Detecting spam comments on Indonesia's Instagram posts

Ali Akbar Septiandri[1] **and Okiriza Wibisono**[2]

School of Informatics, The University of Edinburgh, 11 Crichton St, Edinburgh EH8 9LE, UK

E-mail: [1]`s1500216@sms.ed.ac.uk`, [2]`s1367442@sms.ed.ac.uk`

Abstract. In this paper we experimented with several feature sets for detecting spam comments in social media contents authored by Indonesian public figures. We define spam comments as comments which have promotional purposes (e.g. referring other users to products and services) and thus not related to the content to which the comments are posted. Three sets of features are evaluated for detecting spams: (1) hand-engineered features such as comment length, number of capital letters, and number of emojis, (2) keyword features such as whether the comment contains advertising words or product-related words, and (3) text features, namely, bag-of-words, TF-IDF, and fastText embeddings, each combined with latent semantic analysis. With 24,000 manually-annotated comments scraped from Instagram posts authored by more than 100 Indonesian public figures, we compared the performance of these feature sets and their combinations using 3 popular classification algorithms: Naïve Bayes, SVM, and XGBoost. We find that using all three feature sets (with fastText embedding for the text features) gave the best F_1-score of 0.9601 on a holdout dataset. More interestingly, fastText embedding combined with hand-engineered features (i.e. without keyword features) yield similar F_1-score of 0.9523, and McNemar's test failed to reject the hypothesis that the two results are not significantly different. This result is important as keyword features are largely dependent on the dataset and may not be as generalisable as the other feature sets when applied to new data. For future work, we hope to collect bigger and more diverse dataset of Indonesian spam comments, improve our model's performance and generalisability, and publish a programming package for others to reliably detect spam comments.

1. Introduction

As reported in [1], Instagram has 22 million monthly active users in Indonesia from 500 million users worldwide. With 95 million photos and videos posted every day, Instagram has obviously become a prominent photo and video sharing platform in the world. However, just like e-mails, the high number of activities mean that there is a chance that people could get more attention even when they post totally irrelevant comments which we also know as spam.

Indonesia, in particular, has this problem where some users put advertising on public figures' posts as comments. This problem has gone too far to the fact that the first 50 comments on a public figure's post could be all spam[1]. Like e-mail spam, we believe that these comments can be very annoying and should be automatically filtered.

To the best of our knowledge, we have not seen any investigation on Instagram's spam on Indonesian accounts before. The case might be platform specific, but the insight from this study

[1] e.g. username: @cita_citata, post_id: BIjHGG3BUEK

could hopefully be used to similar cases, e.g. spam comments on blogs or e-mail spam. This is because the machine learning algorithms we used in our study which have been proven to work well in general. Our focus is less on finding the algorithm that works best on our dataset and more on identifying the salient features to detect spam comments.

2. Related Work

While we might be able to detect spam to some extent by using origin-based filters (e.g. using IP or email address), in the case of spam email, content-based filtering is more common to do [3]. The same thing also applies to spam comments, as people can easily create new accounts for this spamming purpose. However, content-based filtering also comes with some problems. For instance, spammers could obfuscate their message (e.g. by writing "f r 3 3" instead of "free") to trick the filter [2].

Machine learning approach, which can automatically filter the spam by building adaptive model, has become more popular nowadays. Before sending the features to be learned by the algorithms, according to [2], the structure of a spam filter can be grouped into:

(i) tokenization, which extracts the words in the message body;
(ii) lemmatization, reducing words to their root forms;
(iii) stop-word removal, eliminating some words that often occur in many messages;
(iv) representation, which converts the set of words present in the message to a specific format required by the machine learning algorithm used.

Nevertheless, as it was also pointed out in the study, not all of the steps are mandatory.

As the name might suggest, words are the features in content-based spam filtering. The problem now is how to represent them properly for the algorithms. Some of the representation that has been used in content-based spam filtering are bag-of-words (BoW) [5], term frequency-inverse document frequency (TF-IDF) [6], and binary representation of word occurences [4]. In [4], it was also considered to use of upper case words as one of the features.

Since using BoW and TF-IDF can result in a sparse matrix, we can reduce the dimension by using Latent Semantic Analysis (LSA). As explained in [9], this method applies Singular Value Decomposition (SVD) to the matrix so that we can learn "expected contextual usage of words in passages of discourse." LSA also helps us to make the learning process faster.

In more recent work [7], we can see the improved version of skip-gram model [8], "where each word is represented as a bag-of-character n-grams." This method enriches the word vectors with subword information on rare words. It also has the advantage of not needing any preprocessing of the data.

3. Methodology

3.1. Features

We used several techniques for representing the comments as follows:

(i) Binary Bag-of-Words with LSA;
(ii) TF-IDF with LSA;
(iii) Word2Vec using skip-gram model.

The first two representations were made using the library provided in scikit-learn[2] [10]. We set the minimum document frequency (`min_df` parameter) to be 5. We then reduced the dimension of BoW and TF-IDF representations to 100 using LSA. The chosen dimension is the same as

[2] `http://scikit-learn.org/stable/`

the default output dimension in the Word2Vec library we used in this study, i.e. fastText[3] [7]. We did not apply any lemmatization or stop-word removal in our study.

The word vectors produced by Word2Vec are 100-dimension vectors for each word in the document. We then took the average of all word vectors that constitutes each document, yielding 100-dimension document vectors. These vectors can then be used as the features in our experiment.

These features alongside some numerical features that can be easily extracted from the text, e.g. number of tokens, number of upper case words, number of numerical characters, percentage of emoji, and the length of the text (see Table 1), were then passed onto a number of machine learning algorithms. We call these as *basic* features in our experiemnt. We also utilised some hand-engineered keyword patterns in our experiment (see Table 2), which later on in this paper are known as *keywords* features.

Table 1. List of basic features

Name	Description
`n_token`	Number of tokens
`n_capital`	Number of upper case words
`n_emoji`	Number of emojis
`n_unique_emoji`	Number of unique emojis
`n_number`	Number of numerical characters
`n_mention`	Number of mentions
`%_capital`	Percentage of upper case words
`%_number`	Percentage of numerical characters
`%_emoji`	Percentage of emojis
`%_unique_emoji`	Percentage of unique emojis
`log_char`	$log(number of characters)$
`has_phone_number`	Contains phone number
`has_bbm_pin`	Contains BlackBerry® Messenger PIN

3.2. Algorithms

In this study, we used Naïve Bayes (NB), Support Vector Machine (SVM) with RBF kernel, and XGBoost as the algorithms to classify the data. We based our selection of algorithms on a generic review of machine learning algorithms performance in [11] where SVM with RBF kernel and Random Forest turned out to be performing quite well in many cases. XGBoost [12] as a variation of decision trees method that shares some similarities with Random Forest, such as in the column sampling method and ensemble trees concept, became our choice of implementation because it has been proven to be successful in many machine learning competitions. On the other hand, we also incorporated the results from Naïve Bayes algorithm as the baseline for our study because of its simplicity and efficacy in practice [13].

We held out 20% of the dataset to be used as the test set later on in our study. We evaluated our models using cross-validation and concurrently tuned the hyperparameters. Using consistent random number generator, the hyperparameters we tested in our study can be seen in Table 3. The best models were then tested on the test set to know how well our model performs in general.

[3] `https://github.com/facebookresearch/fastText`

Table 2. Hand-engineered keyword patterns

Regular Expression
`bbm|pin bb|line|whatsapp`
`(ch?ec?k|intip|liat|kepoin)( out)?( (our|my))?  (ig|insta|koleksi|profil)`
`cekidot`
`dada|herba|langsing|payudara|pemutih|penggemuk|peninggi|tahan lama|tinggi badan`
`efek samping`
`follow`
`free (delivery|ongkos|ongkir|pengiriman)|gratis|murah|promo|terjangkau`
`garansi|kualitas`
`impor`
`invit`
`jerawat`
`jual|sell`
`langganan`
`luar biasa`
`mampir`
`nyaranin`
`order`
`password`
`penghasilan`
`produk`
`s(o|u|ou)venir`
`stock|stok`
`yu+k`

Table 3. List of hyperparameters

Classifier	Hyperparameters	Values tested
Naïve Bayes	`fit_prior`	[True, False]
SVM	`C`	[2^{-5}, 2^{-3}, 2^{0}, 2^{3}, 2^{5}, 2^{7}, 2^{9}]
	`gamma`	[2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{0}, 2^{3}, 2^{5}, 2^{7}, 2^{9}]
XGBoost	`n_estimators`	[10, 20, 50, 100]
	`subsample`	[0.5, 0.6, 0.7, 0.8, 0.9, 1.]
	`colsample_bytree`	[0.5, 0.6, 0.7, 0.8, 0.9, 1.]
	`reg_alpha`	[0., 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1., 5., 10.]
	`reg_lambda`	[0., 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1., 5., 10.]
	`scale_pos_weight`	[1., 2.28, 3.55, 4.83, 6.11, 7.38, 8.66, 9.94, 11.22, 12.49]

3.3. Evaluation Metrics

Since the distribution of the classes in the dataset is imbalanced, we used precision, recall, and F_1-score to evaluate our models. These metrics are formulated as:

$$\textbf{precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\textbf{recall} = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \cdot \frac{\textbf{precision} \cdot \textbf{recall}}{\textbf{precision} + \textbf{recall}} \tag{3}$$

where TP, TN, FP, FN denote true positives, true negatives, false positives, and false negatives respectively.

In this study, we focused on getting the highest F_1-score. Therefore, most of the comparison would be based on F_1-score. We only provide the precision and recall for the best model achieved.

4. Experiments

4.1. Dataset

We collected 24,602 comments from 500 posts authored by 104 Indonesian public figures. We annotated those comments ourselves by looking at the original post to see the relevance of the comments. We focused on comments that are promoting websites or products to be categorised as spam. Eventually, we got 22,743 ham and 1,859 spam in the dataset.

4.2. Results

Our preliminary result from using only basic features as mentioned in Table 1 was promising. We got 0.7775 of F_1-score using XGBoost. We can see in Figure 1 that the logarithmic value of number of characters by itself can be a discerning feature.

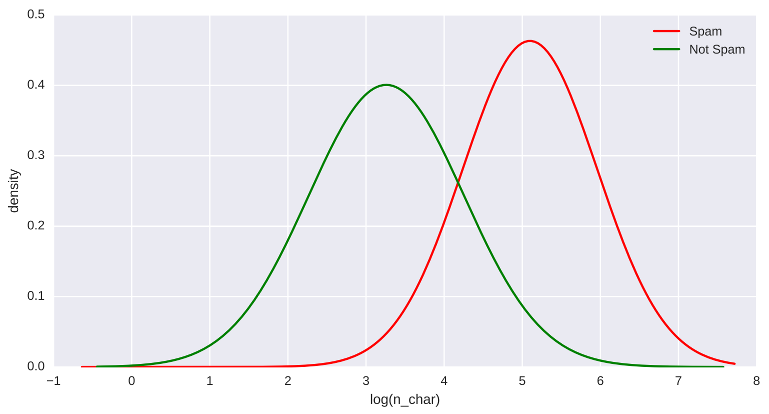

Figure 1. Difference in length on spam vs ham

We achieved an even better result using the features from hand-engineered keyword patterns (see Table 4). However, as the keywords can be altered in the future by the spammers to trick the classifiers, we cannot rely solely on this result. Thus, we continued our experiments with the representations mentioned in subsection 3.1.

By using only word vectors extracted from the text, we can see in Table 5 that the algorithms, with the exception of Naïve Bayes, managed to beat the corresponding algorithms with our hand-engineered keywords. Using word vectors from fastText resulted in the best models from three representations.

Our next effort was combining the features in Table 1 with the word vectors. We wanted to see how adding the basic features and our hand-engineered keyword patterns could improve the models. The result for this experiment can be seen in Table 6. Note that "+b" means we were

Table 4. F_1-scores of preliminary results

Classifier	Basic	Keywords	Basic + Keywords
Naïve Bayes	0.6667	0.8391	0.8772
SVM	0.7688	0.8726	0.9093
XGBoost	0.7775	0.8655	0.9089

Table 5. F_1-scores on word vectors

Classifier	BoW	TF-IDF	fastText
Naïve Bayes	0.5361	0.2385	0.8249
SVM	0.9074	0.9074	0.9398
XGBoost	0.9121	0.9089	0.9248

adding basic features, while "+k" means we were adding keyword patterns features when we trained the models.

Table 6. F_1-scores on combined features

Classifier	BoW+b	BoW+b+k	TF-IDF+b	TF-IDF+b+k	fastText+b	fastText+b+k
Naïve Bayes	0.5802	0.7366	0.7331	0.8818	0.8621	0.8801
SVM	0.9399	0.9309	0.9373	0.9349	0.9523	0.9601
XGBoost	0.9268	0.9381	0.9377	0.9436	0.9512	0.9512

We can see that adding our proposed basic features improved all the F_1-scores. However, adding keyword patterns to word vectors with basic features did not improve the performance significantly. We confirmed this by using McNemar's test defined as follows:

$$\chi^2 = \frac{(b-c)^2}{b+c} \tag{4}$$

where b and c denote the number of difference in class predictions (test 1 negative and test 2 positive, and vice-versa).

The p-value we got from McNemar's test on SVM classifiers is 0.22, which means the difference is not statistically significant ($p > 0.05$). This result indicated that using word vector representation with semantic analysis is sufficient to detect important features from the text. Having said that, the best F_1-score in our experiment was from adding basic and keyword patterns features to SVM with fastText: 0.9601.

5. Conclusions and Future Work

Our experiments show that employing fastText can produce robuster models. This technique can also minimise the time it takes for the feature extraction step. However, there is still a possibility to improve the document vectors by taking the maximum or signed maximum values for each dimension of the word vector instead of averaging them out.

This study also corroborates the thorough investigation in [11] where it is stated that SVM-RBF and Random Forest performs quite well on many cases. These algorithms combined with fastText and our proposed basic features and keyword patterns turned out to be the best models to identify spam comments. In our experiment, SVM and XGBoost got the best F_1-scores of 0.9601 and 0.9512 respectively.

References

[1] Fajrina H N 2016 Ada 22 Juta Pengguna Aktif Instagram dari Indonesia URL `http://www.cnnindonesia.com/teknologi/20160623112758-185-140353/ada-22-juta-pengguna-aktif-instagram-dari-indonesia/`
[2] Guzella T S and Caminhas W M 2009 *Expert Systems with Applications* **36** 10206–10222
[3] Nosseir A, Nagati K and Taj-Eddin I 2013 *IJCSI International Journal of Computer Science Issues* **10** 1694–0814
[4] Drucker H, Wu D and Vapnik V N 1999 *IEEE Transactions on Neural networks* **10** 1048–1054
[5] Metsis V, Androutsopoulos I and Paliouras G 2006 *CEAS* pp 27–28
[6] Joachims T 1996 A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep. DTIC Document
[7] Bojanowski P, Grave E, Joulin A and Mikolov T 2016 *arXiv preprint arXiv:1607.04606*
[8] Mikolov T, Sutskever I, Chen K, Corrado G S and Dean J 2013 *Advances in neural information processing systems* pp 3111–3119
[9] Landauer T K, Foltz P W and Laham D 1998 *Discourse processes* **25** 259–284
[10] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M and Duchesnay E 2011 *Journal of Machine Learning Research* **12** 2825–2830
[11] Fernández-Delgado M, Cernadas E, Barro S and Amorim D 2014 *J. Mach. Learn. Res* **15** 3133–3181
[12] Chen T and Guestrin C 2016 *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '16 (New York, NY, USA: ACM) pp 785–794 ISBN 978-1-4503-4232-2 URL `http://doi.acm.org/10.1145/2939672.2939785`
[13] Rish I 2001 *IJCAI 2001 workshop on empirical methods in artificial intelligence* vol 3 (IBM New York) pp 41–46