Towards Automated Comprehensive Feature Engineering for Spam Detection

Fred N. Kiwanuka[1], Ja'far Alqatawna[1,2], Anang Hudaya Muhamad Amin[1], Sujni Paul[1] and Hossam Faris[2]

[1]*Computer Information Science, Higher Colleges of Technology, Dubai, U.A.E.*

[2]*King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan*

Keywords: Spam Detection, Dataset Processing, Automated Feature Engineering, Classification, Spam Features, Data Mining, Machine Learning, Python E-mail Feature Extraction and Classification Tool (CPyEFECT).

Abstract: Everyday billions of emails are passed or processed through online servers of which about 59% is spam according to a recent research. Spam emails have increasingly contained viruses or other harmful malware and are a security risk to computer systems. The importance of spam filtering and the security of computer systems has become more essential than ever. The rate of evolution of spam nowadays is so high and hence previously successful spam detection methods are failing to cope. In this paper, we propose a comprehensive and automated feature engineering framework for spam classification. The proposed framework enables first, the development of a large number of features from any email corpus, and second extracting automated features using feature transformation and aggregation primitives. We show that the performance of classification of spam improves between 2% to 28% for almost all conventional machine learning classifiers when using automated feature engineering. As a by product of our comprehensive automated feature engineering, we develop a Python-based open source tool, which incorporates the proposed framework.

1 INTRODUCTION

In the context of email systems, spam can be defined as unsolicited messages carrying commercial ads, offers and/or malicious content (Alqatawna et al., 2015; Guzella and Caminhas, 2009; Zaid et al., 2016). Annual statistics and research studies show that spammers keep updating their techniques and tricks to circumvent current anti-spam solutions resulting in a growing volume of spam messages (Kaspersky, 2017; Ruano-Ords et al., 2018). This ongoing challenge keeps the door open for more studies to better understand the nature of these unwanted messages and improve detection methods. Data mining techniques can be used to study and analyse the spam features, which would help in building spam classification models and significantly improve their detection rates. The detection is based on the assumption that spam's content differs from that of a legitimate email in ways that can be quantified(Caruana and Li, 2008). This usually requires *Features Engineering*; a process of identifying representative characteristics or attributes that can be used to build detection models(Koitka and Friedrich, 2016).

The feature engineering process is considered the most expensive and time consuming part of the data mining process as it is usually performed manually and requires domain knowledge(Domingos, 2012). The traditional approaches to feature engineering is to build features one at a time using domain knowledge, a tedious, time-consuming, and error-prone process known as manual feature engineering. Over the last few years there has been an emerging trend to improve and automate this process which would reduce exploration time and give researchers the chance to try more ideas(Kanter and Veeramachaneni, 2015a; Lam et al., 2017b). Automated feature engineering improves upon the manual feature engineering workflow by automatically extracting useful and meaningful features from datasets. Automatic feature engineering cuts down on the time spent on feature engineering,as well as reduces data leakage by filtering time-dependent data.

This paper contributes to this direction by proposing an automated feature engineering approach in the context of spam detection. In our previous work we extensively studied the characteristics of Spam email and conducted a review of the existing literature to get insight into the various aspects of spam and spam-

Kiwanuka, F., Alqatawna, J., Amin, A., Paul, S. and Faris, H.
Towards Automated Comprehensive Feature Engineering for Spam Detection.
DOI: 10.5220/0007393004290437
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 429-437
ISBN: 978-989-758-359-9

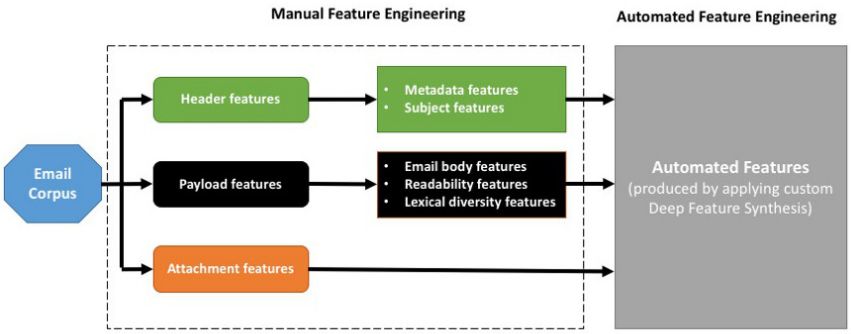

Figure 1: Comprehensive Feature Engineering Process.

mer techniques (Alqatawna et al., 2015; Alqatawna et al., 2016; Hijawi et al., 2017a; Faris et al., 2015; Halaseh and Alqatawna, 2016). We developed a comprehensive list of spam features that we used for spam detection. As part of that work, we developed E-mail Features Extraction Tool (EMFET) based on the .NET platform that only extracts features (Hijawi et al., 2017b; Hijawi et al., 2017a). In this paper, we extend this work by automating and improve the feature engineering process. Additionally, we integrate that with the remaining data mining steps of classification and feature selection. The proposed approach is implemented as an open source Python E-mail Feature Extraction and Classification Tool (CPyEFECT). CPyEFECT tool provides a flexible and automated way to perform spam feature engineering by extract a large number of features from any email corpus to produce cleansed dataset, run classification, and to select the most important features.

The rest of the report is organized as follow. A background literature is presented in section 2. The proposed approach presented in section 3 followed by implementation in section 4. Testing and results are shown in section 5. Finally, section 6, presents the conclusions and future works.

2 LITERATURE REVIEW

Spam detection techniques have evolved as quickly as the ever-increasing tide of spam, with different objectives, evaluation methods, and measures of success. A large number of spam filtering aimed at helping eliminate spam problem exists. Existing spam filters use different techniques for analyzing email and determining if it is indeed spam. The accuracy of spam filters are evaluated based on two dimensions: How much spam you successfully filter out, and how little legitimate messages you accidentally delete (Blanzieri and Bryl, 2008). Maintaining accuracy can be difficult because spam is constantly changing. Spam detection has focused mainly on improved machine learning techniques to provide effective spam filters (Tretyakov, 2004). Feature engineering for spam has received less attention. The importance of feature engineering in predictions can not be understated. Recent research has focused on feature engineering. Unfortunately, there is not a lot in terms of feature engineering exists for spam.

The work in (Alqatawna et al., 2015) attempts to focus on feature engineering to handle the problem of spam detection. The develop a substantial set (140) of features that can be used to detect spam with considerably good performance. Feature engineering for spam on twitter is studied in (Herzallah et al., 2018). Similar interesting work in (Scott and Matwin, 1999) proposes feature engineering for text based classification. Spam feature engineering is purely text based. (Scott and Matwin, 1999) proposes, feature engineering techniques for text based data, and describe 8 automatically-generated representations based on words, phrases, synonyms and hypernyms. Unfortunately the majority of features in this work are not feasible for spam emails.

In recent years there has been progress in automating many machine learning aspects including model selection and hyperparameter tuning, however, feature engineering, which is considered the most important aspect of the machine learning pipeline, has received less attention. That is why recent work on automatic feature engineering (Kanter and Veeramachaneni, 2015b), (Khurana et al., 2016), (Katz et al., 2016), (Lam et al., 2017a) has received quite some attention in literature. Automated feature engineering is a relatively new technique. The traditional approaches to feature engineering is to build features one at a time using domain knowledge, a tedious, time-consuming, and error-prone process known as manual feature engineering. Automated feature engineering improves upon the manual feature engineer-

Table 1: The Header Features.

ID	Feature Details	Type	Ref.
1-2	Year, Month	Metadata	(Alqatawna et al., 2015)
3-6	Day, Hour, Minute, Second	Metadata	(Tran et al., 2013; Alqatawna et al., 2015)
7-22	From [Google?, AOL?, Gov?, HTML?, MIL?, Yahoo?, Example?] To [Hotmail?, Yahoo?, Example?, MSN?, Localhost?, Google?, AOL?, Gov?, MIL?]	Metadata	(Alqatawna et al., 2015)
23	Count of "To" Email	Metadata	(Tran et al., 2013)
24-33	Replay to [Google?, Hotmail? MIL?, Yahoo?, AOL?, Gov?], X-Mailman-Version, Exist [Text/Plain?, Multipart/Mixed?, Multipart/Alternative?]	Metadata	(Alqatawna et al., 2015)
34-49	No. of [characters, capitalised words], No. of words [in all uppercase, that are digits, containing only letters, containing letters and numbers, that are single letters, that are single digits, that are single characters], Max ratio of [uppercase to lowercase letters of each word, uppercase letters to all characters of each word, digit characters to all characters of each word, non-alphanumerics to all characters of each word], Min character diversity of each word, Max of the [longest repeated character, character lengths of words]	Subject	(Tran et al., 2013)

ing workflow by automatically extracting useful and meaningful features from datasets. Automatic feature engineering cuts down on the time spent on feature engineering,as well as reduces data leakage by filtering time-dependent data. Automatic feature engineering has seen tremendous success in image and video processing through deep learning techniques. Deep learning through its explicit feature construction has also been used in many other fields like speech processing and recently text classification with considerable success. The main problem with deep learning as highlighted by many authors relate to, higher computational burden, requires large datasets in order to prevent overfitting and unintrepretable features.

In this paper (Hanif Bhuiyan, 2018) have presented the classification, evaluation and comparison of different email spam filtering system and summarize the overall scenario regarding accuracy rate of different existing approaches.A Naives Bayesian Classifier is used in this paper (G.Vijayasekaran, 2018) with three layer framework that includes classifier and anomaly detector for spam classification for bulk emails. The idea of automatic feature engineering for transaction and relational data was first fronted in (Kanter and Veeramachaneni, 2015b) with the aim of deriving predictive modelling from raw data automatically. The authors propose and develop the Deep Feature Synthesis algorithm for automatically gener-

ating features for relational datasets. The algorithm captures features that are usually supported by human intuition. The beauty about the deep feature synthesis is that one can combine raw data with what you know about your data to build meaningful features for machine learning and predictive modelling. Using aggregation and transformation primitives, a large set of features can be developed. In this paper, we adopt some of the components of this work to spam detection. While (Kanter and Veeramachaneni, 2015b) is designed for relational and transaction data, spam emails do not have the notion of relationships.

3 THE PROPOSED SPAM FEATURE ENGINEERING APPROACH

In this section, we present our proposed comprehensive feature engineering approach which can be described as a process with two phases; (1) a conventional or manual features engineering phase to produce the initial set of spam features from raw emails(text-based features), where we develop 148 features, and (2) an automated feature engineering through transformation primitives of the manual features whose number depends on the number of primitives used. These two phases are shown in figure 1 and discussed in the following subsections.

3.1 Conventional Manual Feature Engineering

This conventional feature engineering phase has evolved out of several years of research in the context of spam detection to investigate malicious spam content and type of attacks carried over email messages(Alqatawna et al., 2015; Alqatawna et al., 2016; Hijawi et al., 2017a; Faris et al., 2015; Halaseh and Alqatawna, 2016). In this phase we identify 148 features which we extract from SpamAssassin corpus[1]. In particular, we extract: header features, payload (body) and attachment. As shown in figure 1, the features comprise of 50 header features, 4 attachment features, and 94 Payload features.

Header features are shown in table 1, these feature are extracted from the contents of header of the email message (sender, recipient, date and subject). Other features are extracted from the email payload which is the part of transmitted data that is the actual intended message. Features in this part are categorized into three sub categories: Email body features, Readability features, and Lexical diversity features. Email body features are extracted from the body of which contains unstructured data such as images, HTML tags, text, and other objects. This features set contains 64 features, which are described in table 2. Four attachment related features are also extracted; one feature related to the number of all attachment files in an email and other three features related to the number of unique content types of attachment files in an email(text/plain, multipart/mixed and multipart/alternative).

Readability features represent the difficulty properties of reading a word, a sentence or a paragraph in the given email's body (Shams and Mercer, 2016; Al-Shboul et al., 2016). Readability features are extracted based on the syllables — sequence of speech sounds —, which are used to distinguish between the simple and complex words. This set of features contains 26 features that measure the difficulty of reading the email's body.

Lexical Diversity Features are extracted based on the vocabulary size in the text, where the word occurrences are counted using different constraints (Tran et al., 2013; Tweedie and Baayen, 1998; Choi, 2012). This set of features contains seven features as follows: Vocabulary Richness,Hapax legomena, Hapax dislegomena, Entropy measure, YuleK, YuleI, SichelS, Honore, and Unusual words

3.2 Automatic Feature Engineering

Manual feature engineering process depends on domain knowledge, intuition, and data manipulation. This makes it extremely tedious and the final features are limited both by human subjectivity and time. Automated feature engineering aims at automatically creating many candidate features out of already available features from which the best can be selected and used for training.

Once we have built the manual features like in section 3.1, we employ, Deep Feature Synthesis (DFS) as in (Kanter and Veeramachaneni, 2015b), to perform automated feature engineering. Deep feature synthesis stacks multiple transformation and aggregation operations to create features from data spread across many tables. Whereas **DFS** is designed for transaction and relation data with multi-tables, in this research, we only have one table, we customise DFS, to support one table. DFS is also not designed for text data like in this research and requires to first build an initial set of features upon which to build the automatic features.

[1]SpamAssassin corpus. http://spamassassin.apache.org/publiccorpus/

Table 2: Email Body Features.

ID	Feature Details	Ref.	ID	Feature Details	Studies
1	Count of Spam Words	(Shams and Mercer, 2016; Alqatawna et al., 2015; Al-Shboul et al., 2016)	31	Number of question marks	(Alqatawna et al., 2015; Al-Shboul et al., 2016)
2	Count of Function Words	(Shams and Mercer, 2016; Al-Shboul et al., 2016)	32	No. of multiple question marks	(Alqatawna et al., 2015)
3	Count of HTML Anchor	(Tran et al., 2013; Shams and Mercer, 2016; Alqatawna et al., 2015; Al-Shboul et al., 2016)	33	No. of exclamation marks	(Alqatawna et al., 2015; Al-Shboul et al., 2016)
4	Count of Unique HTML Anchor	(Tran et al., 2013; Alqatawna et al., 2015)	34	No. of multiple exclamation marks	(Alqatawna et al., 2015)
5	Count of HTML Not Anchor	(Shams and Mercer, 2016)	35	No. of colons	(Alqatawna et al., 2015; Al-Shboul et al., 2016)
6	Count of HTML Image	(Al-Shboul et al., 2016)	36	No. of ellipsis	(Alqatawna et al., 2015; Al-Shboul et al., 2016)
7	Count of HTML All Tags	(Shams and Mercer, 2016; Al-Shboul et al., 2016)	37	Total No. of sentences	(Shams and Mercer, 2016; Alqatawna et al., 2015; Al-Shboul et al., 2016)
8	Count of Alpha-numeric Words	(Tran et al., 2013; Shams and Mercer, 2016; Al-Shboul et al., 2016)	38	Total No. of paragraphs	(Alqatawna et al., 2015)
9	TF-ISF	(Shams and Mercer, 2016)	39	Average No. of sentences per paragraph	(Alqatawna et al., 2015)
10	TF•ISF without stopwords	(Shams and Mercer, 2016)	40	Average number of words pre paragraph	(Alqatawna et al., 2015)
11	Count of duplicate words.	(Al-Shboul et al., 2016)	41	Average No. of character per paragraph	(Alqatawna et al., 2015)
12	Minimum word length	(Al-Shboul et al., 2016)	42	Average No. of word per sentences	(Alqatawna et al., 2015)
13	Count of lowercase letters	(Al-Shboul et al., 2016)	43	No. of sentence begin with upper case	(Alqatawna et al., 2015)
14	Longest sequence of adjacent capital letters	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	44	No. of sentence begin with lower case	(Alqatawna et al., 2015)
15	Count of lines	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	45	Character frequency "$"	(Alqatawna et al., 2015; Al-Shboul et al., 2016)
16	Total No. of digit character	(Alqatawna et al., 2015)	46	No. of capitalized words.	(Tran et al., 2013)
17	Total No. of white space	(Alqatawna et al., 2015)	47	No. of words in all uppercase.	(Tran et al., 2013)
18	Total No. of upper case character	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	48	Number of words that are digits.	(Tran et al., 2013)
19	Total No. of characters	(Tran et al., 2013; Alqatawna et al., 2015)	49	No. of words containing only letters.	(Tran et al., 2013)
20	Total No. of tabs	(Alqatawna et al., 2015)	50	No. of words that are single letters.	(Tran et al., 2013)
21	Total No. of special characters	(Alqatawna et al., 2015)	51	No. of words that are single digits.	(Tran et al., 2013)
22	Total number of alpha characters	(Alqatawna et al., 2015)	52	Number of words that are single characters.	(Tran et al., 2013)
23	Total No. of words	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	53	Max ratio of uppercase letters to lowercase letters of each word.	(Tran et al., 2013)
24	Average word length	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	54	Min of character diversity of each word.	(Tran et al., 2013)
25	Words longer than 6 characters	(Alqatawna et al., 2015)	55	Max ratio of uppercase letters to all characters of each word.	(Tran et al., 2013)
26	Total No. of words (1 - 3 Characters)	(Alqatawna et al., 2015)	56	Max ratio of digit characters to all characters of each word.	(Tran et al., 2013)
27	No. of single quotes	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	57	Max ratio of non-alphanumerics to all characters of each word.	(Tran et al., 2013)
28	No. of commas	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	58	Max of the longest repeating character.	(Tran et al., 2013)
29	No. of periods	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	59	Max of the character lengths of words.	(Tran et al., 2013; Al-Shboul et al., 2016)
30	No. of semi-colons	(Alqatawna et al., 2015; Al-Shboul et al., 2016)	-	-	-

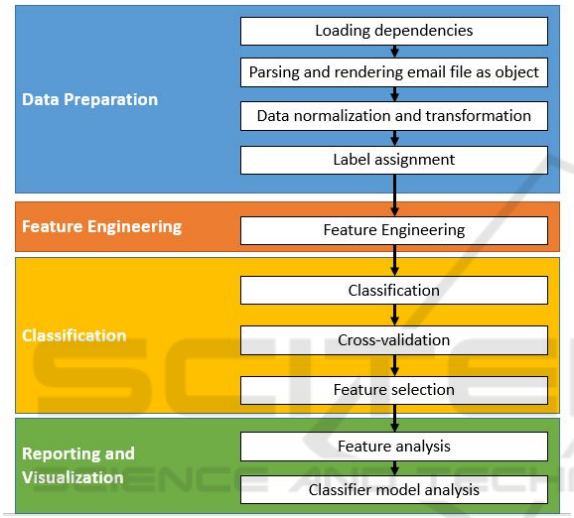

Figure 2: The implementation structure of CPyEFECT.

We recursively apply a set of predefined mathematical transformations on the manual features from section 3.1 to obtain new features. The number of features computed depends on the number of manual features computed, the number of transformation primitives selected and the depth.

Let β be the total number of features computed, N be the number of manual features computed and m be the number of primitives selected and the depth be d. The number of features computed is given as:

$$\beta = \left(\frac{Nm(N+1)}{2}\right)^{d} \quad (1)$$

Comprehensive Features Engineering. This is a two fold phase. First, manual features were developed as in section 3.1, then the automated feature engineering process is done as described in section 3.2. As highlighted in (Kanter and Veeramachaneni, 2015b), the feature space grows very quickly with increase in the number of primitives used as well as depth.

The number of features exponentially grows with an increase in the depth. We utilize transformation primitives rather than aggregation. The number of potential transformations is indefinite. The larger the number of transformations, the larger the number of features generated. In principle, we can generate new features using different combinations of the manual features. The only draw back is the exponential growth of feature computation. The simplest way to reduce the overall classification complexity given the huge feature space is feature selection which we utilize in this research. There are other ways of reducing this complexity like those employed in (Khurana et al., 2016), and (Katz et al., 2016), where the authors propose mechanisms of finding the best transformation and aggregation combinations that give the best classification performance as well as speeding up the feature computation.

For our case, we limit the depth at two and generated 6121 features. The computational burden of computing the features increases substantial with a higher depth. For example, at a depth of 2, it took 6.4 minutes to compute the features while depth of 3, it took at least 9 hours. However, an increase in depth to 3 did not improve the classification results substantially. In this research, through try and error as well as domain knowledge of classifiers, we find transformation primitives that give the best classification results. However, this can be computed automatically like in (Khurana et al., 2016), and (Katz et al., 2016) but at a substantial cost.

4 IMPLEMENTATION

In this section, we describe the implementation of the proposed comprehensive feature engineering framework and provide a performance evaluation for classification of spam emails. We developed

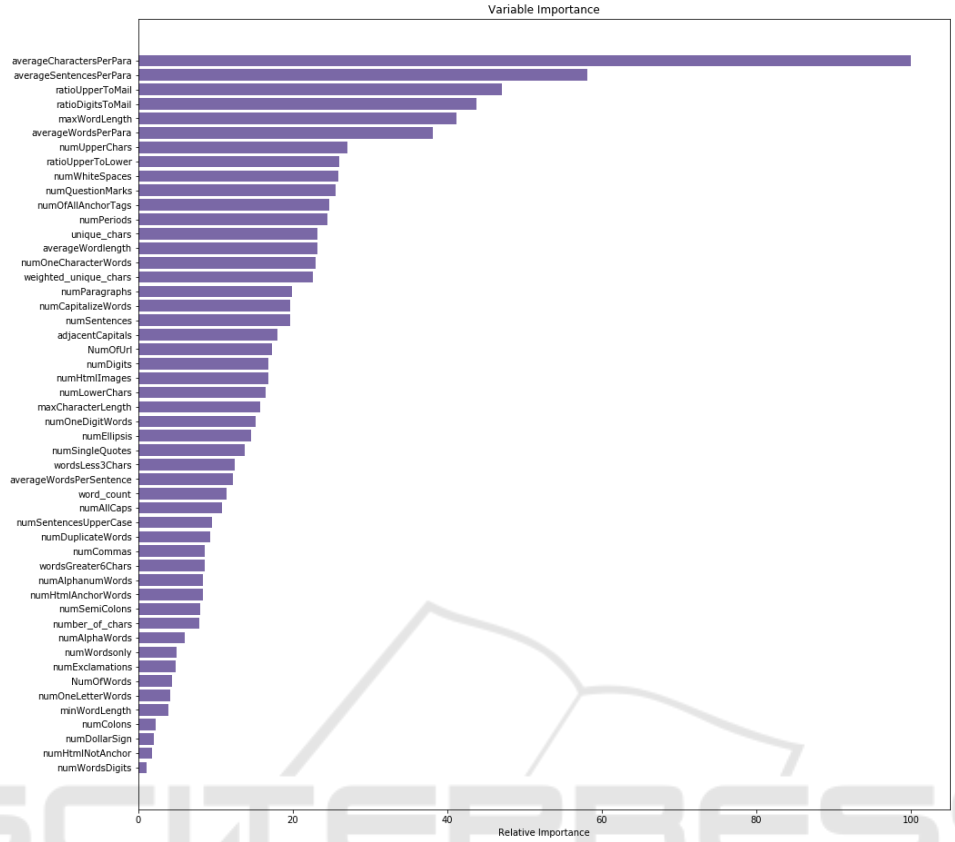

Figure 3: Most informative 50 features for Naive Bayes.

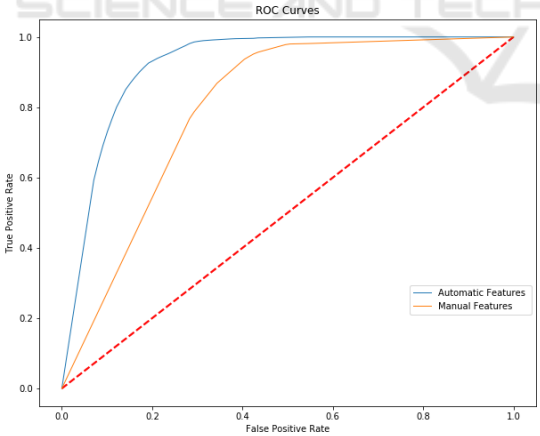

Figure 4: A Comparison of ROC curves for Naive Bayes Classifier.

a tool that incorporates the comprehensive feature engineering framework. There are four functions to cover the various processes required to study email spam detection.

Data Preparation. The tool uses the python *email* package to decompose MIME structured files. Email message is read as a text from the individual files in the corpus to produce the object structure of the email message. This decomposition facilitates extraction of the required features and preparation of dataset for subsequent phases. Each email is split into their main parts which are: `From`, `To`, `Date`, `Subject`, `ContentType`, `CCC`, `Reply` and `Body`. The manual feature engineering as described in section 3.1 depends on these parts to extract the features.

Classification. We implemented nine conventional machine learning algorithms. These included: Extra Tree Classifier, Gradient Boosting Classifier, Support vector machine(SVM), Random Forest Classifier, K Nearest Neighbor Classifier, Logistic Regression, Naive Bayes, AdaBoost Classifier, and Decision trees). Additionally, various validation measures are available to test the produced models.

Feature Selection. Thousands of features were computed at a less than proportional increase in computational cost. We computed the most important features for the classifier with the best performance. Fig. 2 describes the implementation structure of the framework.

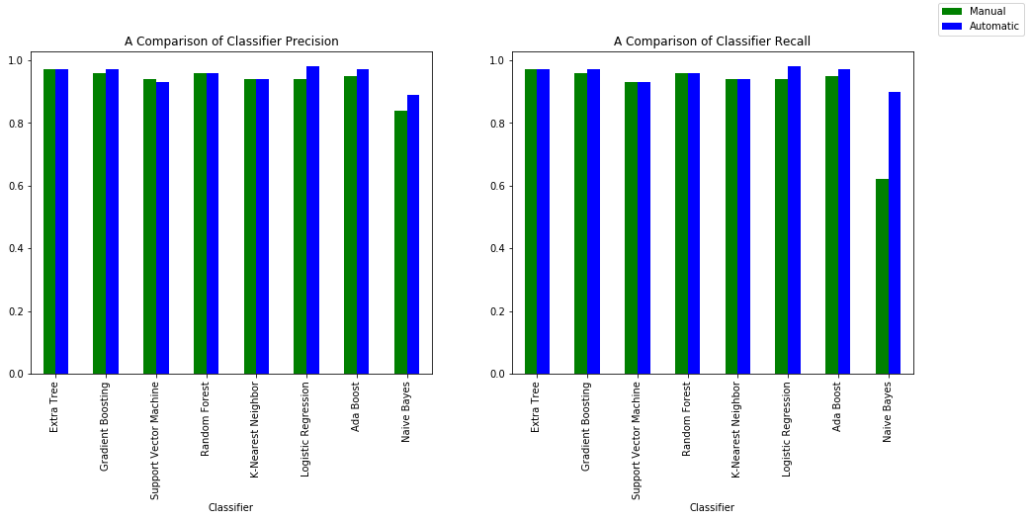

Figure 5: A comparison of classifier precision and recall for various algorithms.

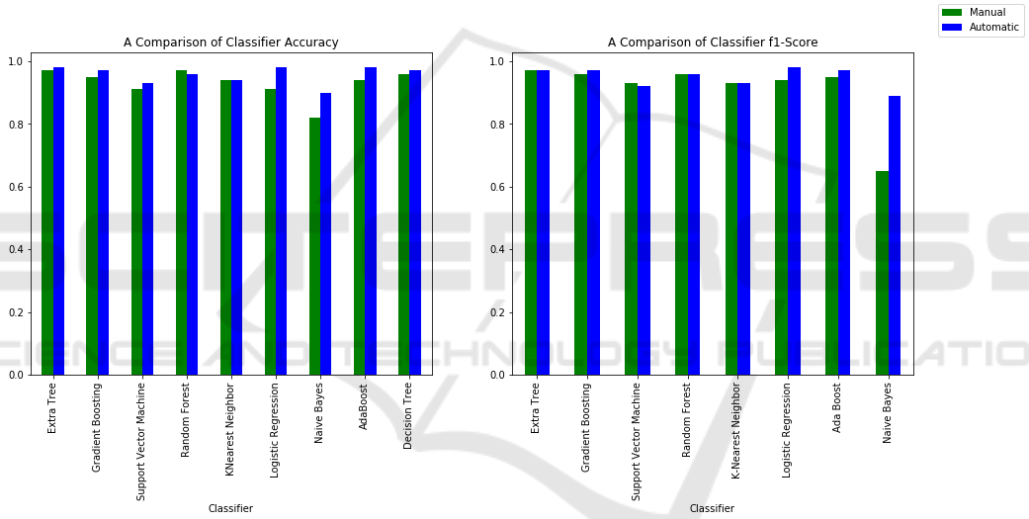

Figure 6: A comparison of classifier accuracy and f-score for various algorithms.

5 RESULTS

In this section, using the *SpamAssasin* dataset, we apply different classical classification methods, such as decision tree, neural network, support vector machines, random forests, gradient boosting, logistic regression and k-nearest neighbors, to identify whether an email is spam or ham. We do this first for the manually-engineered features, consequently we experiment with the automatically generated features. In the manual classification, each email is represented by a vector of 148 features, while the automatic feature classification, email is represented by 6121 features. We experimented on 1800 emails with 950 ham and 850 spam.

For each set of features we computed **Recall, Accuracy, Precision, F-Score, ROC curves, feature selection, confusion matrix**. We would be happy to provide all the results but for space limitation.

From Figure 6, Figure 5 and Figure 4, its clear that automatic feature engineering improves the performance of classifiers. This is especially so in the case of weak classifiers for the manual features like Naive Bayes. Using the manual features the accuracy was 82% while the automatically generated features there was 8% increase in the accuracy to 90%. The f-score improved by 24%, precision improved by 5%, while recall improved by 28%. This improvement was also reflected by other classifiers in the magnitude of between 1.5% to 12%.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we present an automated comprehensive email feature engineering framework that has been developed for the purpose of spam detection and classification. The framework incorporates a scalable mechanism for automated feature engineering and classification algorithms for spam classification. Currently, the proposed framework is capable of producing high accuracy for spam classification with 148 manual email features used. The automated feature engineering scheme further improves the classification accuracy of some of the classifiers up to 12%, with more features being added into the analysis.

For future work, we propose to look into the process optimization of the developed framework, to enable more efficient feature engineering and classification processes.

REFERENCES

Al-Shboul, B. A., Hakh, H., Faris, H., Aljarah, I., and Al-sawalqah, H. (2016). Voting-based classification for e-mail spam detection. *Journal of ICT Research and Applications*, 10(1):29–42.

Alqatawna, J., Faris, H., Jaradat, K., Al-Zewairi, M., and Omar, A. (2015). Improving knowledge based spam detection methods: The effect of malicious related features in imbalance data distribution. *International Journal of Communications, Network and System Sciences*, 8(5):118–129.

Alqatawna, J., Hadi, A., Al-Zwairi, M., and Khader, M. (2016). A preliminary analysis of drive-by email attacks in educational institutes. In *Cybersecurity and Cyberforensics Conference (CCC)*, pages 65–69. IEEE.

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

Caruana, G. and Li, M. (2008). A survey of emerging approaches to spam filtering. *ACM Comput. Surv.*, 44(2):9:1–9:27.

Choi, W. H. (2012). Finding appropriate lexical diversity measurements for small-size corpus. In *Applied Mechanics and Materials*, volume 121, pages 1244–1248. Trans Tech Publ.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

Faris, H., Aljarah, I., and Alqatawna, J. (2015). Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection. In *Applied Electrical Engineering and Computing Technologies (AEECT), 2015 IEEE Jordan Conference on , vol., no., pp.1-5*, pages 1–5. IEEE.

Guzella, T. S. and Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206 – 10222.

G.Vijayasekaran, S. (2018). Spam and email detection in big data plaftorm using naive bayesian classifier. *International Journal of Computer Science and Mobile Computing*, Vol. 7, Issue. 4.

Halaseh, R. A. and Alqatawna, J. (2016). Analyzing cybercrimes strategies: The case of phishing attack. In *Cybersecurity and Cyberforensics Conference (CCC)*, pages 82–88. IEEE.

Hanif Bhuiyan, Akm Ashiquzzaman, T. I. J. S. B. . J. A. (2018). A survey of existing e-mail spam filtering methods considering machine learning techniques. *Global journal of computer science and technology: Software and Data Engineering*, 18 issue 2 Version 1.0.

Herzallah, W., Faris, H., and Adwan, O. (2018). Feature engineering for detecting spammers on twitter: Modelling and analysis. *Journal of Information Science*, 44(2):230–247.

Hijawi, W., Faris, H., Alqatawna, J., Ala'M, A. Z., and Aljarah, I. (2017a). Improving email spam detection using content based feature engineering approach. In *Applied Electrical Engineering and Computing Technologies (AEECT)*. IEEE.

Hijawi, W., Faris, H., Alqatawna, J., Aljarah, I., Al-Zoubi, A., and Habib, M. (2017b). Emfet: E-mail features extraction tool. *arXiv preprint arXiv:1711.08521*.

Kanter, J. M. and Veeramachaneni, K. (2015a). Deep feature synthesis: Towards automating data science endeavors. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE.

Kanter, J. M. and Veeramachaneni, K. (2015b). Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.

Kaspersky (2016 (accessed May 20, 2017)). *Spam and phishing in Q3 2016*.

Katz, G., Shin, E. C. R., and Song, D. (2016). Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984.

Khurana, U., Turaga, D., Samulowitz, H., and Parthasrathy, S. (2016). Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 1304–1307.

Koitka, S. and Friedrich, C. M. (2016). Traditional feature engineering and deep learning approaches at medical classification task of imageclef 2016. In *CLEF (Working Notes)*, pages 304–317.

Lam, H. T., Thiebaut, J., Sinn, M., Chen, B., Mai, T., and Alkan, O. (2017a). One button machine for automating feature engineering in relational databases. *CoRR*, abs/1706.00327.

Lam, H. T., Thiebaut, J.-M., Sinn, M., Chen, B., Mai, T., and Alkan, O. (2017b). One button machine for au-

tomating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*.

Ruano-Ords, D., Fdez-Riverola, F., and Mendez, J. R. (2018). Concept drift in e-mail datasets: An empirical study with practical implications. *Information Sciences*, 428:120 – 135.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 379–388, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Shams, R. and Mercer, R. E. (2016). Supervised classification of spam emails with natural language stylometry. *Neural Computing and Applications*, 27(8):2315–2331.

Tran, K.-N., Alazab, M., and Broadhurst, R. (2013). Towards a feature rich model for predicting spam emails containing malicious attachments and urls. In *Eleventh Australasian Data Mining Conference Canberra, ACT*, volume 146.

Tretyakov, K. (2004). Machine learning techniques in spam filtering. Technical report, Institute of Computer Science, University of Tartu.

Tweedie, F. J. and Baayen, R. H. (1998). How variable may a constant be? measures of lexical richness in perspective. *Computers and the Humanities*, 32(5):323–352.

Zaid, A., Alqatawna, J., and Huneiti, A. (2016). A proposed model for malicious spam detection in email systems of educational institutes. In *Cybersecurity and Cyberforensics Conference (CCC)*, pages 60–64. IEEE.