# Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement

Grano, Giovanni ; Di Sorbo, Andrea ; Mercaldo, Francesco ; Visaggio, Corrado A ; Canfora, Gerardo ; Panichella, Sebastiano

# Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement

Giovanni Grano
University of Zurich
grano@ifi.uzh.ch

Andrea Di Sorbo
University of Sannio
disorbo@unisannio.it

Francesco Mercaldo
Ist. di Informatica e Telematica, CNR
francesco.mercaldo@iit.cnr.it

Corrado A. Visaggio
University of Sannio
visaggio@unisannio.it

Gerardo Canfora
University of Sannio
canfora@unisannio.it

Sebastiano Panichella
University of Zurich
panichella@ifi.uzh.ch

## ABSTRACT

Nowadays, Android represents the most popular mobile platform with a market share of around 80%. Previous research showed that data contained in user reviews and code change history of mobile apps represent a rich source of information for reducing software maintenance and development effort, increasing customers' satisfaction. Stemming from this observation, we present in this paper a large dataset of Android applications belonging to 23 different apps categories, which provides an overview of the types of feedback users report on the apps and documents the evolution of the related code metrics. The dataset contains about 395 applications of the F-Droid repository, including around 600 versions, 280,000 user reviews and more than 450,000 user feedback (extracted with specific text mining approaches). Furthermore, for each app version in our dataset, we employed the Paprika tool and developed several Python scripts to detect 8 different code smells and compute 22 code quality indicators. The paper discusses the potential usefulness of the dataset for future research in the field.

Dataset URL: https://github.com/sealuzh/user_quality

## CCS CONCEPTS

• **Software and its engineering** → *Software development process management*; *Software evolution*;

## KEYWORDS

Software Quality, App Reviews, Mobile Applications, Software Maintenance and Evolution

**ACM Reference format:**

Giovanni Grano, Andrea Di Sorbo, Francesco Mercaldo, Corrado A. Visaggio, Gerardo Canfora, and Sebastiano Panichella. 2017. Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn



*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Mobile app stores, such as Google Play and Apple App Store, represent a rich source of information for software engineering researchers and developers interested in better understanding how mobile software is created and maintained [9], since these markets provide an open access to huge numbers of software applications together with consumers' feedback [11]. Indeed, previous research showed that data contained in both user reviews and code change history of mobile apps represent a rich source of information for the development of mobile software, for reducing software maintenance effort and increasing customers' satisfaction [5, 12, 14].

In this context users' feedback are particularly important since software maintenance and evolution of mobile applications are strictly guided by requests contained in user reviews [3, 6, 11]. For instance, investigating the types of feedback users report on the apps they are using can give valuable information about the features on which they pay more attention or help better understanding the most common issues related to a specific app category [5, 11, 16]. Beside that, a more in-depth analysis on the code changes performed by developers when integrating users' feedback in the code base of mobile applications can provide key insights on how developers evolve these apps to gain an higher customers satisfaction (e.g., for increasing downloads of a given app). Unfortunately, app stores lack functionalities to organize informative user reviews feedback toward proper software maintenance tasks or filter them according to the treated topics [1].

In this paper we propose a dataset containing 288,065 reviews extracted from Google Play[1] related to 395 open source apps mined from F-Droid[2]. Every review is connected with a specific version of the app and then split into atomic sentences. Each of the obtained sentences is labelled with the intention and the topics it deals with, relying on the two-dimension URM taxonomy proposed by Di Sorbo *et al.* [5]. Moreover, for each of the app versions, 22 code quality metrics (e.g., object-oriented and Android-oriented metrics) and 8 different code-smells have been computed. The goal of this work is to provide data that researchers may promptly use to conduct experiments aimed at better (i) understanding how specific aspects related to code quality could affect app reviews and star-ratings, (ii) comprehending how developers react to specific user review feedback when evolving their mobile applications. To the best of authors' knowledge, this is the first attempt to create

[1] https://play.google.com/store/apps
[2] https://f-droid.org

**Table 1: Intention Categories Definition**

| | |
|---|---|
| **Information Giving** | Sentences that inform other users or developers about some aspect of the app. |
| **Information Seeking** | Sentences describing attempts to obtain information or help from other users or developers. |
| **Feature Request** | Sentences expressing ideas, suggestions or needs for enhancing the app. |
| **Problem Discovery** | Sentences reporting unexpected behavior or issues. |
| **Other** | Sentences not belonging to any of the previous categories. |

a publicly available data collection containing such a substantial amount of data, in which reviews related to specific app releases are labeled according to software maintenance categories (i.e., types of user feedback) and apps are analyzed by static analysis tools for computing their software quality.

## 2 RELATED WORK

Despite app stores represent a relatively recent phenomenon, they immediately captured the interest of the software engineering community and, nowadays, there are already over 180 papers devoted to their study[9]. As a consequence, several datasets involving a quite high numbers of apps with structured (e.g., source code) and unstructured information (e.g., commits messages) have been proposed in the literature. For instance, the paper by Krutz *et al.*[8] provided a dataset that reports results obtained by several static analysis tools on 4,416 different versions of 1,179 open-source android applications combined with data of version control commits related to these applications. Collections containing huge amounts of app reviews have also been published for pursuing different research goals. For example, the *Data Set for Mobile App Retrieval*[3] includes 1,385,607 user reviews of 43,041 mobile apps and it has been mainly used to run experiments about accuracy improvements in mobile app retrieval[15]. The *SoftWare Marketplace* (SWM) review dataset[4] contains 1,132,373 reviews from 15,094 apps and has been involved in research works aimed at detecting spam or fake reviews[2, 18, 19]. Other existing public available data[5] could be used to build and test sentiment analysis algorithms, since they contain reviews clustered according to the sentiment expressed in them (i.e., negative and positive sentiment). Nevertheless, to the best of our knowledge, no previous work provided a comprehensive dataset that, at the same time, (i) sheds the light on the types of feedback users report for different versions of several apps and, (ii) combines such information with software quality indicators computed on the app versions they are referring to.

## 3 DATASET CONSTRUCTION

Our dataset was built in two phases: (i) in the data collection phase we analyzed the F-Droid repository and the Google Play store for collecting the app versions data and the information related to their user reviews; (ii) in the analysis phase we examined the Android package (i.e., the apk) of the mined apps using several static analysis scripts/tools and labeled the extracted reviews through the use of two automated classifiers.

[3]https://sites.google.com/site/daehpark/Resources/data-set-for-mobile-app-retrieval
[4]http://odds.cs.stonybrook.edu/swmreview-dataset/
[5]https://github.com/amitt001/Android-App-Reviews-Dataset

**Table 2: Topic Definitions**

| Cluster | Description |
|---|---|
| **App** | sentences related to the entire app, *e.g.*, generic crash reports, ratings, or general feedback |
| **GUI** | sentences related to the Graphical User Interface or the look and feel of the app |
| **Contents** | sentences related to the content of the app |
| **Pricing** | sentences related to app pricing |
| **Feature or Functionality** | sentences related to specific features or functionality of the app |
| **Improvement** | sentences related to explicit enhancement requests |
| **Updates/ Versions** | sentences related to specific versions or the update process of the app |
| **Resources** | sentences dealing with device resources such as battery consumption, storage, etc. |
| **Security** | sentences related to the security of the app or to personal data privacy |
| **Download** | sentences containing feedback about the app download |
| **Model** | sentences reporting feedback about specific devices or OS versions |
| **Company** | sentences containing feedback related to the company/team which develops the app |
| **Other** | sentences not treating any of the previous topics |

### 3.1 Data Collection Phase

In this phase, we primarily built a web crawler (available in the dataset URL) to collect from the F-Droid repository the meta-data (package name, available versions, release date of each version) and the apks of each app. The crawler initially mined data for 1,929 different apps. The versions of each mobile application have been ordered according to the release date (i.e., from the oldest to the latest version). All the apps (i) not appearing in the Google Play Store and (ii) whose latest version was released before the year 2014 (*i.e.*, this could indicate that the app is no longer maintained) have been discarded. A second scraper tool[6] was built to download from Google Play Store all the user reviews related to the remaining 965 apps. It relies on Phantom JS[7] and Selenium[8] in order to navigate the Play Store web site and extract reviews from the resulting HTML code. We set up a *cronjob* in order to mine new reviews 4 times a week. The tool totally gathered 297,323 app reviews, and for each user comment it also extracted (i) the package name of the app to which the review refers, (ii) the review content, (iii) the related star-rating assigned by the user to the app, and (iv) the posting date of the review. Relying on the release date of each applications' version and on the review's posting date of each user comment, we assigned each review to one of the app versions as described below. Given a generic version of an app, $V_i$, and the next version of the same app, $V_{i+1}$, the reviews assigned to the version $V_i$, i.e., $R_i$, are collected considering the reviews whose posting date occur after the release date of $V_i$ and before the release date of $V_{i+1}$. Despite this assumption may produce for some reviews an assignment to a wrong app version, Pagano et Maalej [10] empirically demonstrated that user feedback is mostly triggered by new releases, i.e., usually in the first few days after the download of a new app version. We discarded 8,758 reviews (because their publication date was too old for assigning them to any of the available versions) obtaining a dataset containing 288,565 reviews belonging to 710 different versions. Then we decided to keep in the collection exclusively the app versions having at least 10 reviews assigned (according to previous studies [17]), discarding all the remaining ones. At the end of this filtering process we obtained a dataset of 288,065 reviews related to 629 versions of 395 different apps.

[6]https://github.com/sealuzh/user_quality/tree/master/tools
[7]http://phantomjs.org/
[8]http://www.seleniumhq.org/

## 3.2 Analysis Phase

In this phase we classified user reviews' feedback according to software maintenance and evolution categories and computed software quality indicators for each version of the mined apps. To achieve these goals, we employed some existing tools recently presented in literature. In the following we briefly explain how we performed the two tasks.

*3.2.1 User Reviews Classification.* In order to classify users' comments according to software maintenance and evolution categories, we selected as a conceptual framework the URM taxonomy proposed by Di Sorbo *et al.* [5] which represents a robust and suitable model for representing user comments in meaningful maintenance tasks. This model assigns each review to (i) one of the *intention* categories showed in Table 1 and (ii) one or more topics detailed in Table 2. For performing the two-dimensions classification encompassed by the model, we employed:

- ARDOC, a reviews classifier previously defined by Panichella *et al.* [13], which combines Natural Language Processing (NLP), Sentiment Analysis (SA), and Text Analysis (TA) techniques in order to automatically mine *intentions* in user reviews, according to the categories defined in Table 1. This classifier has shown to achieve high precision (ranging between 84% and 89%) and recall (ranging between 84% and 89%) in categorizing reviews from real-word applications. We used for our purposes the original implementation of the tool, freely accessible as a Java library [13].
- The topic classifier module based on topics-related keywords and n-grams, used in the SURF summarizer tool[5], which is able to assign to each sentence in the review one (or more) of the topics defined in Table 2. This classifier has shown to achieve a classification accuracy of 76%.

*3.2.2 Quality Analysis of Applications' Code.* To evaluate the code quality of Android applications, we developed Python scripts that compute a set of code quality indicators (all the quality metrics that our scripts are able to compute are detailed in the Appendix available online[9]). In particular, the apks of all the mined versions with at least 10 reviews assigned have been disassembled, in order to obtain a set of human readable Dalvik bytecode `.smali` files from the binary Dalvik bytecode format `.dex` ones. To accomplish this task we used *apktool*[10], a tool for reverse engineering which allows to decompile and recompile Android applications. Thus, we developed a set of Python scripts[11] able to parse the `.smali` files and automatically compute the suite of code metrics for each of the available apks in our dataset.In our analysis, we compute the metrics by parsing `.smali` classes (and not java ones), in order to consider code optimizations eventually applied by the compiler. In addition, we enrich our analysis by detecting code smells in the selected apks employing Paprika[7], a tooled approach which decompiles the application with Soot[12] and performs the detection of 4 categories of Object-Oriented code smells (i.e., *Blob Class (BLOB), Swiss Army Knife (SAK), Long Method (LM) and Complex Class (CC)*) and 4 categories of Android anti-patterns (i.e., *Internal Getter/Setter (IGS), Member Ignoring Method (MIM), No Low Memory Resolver (NLMR), Leaking Inner Class (LIC)*). We computed and stored all the code smells above for each of the available versions of apps in our set, except for few ones that Paprika was not able to work with.

**Table 3: Applications and Versions for each App Category**

| App Category | Apps | Total apks | Total reviews |
|---|---|---|---|
| Books & Reference | 19 | 35 | 15,892 |
| Business | 1 | 1 | 1,172 |
| Comics | 4 | 5 | 2287 |
| Communication | 33 | 64 | 31,219 |
| Education | 12 | 14 | 1,291 |
| Entertainment | 5 | 5 | 2,584 |
| Finance | 7 | 10 | 621 |
| Games | 30 | 44 | 20,378 |
| Health & Fitness | 3 | 4 | 1,149 |
| Libraries & Demo | 3 | 5 | 990 |
| Lifestyle | 4 | 6 | 246 |
| Maps & Navigation | 10 | 15 | 1,411 |
| Music & Audio | 17 | 32 | 3,025 |
| News & Magazines | 6 | 9 | 1,988 |
| Personalization | 18 | 26 | 12,037 |
| Photography | 7 | 10 | 4,275 |
| Productivity | 45 | 80 | 8,361 |
| Shopping | 2 | 2 | 2,647 |
| Social | 7 | 11 | 6,146 |
| Tools | 139 | 214 | 151,509 |
| Travel & Local | 9 | 12 | 984 |
| Video Players & Editors | 12 | 22 | 15,352 |
| Weather | 2 | 3 | 2,501 |
| **TOTAL** | **395** | **629** | **288,065** |

## 4 ANALYTICS & DATA SHARING

To offer a more complete overview of the dataset, in this section we provide some statistics about the collected data, as well as information about its final structure. In detail, Table 3 reports respectively the number of apps collected, the apks available and the reviews mined for each of the covered Google Play categories.

Analyzed apks contain a total of 100,638,277 byte-code instructions, 832,347 classes, and 6,375,906 methods. Each apk ranges from a minimum of 10 to a maximum of 103,535 reviews assigned. On average, each apk has about 458 reviews assigned and in turn each user comment is composed generally by 1.57 sentences. Furthermore, every one of them belongs to one of the intention categories (see Table 1), and, on average, deals with about 1.34 of the maintenance topics (see Table 2). To provide a more detailed description of the reviews' dataset, Table 4 reports, for each of the maintenance topics, the amounts of (i) sentences discussing the topic, (ii) sentences of the *Feature Request* (FR) intention category dealing with the topic, (iii) sentences of the *Problem Discovery* (PD) intention category treating the topic, (iv) sentences of the *Information Seeking* (IS) intention category dealing with topic, (v) sentences of the *Information Giving* (IG) intention category discussing the topic, and (iv) sentences of the *Other* intention category treating the specific topic. Moreover, for the 629 apks of our dataset, we were able to detect a total of (i) 3,263 *Blob Classes*, (ii) 44,834 *Long Methods*, (iii) 432 *Swiss Army Knives*, (iv) 8,640 *Complex Classes*, (v) 9,012 *Internal*

[9] https://github.com/sealuzh/user_quality/wiki/Code-Quality-Metrics
[10] http://ibotpeaches.github.io/Apktool/
[11] https://github.com/sealuzh/user_quality/tree/master/code_metrics_scripts
[12] https://sable.github.io/soot/

**Table 4: Number of sentences discussing each topic**

| Topic | Sentences | FR | PD | IS | IG | Other |
|---|---|---|---|---|---|---|
| App | 117,409 | 4,879 | 11,089 | 1,600 | 11,943 | 87,898 |
| GUI | 37,620 | 3,381 | 5,034 | 705 | 3,560 | 2,4940 |
| Contents | 16,819 | 1,315 | 1,973 | 434 | 1,620 | 11,477 |
| Download | 7,853 | 333 | 1,346 | 363 | 830 | 4,981 |
| Company | 1672 | 118 | 190 | 57 | 152 | 1,155 |
| Feature/Functionality | 173,847 | 15,480 | 27,810 | 4,342 | 14,972 | 111,243 |
| Improvement | 8,281 | 1,005 | 304 | 54 | 755 | 6,163 |
| Pricing | 4,016 | 142 | 216 | 62 | 559 | 3,037 |
| Resources | 3071 | 155 | 375 | 50 | 263 | 2228 |
| Update/Version | 21,669 | 1,358 | 3,886 | 548 | 2,423 | 13,454 |
| Model | 22,044 | 1,308 | 3,397 | 459 | 2,055 | 14,825 |
| Security | 2,392 | 212 | 313 | 65 | 218 | 1,584 |
| Other | 189,784 | 630 | 2,019 | 1,402 | 2,842 | 182,891 |
| **TOTAL** | **606,477** | **30,316** | **57,952** | **10,141** | **42,192** | **465,876** |

*Getters/Setters*, (vi) 6,768 *Member Ignoring Methods*, (vii) 2,280 *No Low Memory Resolvers*, and (viii) 23,293 *Leaking Inner Classes*.

**Dataset Schema.** We provide our dataset both as relational DBMS and in CSV format. In our repository's wiki page[13] we show and ER Diagram and we accurately describe its structure. The **Apk** table encompasses the metadata (*i.e.*, package name, app category, version code, release date) related to each apk involved in the dataset. The **Review** table contains all the extracted reviews with the correspondent apk id. Each review is linked with one or more records of the **Sentence** table, which reports the classification outputs (*i.e.*, its intention and topics) about each atomic sentence. The **CodeMetrics** table provides the code quality metrics for each apk, while the **Smell** table holds the quantities of code smells detected in each app version. Finally, the **UserMetrics** table summarizes through a set of cumulative metrics (*e.g.*, total number of reviews, overall number of sentences, the amount of sentences falling in each intention and topic category, etc.) the processed review data related to each apk.

**Research Opportunities.** Recently, researchers have started to investigate the impact of specific code quality metrics in the market success of mobile applications [4]. To the extent of our knowledge, there are no studies that specialize more in depth this analysis considering the different app categories (i.e., social, tools, games and so on) and the semantic analysis of user feedback related to different versions of the same app. In this context, we believe that this dataset can be successfully exploited for many different purposes, for example: (i) to study more in depth the relationships between code quality metrics and the success within different app categories, (ii) to investigate the influence of code quality on app rating and user satisfaction, (iii) to have a look at the consequences on code quality when integrating specific user feedback in the code base, (iv) to study the evolution trends of quality metrics and code-smells between different versions of the same app in presence of particular kinds of user feedback.

## 5 CONCLUSIONS

The dataset we provide comprises 395 different apps from F-Droid repository, including code quality indicators of 629 versions of these apps. It also encloses app reviews related to each of these versions, which have been automatically categorized classifying types of user feedback from a software maintenance and evolution perspective. A total of 288,065 user reviews and more than 450,000 user feedback have been gathered, for enabling future research aimed at supporting developers evolving and maintain mobile applications in a faster and more efficient way, increasing users' satisfaction. The data provided are useful for understanding potential correlations between the various collected data metrics, not only looking into individual apps, but also analyzing them in aggregation.

[13] https://github.com/sealuzh/user_quality/wiki/Database-Schema

## REFERENCES


[1] U. Abelein, H. Sharp, and B. Paech. Does involving users in software development really influence system success? *IEEE Software*, 30(6):17–23, 2013.

[2] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[3] A. Ciurumelea, A. Schaufelbuhl, S. Panichella, and H. C. Gall. Analyzing reviews and code of mobile apps for better release planning. In *SANER*, pages 91–102. IEEE Computer Society, 2017.

[4] L. Corral and I. Fronza. Better code for better apps: A study on source code quality and market success of android applications. In *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems*, MOBILESoft '15, pages 22–32, Piscataway, NJ, USA, 2015. IEEE Press.

[5] A. Di Sorbo, S. Panichella, C. Alexandru, J. Shimagaki, C. Visaggio, G. Canfora, and H. Gall. What would users change in my app? summarizing app reviews for recommending software changes. In *Foundations of Software Engineering (FSE), 2016 ACM SIGSOFT International Symposium on the*, pages 499–510, 2016.

[6] E. Guzman, O. Aly, and B. Bruegge. Retrieving diverse opinions from app reviews. In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10, 2015.

[7] G. Hecht, O. Benomar, R. Rouvoy, N. Moha, and L. Duchien. Tracking the software quality of android applications along their evolution (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 236–247, Nov 2015.

[8] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz, J. Peterson, A. Filipski, and J. Smith. A dataset of open-source android applications. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 522–525, Piscataway, NJ, USA, 2015. IEEE Press.

[9] 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*, PP(99):1–1, 2016.

[10] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE 2013)*. IEEE Computer Society, 2013.

[11] F. Palomba, M. Linares-Vasquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 291–300, Sept 2015.

[12] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. C. Gall, F. Ferrucci, and A. D. Lucia. Recommending and localizing change requests for mobile apps based on user reviews. In *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*, pages 106–117, 2017.

[13] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall. Ardoc: App reviews development oriented classifier. In *Foundations of Software Engineering (FSE), 2016 ACM SIGSOFT International Symposium on the*, pages 1023–1027, 2016.

[14] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, pages 281–290, 2015.

[15] D. H. Park, M. Liu, C. Zhai, and H. Wang. Leveraging user reviews to improve accuracy for mobile app retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 533–542, New York, NY, USA, 2015. ACM.

[16] A. D. Sorbo, S. Panichella, C. V. Alexandru, C. A. Visaggio, and G. Canfora. SURF: summarizer of user reviews feedback. In *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume*, pages 55–58, 2017.

[17] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan. What are the characteristics of high-rated apps? a case study on free android applications. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, ICSME '15, pages 301–310, Washington, DC, USA, 2015. IEEE Computer Society.

[18] J. Ye and L. Akoglu. Discovering opinion spammer groups by network footprints. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 267–282. Springer, 2015.

[19] J. Ye, S. Kumar, and L. Akoglu. Temporal opinion spam detection by multivariate indicative signals. In *Tenth International AAAI Conference on Web and Social Media*, 2016.