Comparative Evaluation of SMS Spam Filtering Techniques

Diego Antognini and **Panayiotis Danassis**
École Polytechnique Fédérale de Lausanne
Email: {diego.antognini, panayiotis.danassis}@epfl.ch

Abstract

SMS spam is an increasing threatening problem, especially in Asia and in developing countries, where the volume of SMS spam messages has dramatically increased over the past few years. The SMS spam filtering problem can be approached by a plethora of different techniques, ranging from simple solutions such as white and black listing, to more elaborate content-based filtering, such as the methods employed to combat email spam messages. Nevertheless, compared to the email spam filtering problem, the sms one presents many unique challenges. This is partly due to lack of benchmark datasets, but mainly due to the limited size of a standard SMS along with the abundance of idioms and abbreviations. In addition, computational resources on mobile phones are much lighter than these used in current server. As such, in this work we focus on both message representation techniques along to classification algorithms, and present a brief comparative evaluation of different state-of-the-art techniques. Our best performing model (RNN) achieves 0.978 F1 score. Amongst the best performing models was the naive Bayes classifier which, with proper message modeling, managed to achieve 0.971 F1 score, but with significantly less computational complexity and memory requirement.

1 Introduction

Spam refers to unsolicited electronic messages delivered to a large number of recipients customarily via email. In recent years returns from the email spaming are diminishing due to robust and effective filtering and user awareness (Delany, Buckley, and Greene 2012). Coupled with the emergence of low cost or free SMS messaging services, it has lead a growing number of marketers to use text messages (SMS) to target subscribers, especially in Asia and in developing countries (Gómez Hidalgo et al. 2006) (Yadav et al. 2011).

A spam classifier or filter, aims in recognizing and preventing the delivery of the aforementioned unsolicited messages. There is a vast literature in email spam filtering (e.g. see (Cormack and others 2008) (Blanzieri and Bryl 2008)), and state-of-the-art email spam filters are remarkably effective. Nevertheless, the sms spam filtering problem presents many unique challenges, and blindly adopting such techniques is not sufficient. This is partly due to lack of benchmark datasets, which further complicates the application of content-based filtering algorithms, but mainly due to the limited size of a standard SMS along with the abundance of idioms, informal speech (slang), and abbreviations [1]. As such, special care is required on both the message representation and the employed classification algorithm.

In this work we present a brief comparative overview on well established content-based filtering algorithms, ranging from simple models such as the naive Bayes classifier, to more complicated ones like a convolutional neural network. Moreover, we examine various message representation models, ranging from simple vector representations, to word and sentence embeddings. The latter constitute distributed vector representations able to capture a large number of syntactic and semantic word relationships, and have recently shown to be highly successful in a plethora of natural language processing applications (Mikolov et al. 2013) (Bojanowski et al. 2016) (Pagliardini, Gupta, and Jaggi 2017).

The rest of the paper is organized as follows. Section 2 provides a brief overview of SMS filtering techniques, Section 3 presents the evaluated models, both for message representation and classification, and Section 4 provides simulation results. Finally, Section 5 concludes the paper.

2 Related Work

In this section we provide a brief overview of related work in the area of SMS spam classification.

In (Gómez Hidalgo et al. 2006) the authors have tested a number of message representation techniques and machine learning algorithms, in terms of effectiveness. They have identified the tokenization step as the most important one, since a bad representation may lead to a poor classifier. Contrary to our work, were we have employed word and sentence embeddings which can capture syntactic and semantic word relationships (symmetrically sentence), the authors feed their model with a big number of attributes and used the information gain metric (Yang 1999) as an attribute selection mechanism. They have identified as the most suitable learning algorithm the Support Vector Machines (SVM).

In (Xu et al. 2012) the authors follow an orthogonal approach and utilize non-content features from static, network and temporal views. Subsequently, they incorporated the

[1] Here are two examples of such messages, drawn from the employed dataset: '*Ok lar... Joking wif u oni...*', and '*dun say so early hor... U c already then say...*'.

aforementioned features to an SVM classifier, with a gain of $\approx 8\%$ in AUC (Area Under the ROC Curve).

Finally, a combination of content-based and non content-based features was studied in (Sulaiman and Jali 2016), in which the authors also focused on the effect of the employed model in the battery and processing performance.

In this work, we limit ourselves to content-based filtering with emphasis on the message representation. Following the recent success in a variety of natural language processing applications (Mikolov et al. 2013) (Bojanowski et al. 2016) (Pagliardini, Gupta, and Jaggi 2017), we employ word and sentence embeddings to capture syntactic and semantic word and sentence relationships.

The aforementioned embeddings incorporate a general syntactic and semantic meaning and thus might not representative of the test message domain. As future work, we could employ domain specific knowledge and corpora in order to capture the abundance of idioms, informal speech (slang), and abbreviations found in such domains.

3 Overview of Employed Models

In this section we present a brief theoretical overview of the employed models for both message representation and the subsequent classification. Unless stated otherwise, each sentence is preprocessed via Stanford CoreNLP tools (Manning et al. 2014): tokenization, word lowering, part-of-speech tags, lemmatization and stopwords removal. At the end, only lemmas are kept and a vocabulary of lemmas is built.

3.1 Message Representation Models

Naive Vector Representation The naive representation model results in a vector of integers, each one representing an index corresponding to the lemma in the vocabulary.

Bag of Words The bag of words (BoW) model results in a vector of integers, where each dimension corresponds to the frequency of a lemma in the current sentence. The aforementioned vector has fixed size, hence every sample has the same dimension (i.e. same vocabulary), which is not the case in the naive representation. Moreover, each vector has values of the same range, and it merges multiple occurrences together. Finally, the BoW representation can be used to compute similarities between two messages (e.g. using the cosine similarity measure).

Bag of Words with TF-IDF This model is similar to the BoW model yet, instead of having discrete values, it takes into account the importance of each word in our corpus using the TF-IDF statistic (Salton and Buckley 1988). This is a common, simple and efficient technique used in Information Retrieval.

Word Embeddings Word embeddings generate a mapping from a word in our corpus to a high dimension continuous vector (e.g. 300 dimensions). They are able to capture a large number of syntactic and semantic word relationships, and have recently shown to be highly successful in a plethora of natural language processing applications (Mikolov et al. 2013) (Bojanowski et al. 2016). We have utilized Facebook's AI Research (FAIR) fastText library (Bojanowski et al. 2016) to learn the word embeddings. The intuition for training consists of moving a fixed-size window, take a specific word in the middle and predict the probabilities of every words to be in nearby (i.e. within the window). One of the novelty of this model, besides being fast, is they can handle Out-Of-Vocabulary (OOV) words as it also computes n-gram embeddings. Therefore, when an unknown word is encountered, the model sums all the n-gram embeddings of the given word.

As this representation leads to a $3 - D$ matrix, we only combine them with advanced neural network classification models as most of the other methods can not manage this kind of high dimensional data easily. Finally, we also update our word embeddings during the back-propagation phase, in order to fine tune them to our specific domain.

Sentence Embeddings Sentence Embeddings generate a mapping from sentences to high dimensional vectors (e.g. 600 dimensions). Generating semantic embeddings of longer pieces of text (e.g. whole sentences) has become possible due to the utilization of neural network models, and recent studies have shown that such models can outperform state-of-the-art unsupervised models (Pagliardini, Gupta, and Jaggi 2017). The model is similar to the one for word embeddings however, the differences rely on the size of the window being the whole sentence but also different tricks to make the training converge.

Topic Models Topic models aim to find patterns of words in our message collection using (hierarchical) probabilistic models. Specifically, we have utilized the Latent Dirichlet Allocation topic model from (Blei and Lafferty 2006). The latter is a generative probabilistic model considering documents as a mixture of topics (known a priori) and topics as a mixture of words. Each topic has a multi-nomial distribution sampled from a Dirichlet distribution (with parameter α). Same applies for the word distribution given the topic (with parameter β).

3.2 Classification Models

Naive Bayes The naive Bayes classifier determines the class $c \in \mathcal{C}$ of an input feature vector $\mathbf{h}$ by seeking the value that maximizes Equation 1. We have employed a Gaussian, a Bernoulli and a multinomial naive Bayes classifier. Despite the unrealistic independence assumption, the naive Bayes classifier is highly scalable and have shown to perform well in both SMS and e-mail spam detection (Gómez Hidalgo et al. 2006) (Androutsopoulos et al. 2000).

$$\max_{c \in \mathcal{C}} \pi_c \mathbb{P}(\mathbf{h} = h | \mathbf{c} = c) \tag{1}$$

Logistic Regression This model implements regularized logistic regression using a limited-memory BFGS solver (Liu and Nocedal 1989). The L-BFGS solver is an optimization algorithm in the family of quasi-Newton methods which does not require to analytically compute the Hessian of a function. Instead it computes an estimation which uses to steer its search through variable space. The goal is to find

the class that minimizes $f(.)$, where f is an L2-regularized cost function (Equation 2). In addition, it provides a high-level of interpretability.

$$c^* = \arg\min_{c \in \mathcal{C}} f(c) \tag{2}$$

Decision Tree This predictive model implements a decision tree classifier using the Gini impurity metric. Gini impurity measures the probability that a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. This model recursively discretizes the space with respect to the feature obtained via Gini. Therefore, this leads to a highly interpretable model.

Random Forest We utilize a random forest meta-classifier which fits a number of decision tree classifiers on various sub-samples of the dataset and the feature set. It uses averaging to improve the predictive accuracy, reduce variance, and control over-fitting. Moreover, random forests do not sample the entire set of features. Unsurprisingly, we expect random forests to perform better than a single decision tree.

Adaptive Boosting The adaptive boosting (AdaBoost) algorithm (Freund and Schapire 1997) produces an accurate classification rule by combining rough and moderately inaccurate learning algorithms ('weak learners') into a weighted sum that represents the final output of the boosted classifier. We have deployed a variant called AdaBoost-SAMME (Hastie et al. 2009), using a decision tree classifier as the base estimator from which the boosted ensemble is built. We expect it performs on par with random forests.

Support Vector Machine As support vector machine have been identified as the most suitable learning algorithm ((Yang 1999)), we naturally have utilized it and compared two support vector machine classifiers, one with a linear and one with radial basis function (rbf) kernel. It uses the Hinge loss as objective function and tries to maximize the margin (i.e. hyperplane) between all the samples. In order to achieve this, it leverages kernel function mapping the original dimensional space into much higher. Functions can be used as long as their kernel matrix is symmetric positive semi-definite, which is the case with rbf and linear kernels.

Feedforward Neural Network This model implements a multi-layer perceptron classifier with a rectified linear unit activation function. The weight optimization is being performed using Adam (Kingma and Ba 2014), an algorithm based on adaptive estimates of lower-order moments. Adam has the advantages of being computationally efficient, improves convergence time and has low memory requirements, which is ideal for deployment in real scenarios with large datasets used by telecommunication providers, and/or in mobile devices. In order to improve generalization we use, in addition to L2 regularization, dropout (Srivastava et al. 2014) and we clip the gradient norm to 1.0 to avoid gradient exploding. Dropout cancels the activation of a neuron with a probability p. We have employed the aforementioned techniques to the subsequent neural network models as well.

Recurrent Neural Network A recurrent neural network (RNN) is a neural network where connections between units form a directed graph along a sequence. As sentences are a sequence of words, this makes RNNs an ideal candidate model to leverages this kind of sequential data. In order to handle the gradient vanishing / exploding problem, we utilize the Long-Short Term Memory units (LSTM) of (Hochreiter and Schmidhuber 1997). Our model consists of a single LSTM layer followed by max pooling and a simple feedforward neural network.

Convolutional Neural Network Convolutional neural networks (CNN) is a class of neural network where connectivity pattern between neurons partially overlaps, which makes them ideal for certain domains (e.g. image processing). It relies to translation and rotation invariance properties. Although this might not make sense to apply them on text rather than images, as translation and rotation are not defined for texts, surprisingly CNNs are also considered state-of-the-art in text classification (Cao et al. 2017) (Kim 2014) (Zhang and LeCun 2015) (Xiao and Cho 2016) and much more text applications. Motivated by their recent success in such similar domains, we have implemented a CNN classifier based on (Kim 2014).

4 Experimental Evaluation

We have conducted a series of simulations, and in this section we present a comparative overview of the presented representations and models applied to the SMS spam filtering problem. All the models described in Section 3 were trained and evaluated on a Intel Xeon E5-2640V4, 2.4GHz system with 32GB of RAM and a GeForce Titan Xp GPU to improves training and inference times for advanced neural networks.

4.1 Dataset

To train and evaluate our models, we have utilized the *SMS Spam Collection v. 1* [2]. The SMS Spam Collection is a free corpus, collected for research purposes. It consists of $5\,574$ messages in English, tagged according to being ham (legitimate) or spam ($|\mathcal{C}| = 2$). As spam classification is usually similar to an anomaly detection problem, the dataset is highly imbalanced as only 13.5% is considered as spam.

4.2 Simulation Results

Message Representation We first begin by examining the impact of the message representation model. Figure 1 presents a 2-D visualization of the SMS Spam Collection dataset. We have employed the t-distributed stochastic neighbor embedding (t-SNE) algorithm (Maaten and Hinton 2008), a nonlinear dimensionality reduction technique well-suited for visualizing high dimensional data in a low-dimensional space. We choose this latter as opposed to Principal Component Analysis (PCA) due to its nonlinear nature.

[2] `https://www.kaggle.com/uciml/sms-spam-collection-dataset`, `http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`

Sub-figures 1a - 1f depict each of the message representation model of Section 3.1, while sub-figure 1g depicts the concatenation of the bag of words and topics models.

The aforementioned figures make immediately apparent the importance of the message representation model. Models like the BoW (sub-figure 1b) result to almost linearly separable data, which suggest that with a strong message representation model, even a linear classifier (e.g. logistic regression) might be enough. According to our employed t-SNE visualization algorithm, the best message representation model is the topics model with four topics (sub-figure 1d). To better visualize the patterns discovered by the topics models and understand why this can lead to a better clattering, Figure 2 depicts a word cloud of the words in each of the aforementioned four topics. The size of each word indicates its frequency. Topics #2 & #3 have high number of tokens like 'phone numbers', 'urls', 'free', 'win', 'text', 'call', etc., which are characteristic in spam messages. On the other hand, topics #1 & #4 contain innocuous words. Specifically, topic #1 seems to contains 'regular' words, and topic #2 slang and abbreviations; both of which are common in ham messages.

Classification Deciding on a message representation model is the first part of the final classification model. In this section, we will examine the second part, which is the classifier. Table 1 presents an overview of the F1-score achieved by each combination of the employed classification / message representation model. The F1 score is the harmonic mean of precision and recall (Equation 3). It is a real number in $[0, 1]$, with 1 being the optimal value (perfect precision and recall). In other words, it represents the tradeoff between precision (fraction of the predicted classes being correct) and recall (fraction of the predicted class being among the predicted set of golden labels).

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

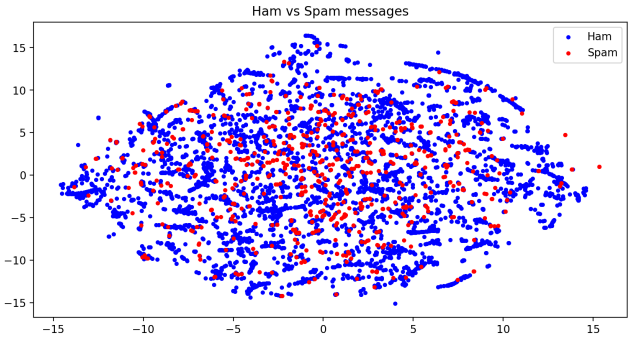
Along with the F1 score, another important metric for our models is their computational complexity. To evaluate the latter, Table 2 presents an overview of the training and testing time required by each combination of the employed classification / message representation model.

The highest performing models in terms of the F1 score are the Recurrent Neural Network and the Convolutional Neural Network, achieving 0.978, and 0.976 respectively. The aforementioned models use the word embeddings to represent a message. Hence, our results are in par with recent results in other NLP domains. The models that are regarded as the best (RNNs, CNNs, and Word Embeddings), are the ones that achieved the highest scores in our SMS spam classification scenario as well. Rather surprising though, the multinomial naive Bayes classifier was able to achieve similar performance, specifically 0.971 in F1 score (less than 1% worse than the best performing model). The latter corroborates the fact that the message representation model has a significant impact on the overall performance. The naive Bayes classifier achieved it's best performance using the concatenated Bag of Words and Topics models. Unsurprisingly, with respect to the training and testing speed, the simplest models achieve the best times whereas advanced ones require more time to train and infer due to their complexity.

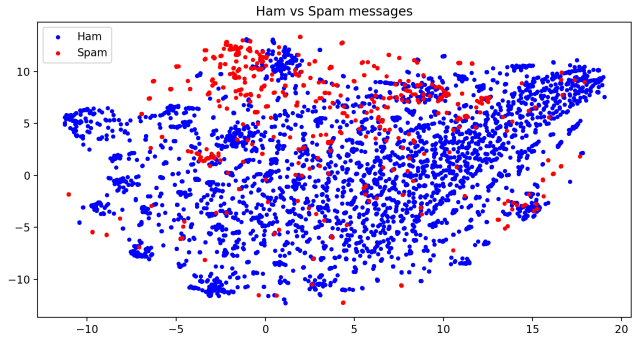
Furthermore, the naive Bayes classifier is significantly less computational expensive, since it required 98% less training time, and 44% less testing time than the fastest high performing model (RNN). Moreover, it only uses CPU and do not leverage GPUs. As such, it provides the ideal tradeoff between high performance (F1 score) and low computational complexity. Finally, contrary to the cumbersome neural network models, the naive Bayes classifier can easily handle on-line updates (Chan, Golub, and LeVeque 1982). With respect to the BoW and Topics models message representation model, we can easily update it by adding an extra dimension when a new word is encountered. The number of topics should not change that much but the topic model can also support online updates (Canini, Shi, and Griffiths 2009).

5 Conclusion

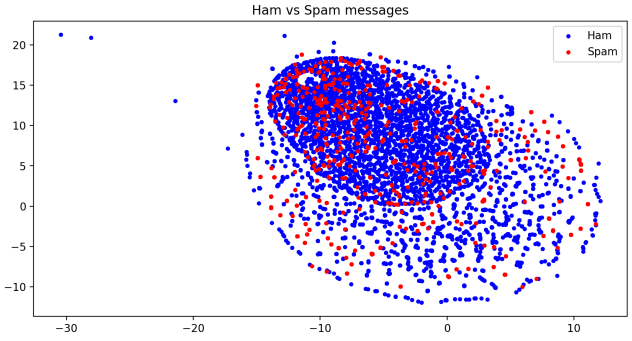
As the cost of sending messages over the telecommunication network decreases, SMS messaging is becoming a perfect domain for abuse. SMS spaming is thriving, especially in Asia and in developing counties. The unique text style in SMS messaging, requires additional effort in terms of message representation models, which in turn have a significant effect in the performance of the employed spam classifiers. In this work, we have presented a brief comparative overview of various message representation models, ranging from simple vector representations to state-of-the-art word and sentence embeddings, and content-based filtering models. Our best performing model (RNN) achieves 0.978 F1 score. Amongst the best performing models was the naive Bayes classifier with BoW and Topics models which managed to achieve 0.971 F1 score, but with significantly less computational complexity, resource requirements, training and inference time. More over, both the naive Bayes classifier and the message representation model can be easily updated with online learning. The latter constitutes the naive Bayes a promising candidate for on-device implementation of SMS spam filtering.



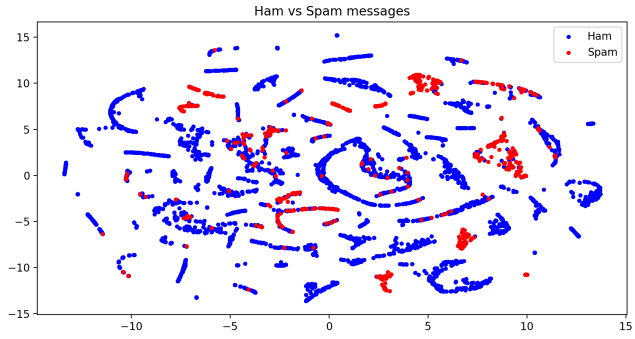
(a) Naive Vector Representation



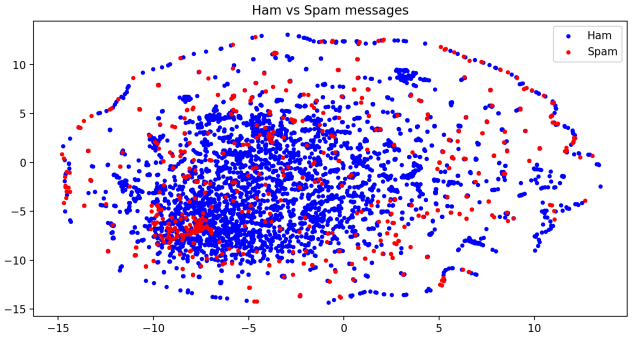
(b) Bag of Words



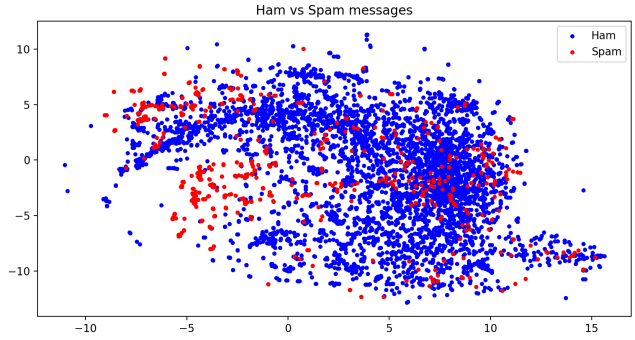
(c) Bag of Words with TF-IDF



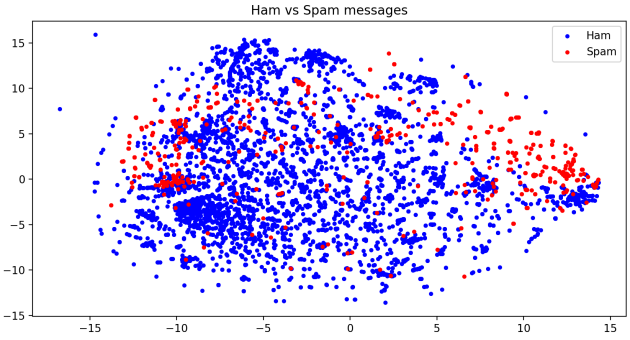
(d) Topic Models



(e) Word Embeddings



(f) Sentence Embeddings



(g) Bag of Words & Topic Models

Figure 1: 2-D visualization using t-SNE of the SMS Spam Collection dataset for each of the message representation models of Section 3.1.

Figure 2: Word cloud indicating the frequency of each word in a 4-topic message representation model.

Table 1: Overview of the F1-score achieved by each combination of the employed classification / message representation model. Bold numbers represent the top three models achieving the highest scores while * denotes model that could not be trained in less than 24 hours or converge.

	Naive	BoW	BoW / TF-IDF	Word Embeddings	Sentence Embeddings	Topics	BoW & Topics
Naive Bayes (Bernoulli)	0.707	0.951	0.951		0.669	0.460	0.951
Naive Bayes (Multinomial)	0.682	0.962	0.931		N/A*	0.460	**0.971**
Naive Bayes (Gaussian)	0.159	0.748	0.745		0.573	0.837	0.748
Logistic Regression	0.600	0.966	0.922		0.912	0.880	0.964
Decision Tree	0.799	0.960	0.957		0.793	0.888	0.950
Random Forest	0.826	0.964	0.960		0.834	0.895	0.956
AdaBoost	0.836	0.959	0.959		0.905	0.878	0.951
SVM (Linear)	N/A*	0.968	0.958		0.918	0.878	0.962
SVM (RBF)	0.691	0.460	0.460		0.805	0.876	0.460
Feed-forward Neural Net	0.659	0.951	0.948		0.931	0.460	0.956
Recurrent Neural Net				**0.978**			
Convolutional Neural Net				**0.976**			

Table 2: Overview of the training and testing time (in seconds) required by each combination of the employed classification / message representation model. Bold numbers represent the top three models achieving the highest F1 scores while * denotes model that could not be trained in less than 24 hours or converge.

	Naive		BoW		BoW / TF-IDF		Word Embeddings		Sentence Embeddings		Topics		BoW & Topics	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Naive Bayes (Bernoulli)	0.011	0.001	0.373	0.092	0.353	0.086			0.059	0.013	0.001	0.000	0.607	0.144
Naive Bayes (Multinomial)	0.004	0.000	0.172	0.033	0.164	0.032			N/A*	N/A*	0.001	0.000	**0.229**	**0.052**
Naive Bayes (Gaussian)	0.011	0.002	0.468	0.118	0.500	0.126			0.038	0.008	0.001	0.000	0.654	0.181
Logistic Regression	0.097	0.000	0.925	0.024	0.901	0.032			0.200	0.001	0.021	0.000	0.963	0.050
Decision Tree	0.041	0.000	9.982	0.030	6.426	0.026			1.936	0.001	0.013	0.000	12.060	0.045
Random Forest	0.085	0.007	4.190	0.136	2.616	0.087			0.902	0.008	0.107	0.006	4.380	0.188
AdaBoost	0.411	0.013	17.084	0.205	23.387	0.216			14.000	0.024	0.169	0.009	23.845	0.219
SVM (Linear)	N/A*	N/A*	17.196	3.791	28.913	7.335			2.415	0.356	0.035	0.005	15.208	3.148
SVM (RBF)	3.513	0.834	45.042	11.604	40.929	9.913			3.866	0.780	0.099	0.016	41.349	10.146
Feed-forward Neural Net	0.179	0.002	4.149	0.049	13.387	0.062			1.565	0.011	0.058	0.001	6.795	0.050
Recurrent Neural Net							**11.210**	**0.093**						
Convolutional Neural Net							**21.489**	**0.225**						

References

Androutsopoulos, I.; Koutsias, J.; Chandrinos, K. V.; Paliouras, G.; and Spyropoulos, C. D. 2000. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*.

Blanzieri, E., and Bryl, A. 2008. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review* 29(1):63–92.

Blei, D. M., and Lafferty, J. D. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, 113–120. ACM.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Canini, K.; Shi, L.; and Griffiths, T. 2009. Online inference of topics with latent dirichlet allocation. In *Artificial Intelligence and Statistics*, 65–72.

Cao, Z.; Li, W.; Li, S.; and Wei, F. 2017. Improving multi-document summarization via text classification. In *AAAI*, 3053–3059.

Chan, T. F.; Golub, G. H.; and LeVeque, R. J. 1982. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, 30–41. Springer.

Cormack, G. V., et al. 2008. Email spam filtering: A systematic review. *Foundations and Trends® in Information Retrieval* 1(4):335–455.

Delany, S. J.; Buckley, M.; and Greene, D. 2012. Sms spam filtering: methods and data. *Expert Systems with Applications* 39(10):9899–9908.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55(1):119–139.

Gómez Hidalgo, J. M.; Bringas, G. C.; Sánz, E. P.; and García, F. C. 2006. Content based sms spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, 107–114. ACM.

Hastie, T.; Rosset, S.; Zhu, J.; and Zou, H. 2009. Multi-class adaboost. *Statistics and its Interface* 2(3):349–360.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1746–1751.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Liu, D. C., and Nocedal, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming* 45(1-3):503–528.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–1958.

Sulaiman, N. F., and Jali, M. Z. 2016. A new sms spam detection method using both content-based and non content-based features. In *Advanced Computer and Communication Engineering Technology*. Springer. 505–514.

Xiao, Y., and Cho, K. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR* abs/1602.00367.

Xu, Q.; Xiang, E. W.; Yang, Q.; Du, J.; and Zhong, J. 2012. Sms spam detection using noncontent features. *IEEE Intelligent Systems* 27(6):44–51.

Yadav, K.; Kumaraguru, P.; Goyal, A.; Gupta, A.; and Naik, V. 2011. Smsassassin: Crowdsourcing driven mobile-based system for sms spam filtering. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, 1–6. ACM.

Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval* 1(1-2):69–90.

Zhang, X., and LeCun, Y. 2015. Text understanding from scratch. cite arxiv:1502.01710Comment: This technical report is superseded by a paper entitled "Character-level Convolutional Networks for Text Classification", arXiv:1509.01626. It has considerably more experimental results and a rewritten introduction.