Spam Detection Framework for Android Twitter Application Using Naïve Bayes and K-Nearest Neighbor Classifiers

Aryo Pinandito[1], Rizal Setya Perdana[2], Mochamad Chandra Saputra[3], Hanifah Muslimah Az-zahra[4]
Information System Department, Computer Science Faculty, Universitas Brawijaya
Malang, East Java, Indonesia
{aryo[1], rizalespe[2], andra[3], hanifah.azzahra[4]}@ub.ac.id

ABSTRACT
Twitter currently is one of the leading social networks worldwide based on the amount of monthly active users after Facebook and Instagram. People uses Twitter mostly to find out more information about breaking news or keeping up with news in general by following trending topics. As Twitter become a source of news breaks contents in form of comments and replies to share the newest ideas. Therefore, several mobile applications that utilize Twitter API has been developed to provide a convenient way in providing trending topics to their user. Twitter trending topics offers an effective opportunity in marketing point of view for online marketers to promote their marketing contents. Spam contents in Twitter were found to be distracting and annoying for certain users, thus mobile application to deliver spam-free Twitter trending topics contents is needed. This research designs an Android application framework that allow developers to build their own implementation of spam detection classifier for Twitter contents as application library. This research implements two classification methods, i.e. Naïve Bayes and K-Nearest Neighbor, to identify spam in Twitter trending topics. The Naïve Bayes and K-Nearest Neighbor classification methods are able to detect spam and ham contents with 82% and 71% accuracy respectively.

CCS Concepts
• Information systems → Mobile information processing systems

Keywords
Spam; Bayes; KNN, Twitter; Android;

1. INTRODUCTION
Nowadays, social networks and their associated mobile and messaging applications are become more popular than ever. People uses Twitter mainly to be alerted to or find out more about breaking news, keeping up with news in general, and even following trending topics [1]. It is found that YouTube, Facebook, and Instagram users are more likely to find their news online mostly by chance while LinkedIn, Twitter, and Reddit users are more likely to be related with news seekers and non-seekers [2].

ICSCA 2017, February 26-28, 2017, Bangkok, Thailand

DOI: http://dx.doi.org/10.1145/3056662.3056704

Therefore, Twitter considered as an important social media for users, business, researchers, and public at large [3] and also Twitter had become more than just a breaking news social media service, but changed the way users to stay in touch with public updates.

Twitter had published their Trends feature since the summer of 2008. The feature is designed to help people discover the most breaking news, which matter for a specific user, from across the world in real time. The topics were automatically generated by their proprietary algorithm and tailored based on location and people that being followed [4]. Twitter now become a source of content as news breaks on Twitter as comments and replies to share the newest ideas on the latest breakthrough are probably hitting Twitter first before other social medias.

With the widespread use of social media, it is no secret that social media contents are full of messy, useless, and irrelevant data. As online marketing goes to social media such as Twitter, some people find themselves are spending too much time to read or listen to irrelevant and undesired contents. Even though spam rates across social media platforms is less than 10%, Twitter conversations of product brands are dominated by spam as described in [5]. The term 'spam' itself was widely agreed as unsolicited electronic messages which is sent in bulk [6]. In contrary, 'ham' is more generally desired message and is not considered as spam [7]. Several online marketers took advantages from Twitter trending topics feature by injecting several trending topic keywords into their marketing contents. This way, they could promote their marketing contents to quickly reach more potential customers. Unsolicited contents with repeating actions that negatively impact other users and/or potentially dangerous can be generally described as spam. Therefore, people who intends to get current updates from trending topic contents were often distracted or even annoyed by negative contents.

Aside from providing social media services through its website and mobile applications, Twitter also provides social media content and data through an Application Programming Interface (API). The API enables mobile application developers, especially on the iOS and Android platforms, to integrate and elaborate their mobile application functionalities with Twitter. Therefore, many news-related mobile applications, which utilizes APIs and online services from social media services, e.g., Twitter, Instagram, and Facebook that are emerging.

There are several classification methods to detect and identify spam and malicious contents such as K-Nearest Neighbor, Fuzzy K-Means, Extreme Learning Machine (ELM), and Support Vector

Machine (SVM) methods as presented in [8] and [9]. Both researches reveal an excellent accuracy in detecting spam. Another method such as improved Naïve Bayes method to filter out spam contents also has good classification accuracy [10][11][12]. This paper presents design of Android-based spam detection application framework that allow Android developers to use or implement their own spam classifiers or detection method as application libraries or modules to identify spam messages in their Twitter contents. The performance and accuracy of implemented libraries based on the amount of training data or referenced contents are also presented.

2. LITERATURE REVIEW

2.1 Naïve Bayes Classification

In a text-based document classification, a document can be classified by probabilistically evaluating its consisting words. Naïve Bayes is one popular method that can be used to classify documents on a probabilistic basis. Naïve Bayes method is a specific type of Bayesian theory that assume every conditions or classes are independent among each other. Words that compose a document in a class or category are not affecting each other classes or categories. Therefore, the probabilities of evaluated document belong to a particular category had to be separately calculated. However, there are only two identified categories in this research case, i.e. spam and ham.

The probability of a document based on its composing words ($W_1 \ldots W_n$) belongs to a particular category *(C)* can be calculated using (1). This probability value is also known as document posterior probability for category C.

$$P(C \mid W_1 \ldots W_n) = P(C) \prod_{i=1}^{n} P(W_i \mid C) \quad (1)$$

The likelihood for a word belongs to a particular category is obtained from the division between the occurrence frequency of the word in a document (n_k), the amount of words in observed category (n_c), and the number of words variation *(n)* from all documents in the training data for a specified category as in (2). However, prior probability for a particular category was obtained from the number of documents (n_d) in a specified category divided by the total number of documents (n_t) from all categories in provided training data. as shown in (3).

$$P(W_j \mid C_i) = (1 + n_k) / (n_c + n) \quad (2)$$

$$P(C_i) = | n_d | / | n_t | \quad (3)$$

Once the posterior values of the evaluated document against every category has been calculated, the document is classified based on its maximum-a-posteriori value (v_{MAP}) as in (4). Therefore, documents are classified into a category where its calculated posterior values are maximized.

$$v_{MAP} = \text{argmax } P(C_i) \Pi P(W_j \mid C_i) \quad (4)$$

2.2 K-Nearest Neighbor and TF-IDF

K-Nearest Neighbor (KNN) is a popular text classification method that enables text to be classified according to several parameters. KNN often combined with TF-IDF method to reflect the importance of a word to a document in a collection or corpus as in [13]. TF-IDF value is often used as a weighting factor in information retrieval and text mining.

One of spam detection libraries that being implemented in this research implements KNN classification method to identify spam and ham tweets in a collection of tweets obtained from the Android application. The implemented classification method utilizes TF-IDF method to calculate the value of a document against spam or ham category based on its consisting words against referenced documents for a particular category.

Weight value of word *i* in a document *j* for category *k* (a_{ijk}) were computed using TF-IDF method based on its frequency in the document (tf_{ij}) and its inverse document frequency (idf_i) as in (5). The inverse document frequency of word *i* can be calculated from the number of documents for a category *k* (N_k) and the number of documents that contain the word *i* (df_i) as in (6).

$$a_{ijk} = tf_{ij} \, . \, idf_i \quad (5)$$

$$idf_i = log \, (N_k / df_i) \quad (6)$$

Thus, the weight value of a document *j* to category *k* (w_{jk}) can be calculated by summing over the weight of words (a_{ijk}) for category *k* that compose the document as in (7).

$$w_{jk} = \sum_{i=1}^{n} a_{ijk} \quad (7)$$

To classify a document, it is necessary to determine the distance vectors between evaluated document and determine the required number of documents in collections, i.e. the K value in KNN, that is closest to the evaluated document. The distance vector between two documents *x* and *y* is calculated by their weight value for each category *k* as in (8).

$$d(x, y) = \sqrt{\sum_{k=1}^{n} (w_{xk} - w_{yk})^2} \quad (8)$$

3. METHODOLOGY

This research design an Android application framework that allow developers to build their own implementation of spam classifier for Twitter contents. This research also develops several spam classifier libraries based on the designed application framework for use in an Android application. The classification libraries should be able to accept Twitter contents provided by the application, perform classification activities and reinstate the contents provided in two categories, i.e. spam and ham.

During implementation, Twitter trending topics and contents were limited to contents that were originating from Indonesia and were specifically written in Bahasa Indonesia. This research uses 100 Twitter contents (tweets) obtained from Twitter Search API that were manually labeled as spam or ham by hand as training data or referenced documents for the classification library specific use. This research also uses another 100 tweets for the accuracy and testing purposes.

3.1 Application Framework Design

The main purpose for the application is to present spam-free Twitter contents timeline by performing several activities to obtain tweets from various sources, e.g. from Twitter API, classify the tweets into spam and ham by utilizing classification method, and provide ham tweets.

Naïve Bayes and K-Nearest Neighbor (KNN) were taken as implementation target to classify spam and ham in tweets provided by the Android application. The classification method will be implemented as application module or library thus it could

be dynamically utilized by Android host-application. Both Naïve Bayes and KNN has similar required tasks to classify documents. They require data preparation or training, document preprocessing to improve accuracy, and the document labeling or classification process itself. Therefore, the main workflow of the Android application that utilizes the classification library depicted in UML Activity Diagram as shown in Figure 1.

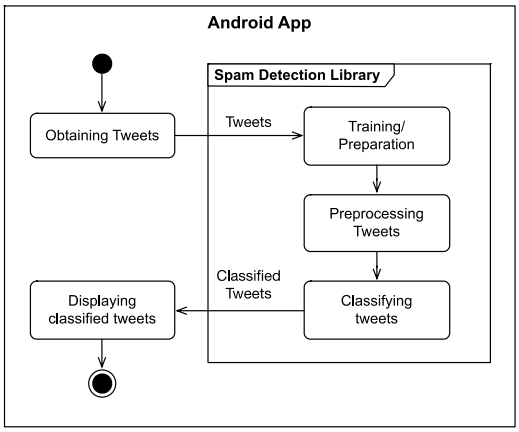

Figure 1. Android application workflow activity diagram

The Android host-application which host the spam detection library performs several steps in displaying spam-filtered tweets to user. First, the application should obtain Twitter contents from external sources such as Twitter Search API. Second, it provides obtained tweets to the classifier to be classified into spam and ham. Lastly, the application displays spam-free tweets to users.

The library performs three generic tasks to classify tweets, i.e. preparation, preprocessing, and document classification/labeling. Every classification method requires their own preparation or training procedures so that they are able to correctly and accurately classify the documents provided. The documents have to be preprocessed to filter stop words, symbols, and other non-printing characters to avoid natural language computing process problems.

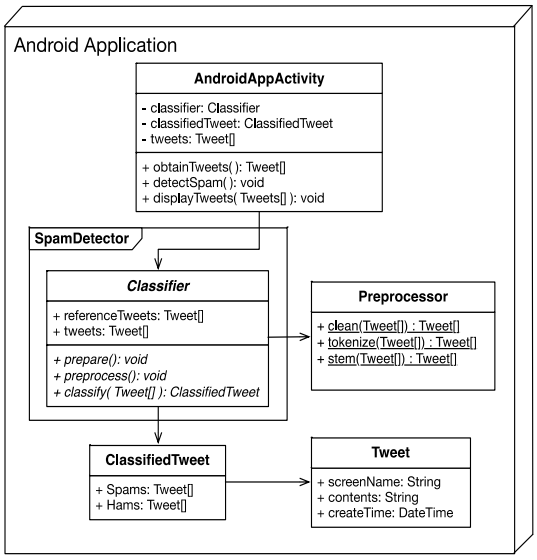

Figure 2. Android application framework design

The preprocessing process is a generic task that clean, tokenize, and stemming documents into several words which ready to be further processed into the classification process. There are also several stemming methods for a particular language available to stem words found in the documents. Therefore, the separation of preprocessing task from main classifier process would be a better design decision because this would provide more flexibility for application developers in implementing a specific algorithm for a particular process.

Application behaviors model as depicted in Class Diagram in Figure 2 shows application structure design of Android application that implemented in this research. In concept of Android mobile application, an Activity is a focused thing that user can interact with and managed as an activity stack [14]. The design should separate the Android application activity logic from the main computing process. Therefore, in the framework depicted in Figure 2 should allow the library to be dynamically utilized by the applications and allow developers to develop and implement their own method or algorithm.

3.2 Host-Application Implementation

This research implements an Android application as library's host-application during development and testing of the framework. The application obtains Twitter contents from Twitter Trends API to retrieve trending topic keywords list in Indonesia. The API returns current top 50 trending topics for a specific location parameter value namely Where on Earth ID (WOEID). Users are allowed to choose one trending topic as parameter for the application to retrieve keyword-related tweets from Twitter Search API.

The application instantiates one of the classifier implementation library, provide it with obtained tweets from Twitter Search API and ask the library to return spam and ham from the documents provided before showing spam free tweets to user. The implementation of a particular classifier should extend Classifier abstract class and implements all blueprints methods as specified in Classifier class. However, for the development and testing purposes, instead of showing only ham tweets, the application put spam and ham tweets into two different screens and put time execution measurement at several critical points such as at the beginning or the end of the preparation, preprocessing, or the classification process. Therefore, its characteristics could be further investigated.

3.3 Naïve Bayes Classifier Implementation

The preparation task of Naïve Bayes Classifier is implemented by generating a classifier model based on documents found in training dataset. The model stores every word along with its frequency of occurrence in spam and ham documents. The task also computes every word prior values for ham and spam category using (2).

The preprocessing task was implemented by a Preprocessor helper class that removes symbols and non-printing characters from documents, tokenizing it into words, removing meaningless stop words, and stemming its words into their stem-words. The implementation of classifier process for Naïve Bayes Classifier was computing document's ham posterior value and document's spam posterior value using (1). Its category was determined by its maximum-a-posteriori value as in (4).

3.4 KNN Classifier Implementation

The preprocessing task used in the KNN classifier reuses the Preprocessor helper class that implemented in Naïve Bayes classifier to provide the same functionality. For the preparation task, the classifier computes every reference documents (training dataset) weight for spam and ham category against spam and ham

documents collection respectively. Therefore, every document *j* in the dataset would have weight for spam (w_{jspam}) and weight for ham (w_{jham}).

The implementation of classifier process for KNN Classifier were computing every evaluated document x_i for their weight for spam (w_{ispam}) and weight for ham (w_{iham}), measure their distances with all referenced documents using based on their weights using (8), choose K value to pick K-closest documents and label/classify document x_i based on the most category in K-closest documents.

3.5 Accuracy and Performance Measurement

Prior conducting the measurement of system accuracy in identifying spam tweets, several white box and black box tests were conducted against the Android application. The application was tested to ensure that the application should be able to obtain tweets that related to user-specified trending topic using Twitter Search API and display Tweets into spam and ham category properly. White box testing was mainly focused in spam classification process to ensure that mathematical classification computation implemented correctly.

Several tests to measure system accuracy in recognizing spam in trending topic tweets were conducted using 10 to 100 of training data (n), which are having pre-classified label value, in the increments of 10 against 100 tweets as testing data. The percentage measurement resulting from comparing positive and false positive. The comparative accuracy measurement between Naive Bayes and K-NN is shown in Figure 3 and Figure 4.

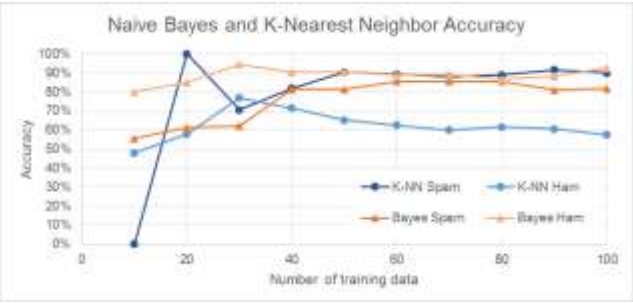

Figure 3. Naive Bayes and K-Nearest Neighbor Accuracy

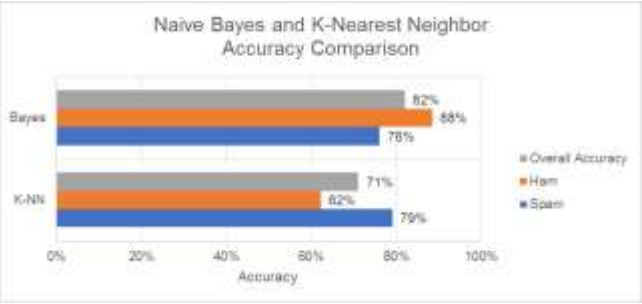

Figure 4. Naive Bayes and K-Nearest Neighbor Accuracy Comparison

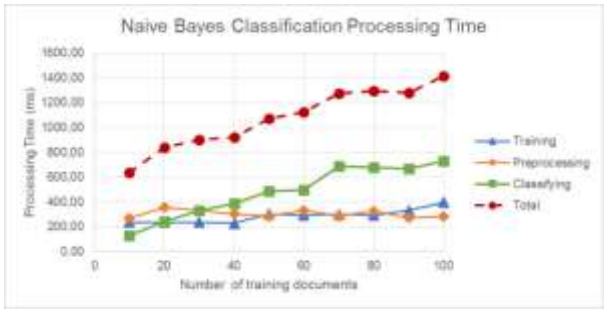

Figure 5. Naive Bayes Classification Processing Time

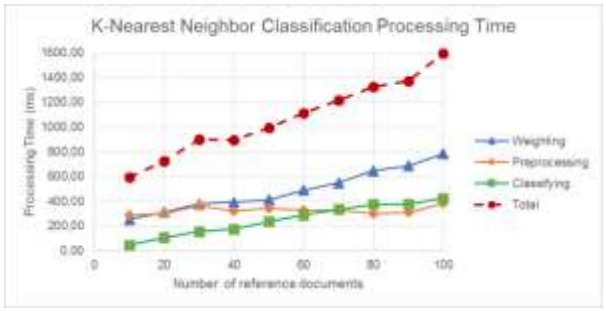

Figure 6. K-Nearest Neighbor Classification Processing Time

Figure 7. Naive Bayes and K-Nearest Neighbor Classification Processing Time Comparison

The performance test measures processing time of each processing stage in the implemented library, i.e. preparation, preprocessing, and classification stage performed by Naive Bayes and K-Nearest Neighbor methods. The tests were performed and measured on a physical Motorola Nexus 6 mobile device. The measurement result are shown in Figure 5, Figure 6, and Figure 7.

4. RESULTS AND DISCUSSION

An Android application that utilizes both Naïve Bayes and KNN classifier libraries has been successfully developed. The application is able to obtain tweets from Twitter Trends and Search API, identify spam contents from obtained tweets by utilizing one of the implemented spam detection libraries. Therefore, the designed framework allows application developers to implement several spam detection methods for use in an Android mobile application and optimize their application libraries usage. Developers are also allowed to develop their own preprocessing process in supporting their spam detection method. The blueprint of the classification library allow developer to develop their own spam detection library by accepting Tweet documents, process the documents into three sequential stages, i.e. preparation, preprocessing, and document classification, and reinstate classified Tweet documents.

Based on performance measurement conducted on the developed Android application that utilizes both implemented spam detection libraries, there are no significant performance differences between Naïve Bayes and KNN classifier method in all number of training or referenced documents as shown in Figure 7. Both methods processing time requirements are growing linearly as the number of training data or the number of referenced documents increases.

The increasing of the number of training data or referenced documents affect the preparation and classification stages in both methods. However, the preprocessing stage tend to have constant processing time in all number of training data or referenced documents. For a similar performance cost, Naïve Bayes tend to have better accuracy compared to KNN classification method. The number of training data or referenced documents affects overall accuracy in detecting spam and ham. The more training data or referenced documents involved in the computation process, the higher the detection accuracy that the system could get.

The same behavior does not occur in identifying ham tweets. The accuracy measurement data shows a relatively linear accuracy level at all level of the amount of training data. Ham tweets tend to have more unique, varying, and non-repetitive words than spam tweets other than have Uniform Resource Locators (URLs) or links in their contents. Therefore, ham tweets will be most likely reducing the likelihood probability value of every word in ham category. The accuracy test shows that the overall accuracy when the amount of training data increase. The accuracy level tends to increase logarithmically and begin to reach a value that close to the value of 90% at a certain amount of training data.

As this research future works, the implementation of Naive Bayes and KNN classification methods could be further realized as an independent Java Android library file, thus other developers who will develop Google Android-based applications that require spam and ham detection functionalities may simply include and use libraries they need. It is also possible to implements another classifier such as Support Vector Machine (SVM) or Random Forest classification as they were also known to perform very well in text classification [15] [16]. Allowing users to mark false positive contents into as additional spam and ham reference data would improves the spam and ham detection accuracy.

5. CONCLUSION

A Google Android mobile application, which implements the designed application framework has been successfully developed. The design allow one Android application utilizes more than one spam classification method to detect spam in Twitter contents. The methods are implemented in form of class library that extends Classifier class as its blueprint. The library should accept Twitter contents, classify the contents into spam or ham, and return them. The framework design allows spam classification method being implemented as a framework library and utilized as an Android Studio application project module to provide spam filtering functionality inside of Android application.

Naïve Bayes classification tend to have better accuracy in detecting ham over spam than K-Nearest Neighbor classification method. Conversely, K-Nearest Neighbor classification using TF-IDF weighting method have better accuracy in detecting spam than Naïve Bayes classification.

6. ACKNOWLEDGMENTS

This study is a partial research that is supported by Media, Game, and Mobile Technologies (MGM) Research Group of Computer Science Faculty, Universitas Brawijaya. Gratitude also given to all of our colleagues and participants in their contributions to this study.

7. REFERENCES

[1] Tom Rosenstiel, Jeff Sonderman, Kevin Loker, Maria Ivancin and Nina Kjarval. 2015. Twitter And The News: How People Use The Social Network to Learn About The World. (September 2015). Retrieved Oct 5, 2015 from https://www.americanpressinstitute.org/publications/reports/survey-research/how-people-use-twitter-news/

[2] Jeffrey Gottfried and Elisa Shearer. 2016. News Use Across Social Media Platforms 2016. (May 2016). Retrieved Oct 5, 2015. http://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/

[3] Grant Stafford and Louis Lei Yu. 2013. An Evaluation of the Effect of Spam on Twitter Trending Topics. In *2013 International Conference on Social Computing (SocialCom)*. Alexandria, VA, 373 – 378. IEEE. DOI= http://dx.doi.org/10.1109/SocialCom.2013.58

[4] Christian Arno. 2013. Global Social Media Trends in 2013. (Februari 2013). Retrieved Oct 5, 2015 from https://searchenginewatch.com/sew/how-to/2242467/global-social-media-trends-in-2013

[5] Networked Insights. 2015. How Dirty is Big Data?. March 2015. Retrieved Oct 5, 2015 from http://info.networkedinsights.com/Dirty-Data-LP.html

[6] Guido Schryen. 2007. *Anti-Spam Measures: Analysis and Design*. Springer-Verlag, Heidelberg.

[7] Christine Barry. 2013. Ham v Spam: what's the difference? (October 2013). Retrieved December 12, 2016 from https://www.barracuda.com/blogs/pmblog?bid=2152#.WGpQuVV97Dc

[8] Saini Jacob Soman and S. Murugappan. 2014. Detecting malicious tweets in trending topics using clustering and classification. In 2014 International Conference on Recent Trends in Information Technology (ICRTIT). IEEE. DOI= https://doi.org/10.1109/ICRTIT.2014.6996188

[9] Xianghan Zheng, Zhipeng Zeng, Zheyi Chen, Yuanlong Yu, and Chunming Rong. 2015. Detecting spammers on social networks. *Neurocomput.* 159, C (July 2015), 27-34. DOI=http://dx.doi.org/10.1016/j.neucom.2015.02.047

[10] Lin Li and Chi Li. 2015, Research and improvement of a spam filter based on Naive Bayes. In 7th IEEE International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). IEEE. DOI=https://doi.org/10.1109/IHMSC.2015.208

[11] Daniel Lowd and Pedro Domingos. 2005. Naive Bayes models for probability estimation. In *Proceedings of the 22nd international conference on Machine learning* (ICML '05). ACM, New York, NY, USA, 529-536. DOI=http://dx.doi.org/10.1145/1102351.1102418

[12] Srilaxmi Gandra. 2014. *Implementation Of Prototype to Detect Spam In YouTube Using The Application TubeKit And Naïve Bayes Algorithm*. Graduate Project Report. Texas A&M University, Corpus Christi, TX.

[13] Bruno Trstenjak, Sasa Mikac, Dzenana Donko. 2014. KNN with TF-IDF Based Framework for Text Categorization. In *2013 24th DAAAM International Symposium on Intelligent*

Manufacturing and Automation. 1356-1364. DOI=http://dx.doi.org/10.1016/j.proeng.2014.03.129

[14] Android Developers. 2016. Activity (Android Reference). Retrieved Sep 23, 2016 from https://developer.android.com/reference/android/app/Activity.html

[15] Ka-Chun Wong and Zhaolei Zhang. 2014. SNPdryad: predicting deleterious non-synonymous human SNPs using only orthologous protein sequences. *Bioinformatics* 30, 8 (April 2014), 1112-1119. DOI=http://10.1093/bioinformatics/btt769

[16] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim. 2014. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research* 15 (Oct. 2014). 3133-3181.