Email Classification Dataset: Spam or Ham

Alberto Gadda, Paolo Guerini Rocco

CdLM Data Science, Università degli studi di Milano Bicocca

Abstract

In the context of electronic communication, spam and ham refer to the content of an email message. Spam is defined as unsolicited, bulk email that is typically sent indiscriminately to a large number of recipients and is considered a nuisance due to its volume and lack of relevance to the user. In addition, spam can also include malicious or harmful content, such as viruses or malware. On the other hand, ham refers to legitimate email that is not spam, as it is relevant and desired by the recipient. For this reason, it is essential to enable email providers to accurately identify and separate ham from spam in order to keep the user mailbox safe and clean. This paper aims to tackle exactly this classification issue, presenting also a deep dive on the similarities and differences between the topics treated in each type of email.

Introduction

In the context of electronic communication, spam and ham refer to the content of an email message. Spam is defined as unsolicited, bulk email that is often sent for commercial purposes, such as advertising a product or service. This type of email is typically sent indiscriminately to a large number of recipients and is considered a nuisance due to its volume and lack of relevance to the user. In addition, spam can also include malicious or harmful content, such as viruses or malware. These emails are often sent with the intention of causing damage to the recipient's computer or network and may successfully harm the recipient in case of distraction or unawareness. A common type of malicious email is called *phishing*, which is a type of cyber attack that involves sending fraudulent emails that appear to be from a legitimate source. These emails often try to convey a sense of urgency in the recipient to click links or open attachments that either install malware on the recipient's computer or trick them into entering sensitive information in a fake website [3].
The exact opposite of spam is ham, which refers to legitimate emails that are relevant to the recipient. Ham is generally intended as anything the recipient has opted to receive, such as personal emails from friends, family or any other trusted senders [3].

To provide a perspective on the extent of this phenomenon, a report published by the Statista Research Department in July 2022 [1] presents evidence that the global spam volume, despite a 20% decrease from January 2014 to December 2021, still accounts for almost half the total email volume, precisely 45.37% (Fig. 1). This highlights the necessity for email providers to accurately identify and separate ham from spam in order to keep the user mailbox safe and clean.

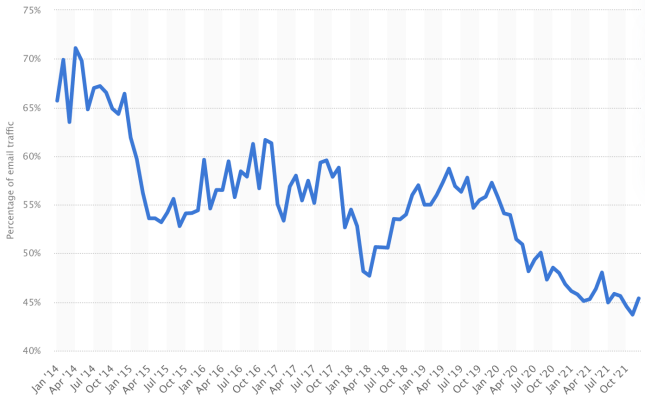

Figure 1. Global spam volume as a percentage of total e-mail traffic from January 2014 to December 2021, courtesy of statista.com.

This paper aims to tackle exactly this classification issue, presenting also a deep dive on similarities and differences between the topics treated in each type of email. The dataset chosen for the task is titled "Email Spam Dataset" [2] and can be found on kaggle.com.

EDA and Data Preprocessing

The "Email Spam Dataset" dataset [2] chosen for the task is comprised of three different CSV collections of labeled (spam/ham) emails:

- *completeSpamAssassin* (6046 rows, 31% being spam)
- *enronSpamSubset* (10000 rows, 50% being spam)
- *lingSpam* (2605 rows, 17% being spam)

The three components share the same structure and add up to a total of 18651 rows, 75% of which were used for the training set and 25% of which were separated via stratified sampling over the label to constitute the test set. The training set contains 13988 records, 39% of which being spam, divided in two columns[1]:

- *Body*: string variable (email content)
- *Label*: binary response variable (1=spam/0=ham)

There are a total of 735 duplicates and no missing values, although there are 405 emails with "[empty]" body pertaining to both spam and ham. All of these uninformative instances were removed to prevent harm to the learning process.
Since all the bodies from the second and third datasets started with the string "Subject:" followed by the email subject, this string was removed and the subject was kept as part of the email body to match the structure of the first dataset.

[1] Excluding extra columns containing indexes.

The distribution of the length of the documents (Fig. 2) is extremely heavy-tailed on the right-hand side, with median equal to 927 and values in interval $[0, 129635]$.

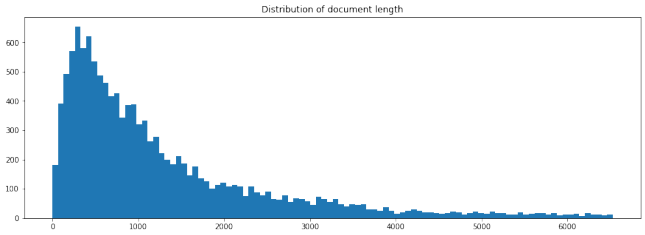

Figure 2. Length distribution (character count) of the documents.

The comparison between spam and ham body lengths (Fig. 3) shows a substantial tendency of the former being shorter than the latter, especially in regards to character and word count.

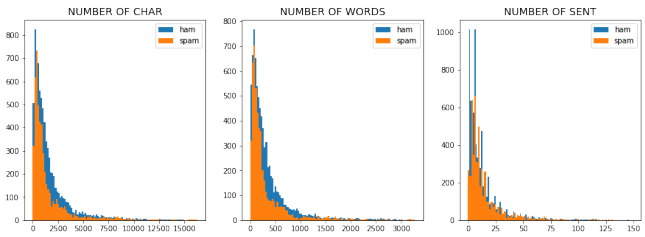

Figure 3. Length comparison between spam and ham over character, word and sentence count.

The preprocessing steps applied to each email body started with the tokenization, i.e the identification of units from raw text [4]. In the case of English language, the easiest way is to separate tokens based on the occurrence of whitespaces. During this process, every character was converted into lowercase and any special characters such as punctuation marks, numbers and escape sequences were removed. Through the use of regular expressions any references to links, email addresses or HTML tags were eliminated to polish the dataset.
After the identification of tokens, all stop words were discarded, i.e. extremely common words that have little to no value in solving the task [5].
Stemming (via Porter Algorithm) was chosen instead of lemmatization to reduce words to their root. Despite being an approach that ignores the semantics and could harm precision [6], it still outperformed lemmatization both in computational cost and final classification performance.
At last, due to the fact that several records (both spam and ham) contained non-English paragraphs, all the content was translated to English. The precise amount of records affected by this issue (roughly 10%) was hard to estimate due to the unstable accuracy provided by the *cld2* Python library, especially in case of mid-sentence language switches. It would often classify these instances as unknown or misidentify the precise boundary of the language switch. The whole training set was translated indiscriminately due to this limitation, also because the computational cost of translating all the emails was not much more impactful than translating only some.
All of the steps above set the ground for a Bag-of-Words representation of each document.

Bag-of-Words Model and its successors

The Bag-of-Words (BoW) model is a vector-based representation for documents in a corpus, where each element denotes the number of occurrences for a word in a given document. BoW has to face many challenges due to its intrinsic extreme sparsity, high dimensionality and inability to retain semantic meaning behind text [7]. Nevertheless, it is still widely used for its simplicity.
A way to enhance the BoW model is to use TF-IDF. This approach applies a weighting scheme on top of the term frequency (TF), the Inverse Document Frequency (IDF), that measures the rarity of a term in the corpus given a term t_i and a document d_j belonging to the corpus D [8]:

$$IDF_{i,j} = \log \frac{|D|}{|d_j \in D : t_i \in d_j|} \tag{1}$$

The TF-IDF value is then computed as $TF \times IDF$.
There is an additional tweak to BoW that attempts to fix its inability to capture context, being the n-gram model. This language model works considering n-long sequences of words [9], generating as a side-effect higher dimensionality and sparsity the more n increases. Due to computational limitations, the value for n was kept equal to 1 (unigram model), with an overall term frequency of 30 in the whole corpus as minimum requirement for a term to be included in the vocabulary (4674 terms met this condition).

Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix A is the factorization of A into the product of three matrices U, V and S. The columns of U and V are orthonormal and S is a diagonal matrix with decreasingly ordered values [10]:

$$A_{(n\times p)} = U_{(n\times k)} S_{(k\times k)} V^T_{(k\times p)} \tag{2}$$

This is useful to map A into a k-dimensional subspace, with $k \ll p$ and k inversely proportional to the information loss caused by the approximation obtained via SVD. The fundamental benefit of this process is to reduce the dimensionality of a sparse matrix, tuning k according to the trade-off between sparsity and information loss.

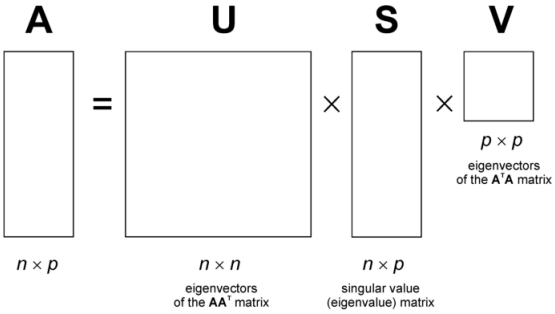

Figure 4. Singular Value Decomposition of a matrix A in case $k = p$ (lossless).

In this paper the chosen value for k was 500, selected against the other tested values of 300 and 1000, based on its classification performance over the training set.

Feature Engineering

After applying feature reduction via SVD and keeping only the first 500 principal components, an additional set of 7 handcrafted features was included as a design choice:

- *link_removed*: if any link was removed
- *email_removed*: if any email address was removed
- *n_char_removed*: number of characters removed
- *lev_dist*: Levenshtein distance from the original body, that calculates the minimum number of deletions, insertions or substitutions required to transform a string into another [11]
- *char*: number of characters before preprocessing
- *words*: number of words before preprocessing
- *sent*: number of sentences before preprocessing

The core intuition behind these handcrafted features is that general purpose information about the structure of email bodies may highlight patterns that are useful for spam detection. These additional features were standardized to match the ranges of values produced by the SVD: data standardization is a feature transformation technique that lets a variable have mean equal to 0 and variance equal to 1. This is especially useful when features have different ranges of values because it could hinder the training process [12].
All the features hereby presented were be used to perform the classification task without any further selections.

Classification

The first task tackled in this paper is classification, defined as the apprehension of a set of rules to classify new data based on some learning process performed over labeled training data [13]. Four different classifiers contained in the *sklearn* Python library were tested to predict whether a document is spam or ham:

- C-Support Vector (*SVC*)
- Multi-layer Perceptron (*MLPClassifier*)
- Gradient Boosting (*GradientBoostingClassifier*)
- Gaussian Naive Bayes (*GaussianNB*)

Before proceeding, let us provide a brief overview of the core intuition behind these classification algorithms.

C-Support Vector: Support vector machines are learning models that construct a hyper-plane or set of hyper-planes in a high or infinite dimensional space to classify data points. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. While trying to find the optimal margin, when a sample is misclassified or within the margin boundary it incurs a penalty controlled by the penalty term C, acting as an inverse regularization parameter [14].

Multi-layer Perceptron: Multi-layer Perceptrons (MLP) are architectures comprised of multiple layers connected to one another. The leftmost layer, known as the input layer, consists of a set of neurons representing the input features. The non-linear layer placed in the middle is called hidden layer and the rightmost one is known as the output layer, whose neurons represent the output features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation, followed by a non-linear activation function. The output layer receives the values from the last hidden layer and transforms them into output values. MLP is trained using gradient descent and the gradients are calculated via backpropagation [15].

Gradient Boosting: Gradient boosting is an ensemble learning algorithm in which a strong model is built using a collection (ensemble) of weak models that perform poorly [16]. At each iteration, a new weak tree is trained specifically on data classified poorly by the previous trees, attempting to outperform them. After training, the new weak tree is added to the model, without altering the weights of the previous trees, and the process repeats [17].

Gaussian Naive Bayes: Naive Bayes based learning algorithms apply Bayes' Theorem, with the assumption of conditional independence between every pair of features given the value of the class variable. The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the said conditional feature distributions, which in case of Gaussian Naive Bayes are assumed to be Gaussian. In spite of their apparently over-simplified assumptions, Naive Bayes classifiers have worked well in many real-world situations, especially document classification and spam filtering [18].

Since the training set has a 39% to 61% ratio of spam records, it not heavily impacted by class imbalance. This implies that the accuracy metric, defined as the ratio between the correct predictions over the total predictions, will be evaluated as it is well-suited for balanced datasets [19].
Users most likely prefer type 2 error (false negative) against type 1 error (false positive), which means that classifying ham emails as spam is much worse than the other way around. Under this reasonable assumption, in the current paper will also be taken into account the precision metric, defined as the ratio between the correct positive predictions over the total positive predictions, since it highlights the amount of type 2 errors [19].
The following section will provide an overview of the performance on the training set for each type of learning algorithm. Together with their training performance will also be shown their performance on the test set, to which were applied the same preprocessing steps of the training set to ensure consistency[2].

[2]The only exception is the standardization of the features contained in the test set. In this case, the values for the mean μ and standard deviation σ to compute the standardization must not be recalculated, but instead need to be identical to the values used for the standardization on the training set.

Classification Performance

The performance on the training set for each type of learning algorithm is the following:

Model	Accuracy	Precision	Recall
C-Support Vector	0.97	0.95	0.98
Multi-layer Perceptron	1.00	1.00	1.00
Gradient Boosting	0.98	0.97	0.97
Gaussian Naive Bayes	0.80	0.77	0.68

Gaussian Naive Bayes has the lowest training accuracy by a wide margin, while the other three models perform similarly and close to perfection. This is not necessarily a favorable outcome though, because it is a rather clear indicator of the presence of overfitting, intended as the loss of generalization capabilities occurring when the model learns to fit the training data too meticulously [20].
Evaluating the performance on the test set is the only way to truly gauge how models are able to generalize and provide correct predictions over a set of never-seen data. The performance on the test set was as follows:

Model	Accuracy	Precision	Recall
C-Support Vector	0.82	0.75	0.80
Multi-layer Perceptron	0.78	0.74	0.67
Gradient Boosting	0.69	0.60	0.61
Gaussian Naive Bayes	0.68	0.59	0.59

The results on the test set show a clear superiority of the C-Support Vector model, better than all the others in every metric. Even if each model scored significantly worse on the test set than on the training set due to overfitting, the C-Support Vector was more able to generalize. This was achieved thanks to a strong regularization parameter value $C = 0.6$. Furthermore, precision is high also because the *SVC* module offers a "balanced" weighting scheme setting that gives to each class (spam/ham) an importance inversely proportional to its frequency, enabling the classifier to focus slightly more on the spam records [14].

Topic Modeling

Topic Modeling is an unsupervised machine learning technique aimed at clustering together word groups and similar expressions that best characterize a set of documents. One of the most widely used method for assigning topics is the Latent Dirichlet Allocation (LDA).

LDA: The LDA is a generative probabilistic model structured as a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a mixture of topics and each topic is modeled as a mixture of words [21]. The LDA does not autonomously detect an optimal value for topic numerosity, since there is no objectively optimal way to choose it. For this reason, the user is required to choose and input manually the desired number of topics to look for, preferably with the help of some evaluation criteria. In this paper, the chosen approach was to use two intrinsic evaluation measures: perplexity and coherence.

Perplexity: Perplexity is calculated performing hold-out classification by means of a topic model. The model is trained in order to quantify the amount of perplexity, that is uncertainty, while making predictions on a test set [22]. To criticise this evaluation measure, a paper by Chang et al. from 2009 demonstrated that often perplexity and human interpretability are positively correlated, implying that topic models performing better on held-out likelihood may infer less semantically meaningful topics [25]. Nonetheless, perplexity was considered in this paper, while keeping particular caution in regard to its limitations.

Coherence: Coherence measures the conditional likelihood of the co-occurrence of words in a topic [22]. In detail, the coherence measure chosen for this paper is the C_v, first introduced in a publication from 2015 by Röder et al. [23] and later in 2017 summarised by Syed and Spruit in four brief steps [24]:

1. segmentation of the data into word pairs
2. calculation of word pair probabilities
3. calculation of a confirmation measure (normalized pointwise mutual information) that quantifies how strongly a word set supports another word set
4. aggregation of individual confirmation measures into an overall coherence score

In this paper, the C_v coherence was computed in respect to the top 10 words per topic identified.
The document representation chosen for topic modeling was a pure BoW model, with minimum required overall term frequency of 5 but also document frequency less than 50%.

Topic Numerosity: As evident in Fig. 5, the optimal topic numerosity according to the coherence curve is 2, whilst perplexity suggests that the higher the number of topics the better. This case perfectly embodies the aforementioned positive correlation between perplexity and human interpretability, since running LDA on more than 10 topics resulted in almost indistinguishable topics with multiple instances of shared words. This lead to discard perplexity as an evaluation measure and to rely solely on the C_v coherence.

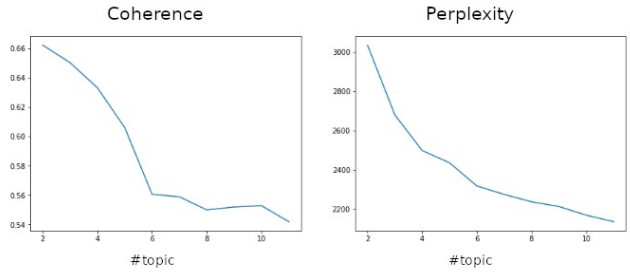

Figure 5. Curves of coherence (left) and perplexity (right) by topic numerosity.

Despite being recommended by the coherence curve a topic numerosity of 2, the threshold was raised to 3 as an attempt to check whether each topic pertains to a single dataset component or if multiple sources share the same topics.

Single topic hypothesis: For this specific dataset, the hypothesis of a single topic per source is inferred by a paper published by Metsis et al. in 2006, containing the origin and a brief commentary of the three dataset components [26]:

- LingSpam is the earliest dataset since it dates back to the year 2000. It consists of a mixture of spam messages received by one of the authors and legitimate messages sent via the Linguist list, a moderated mailing list about the science and profession of linguistics. Its ham messages are more topic-specific than the messages most users receive, despite including also job postings, software availability announcements and even flame-like responses [26][27].

- SpamAssassin contains ham messages that were received by many different users, thus providing more topic variety but also making it inappropriate for experimentation with personalized spam filters [26].

- Enron contains personal emails of 158 employees from the Enron Corporation, in the years leading up to the company's collapse in December 2001 [28]. Six of these employees in particular had a large mailbox, a favorable condition for the development of personalized spam filters [26].

The aforementioned description of the origin for all the three dataset components explains the cause of the single topic hypothesis: LingSpam would probably be about linguistics, Enron regarding business and SpamAssassin yet to be determined.

Topic Modeling Results

The intertopic distance map (Fig. 6) is a visualization in a two-dimensional space useful to represent the topics identified with the LDA. The area of these topic circles is proportional to the amount of words that belong to each topic across the dictionary [29]. For each topic is also displayed a chart with a set of red bars overlaying blue bars (Fig. 7). Blue bars represent the overall term frequency in the corpus for each word, while red bars show the estimated term frequency within the selected topic. If the red bar entirely eclipses the blue bar, it means that such term exclusively belongs to the selected topic [30].
The library for this visualization also offers a slider to adjust the value of the relevance parameter λ in range $[0,1]$, which determines the trade-off between word exclusivity to a topic ($\lambda = 0$) and term frequency within that topic ($\lambda = 1$). The authors of this visualization found in a user study that $\lambda \approx 0.6$ was optimal for interpreting the topics, although they expected this value to change based on the data and individual topics [29].
Fig. 6 and Fig. 7 show the results obtained by performing topic modeling through the use of the LDA algorithm. The total number of topics identified is equal to 3 and for each topic are displayed the top 10 words sorted by relevance, with $\lambda = 0.6$ as suggested.

The intertopic distance map in Fig. 6 shows that the three identified topics are practically the furthest they could be from one another, despite both topic 2 and 3 having a similar value in regard to the second principal component. By inspecting Fig. 7, the difference between topics is straightforward:

- Topic 1 is the most generic of the three, with general purpose words like "email", "use", "list" or "make", alongside with presumably e-commerce related terms like "get", "order", "free" and by extension also "time" (probably intended as limited time offers and similar).

- Topic 2 is clearly related to business with words such as "company", "trade", "deal" or "market". In addition, the main clue is the most relevant word "Enron", the name of the company and also one of the dataset sources. The second word "ECT" primarily refers to the acronym of the company after coining its new name in 1994: "Enron Capital & Trade Resources" (ECT) [31]. This term could also have some degree of intersection with the financial ECT sector, which stands for "Energy, Commodities and Transportation" [32], due to the fact that Enron predominantly traded natural gas as energy commodity [31].

- Topic 3 is undoubtedly related to languages and academic activities, with terms like "language", "paper", "education", "research" and "study".

Figure 6. 2D intertopic distance map of the three topics obtained.

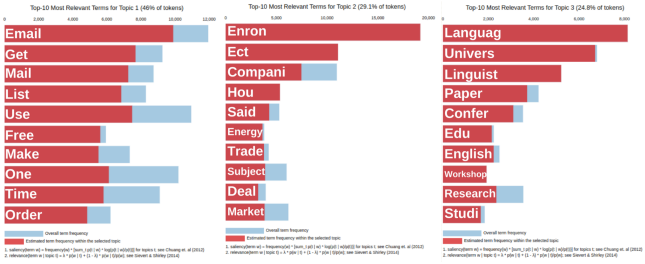

Figure 7. Top 10 relevant terms contained in topic 1 (left), 2 (middle) and 3 (right).

Repeating the same analysis separately for ham emails does not alter the topics by much in terms of interpretation, as shown in Fig. 8. On the contrary, topic modeling on spam emails provides very different results from before[3], as seen in Fig. 9:

- Topic 1 encompasses all sorts of instigation towards generically imprudent user behavior, with words such as "free", "click", "get", "please", together with references to email addresses and links.
- Topic 2 is similar to topic 1, but more directly related to financial fraud. It contains terms like "stock" and invitations to "invest" or "make" something due to time constraints ("within") or external opportunities ("report", "news", "may").
- Topic 3 is more about simulation of trusted institutions, including terms such as "company", "bank", "fund" and "govern" that "grant" or "expect" something from the user or his "account".

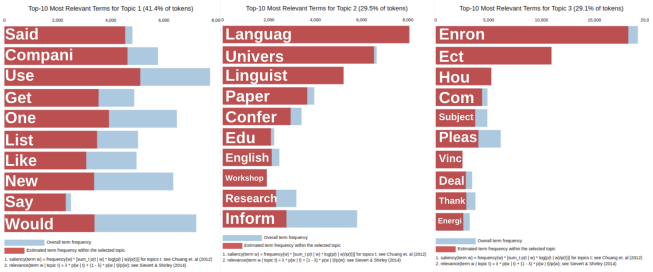

Figure 8. Top 10 relevant terms contained in topic 1 (left), 2 (middle) and 3 (right) for ham emails only.

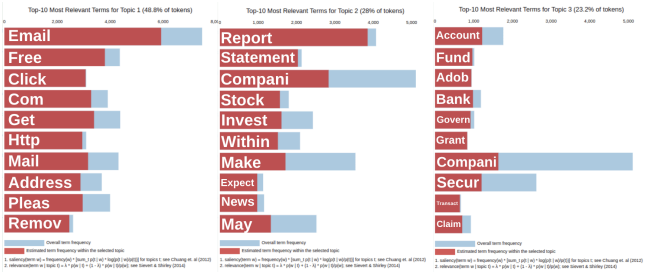

Figure 9. Top 10 relevant terms contained in topic 1 (left), 2 (middle) and 3 (right) for spam emails only.

Given the fact that most of the identified topics, especially in ham emails, appear to be directly related to the dataset descriptions provided in the previous chapter, the last piece of analysis proposed in this paper is to check whether these topics obtained through LDA can be used as informal "document classifiers" to match emails back to their source datasets.
On this regard, Fig. 10 shows how indeed ham topics have almost one to one relationships with their original datasets, whilst this does not hold true for topics identified from spam emails, for which the main themes are in general more recurrent and less domain dependent.

[3]Please note that the ID number assigned to each topic is generated from scratch every new run of the LDA algorithm, so there is no direct topic-topic relationship between different sets of results.

HAM	Dataset	% of Documents from this dataset
TOPIC 1	Assassin	90
TOPIC 2	Ling	85
TOPIC 3	Enron	99

SPAM	Dataset	% of Documents from this dataset
TOPIC 1	Enron	67
TOPIC 2	Assassin	25
TOPIC 3	Enron	88

Figure 10. Analysis on the relationship between topics and dataset sources for ham (left) and spam (right) emails.

Conclusion

The aim of this paper was to tackle classification between spam and ham emails, in order to perform spam filtering and to remove unsolicited content from the user mailbox.
The best performing model was the C-Support Vector, capable of obtaining a significant 0.82 accuracy, 0.75 precision and 0.80 recall over a set of never-seen data. Given this performance, the TF-IDF weighted Bag-of-Words model revealed itself as an effective document representation approach for this dataset, despite its simplicity.
The deep dive on topic modeling though LDA also highlighted the fact that each of the three sources composing the dataset had a strong topical identity. For this reason, it was possible to trace back the origin of most ham emails with an unsupervised approach.

Sitography

1. https://www.statista.com/statistics/420391/spam-email-traffic-share
2. https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset
3. https://securityblog.port.ac.uk/?p=1335
4. https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html
5. https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html
6. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
7. https://ieeexplore.ieee.org/abstract/document/7891009
8. https://hcis-journal.springeropen.com/articles/10.1186/s13673-019-0192-7
9. https://web.stanford.edu/ jurafsky/slp3/3.pdf
10. https://www.cs.cmu.edu/ venkatg/teaching/CStheory-infoage/book-chapter-4.pdf
11. https://people.cs.pitt.edu/ kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm
12. https://courses.cs.washington.edu/courses/cse446/21wi/sections/04/section04.pdf
13. https://www.cs.ucdavis.edu/ vemuri/classes/ecs271/lecture3.pdf
14. https://scikit-learn.org/stable/modules/svm.html#svm-classification
15. https://scikit-learn.org/stable/modules/neural_networks_supervised.html
16. https://blog.paperspace.com/gradient-boosting-for-classification
17. https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn
18. https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes
19. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
20. https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/regularization.pdf
21. https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
22. https://highdemandskills.com/topic-model-evaluation/#h2-2
23. https://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf
24. http://www.saf21.eu/wp-content/uploads/2017/09/5004a165.pdf
25. https://proceedings.neurips.cc/paper/2009/file/f92586a25bb3145facd64ab20fd554ff-Paper.pdf
26. https://www2.aueb.gr/users/ion/docs/ceas2006_paper.pdf
27. http://nlp.cs.aueb.gr/pubs/ir_memory_based_antispam_filtering.pdf
28. https://en.wikipedia.org/wiki/Enron_Corpus
29. https://community.alteryx.com/t5/Data-Science/Getting-to-the-Point-with-Topic-Modeling-Part-3-Interpreting-the/ba-p/614992
30. https://neptune.ai/blog/pyldavis-topic-modelling-exploration-tool-that-every-nlp-data-scientist-should-know
31. https://www.oreilly.com/library/view/enron-ascending/9781118549575/c11.xhtm
32. https://www.acronymfinder.com/Business/ECT.html