Cyberbullying Blocker Application for Android

Pawel Lempa*, Michal Ptaszynski*, Fumito Masui*

*Department of Computer Science
Kitami Institute of Technology
plempa@ialab.cs.kitami-it.ac.jp, {ptaszynski,f-masui}@cs.kitami-it.ac.jp

Abstract

Humiliating and slandering people through Internet existed almost since the beginning of communication through this medium. The introduction of new types of devices connecting to the Internet, such as smartphones and tablet computers further facilitated the process of cyberbullying. To deal with this problem we developed an application for Android smartphones which automatically detects a possible harmful content in text. The application can use one of two methods for detection of harmful messages: a method inspired by a brute force search algorithm applied to language modelling, or a method which uses seed words from three categories to calculate semantic orientation score SO-PMI-IR and then maximize the relevance of categories to specify harmfulness of a message. First tests showed that both methods are working under the Android environment.

1. Introduction

Cyberbullying, understood as humiliating and slandering people through the Internet, has existed almost as long as communication via Internet between people. The appearance of new devices, such as smartphones and tablet computers, which allow using this medium not only at home, work or school but also in motion, sometimes with different access points and simplified anonymity, has further exacerbated that problem. This motivated us to try prevent the problem of cyberbullying with the use of mobile devices. We decided to apply existing methods of detecting harmful contents on the Internet and focused on transferring them to a mobile device.

First step to fulfil this goal was developing a test application for Android devices which we describe in this paper. We created the application for two purposes. Firstly, to test whether it is possible to apply detecting algorithms previously developed on much more powerful machines on mobile devices such as smartphones. Secondly, to perform further test and find out which algorithm would work best. The first results of our study and its possible implications are described in this paper.

The outline of this paper is as follows. Firstly, we describe other available solutions and explain how our software is different from other cyberbullying detection software. Next, we describe our application, starting from the description of its functions and elements it contains, followed by the description of the interface. Further, we describe two methods used for detecting cyberbullying applied in the software. Finally, we describe the preliminary testing meant to verify the performance of the developed application, and discuss the results of those tests.

2. Related Work

With the popularization of mobile devices the problem of cyberbullying has become more noticeable. A number of research teams around the world have attempted to develop solutions for detection and mitigation of this problem (Ptaszynski et al. 2010; Dinakar et al. 2012; Nitta et al. 2013; Kontostathis et al. 2013; Ptaszynski et al. 2015). However, most of the research is still in a developmental phase and is yet to be applied in practice. On the other hand, there have been developed market solutions for the detection and mitigation of online bullying, having a capacity to deal with the problem to some extent. Unfortunately, such methods are usually based on the simplest solutions, which narrows the scope of their applicability.

FearNot! One of the examples of a software with a novel approach is FearNot![1]. The authors describe it as "an interactive drama/video game that teaches children strategies to prevent bullying and social exclusion." The development of this software, which uses psychology-inspired character AI, was supported by EU funded research projects Victec and eCircus. The approach taken by the developers, namely, not to detect and stigmatize cyberbullying behaviour, but rather to educate children not to become bullies, does indicate a deep insight in the problem. Unfortunately, the development of the software has stopped in early 2013.

BullyGuardPro. An example of a potentially effective software could be BullyGuardPro[2]. It is a software aimed at detecting cyberbullying activity around a user allowing her to "effectively respond, diffuse and halt cyberbullying and cyberpredation attacks". The software was developed by Lynne Edwards and April Kontostathis, who lead a team which as some of the first began the research on cyberbullying detection (Kontostathis et al. 2013). Unfortunately, at the time of writing, no details on the technology used in the software, nor its release date is yet known.

Samaritans Radar. On 29th October 2014, "Samaritans"[3], an organization focused on suicide prevention, launched an application called Samaritans Radar. It was a free Internet application for Twitter, which helped users monitors their friends' Tweets. Main function of the application was alerting a user when it spotted anyone in the user's online surroundings, who could be either bullied, depressed or sending disturbing suicidal signals. Unfortunately, due to serious data protection and privacy issues, which were noticed by the users soon after the time of launch, the application was closed permanently on 10 March 2015, six months after its release.

[1] http://sourceforge.net/projects/fearnot/
[2] http://www.bullyguardpro.com
[3] http://www.samaritans.org/

Uonevu. An interesting approach to the detection of cyberbullying in messages has been developed by researchers from Trinity College Dublin and National Anti-Bullying Research and Support Centre, under the codename Uonevu (meaning "bullying" in Swahili language). The software is meant to detect particularly non-literal forms of bullying and negative stereotyping. The software works by applying a semantic knowledge base to associate concepts with each other. And example is associating the concept of "fat/obese people" with the word "pizza", which in a sentence such as "Hey, Jane, are you going to eat a whole pizza tonight?" would indicate an instance of cyberbullying. The major problem here becomes creating a large enough knowledge base of stereotypes. Unfortunately, at the moment of writing the database contains only 57 stereotypes, which is not sufficient for effective functioning of the software. However, the project is still in its developmental phase and could be an interesting solution when finished.

Twitter New Policy. On February 26, 2015, Twitter independently released its new policy regarding safety and misbehaviour among its users[4]. Twitter allowed users to report particular tweets as harassment incidents. This provided the users who became victims of online bullying a tool for personally reacting to the bullying attacks. An account of a confirmed bully becomes locked and can be reopened only under the condition that the bully deletes her harmful tweets. This way of dealing with cyberbullying is not yet a software *per se*, but Twitter aims at detecting bullying messages automatically in the future. For the time being however, they declared increasing the number of staff in their support team focused on handling abuse reports.

ReThink. An example of a recent popular solution which still works, could be ReThink[5]. It is an application for smartphones, which shows a pop-up warning message when user tries to send a message containing harmful contents. The idea of informing a user about possible harmfulness of a message has been noticed even earlier as an effective mean to make user re-evaluate her message before making it publicly available (Masui et al. 2013, Patent Application No. 2013-245813). Unfortunately, though ReThink is a good example of a quick and ad-hoc response to the cyberbullying problem, its algorithm for the detection of harmful contents is based on simple keystroke logging and detecting vulgar and harmful words within the string of characters. This makes it inapplicable for more sophisticated contents, not including vulgar expressions. It also fails when user makes a mistake during writing and e.g., uses a backspace, since the "backspace" character is also recorded and hinders detection of harmful words.

PocketGuardian. PocketGuardian is a newly released (September 2015) software for parental monitoring, that "detects cyberbullying, sexting, and explicit images on children's mobile devices."[6] By using machine learning techniques the software provides a statistical probability that the content (sentence, tweet, e-mail, or image) is inappropriate. An advantage of this software is that it focuses not only on textual content, but includes in its monitoring range the ability to detect explicit images. Its disadvantage could be its price ($3.99 per month). The exact technology behind the software (e.g., applied machine learning algorithms, size of the training lexicon or corpus) is yet unknown. Moreover, as the software is aimed at parents trying to monitor their children's mobile devices, the developers will need to address the questions regarding ethics of the use if such software and its influence on parent-child trust relationship.

In comparison with the software described above, the application presented here distinguishes itself in the following ways. Similarly to ReThink it provides a tool for the user to reflect on their own written messages. However, differently to ReThink, it shows the user which exactly words or sentence patterns were considered as inappropriate. Moreover, it does not only use simple keystroke logs, but uses various Artificial Intelligence methods (at present two) to spot any undesirable contents. It focuses only on textual contents, however, differently to the PocketGuardian we do not plan to make profit of the application. Moreover, since the application is aimed to be used by the user directly, all ethical issues and any influence on parent-child trust relations, as well as any privacy issues, like in the Samaritans Radar application, do not become a problem. The methods applied so far have been in development for over six years, thus the problem of insufficient data, such as in the Uonevu project is also resolved.

3. Application

The application was created for devices supporting Android 4.2.2 (API level 17, codename Jelly Bean) or higher. In this process Java 8 and Android Studio were used. The application contains one activity responsible for the interaction with the user. For the process of checking the text for possible harmful content is starts a background thread. Thanks to that the user still can use his device even if checking process takes a while.

To test which algorithm will be the most sufficient on mobile devices there were implemented two detection methods. User can test both of them in the same application, thanks to which it will be easier to choose the algorithm used for final version of the application.

Figure 1 represents an activity diagram of the detection process. Depending on the choice of the method of detection, application may require an Internet connection.

The application is created so that removing or adding a new method for detecting cyberbullying was possible, and did not affect operation of the application itself.

3.1. Description of Application Interface

Application interface is designed in accordance with the standards for Android operating system (Android Developers, 2015). Interaction with a user is intuitive and all important information is accessible in main window. Figure 2 represents the interface with description of each important element.

Below we describe the interface of the application. The numbers correspond to the numbers in red from Figure 2.

[4]https://blog.twitter.com/2015/update-on-user-safety-features

[5]http://www.rethinkwords.com/

[6]https://gopocketguardian.com/

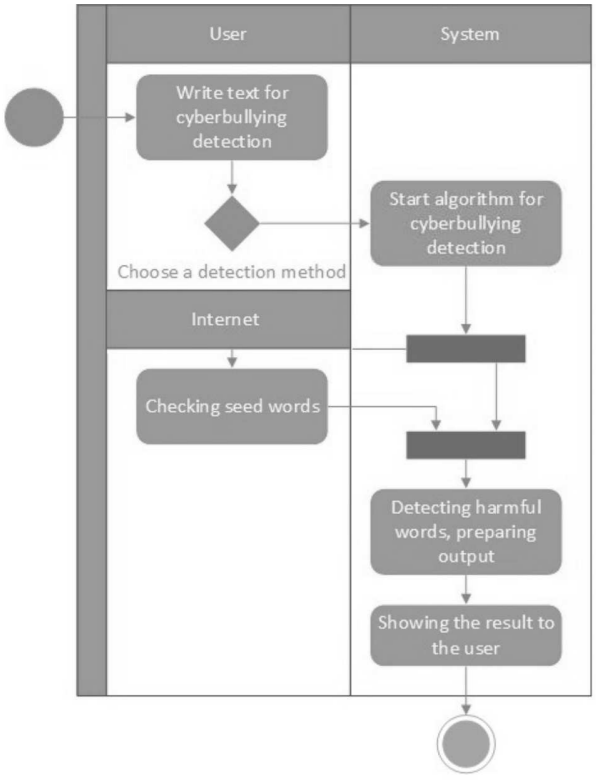

Figure 1: Activity diagram of the application.

1. Name of the application
2. Settings, contains information about application, and short description of used methods for cyberbullying detection
3. Input for text to classify
4. Choice of a method for cyberbullying classification
5. Button "Check" - Clicking on it starts the process of checking the entered text
6. Button "Clear" - Clears the input and feedback fields
7. Field showing the feedback of used method for analysing the text

3.2. Harmful Content Detection Process

Below we describe the process of detecting possible harmful contents.

1. Sentence input. Firstly, user inputs a sentence she wants to check. There is no limits how long it can be. User can write several sentences at once. The only limitation is phone memory for the TextView control class, which is usually set to 9,000 characters. Due to limitations of smartphone screen size, the application allows the maximum of five visible lines of input, for more the user needs to scroll down. However, larger amount of words to check will slow down the detection process for both methods. User can input any contents she wants, in which the system will detect words and patterns recognizable by the used algorithm.

2. Selection of detection method. A user should choose one method from a list of all algorithms. Short description of methods can be found in the Settings tab of the application. The option to choose the method is only available in the initial test version of the application and is intended to help select the best algorithm for mobile devices.

3. Launch of the checking process.

Figure 2: Interface of the developed application with numbers (in red) marking each element of the interface.

4. Feedback. The processing of entered text results in one of two possibilities. Either the selected method does not detected any harmful patterns or it does detect them. In the first case the returned feedback informs the user that no harmful words or patterns were detected in the input. The feedback also contains, information on which classification method was used and the text entered by the user (see right part of Fig. 3). In the second case the feedback contains information about detected harmful words and patterns, which method was used, the entered text and the detected harmful patterns represented in bold and red-coloured font (see centre part of Fig. 3).

4. Methods Description

For test purposes in the developed application we plan to apply a number of different classification methods. In the first version of the application, described here, for the purpose of detection of harmful content in text on mobile devices, two previously developed methods of detection were used. Both of them have been adapted to the Java language and Android environment. The methods can be replaced and updated in the future. We also plan on adding other methods.

4.1. Method A

As the first method, further called 'Method A', we used the one developed by (Ptaszynski et al. 2015). The method classifies messages as harmful or not by using a classifier trained with a language modelling method based on a brute force search algorithm applied to language modelling.

The method uses sophisticated sentence patterns with disjoint elements automatically extracted with a novel language modelling method developed by (Ptaszynski et al. 2011). The patterns are defined as ordered combinations of sentence elements which are used for brute-force searching within input sentences.

In the training of this method, at first, ordered non-repeated combinations were generated from all elements of training sentences. In every n-element sentence there is k-number of combination clusters, such as that $1 \leq k \leq n$, where k represents all k-element combinations being a subset of n. In this procedure all combinations for all values of k were generated. The number of all possible combinations is equal to the sum of all k-element combination clusters (see eq. 1).

$$\sum_{k=1}^{n} \binom{n}{k} = \frac{n!}{1!(n-1)!} + \frac{n!}{2!(n-2)!} + ... + \frac{n!}{n!(n-n)!} = 2^n - 1 \quad (1)$$

Next, all non-subsequent elements were separated with an asterisk ("*"). Occurrences O of patterns from each side of the dataset are used to calculate their weight w_j (eq. 2). Finally, the score of a new input sentence is calculated as a sum of weights of patterns found in the sentence (eq. 3).

$$w_j = \left(\frac{O_{pos}}{O_{pos} + O_{neg}} - 0.5 \right) * 2 \quad (2) \qquad score = \sum w_j, (1 \geq w_j \geq -1) \quad (3)$$

The efficiency of this method is essentially associated with the computing power of the device that is currently running. As all patterns used in classification are stored on the mobile device, the method can operate locally, and does not require an Internet connection.

4.2. Method B

Second method, further called 'Method B', uses a list of seed words from three categories to calculate semantic orientation score SO-PMI-IR and then maximize the relevance of categories with input sentence according to a method developed by (Nitta et al. 2013). There are three steps in the classification of the harmfulness of input:

1. Phrase extraction,
2. Categorization and harmful word detection together with harmfulness polarity determination,
3. Relevance maximization.

This method is an extension of the method proposed by (Turney et al. 2002) to calculate the relevance of words with specified categories according to the equation 4, where p_i is a phrase extracted from the input, w_j are three words that are registered in one category of harmfulness polarity words, $hits(p_i)$ and $hits(w_j)$ are Web search hits for each category for p_i and w_j respectively, $hits(p_i \& w_j)$ is a number of hits when p_i and w_j appear on the same Web page. Finally, $PMI - IR(p_i, w_j)$ is the relevance of p_i and w_j.

$$SO\text{–}PMI\text{–}IR(p_i, w_j) = log_2 \left\{ \frac{hits(p_i \& w_j)}{hits(p_i) hits(w_j)} \right\} \quad (4)$$

Table 1: Comparison of different features for two methods.

	Method A	Method B
Avg. # of words / patterns used in classification	11,832,430	9
Internet connection required	NO	YES
Avg. time for one sentence (in seconds and minutes)	88.46 s (1.47 m)	8.41 s (0.14 m)
Best Precision	89%	91%
Recall at Best Precision	34%	9%
F-score for Best Precission	49%	16%

Turney's method was extended to work not only on words, but on phrases. The phrases are automatically extracted from input sentences using dependency relations. Next, for all of the phrases the relevance is calculated with seed words from multiple categories of harmful words. The degree of association for each category, SO-PMI-IR score is maximized so that the maximum value achieved within all categories is considered as the harmfulness $score$ representative for the input sentence, and is calculated according to the equation 5.

$$score = max(max(SO\text{–}PMI\text{–}IR(p_i, w_j))) \quad (5)$$

This method does not require excessive computer power, and works on a small list of seed words, which can be further refined and optimized as showed by (Hatakeyama et al. 2015). On the other hand, this method requires a stable Internet connection for calculating PMI-IR score of the phrases with each group of seed words.

5. Preliminary Tests

We performed preliminary tests with the developed applications. The tests were not meant to check the validity of the applied methods, as this was already confirmed in previous papers (Nitta et al. 2013; Hatakeyama et al. 2015; Ptaszynski et al. 2015). We verified only whether the applied classification methods performed correctly under the new environment, and if they returned a proper feedback.

For the purpose of testing the application we prepared a set of sentences, contents of the dataset applied in previous research, from which some contained harmful words and some did not. The sentences were entered one by one to the application input field and tested by both methods. Examples of outputs for harmful and non-harmful sentences, are represented in the centre and right part of Fig. 3, respectively. As our final goal, we plan to release the application for multiple languages. However, as both of the applied cyberbullying detection methods work presently for the Japanese language, at the moment the application is also developed for this language. However, for the description purposes in this paper we present both the application interface and the sentence examples translated in English.

Tests were performed on virtual devices emulated by Genymotion engine (Sony Xperia with Android 4.2.2 and Google Nexus 10 with Android 5.0) and on Smartphone LG G2 with Android 5.0.2. The tests were focused on performance of used algorithms on mobile devices, not on usability of the application because present version of the application was created purely for verification if detection algorithms work correctly on mobile devices and which of them is the best to use in the full version of application. Depending on the classification method, the de-

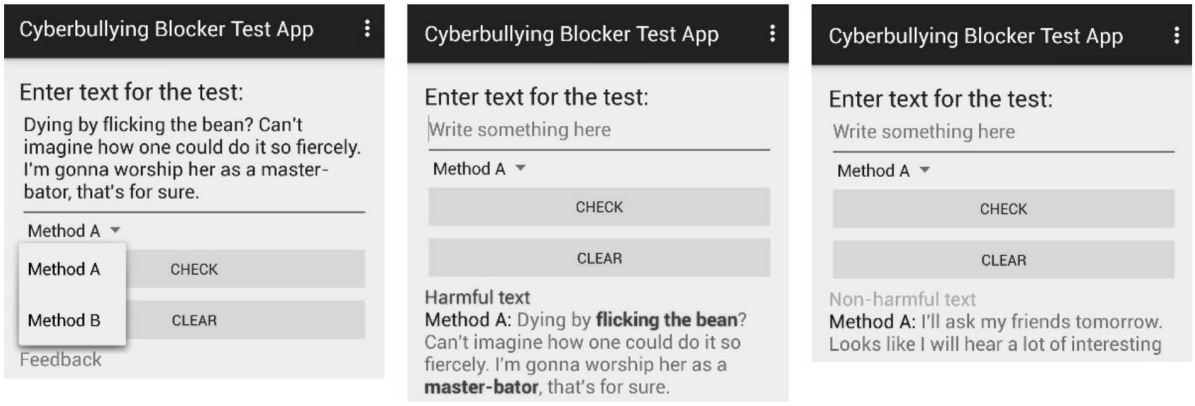

Figure 3: Text input and choice of processing method (left); harmful text output (center); non-harmful text output (right).

tected harmful words can differ. Moreover the processing speed of detection is associated with the type of device used (virtual smartphone, budget smartphone or high-end smartphone) and the length of text.

There was no clear winner. Method A achieves better results and could be advantageous due to its lack of need for Internet connection. However, due to its use of massive numbers of patterns it takes about ten times longer to process one sentence. On the other hand, Method B works faster, but needs Internet connection and achieves worse results. However, if the method itself is improved enough to match Method A, it could be advantageous, since Internet connection has become less a problem these days. Results of the experiments are summarized in Table 1.

6. Conclusions and Future Work

In this paper we presented an Android application for detection of entries that contain cyberbullying – contents humiliating and slandering other people. Cyberbullying is a growing problem in Internet environment, mostly due to the rapid development of mobile devices based on a connection with the Internet.

In the proposed application we applied two methods to find potential harmful words. Both of them have proven themselves in the detection of cyberbullying, as described in previous research. Main difference between them from the point of view of software development, was that Method B to work requires an access to the Internet, while retaining a need for low computing power. Method A, on the other hand, does not require Internet connection, but needs sufficient computing power. However, since future use of this application is inextricably linked with communication via the Internet, and each new generation of smartphones represents a major technological leap, both differences do not affect the usefulness of the methods and in result – the developed application.

In the near future we have three main goals. First is to check how we can improve a performance of detection of the harmful words by using another methods or by optimizing the existing ones – in particular for mobile devices. The second goal is to find the best way to implement our software to communication applications used via Internet, such as Facebook or Twitter, whilst retaining all safety and privacy policies. For this problem for now we have two possible solutions: to prepare a plugin for existing applications or create a virtual keyboard for mobile devices. Finally, the last issue is expanding the scope of our potential users by creating version of the application for other dominant mobile systems, e.g., iOS.

7. References

Android Developers, *http://developer.android.com/*, accessed on 2015.10.21.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Commonsense Reasoning for Detection, Prevention and Mitigation of Cyberbullying, *ACM Transactions on Intelligent Interactive Systems*, Vol. 2, No. 3.

Suzuha Hatakeyama, Fumito Masui, Michal Ptaszynski, Kazuhide Yamamoto. 2015. Improving Performance of Cyberbullying Detection Method with Double Filtered Point-wise Mutual Information, In *Demo Session of The 2015 ACM Symposium on Cloud Computing 2015 (ACM-SoCC 2015).*

Kontostathis, A., Reynolds, K., Garron, A., Edwards, L. 2013. Detecting cyberbullying: query terms and techniques. In *5th Annual ACM Web Science Conference*, pp. 195-204.

Fumito Masui, Michal Ptaszynski, Taisei Nitta. 2013. Intānetto-jō no yūgai kakikomi kenshutsu sōchi oyobi kenshutsu hōhō [Device and method for detection of harmful entries on the Internet] (In Japanese), Patent Application Number: 2013-245813.

Taisei Nitta, Fumito Masui, Michal Ptaszynski, Yasutomo Kimura, Rafal Rzepka, Kenji Araki. 2013. Detecting Cyberbullying Entries on Informal School Websites Based on Category Relevance Maximization. In *Proceedings of IJCNLP 2013*, pp. 579-586.

Michal Ptaszynski, Pawel Dybala, Tatsuaki Matsuba, Fumito Masui, Rafal Rzepka, Kenji Araki, and Yoshio Momouchi. 2010. In the Service of Online Order: Tackling Cyber-Bullying with Machine Learning and Affect Analysis, *Int. Journal of Computational Linguistics Research*, Vol. 1, Issue 3, pp. 135-154.

Michal Ptaszynski, Rafal Rzepka, Kenji Araki, Yoshio Momouchi. 2011. Language combinatorics: A sentence pattern extraction architecture based on combinatorial explosion. In *International Journal of Computational Linguistics (IJCL)*, Vol. 2, No. 1, pp. 24-36.

Michal Ptaszynski, Fumito Masui, Yasutomo Kimura, Rafal Rzepka, Kenji Araki. 2015. Brute Force Works Best Against Bullying, In *IJCAI 2015 Workshop on Intelligent Personalization (IP 2015)*

Peter D. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, In *Proceedings of ACL 2002*, pp. 417-424.