may also contain reviews without a single annotated app feature.

Thirdly, also the type-token ratio of app features is largest for the GUZMAN dataset indicating that the proportion of distinct app features is largest in this dataset. This can be explained by the large number of multi-word app features: the longer the features the more likely they consist of a unique sequence of words.

3.3 Modeling Approach

We adopt the Conditional Random Field (CRF) (Lafferty et al., 2001), a supervised learning method to train the models for all our experiments. CRF is a sequence tagging model which tags each word in the app review text with a label. Similarly to Sanger et al. (2016) we use the BIO labeling scheme, where the tag **B** is used to annotate the first word of each app feature, **I** labels the rest of the words *inside* the app feature and the label **O** is used to tag all words that are *outside* of the app feature. We use an implementation based on CRFSuite[10] which was used as a CRF baseline by Liu et al. (2015) on LAPTOP and RESTAURANT product review datasets. The hand-crafted features used in the CRF model are the same as used by Liu et al. (2015), they are summarized in Table 2.

The study by Liu et al. (2015) showed that using word embeddings (Mikolov et al., 2013) as additional features improves CRF model performance. Therefore, we included word embeddings as features in our experiments. For the datasets in English language (GUZMAN and SHAH datasets), we used SENNA embeddings (Collobert et al., 2011) [11]. For the SANGER dataset in German language, we used the embeddings[12] trained on Wikipedia articles.

[8]From the dataset we obtained from the authors of Guzman et al. (2014) the reviews without any annotated app features had been filtered out.

[9]Available from http://www.romanklinger.de/scare/

[10]https://github.com/pdsujnow/opinion-target

[11]https://ronan.collobert.com/senna/

[12]https://spinningbytes.com/resources/word-

3.4 Evaluation Procedures

We evaluate all results by computing precision, recall and F1-score of the predicted app features. Since app feature annotations themselves may be noisy and ambiguous, which manifests itself in low agreement between annotators (for instance (Guzman and Maalej, 2014) reported an agreement of 53% on annotated app features), we adopt type-based evaluation methods using both exact and partial matching between predicted and human-annotated features (see Part 4 in Table 2).

Exact match requires the predicted and annotated app features to match exactly. For instance, if the annotated feature is *to upload video* then in order to count a match the predicted app feature must consist of exactly the same words. If the model predicts *upload video* as feature leaving the particle *to* untagged the prediction is counted as false positive under the exact match scheme.

Partial match allows a mismatch when comparing predicted app features with human-annotated features in the evaluation set. We allow a difference of one word. Under the partial match scheme, the predicted feature *upload video* will be counted as true positive even if the human-annotated feature is *to upload video*, whereas the predicted feature *video* would be a false positive because it differs from the human-annotated feature by more than one word. Similarly, a predicted feature *failed to upload video* would be counted as true positive under partial match but an even longer predicted feature like *failed to upload video to* will be counted as false positive.

Type-based evaluation counts and evaluates each app feature type only once, regardless of how many times it occurs in the review texts. In order to cluster together different instances of the same app feature type, the features are first lemmatized using *Snowball*[13] stemmer available in NLTK library and then matched based on their lemmas. The type-based evaluation procedure is unbiased by the frequencies of the single app feature types. While the token-based evaluation measures can become artificially high when the annotated training and test set contain a single high-frequency simple one-word app feature, the type-based evaluation gives equal credit to all different app features, regardless of their frequency.

3.5 Data Processing

The data processing steps adopted in our study comprise one pre-processing step and three steps for simulating the changes in AGs.

The *Pre-processing Step* is necessary to unify all experimental annotated datasets to bring them to similar starting point. First of all, GUZMAN dataset can also include annotations of non-consecutive app features. Because the CRF model can only learn app features consisting of consecutive words, we remove all non-consecutive app features from GUZMAN dataset because leaving them in would put the GUZMAN datasets and GUZMAN AGs into a disadvantaged position compared to the SANGER and SHAH datasets annotated with SANGER guidelines where annotated app features always consist of only consecutive words. After that, we remove all reviews from the datasets that do not contain any annotated app feature. We do this because the annotated GUZMAN dataset we obtained from the authors of (Guzman and Maalej, 2014) is a subset of all the reviews originally annotated by Guzman et al. (2014)—the annotated reviews containing no app features were left out. Although removing such reviews biases the datasets, it makes the SANGER and SHAH datasets comparable to the GUZMAN dataset in terms of app feature distribution over reviews.

Simulation Step I removes all annotated features not referring to app functional or non-functional aspects but only to app itself either by the app name or by explicitly using the words such as *app* or *application* and other similar pseudo-features. This step simulates the change in the SANGER AGs such that the command to annotate the references to the app itself are removed. Because GUZMAN AGs do not require to annotate such pseudo-features this simulation step only changes the annotations of the SANGER and SHAH datasets. After this step, the reviews without any annotated app features are removed again to ensure that the feature distributions over all datasets are similar.

Simulation Step II removes all app features that do not contain a noun. Previous studies (Zamani et al., 2014; Keertipati et al., 2016) use this simple heuristic to identify app features from user reviews. This simulation step ensures that annotated app feature should be specific enough and this can only be achieved by requiring the presence of a noun phrase. For instance, an app feature such as *to upload*, which consists of a particle and a verb and does not include a noun, is too non-specific to understand what kind to functionality the feature refers to. Thus, after this simulation step, these kinds of word sequences are not considered as app features anymore, whereas a similar word sequence *to upload video*, which specifies the action with a noun, will be kept. This simulation step mostly removes short generic app features annotated accord-

embeddings/

[13]http://www.nltk.org/_modules/nltk/stem/snowball.html

ing to SANGER guidelines.

Simulation Step III removes all app features that are longer than three words. We believe that useful features cannot be too long because otherwise they become too specific and noisy. We attempt to simulate the change in AGs that would limit the maximum length of an app feature to three words with a very crude heuristic that just removes the longer features from the dataset. Although a better heuristic would be to develop a set of rules to shorten the app features appropriately we opted here for the simplest strategy, believing that it will be good enough for our purpose of testing the potential effect of such a guideline.

3.6 Training Procedure

We use a cross-category validation (CCV) training procedure in our study. The CCV training procedure assumes that the annotated training data consists of app reviews belonging to several different app categories. Then the reviews of each app category are held out in turn and the model is trained on the reviews in the other app categories. Finally the trained model is evaluated on the held-out reviews. We use cross-category validation instead of cross-app validation because both in GUZMAN and SHAH datasets, with one exception we have the reviews of just one app in each of the app categories. In SANGER dataset, although each category contains reviews from several apps, we do not have the app name annotations attached to each review and thus we could not separate the reviews of different apps into different subsets.

4 RESULTS

In this section, we present the results of our study and answer the research question (RQ). For better readability, we only show aggregated results for each dataset in Table 3. Detailed results at app category level can be found in the Appendix[6] (Tables B to E).

Before we describe the contents of Table 3, we summarize characteristics of the labeled datasets used in our study before and after each processing step. The exact numbers can be found in Figure A of Appendix[6]. The annotated datasets at the Baseline, i.e., before applying Step 1 (Pre-processing) correspond to those described in Section 3.2. Several phenomena can be observed when comparing the evolution of dataset characteristics from before Step 1 to after Step 4. Due to the nature of the data processing steps, the numbers of app features steadily decrease in all datasets, both token-wise and type-wise, and several of the characteristics of the four datasets converge. For example, at the Baseline, the type-token ratio of app features varies in the range [0.31, 0.75], while after Step 4, the variation is reduced to the range [0.69, 0.79]. In other words, in Step 4, most of the feature instances occur only once or twice in each of the datasets. Similarly, the average number of features per review, which initially varies in the range [0.31, 1.80], reduces to a range of [0.31, 1.06] after the Step 4. For the GUZMAN, SHAH-I, and SHAH-II datasets the portion of single-word features converges from variation in range [0.32, 0.71] to variation in range [0.31, 0.37]. Only for the SANGER dataset, the portion of single-word features stays high (with a small reduction from 0.84 to 0.76). One explanation for the high portion of single-word features in the SANGER dataset could be that the German language allows for noun compositions replacing multiple-word noun phrases.

Table 3: Model performance after data processing.

Processing Step	Exact Types Prec	Exact Types Rec	Exact Types F1	Partial Types Prec	Partial Types Rec	Partial Types F1
a) GUZMAN dataset:						
Step 1: Pre-processing	45.5	14.6	21.4	68.9	26.2	37.2
Step 2: Simulation I	44.5	14.0	20.7	68.1	25.4	36.1
Step 3: Simulation II	43.3	13.0	19.1	66.5	23.2	33.1
Step 4: Simulation III-3	48.0	15.4	22.5	74.9	29.9	41.8
b) SHAH-I dataset:						
Step 1: Pre-processing	60.1	27.6	37.4	80.0	42.3	55.0
Step 2: Simulation I	50.5	13.2	20.5	69.5	22.5	33.6
Step 3: Simulation II	58.6	12.9	20.9	78.8	22.1	34.1
Step 4: Simulation III-3	60.2	15.2	24.1	82.0	25.8	39.0
c) SHAH-II dataset:						
Step 1: Pre-processing	46.0	14.3	21.4	64.2	21.3	31.5
Step 2: Simulation I	48.9	12.4	18.8	68.2	17.2	25.9
Step 3: Simulation II	40.4	13.2	19.2	56.6	17.7	25.9
Step 4: Simulation III-3	57.8	14.2	22.3	68.0	18.5	28.4
d) SANGER dataset:						
Step 1: Pre-processing	63.5	38.9	47.7	74.7	49.6	59.2
Step 2: Simulation I	60.8	32.1	41.4	71.8	40.7	51.3
Step 3: Simulation II	54.8	32.0	40.0	66.5	40.9	50.3
Step 4: Simulation III-3	57.2	30.6	39.5	68.5	39.0	49.3

In the following, we highlight interesting results shown in Table 3. For each experiment, the table shows precision, recall and F1-measure when applying the evaluation procedures EXACT MATCH (TYPE), and PARTIAL MATCH (TYPE) as described in Section 3.4. The first row in each section of Table 3 (Step 1: Pre-processing) shows the performance after filtering out non-consecutive app features and removing reviews that do not mention app features. For all datasets precision is consistently better than recall for both evaluation procedures. Partial matching, as expected, yields better performance than exact matching. The performance varies largely between datasets. The models built using the GUZMAN and SHAH-II datasets clearly perform worse than those built using

the SHAH-I and SANGER datasets. When looking at the dataset characteristics, one sees the following similarities between the datasets on which the models perform better as compared to the datasets where the models perform worse:

- The SHAH-I and SANGER datasets have a larger share of single-word app features (71% and 84%) than the GUZMAN and SHAH-II datasets (36% and 49%).
- The SHAH-I and SANGER datasets have a lower type-token ratio (0.31 and 0.52) than the GUZMAN and SHAH-II datasets (0.74 and 0.62).

After Step 1, it seems that the language used in the review datasets (German in the case of the SANGER dataset and English in the case of the SHAH-I dataset) does not have a distinguishing impact on model performance. The impact of the two AGs seems to be mixed.

Even though we used the translated SANGER AGs when annotating the SHAH dataset, the performances of SHAH-I-based and SHAH-II-based models are very different. The performance of the SHAH-II-based model is even worse than the performance of the GUZMAN-based model where a different AG was used. We speculated that one possible reason for the difference in performance could be that the SANGER AGs explicitly instruct annotating references to the app itself as a feature. Following this instruction automatically increases the number of single-word features and lowers the type-token ratio as the repeated mentioning of the app itself increases the token count but not the type count. When inspecting the SHAH-II dataset, we noticed that the annotator seemed to have ignored this instruction. Since the frequent annotation of references to the app itself in a review seems to artificially boost the performance of the feature extraction models, while it does not have any practical value to correctly predict the occurrence of a feature referring to the app itself, we decided to remove the annotations of app-references in our datasets.

The second row in each section of Table 3 (Step 2: Simulation I) shows the performance after filtering out app features referring to the app itself. Applying Step 2 simulates a change in the AGs, i.e., the explicit mentioning that references to the app itself should not be annotated. As can be seen in Figure A in the Appendix[6], in Step 2, all datasets have a high type-token ratio in the range [0.68, 0.78]. All English datasets have a low share of single-word features in the range [35%, 38%], while the one German dataset (SANGER) still has a relatively large portion of single-word features (77%). As expected, all results on datasets with previously high performance (SHAH-I and SANGER) drop considerably, especially the recall on the SHAH-I dataset.

The third row in each section of Table 3 (Step 3: Simulation II) shows the performance of our models after removing app features that do not contain a noun. This step was motivated by the assumption that app features not containing a noun (such as *running* or *runs*) are too unspecific to be useful for the developers. Since the SANGER AGs instruct to annotate implicit features represented by a single verb, we expected a significant drop of the number of app features for the SANGER and SHAH datasets and also an over-proportional reduction of the number of single-word app features. Surprisingly, it turned out that the number of app features dropped equally strongly for the GUZMAN dataset, and for all datasets the portion of single-word app features also significantly decreased but not much as compared to Step 2 (Simulation I), while the type-token ratio slightly increased. The German dataset SANGER still has a high portion of single-word app features (now 74%). The average number of app features per review narrows down after this step to the range [0.39,1.42] and is decreasing for all the datasets.

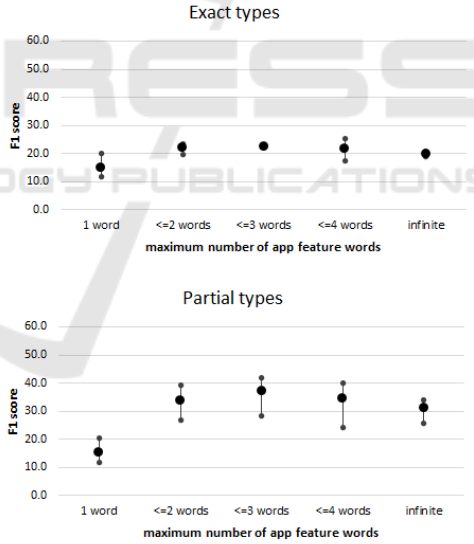

Figure 1: Average f1-score for exact and partial evaluation types when applying different cut-offs to the number of words in app features.

Overall, compared to the performance obtained after Step 2 (Simulation I), the recall remains roughly the same for all datasets. In terms of precision, we expected it to improve. If the annotated feature set contains short and vague verbal aspects that also would be used as non-aspect terms in the text (e.g. *using* or *updating*), it might be very difficult for the model to detect certain instances of these words as features.

We expected that removing such features from the annotated set would thus improve precision. The results show, however, that the precision increases only on the SHAH-I dataset while on all other datasets it dropped. This can happen if the short and vague-meaning verbal features share the same characteristics with the self-references—the distinct number of such features is small but their frequency is high, in which case it is relatively simple for the model to spot them and removing these features from the annotations causes the precision to drop.

The performance results shown for Steps 1 to 3 were achieved with annotations of app features (aspects) consisting of any number of words. Since long aspects potentially have a negative effect on model performance, we investigated whether imposing a maximum length could achieve better performance. Figure 1 summarizes the outcomes of our experiments. For the SANGER dataset, the performance was uniform across all choices of cutoffs. Therefore, we only show the average performances of the three English datasets. The two plots shown in Figure 1 show the minimum, maximum, and average F1-score for app features containing a number of words not greater than 1, 2, 3, 4 and with infinite length. The upper plot corresponds to exact type-based evaluation, while the lower plot represents partial type-based evaluation. In both plots, the best average performance is achieved when the app features consisting of more than three words are removed. The performance of our models after limiting the number of words in app features to a maximum of three words is shown in the last row of each section in Table 3 (Step 4: Simulation III-3). The effect on performance is uniformly positive for all datasets. The precision for partial type-based evaluation is in the range [68%,82%].

In summary, we can state that by simulating the application of modified AGs we achieve a feature prediction precision comparable to that received after the application of the original GUZMAN and SANGER guidelines. The advantage of the models created based on annotated datasets achieved by simulating the modified guidelines is that the predicted app features are more useful for developers since they are crisper (only one to three words of length) and correlate better with actual app features than with pseudo-features such as references to the app itself. The new rules included in the modified (improved) guidelines can be summarized as follows:

- Only annotate app features consisting of consecutive words;
- Do not annotate references to the app itself;
- Only annotate app features containing a noun;
- Restrict the length of the annotated app features to maximum three words.

5 DISCUSSION

In this section, we first explain the value of our proposed new AGs. Then we discuss limitations of our study.

5.1 Usefulness of the New AGs

The main goal of our study was to investigate the impact of AGs and annotated data on extracting app features from app reviews and to improve existing AGs such that (1) the performance of the app feature extraction task gets better in terms of f1 score and (2) the set of extracted app features is more useful to software developers.

Section 4 (Results) presented the step-by-step impact to the performance of the app feature extraction when simulating the effects of changing the used AGs. It turned out that with our proposed new AGs, a small performance improvement over the baseline situation could be achieved. However, this is not the only advantage of our new AGs. In the following we argue that not only the performance of the app feature extraction task can be improved but that the set of annotated and extracted app features itself is more relevant for software developers when using our new AGs.

We will illustrate what we mean by "more relevant for software developers" in two steps with the help of examples. In the first step, we demonstrate that the simulated application of our new AGs actually produces an annotated dataset that contains a larger share of annotated app features that are useful to software developers. In the second step, we demonstrate that this positive effect of our new AGs also propagates to the set of extracted app features.

Table F in Appendix[6] shows samples of app features in the original labeled datasets and in the annotated datasets after the simulated application of our new AGs. We randomly picked one app category from each of the English datasets, i.e., in categories 'Photography' (GUZMAN), 'Social' (SHAH-I), and 'Game' (SHAH-II). We manually classified each app feature as either 'useful' or 'not useful' and then compared how the numbers of useful and not useful app features change after simulated application of our new AGs. Not useful app features have bold font.

We consider an app feature to be useful when it seems to be related to actual functionality of an app.

For example *capture full resolution*, *decorating pictures* and *online scrapbooking* seem to be clearly referencing to some functionality in the app of categories 'Photography' and 'Social'. Aspects are not useful when they are too generic to be connected with a specific functionality (e.g., *share* or *version 1.5.1*). As shown by the study (Groen et al., 2017), non-functional aspects of an app (e.g, *easy to use*) can be identified with high precision using language patterns. Therefore, our concern in this study is to extract app functional aspects.

App features in the upper part of Table F in Appendix[6] correspond to a random sample of those app features that remained in the set of app features after simulated application of the new AGs. The numbers behind each app feature correspond to the token count before and after the simulated application of the new AGs. In some cases the token number changed. App features in the lower part of Table correspond to those app features that were completely removed from the set of app features after simulated application of our new AGs.

Table 4: The number of app features (type) manually classified as either 'useful' or 'not useful' in app categories Photography (GUZMAN), Social (SHAH-I) and Game (SHAH-II).

App category	Before/after simulation	Useful features	Not useful features	Total features
Photography	Before	50	77	127
	After	37	41	78
Social	Before	47	19	66
	After	38	8	44
Game	Before	57	22	79
	After	49	13	62

The ideal impact of the simulated application of our new AGs corresponds to removing all useless app features and keeping only the useful app features. We calculated the impact of our AGs based on the numbers of manually classified 'useful' and 'not useful' app features in three app categories before and after the simulation of new AGs (see Table 4). The actual numbers (based on type count) for each of the three apps are as follows:

- Category 'Photography' (GUZMAN): the percentage of useful app features kept equals 53%; the percentage of useless app features removed equals 47%; the ratio between useful and useless app features improved from 50/77=0.65 before the application of our new AGs to 37/41=0.90 afterwards;
- Category 'Social' (SHAH-I): the percentage of useful app features kept equals 77%; the percentage of useless app features removed equals 58%; the ratio between useful and useless app features improved from 47/19=2.47 before the application of our new AGs to 36/18=4.50 afterwards;
- Category 'Game' (SHAH-II): the percentage of useful app features kept equals 86%; the percentage of useless app features removed equals 41%; the ratio between useful and useless app features improved from 57/22=2.59 before the application of our new AGs to 49/13=3.77 afterwards.

The data shows for each of the three sample cases that the ratio between the number of useful and not useful app features is increasing when applying our new AGs. We computed the percentages based on type counts of app features because there can be cases like, for example, the app feature *editing*. The app feature *editing* occurred 13 times in the app of category 'Photography' before the simulated application of our new AGs and five times afterwards. We assume that eight occurrences of *editing* were removed due to the guideline 'Only annotate app features containing a noun', i.e., because after simulating the application of our new AGs *editing* was predicted to be an app feature when it was used as a noun. Note that the word *editing* when used as a verb is not helpful for software developers because it does not provide information about the purpose or object of editing and thus it is difficult to decide whether the mentioning of *editing* is related to the edit functionality as such or just a special situation in which something was edited. On the other hand, if *editing* is mentioned in the grammatical form of noun, it is more probable that whatever is said in the sentence with the word *editing* refers to the edit functionality in general. A similar case is *pinning* mentioned in the reviews of the app in category 'Social'. Here three of the seven original app feature predictions disappeared after simulated application of our new AGs.

After convincing ourselves that the simulated application of new AGs actually results in more useful app feature annotations, we checked whether this effect also propagates to the set of extracted app features. Table 5 shows the impact on the number of useful and useless app features in model's extracted app features, when training CRF models with the original annotated datasets and when training CRF models using the annotated datasets after the simulated application of our new AGs. We picked the same app categories as before from each of the English datasets, i.e., from categories 'Photography' (GUZMAN), 'Social' (SHAH-I), and 'Game' (SHAH-II). We manually classified each app feature as either 'useful' or 'not useful' and then compare how the numbers of useful and not useful app features change when simulating the application of our new AGs. The actual type counts for each of the three apps are:

- Category ‘Photography’ (GUZMAN): the ratio between useful and useless app features improved from 21/25=0.84 before the application of our new AGs to 17/10=1.7 afterwards;
- Category ‘Social’ (SHAH-I): the ratio between useful and useless app features improved from 17/18=0.94 before the application of our new AGs to 8/6=1.33 afterwards;
- Category ‘Game’ (SHAH-II): the ratio between useful and useless app features improved from 11/21=0.52 before the application of our new AGs to 13/14=0.93 afterwards.

Table 5: Model’s extracted app features (type) are manually classified as either ‘useful’ or ‘not useful’ in app categories Photography (GUZMAN), Social (SHAH-I) and Game (SHAH-II), before and after the simulation of AGs.

App category	Before/After simulation	Useful features	Not useful features	Total features
Photography	Before	21	25	46
	After	17	10	27
Social	Before	17	18	35
	After	8	6	14
Game	Before	11	21	32
	After	13	14	27

Figure 2 summarizes our results and expectations with regards to the effectiveness of our new AGs (based on type-wise analysis of the three selected app categories). Each of the six rectangles corresponds to the total set of annotated (upper row) and extracted (lower row) app features. The blue portion in each rectangle corresponds to the share of useful app features (UFs) while the orange portion corresponds to the share of useless app features (¬UFs). The calculated ratios between UFs and ¬UFs clearly show an improvement for the simulated application of our new AGs not only in the annotated datasets but also in the set of extracted features. This strengthens our expectation that a real application of the new AGs, which presumably yields exclusively useful features in the annotated dataset (thus an exclusively blue rectangle on the right in the upper row of Figure 2) would result in an even further improved ratio between UFs and ¬UFs in the set of extracted app features when comparing to the baseline and the simulated application of our new AGs (as indicated by the small portion of orange in the rectangle on the right in the lower row of Figure 2).

5.2 Limitations of the Study

In some cases it is not fully clear why an app feature was removed or kept. These cases could be due to inaccuracies of the POS tagger used in Step 2 of

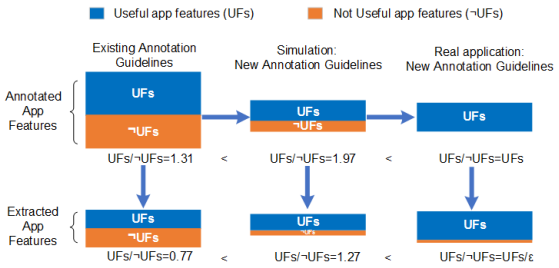

Figure 2: Ratios between useful and not useful app features (annotated and extracted) for three analyzed app categories.

the simulated application of our new AGs. For example, it is unclear why only two out of three occasions of *free* were removed in category ‘Photography’ as it is hard to imagine a context in which *free* could be considered to be a noun in a review text. Overall, our new AGs removed most of the useless app features in app categories ‘Social’ (SHAH-I) and ‘Game’ (SHAH-II). When removing the not useful app features, the lower performance (53%) on app category ‘Photography’ (GUZMAN) is due to a large number of annotations referring to mobile devices, app versions, app updates and non-functional app features.

Note that we only simulated the application of our new AGs on the labeled datasets. We expect that the application of the new AGs by actual people could have resulted in more useful annotations of app features in the first place. The application of our new AGs automatically removes app features that are longer than 3-words. However, in the direct application of our new AGs, a longer app feature might simply have been annotated with fewer words rather than completely been removed as we did in Simulation Step 3. For instance, a 5-word app feature *sorting functionality in board section* annotated in app category ‘Social’ of the SHAH-I dataset could be labeled as an admissible 2-word app feature *sorting functionality*.

Our study is restricted to the use of the CRF model which limits app features to be annotated as consecutive words. Therefore, when limiting annotations to a maximum 3-word app features, it might be impossible to annotate app features consisting of consecutive words; in such cases CRF (or any other sequences tagging model) cannot be applied or we would have to drop those app features (or soften the rule of having maximum 3-words app features). For instance, a 5-word app feature *edit pictures in a high quality* can be reduced to the following two meaningful representations of 3-word app features: *edit high quality* or *edit picture quality*. However, both 3-word app feature representations are not consecutive.

We only found two published AGs associated with publicly available annotated app review datasets. The

problem we encounter is that either annotated datasets were not published or when they had been published it is unclear what annotation rules/guidelines were applied. In other domains, e.g., LAPTOP and RESTAURANT, the standard guidelines and benchmark datasets are contributed by the research community SEMEVAL to perform the task of product feature extraction and its evaluation. Similar to the SEVEMAL research community, the app review mining research community could contribute standardized guidelines and benchmark datasets to help researchers in the development of systems performing fine-grained sentiment analysis at app feature-level.

We created two new labeled datasets SHAH-I and SHAH-II in English, using the English translated version of the German SANGER AGs. The translation from German to English of SANGER guidelines is performed using Google translation service. In order to make sure that the translated guidelines have sufficient quality to be used for the annotation of reviews in English language, one of the author of this paper, who is a native German speaker and have a full command of the English language, read the English translated version of the AGs and found it adequately accurate for the annotation task.

The validity of our results depends partly on the reliability of the annotations of the SHAH-I and SHAH-II datasets. In addition, the assessment of the usefulness of the results produced when using our simulated AGs depends on the reliability of the subjective classification of annotated and extracted app features into 'useful' and 'not useful'. Since each of these tasks was performed by one person, reliability might be limited. However, since we not only used our own annotations (i.e., datasets SHAH-I and SHAH-II) but applied our analyses also to datasets published in the literature and the trend of our results was similar for all our used datasets, we believe that the existing limitations of reliability for the mentioned tasks is not a major threat to validity of our results.

6 CONCLUSION

Previously, several techniques have been used for automatic app feature extraction from app reviews, including (a) unsupervised topic modeling, (b) rule-based methods, (c) supervised machine learning approaches. While unsupervised and rule-based approaches only require annotated data for evaluation purposes, supervised machine learning methods also need it for training the model. In either case, the quality of the annotations can considerably affect the evaluation results. When the annotations contain many complex app features that are difficult to extract, then model performance will be artificially low, especially when these features are infrequent and thus not of much actual interest for app developers. On the other hand, when the annotations contain many short and frequent app features that are easy to detect but not informative for developers then model performance will be artificially high.

In this paper, we study the impact of AGs by controlling other design parameters as much as possible. We used four different labeled datasets annotated with two different AGs. For the app feature extraction technique we adopted the supervised CRF method. To our best knowledge, this is the first study that explores the impact of AGs and labeled datasets for app feature extraction from user reviews. As a result of our study, we propose several changes to the existing AGs to avoid the annotation of useless app features.

Our research question *RQ* started from the observation that the app feature annotations and thus subsequently also automatic feature extraction results varied considerably on different datasets, even though they used the same AGs to annotate the app features and even when the annotated app reviews themselves were identical (SHAH-I vs SHAH-II). We hypothesised that these differences are at least partly due to the AGs that were used to label these datasets. We proposed several changes to the AGs and evaluated the effect of their simulated application using the evaluation results of the CRF modeling. The proposed changes in AGs include (a) instructing to annotate only consecutive words as app features, (b) discouraging the annotations to the references of the app itself, (c) instructing to annotate only noun phrases as app features, i.e. every app feature must contain a noun, and (d) instruct to annotate app features with the length of maximum three words. When simulating the application of our AGs, we were able to retain the precision of the app feature extraction. Moreover, the annotated features in both training and test sets become more informative and less noisy.

The main limitation of our study is the simulation of the new AGs on the labeled review datasets which resulted in removing a number of app features rather than reformulating them according to the new guidelines. Regardless of that, we believe that the real application of the new AGs by human annotators would have produced a set of app features that are useful and refer to the functional aspects of an app. However, this hypothesis is yet to be confirmed by evaluating the proposed AGs by giving them to real human annotators for labeling app features in user reviews.

ACKNOWLEDGMENTS

We are grateful to Emitza Guzman and Mario Sanger for sharing their review datasets. This research was supported by the institutional research grant IUT20-55 of the Estonian Research Council and the Estonian Center of Excellence in ICT research (EXCITE).

REFERENCES

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

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (slmost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Groen, E. C., Kopczyska, S., Hauer, M. P., Krafft, T. D., and Doerr, J. (2017). Users the hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 80–89.

Guzman, E., Aly, O., and Bruegge, B. (2015). Retrieving diverse opinions from app reviews. In *Proceedings of ESEM'15*, pages 1–10. IEEE.

Guzman, E. and Maalej, W. (2014). How do users like this feature? a fine grained sentiment analysis of app reviews. In *Proceedings of RE'14*, pages 153–162. IEEE.

Johann, T., Stanik, C., B., A. M. A., and Maalej, W. (2017). Safe: A simple approach for feature extraction from app descriptions and app reviews. In *Proceedings of RE'17*, pages 21–30. IEEE.

Kang, Y. and Zhou, L. (2017). Rube: Rule-based methods for extracting product features from online consumer reviews. *Information & Management*, 54(2):166–176.

Keertipati, S., Savarimuthu, B. T. R., and Licorish, S. A. (2016). Approaches for prioritizing feature improvements extracted from app reviews. In *Proceedings of EASE'16*, page 33. ACM.

Kurtanović, Z. and Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. In *Proceedings of RE'17*, pages 490–495. IEEE.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML'01*, pages 282–289.

Liu, P., Joty, S. R., and Meng, H. M. (2015). Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of EMNLP'15*, pages 1433–1443.

Lu, M. and Liang, P. (2017). Automatic classification of non-functional requirements from augmented app user reviews. In *Proceedings of EASE'17*, pages 344–353. ACM.

Luiz, W., Viegas, F., Alencar, R., Mourão, F., Salles, T., Carvalho, D., Gonçalves, M. A., and Rocha, L. (2018). A feature-oriented sentiment rating for mobile app reviews. In *Proceedings of WWW'18*, pages 1909–1918, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Maalej, W., Nayebi, M., Johann, T., and Ruhe, G. (2016). Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54.

Malik, H., Shakshuki, E. M., and Yoo, W.-S. (2018). Comparing mobile apps by identifying hot features. *Future Generation Computer Systems*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS'13*, pages 3111–3119.

Pagano, D. and Maalej, W. (2013). User feedback in the appstore: An empirical study. In *Proceedings of RE'13*, pages 125–134. IEEE.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., and Gall, H. C. (2015). How can i improve my app? classifying user reviews for software maintenance and evolution. In *Proceedings of ICSME'15*, pages 281–290. IEEE.

Pavlopoulos, J. and Androutsopoulos, I. (2014). Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method. In *Proceedings of the 5th Workshop on Language Analysis for Social Media*, pages 44–52.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., et al. (2016). Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of SemEval'16*, pages 19–30. ACL.

Poria, S., Cambria, E., and Gelbukh, A. (2016). Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Sänger, M., Leser, U., Kemmerer, S., Adolphs, P., Klinger, R., Calzolari, N., Choukri, K., Declerck, T., Grobelnik, M., and Maegaard, B. (2016). Scare-the sentiment corpus of app reviews with fine-grained annotations in german. In *Proceedings of LREC'16*.

Shah, F. A., Sabanin, Y., and Pfahl, D. (2016). Feature-based evaluation of competing apps. In *Proceedings of the International Workshop on App Market Analytics*, pages 15–21. ACM.

Vu, P. M., Nguyen, T. T., Pham, H. V., and Nguyen, T. T. (2015). Mining user opinions in mobile app reviews: A keyword-based approach. In *Proceedings of ASE'15*, pages 749–759. IEEE.

Zamani, S., Lee, S. P., Shokripour, R., and Anvik, J. (2014). A noun-based approach to feature location using time-aware term-weighting. *Information and Software Technology*, 56(8):991–1011.

Zhang, L. and Liu, B. (2014). Aspect and entity extraction for opinion mining. In *Data Mining and Knowledge Discovery for Big Data*, pages 1–40. Springer.