Engineers have been working hard on improving the accuracy and efficiency of machine learning models since people started to use the data left on the internet as a valuable resource for analytics. We have seen many modern machine learning applications in real life and research projects, such as advertisement services, recommendations systems, artificial intelligent robotics, and self-driven vehicles, etc. For many of these applications, we not only care about the ultimate output of the machine learning models but also expect for explanations for results. This paper aims to present an approach to explaining machine learning algorithms and relative evaluation process. We use a random forest model to perform the text classification and choose the LIME (Local Interpretable Model-agnostic Explanations) to work as the explainer and the PMI (Pointwise Mutual Information) as the evaluation baseline.
UNDERSTANDING AND EVALUATING A TEXT CLASSIFICATION MODEL USING INTERPRETABLE MACHINE LEARNING METHODS

by

Jiesong He

A Master’s paper submitted to the faculty of the School of Information and Library Science of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in Information Science.

Chapel Hill, North Carolina

Approved by

Yue Wang
# Table of Contents

1. INTRODUCTION ............................................................................................................. 2

2. LITERATURE REVIEW .................................................................................................... 4

   2.1 WHAT IS MACHINE LEARNING? ................................................................................. 4
   2.2 A BRIEF SUMMARY FOR MACHINE LEARNING ROUTINE ............................................ 6
   2.3 INTERPRETABILITY OF MACHINE LEARNING MODELS ............................................... 8

3. METHODOLOGIES ........................................................................................................... 9

   3.1 INTRODUCTION TO METHODOLOGIES ADOPTED IN THE EXPERIMENT ..................... 9
   3.2 DATASETS ..................................................................................................................... 10
   3.3 RANDOM FOREST MODEL ............................................................................................ 11
   3.4 LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS (LIME) ............................... 13
   3.5 THE BASELINE: POINTWISE MUTUAL INFORMATION (PMI) ........................................ 14
   3.6 METRICS SEARCHING AND SELECTION ................................................................ 15
   3.7 TEXT DATA PRE-PROCESSING ................................................................................. 18
   3.8 EVALUATION APPROACH ......................................................................................... 19

4. CONCLUSIONS .................................................................................................................. 20

5. LIMITATIONS .................................................................................................................. 23

6. FUTURE WORKS ............................................................................................................... 24

REFERENCES ..................................................................................................................... 26

APPENDIX .......................................................................................................................... 28
1. Introduction

With the development of machine learning technology, machine learning is gradually being applied in more and more fields. For example, when we open the app store, the platform recommends potential apps we might be interested in. When users post in forums, there are also machine learning models preventing users from sending words in forbidden lists. Other popular applications are self-driving vehicles, IoT applications, and intelligent robotics etc.

Nowadays, many researchers and their papers have discussed in detail how to improve the accuracy of machine learning algorithm predictions and how to implement and deploy machine learning algorithms. They posed extraordinary impacts on the development of machine learning in academia and industry. In addition, another interesting and noteworthy topic is how to interpret, understand and evaluate a machine learning algorithm.

For linear models, such as linear regression models and multivariate regression models, we can use a vector of parameters to express machine learning models. Through the regression equation, we can understand and evaluate how the model achieves output based on input. However, in most machine learning algorithms, our models are often nonlinear, such as deep learning models, random forests models and so on. For these algorithm models, the regression models are usually nonlinear structures and we can only
get a “black-box” system, with which human-interpretable details about the regression are not available.

For some domain, such as advertisement or recommendation system, the transparency of the machine learning algorithms is not the first priority when considering implementing one, although it’s good have it. However, there are also many use cases where interpretable machine learning algorithms are desirable. For example, in some forums or social networks, the posting system detects whether the content contains forbidden words before posting the user’s comments. If it does, the system will prompt a notification asking users to modify their comments before posting again. In this context, the machine learning can perform better if it tells users which words are forbidden in their comments so that users can appropriately express their comments in a more efficient way.

Moreover, in other domains of application, developers rely on the interpretability of machine learning algorithms to prevent critical failures caused by their projects. For example, in the domain of self-driving car, engineers have high standard of requirements on the safety issue. Reliable test results alone are not sufficient for implementing an algorithm. The level that understanding engineers need to achieve is to explain and evaluate the black box system they are intended to implement. Exploring the interpretability of a machine learning model also plays an extremely important role in other domains such as medicine and health care.

This paper will be based on an example use case of the interpretability of machine learning algorithms in the domain of natural language processing, summarizing a methodology to explain machine learning algorithms. This methodology will also help to
provide an approach to understanding and evaluating a machine learning algorithm before deciding to adopt it and implement it.

In terms of the method of interpreting a machine learning algorithm, Christoph Molnar (2020) made some very inspiring introduction to the interpretability of linear and non-linear machine learning models. For a linear regression or logistic regression model, the interpretability can be represented as a vector of weights which measure the importance of each feature. For those machine learning models that don’t present apparent interpretability, one of the approaches is the Local Surrogate (LIME) method. LIME separate the interpretability of machine learning models into explaining multiple linear models, each approximating a local segment of the underlying non-linear regression function. More specifically in the approach shown by this paper, LIME outputs a list of words that are determined as the most contributing ones for the classification result made by the random forest model.

To evaluate the interpretability of LIME, the pointwise mutual information (PMI) was chosen as a measurement of the association of words to classes. By comparing the outputs from LIME and PMI scores of each word in the documents, the capacity of LIME to interpret a classification model can be observed.

2. Literature Review

2.1 What is machine learning?

Over the past two decades, scientists and engineers have contributed a lot to the machine learning, an interdisciplinary domain, which can be roughly concluded as a discipline focusing on two fundamental questions: (1) How can one construct computer systems that automatically improve through experience? (2) What are the fundamental statistical-
computational-information-theoretic laws that govern all learning systems, including computers, humans, and organizations?

Many developers of artificial intelligence systems realized that, for the majority of applications, it can be far simpler and more feasible to construct a system by providing it with examples of input-output pairs than to manually implement algorithms to calculate the outputs based on anticipated possible inputs. The methodology of training a system, i.e. machine learning, has been adopted in both software development and laboratory practice. For example, software engineers use machine learning algorithms to develop software for computer vision, natural language processing, pattern recognition, and robot controls, etc. Scientists and engineers also designed and implemented machine learning methods to analyze big-scale and high-throughput experimental data.

A learning problem can be defined as a process of analyzing existing data, i.e., training data, to generate a model that determines the output for a new dataset. For example, the house price can be influenced by multiple variables, such as location, square footage, age, and number of rooms, etc. The training data in this example is the dataset we already have in hand that contains the values of variables mentioned above, which are referred to as features in machine learning. The other part of the training data is the label, which is the house price in this example. Therefore, we regard the house price as a dependent variable determined by the combined influence of multiple features. Briefly, to instantiate the function between features and labels is a very common learning problem in supervised learning. Other types of learning problems are also similar to this question. Unsupervised learning refers to a machine learning model that doesn’t have labels at all, in which case the target of the machine learning process is to cluster a dataset rather than generate a
label for each entries. Semi-supervised learning refers to a machine learning process with a small set data with labels and a large set of data without labels.

Jordan et al. (2015) describe machine learning as an interdisciplinary science, and they believe that machine learning sits on the crossroads of computer science, statistics and other disciplinaries concerned with automatic optimization, inference technology and decision-making science. On the other hand, the applications of machine learning also merge in various domains. For example, search engines generate searching results based on their understanding of the relevance between input and target websites. Recommendation systems profiles the users and prompt to users customized contents based on their usage habits. We can also find computer vision applications in medical practices and self-driving vehicles.

2.2 A brief summary for machine learning routine

As a conclusion, machine learning is a method that computers use to determine and improves predictions or performances based on a collection of datasets, which is also called an experience in machine learning problems.

To train and improve a machine learning system, machine learning engineers generally use three steps.

Step 1: Data collection. Every machine learning algorithm is based on learning by experience, which is also the training data. In house price predictions, examples of features of a house price problem and the value of house prize are experience for systems to learn from. In NLP problems, the experience is words in a document and the classification of the document. The more training data we can collect, the more effectively and more accurately we can train the machine learning system.
Step 2: Use collected data to train machine learning models. In the training step, machine learning engineers have a plenty of approaches and methods to leverage. However, in supervised learning, the ultimate goal is quite the same—to find the most suitable parameters, i.e. weights for features of training data. Such training methods include gradient ascent in maximizing the likelihood function, gradient descent in minimizing loss function, newton method in logistic regression, and quadratic programming in SVM etc.

Step 3: Predict or perform based on new data. After trained and improved the machine learning system, computers can use the algorithms to work on new data and generate an output of prediction or performance. This is also the phase engineers put the trained model into practice.

Molnar (2020) illustrates the training and application process of a machine learning model as the following figure.

![Figure 1: The process of training and applying a machine learning model](image-url)
2.3 Interpretability of machine learning models

For most machine learning models, the algorithm engine works more like a black box. Developers use training datasets to train the model iteratively, and in the end achieve the most optimal parameters for the algorithms at their bests.

As for how to see through the black box, it’s an issue of the interpretability of the machine learning models. While extra dedication is required to solve the interpretability issue, a black box system is totally acceptable in many user cases. Thus, there’s a trade-off between devoting to interpreting a machine learning model or saving the time costs for other effort in improving accuracy of the machine learning model or other perspectives. For example, in user cases such as commercial user profiling or recommendation systems, a classification label is a desired result in most instances. However, in user cases such as forbidden words detection, self-driving vehicles or medical applications, etc., machine learning engineers are expected to provide an explanation in terms of how the labels are classified or why the decisions are made.

2.4 A more detailed example of machine learning applications in NLP

Considering that this paper would focus on machine learning applications in natural language process, a deeper dive into an NLP example to try to understand the general intuition of using machine learning models to classify a document into given types may be useful. Assume that we have a list of documents and the classification label of each document. These documents are training data for generating the classification model.

Given this information, we can calculate the probabilities of word appearance in different classification. As illustrated in left column of the figure, we have four classes for training document, and based on the training data, the probabilities of word appearance have been
calculated in each document class as well as the probabilities of each class of the
documents in the training data. These kinds of probabilities are called parameters for a
machine learning model. If we have a new document that need to be classified, the model
we trained will determine which class the document will go to according the model and
parameters we’ve decided.

3. Methodologies

This section aims to present the framework of the research by providing details and
explanations on the datasets used through the research, the random forest model, LIME
method, and the PMI measurement.

3.1 Introduction to methodologies adopted in the experiment

To validate whether an interpretation model is effective enough to explain a machine
learning algorithm or determine which interpretation model to use, it’s important to
develop a reliable method to evaluate candidate interpretation models. In the domain of text classification, popular classifiers are Naive Bayes classification, Random Forest model, SVM and Neural Networks, etc. Christoph Molnar, in his e-book, introduced multiple machine learning interpretation methods such as Partial Dependence Plot (PDP), Accumulated Local Effects (ALE), and Local Interpretable Model-agnostic Explanations (LIME), etc. This paper tries to present a methodology for machine learning practitioners to evaluate classification algorithms. While a random forest model is chosen as the underlying classification model and the LIME method is selected to be the explainer of the machine learning model, this paper presents methodologies to explain a machine learning model and to evaluate if the explainer is effective enough to be accepted, serving as a roadmap for evaluations of other possible underlying machine learning models and explainers.

3.2 Datasets

In Christoph Molnar’s book *Interpretable Machine Learning-A Guide for Making Black Box Models Explainable*, he provided several collections of datasets for readers to play around. One of them is about the comments on YouTube, and this datasets collection is used for spam research and chosen as the training data in the research of this paper. In each dataset of this collections\(^1\), instances have features as comment_id, author, date, and comment. Instances are classified with labels of spam (1) and ham (0) for the training. According to the description of the producer of the datasets, labels were made with a collaborative tagging tool called Labelling\(^2\).

---

\(^1\) YouTube comments. Available at http://www.dt.fee.unicamp.br/~tiago/youtubespamcollection/

\(^2\) Labelling. Available at http://lasid.sor.ufscar.br/ml-tools/
The statistics for classifications of comments in the YouTube videos are shown in the following table.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>YouTube ID</th>
<th># Spam</th>
<th># Ham</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psy</td>
<td>9bZkp7q19f0</td>
<td>175</td>
<td>175</td>
<td>350</td>
</tr>
<tr>
<td>KatyPerry</td>
<td>CevxZvSJLk8</td>
<td>175</td>
<td>175</td>
<td>350</td>
</tr>
<tr>
<td>LMFAO</td>
<td>KQ6zr6kCPj8</td>
<td>236</td>
<td>202</td>
<td>438</td>
</tr>
<tr>
<td>Eminem</td>
<td>uelHwf8o7_U</td>
<td>245</td>
<td>203</td>
<td>448</td>
</tr>
<tr>
<td>Shakira</td>
<td>pRpeEdMmmQ0</td>
<td>174</td>
<td>196</td>
<td>370</td>
</tr>
</tbody>
</table>

Table1: Statistics for the datasets used in the experiment

3.3 Random forest model

In the experiments of this paper, the underlying model of the interpretability research is a random forest classification model. To obtain an overview understanding of random forest classification, let’s first spend some time with some notions of decision tree models.

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility (Wikipedia). An illustration for a decision tree model is given in the Figure 3.
In the Figure 3, we have outlook, humidity, and wind as decision nodes. Based on the value of decision nodes, we approach to the end nodes with value of Yes or No. Applying to the experiment in this paper, decision nodes are features, or say words, in the documents, and the values of nodes are TF-IDF scores for the words. The end nodes are labels of spams or hams.

Based on the understanding of decision trees, random forest provides nothing but a voting result across multiple decisions trees. More specifically, random forest models extract samples from the training data. After that, the algorithm grows a classification tree from each sample data and predict on new data by aggregating predictions of decision trees.

The scikit-learn, a python library, offers a module called RandomForestClassier\(^2\) to generate a random forest effectively.

---

1. Decision tree. Available at https://www.geeksforgeeks.org/decision-tree/
3.4 Local interpretable model-agnostic explanations (LIME)

Many of text classification models are black boxes to users. For example, when a random forest model is trained and applied to classify a document, the output is nothing else but a class label. If developers would like to know the evidence for the classification, the classifier alone is not capable of doing so.

Among machine learning interpretation models that are developed to explain the classifier and decrypt the classification algorithms, the Local Interpretable Model-agnostic Explanations (LIME) provides a solution that explains the classification results by outputting a list of words in the document, sorted by the contribution to the classification.

In the book *Interpretable machine learning: A guide for making black box models explainable*, the author provided some examples for the mechanism of LIME to explain machine learning models for text and image classification. The following paragraph shows how LIME works on text classification tasks.

Let’s first list the classification results for two documents, where label 0 refers to hams and 1 refers to spams.

<table>
<thead>
<tr>
<th>Content</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>173 For Christmas Song visit my channel! ;)</td>
<td>1</td>
</tr>
<tr>
<td>267 PSY is a good guy</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Classification results from a text classifier

Based on the original documents, we generate a set of variations of the original document by removing one or multiple words from it. For example, one variation of the document “For Christmas Song visit my channel! ;)” could be “For Song visit my channel! ;)” or
“For Christmas my channel! ;)”. To represent the document by features of words, an example of variations is presented in Table 3.

<table>
<thead>
<tr>
<th>For</th>
<th>Christmas</th>
<th>Song</th>
<th>visit</th>
<th>my</th>
<th>channel!</th>
<th>;)</th>
<th>prob</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.17</td>
<td>0.57</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.17</td>
<td>0.71</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.99</td>
<td>0.71</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.99</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 3: Representations of document variations in LIME model

In the table, the column “prob” refers to the probability of a variation is predicted as spam, the column “weight” stands for the proximity of the variation to the original document. For example, if one word is removed from a document of seven words, the proximity of the variation to the original document is $1 - 1 / 7 = 0.86$.

After observing the variation’s probability of being classified as a spam, we can infer that “channel!” is a word that contributes to the classification results of “spam”, to a large extent. After removing it from the document, the probability for the document to be labeled with “spam” decreases sharply. LIME model quantifies the contribution of each word in the list and outputs a list word sorted by the contribution score.

### 3.5 The baseline: pointwise mutual information (PMI)

To determine whether an interpretation model explains an underlying machine learning algorithm accurately enough, we firstly need to define a baseline for the evaluation process. The Pointwise Mutual Information (PMI) is a score that measures the association of a word with a class and in this paper’s experiments is calculated as
\[ PMI(word, class) = \log \frac{P(word, class)}{P(word)P(class)} \]

where \( P(w, c) \) refers to the number of documents that contain the word \( w \) and belongs to the class \( c \), \( P(w) \) refers to the number of documents that contain the word \( w \), and \( P(c) \) refers to the number of documents that belong to the class \( c \), with all the numbers of documents normalized by dividing the total number of document \( N \). A higher PMI score of a combination of a word and a class indicates a stronger correlation of the word to the corresponding class.

As the LIME method generates a list words that are determined as the most critical words for the classification, the designed baseline also outputs a list of words, which are sorted by PMI scores.

3.6 Metrics searching and selection

Through the research, especially after the explanation by LIME model is obtained, multiple metrics are required to measure and evaluate the performance of the text classifier and the explainer.

For example, after the explainer generates a list of words that are considered as most contributing words in the documents to the classification, we expect to have a baseline to evaluate how good are the explanations provided by LIME. Similar to TF-IDF, which is a measurement of how much a word is associated with a document, a metric that measures how much a word is associated with the classification label is expected to be used as the evaluation baseline. After several initial practices by myself and some experiments on suggestions from my paper supervisor, I decided to calculate the pointwise mutual information (PMI) score for every word in a document, according to the classification of the document, and sort the words by the PMI scores in a descending order.
By this step, we have a list of words that determined by LIME as the most contributing words to the classification of the document and another list of words in the document sorted by the PMI scores in a descending order. The next step is to evaluate the performance of LIME model based on the two lists. While the number of words in the LIME list is determined by the parameter passed into LIME method, PMI list contains all the words in the documents.

Original document: Hey subscribe to me

| LIME list: | subscribe | to | me | Hey |
| PMI list:  | subscribe | Hey | me | to |

Figure 4: Alignment comparison for entirely overlapped lists

Original document: Come and check out my new youtube chhannel, going to be posting daily!

| LIME list: | check | out | my | new | going |
| PMI list:  | posting | daily | chhannel | check | my | out | to | new | and | be | going | youtube | come |

Figure 5: Alignment comparison for partially overlapped lists

To quantify how much these two lists are aligned with each other, we have two approaches. The first approach is to calculate F1 scores by updating precision and recall. For example, if we have n words in the LIME list, and N words in the PMI list, we firstly take the first n, setting the value n to a variable \( idx \), words in PMI list and calculate the precision and recall. And we increase \( idx \) by 1 and keep doing this until all the words in LIME list can be found in the sublist of PMI list. The final F1 score is the measurement of alignment when we find all LIME words in the sublist of PMI list.
One of the advantages of evaluation based on F1 scores is that this method comprehensively takes into consideration both precision and recall rates. However, the relative orders of words in two lists are neglected in this method. If we consider the relative orders of words as the first priority when measuring the alignment of LIME list and PMI list, Kentall's tau is one of the metrics that could be used, which is also the practice in the experiment of this paper. Firstly, we traverse every word in the PMI list and extract words that also included in LIME list, keeping the same order in PMI list. Secondly, we import the Kentall's tau library from SciPy and calculate Kentall's tau for each document, using it as the performance measurement.

```python
# prepare an interim_list that only contains LIME words, keeping the same order in pmi_list
interim_list = []
for word in pmi_list:
    if word in lime_list and word not in interim_list:
        interim_list.append(word)

# import Kentall's tau method from SciPy
import scipy.stats as stats

# calculate precision, recall and the F1 score
precision = n / sublist_len
recall = sublist_len / N
F1_score = 2 * precision * recall / (precision + recall)
```

Kentall's tau is preferred in the experiment because the relative order of contributing words is an important factor to evaluate the performance of LIME model. Especially
among the datasets, some documents only contain a small number of words. For example, the 10th document in the dataset is “Hey subscribe to me”, the LIME list for this document is ['subscribe', 'to', 'me', 'Hey'], and the PMI list is ['subscribe', 'Hey', 'me', 'to']. If the evaluation metric is F1 score, the performance would be evaluated as perfect, which is definitely not true. Kentall's tau is able to detect the discrepancy among the orders of these two lists and, therefore, a better choice for the performance metric.

3.7 Text data pre-processing

As in many other researches in the machine learning domain, the data pre-processing section plays a very important role through the experiment.

After obtaining the raw data of five datasets, the first practice is to merge the five datasets into one Pandas dataframe. The next step is the feature extraction, where TF-IDF method is used to vectorize a document. After we convert the raw documents into vectors, we can use these instances to train the text classification model, a random forest model in the experience, and adopt an interpretation model to generate the evidence of classification.

The text data required when training classifier model and interpreter are raw document, which means except for writing raw data into Pandas dataframes, no other manipulations performed on the raw data. However, the raw data is not suitable for the PMI calculation, as no Python library for this calculation is adopted and all the PMI calculation is based on iterations and counting on words and documents in the datasets. To achieve such a goal, the contents of comments are converted into list of words through the PMI calculation process.
3.8 Evaluation approach

For each document, the next step is to define an approach to evaluating how well the LIME method works. After we generate a list of words from LIME and a list of words in document sorted by the PMI score, we can evaluate the LIME method by comparing the relative orders of words in LIME output and words sorted by PMI scores.

For example, one of the comments in the dataset is “Subscribe to my channel”. By the classifier in the experience, this comment is classified as a spam. While LIME explains the classification results by the word list ['Subscribe', 'channel', 'my', 'to'], the list of words sorted by PMI scores is ['Subscribe', 'channel', 'my', 'to']. In this case, we can observe a perfect alignment between the two list, and therefore the LIME method provides a good explanation for the classification of this case. For another sentence, “Hey subscribe to me”, the LIME list is ['subscribe', 'to', 'me', 'Hey'] and the PMI list is ['subscribe', 'Hey', 'me', 'to']. Unlike the first case, the LIME explanation for the second sentence shows some discrepancy comparing to the baseline.

To quantify the alignment between LIME explanations and the baseline, Kentall's tau was used as a metric to measure how much are aligned the relative orders of words in LIME explanations and words in PMI sorted list. With Python, we can call the calculation method from the SciPy library by the command `scipy.stats.kendalltau(x, y, initial_lexsort=True)`, where \( x \) and \( y \) refer to arrays of rankings and \( initial\_lexsort \) is an optional bool parameter, assigning whether to use lexsort or quicksort as the sorting method for the initial sort of the inputs with lexsort (True) by default.

For the calculation results, tau values close to 1 indicate strong agreement, values close to -1 indicate strong disagreement.
4. Conclusions

When attempting to calculate the Kentall's tau for each word in every document to evaluate the interpreter in a comprehensive manner, the program failed to generate the final results. Therefore, 50 of random document ids are selected from the corpus to exam the performance of LIME model on a random forest classifier. Remind that the Kentall's tau should be within a range of -1.0 to 1.0, with -1.0 representing a week alignment of word order and 1.0 representing a strong alignment. The distribution of Kentall's tau calculated from the sample in the experiment, with an average value of 0.0946, is presented in Figure 7.
Figure 7: the distribution of Kentall’s tau for 50 random comments

We can observe from the graph that the values of Kentall’s tau are concentrated within the range of -0.25 to 0.25, where a strong alignment of relative orders of words in LIME list and words in PMI ranked word list cannot be observed.

One of the factors that probably affects the performance of the interpreter might be the length of document. To check whether the length of documents substantially leads to variations in the interpreter’s performance, samples are extracted separately from a document set that only contains documents with more than 10 words and a document set that only contains short documents. The average Kentall's tau for long document sample and short document sample are 0.0160 and 0.1880. The distribution of performance are presented in the following chart.
Figure 8: the distribution of Kentall’s tau for long comments, short comments and mixed samples

Although the average value of Kentall’s tau for short comments, 0.1880, is slightly better than the other two average score, 0.0946 for mixed random comments and 0.0160 for long comments, either the average value or the distribution can hardly prove the existence of a good alignment between the relative word orders of LIME word list and PMI ranked list. We can know from the statistic results and the distribution of Kentall’s tau that LIME tells a different story from what the PMI does.

Another observation is that we can find more high-value Kentall’s taus in short comments than in long comments. As the LIME model is a document specified interpreter and the PMI is a global measurement, the alignment between LIME and PMI should be weaker in long comments, where we have more features and the local model is therefore inclined to over-fitting.
Examples of Kentall’s taus in short and long comments and corresponding LIME and PMI sorted word lists are presented in Figure 9 and Figure 10, where a high Kentall’s tau value stands for a good alignment of relative word orders in the LIME list and the PMI ranked list.

Original document: Subscribe To Më Please Guys

LIME list: Subscribe Please To Më Guys

PMI list: Please Subscribe To Guys Më

Kentall’s tau: 0.60

Figure 9: the Kendall’s tau in a short comment example

Original document: ayyy can u guys please check out my rap video im 16 n im juss tryna get some love please chrck it out an thank u

LIME list: check please out love my

PMI list: out my video check it please can
guys get love u some im thank
an rap n tryna juss chrck ayyy

Kentall’s tau: -4.0

Figure 10: the Kendall’s tau in a long comment example

5. Limitations

As the LIME method explains a text classifier by generating variations of the original documents, we could conclude that LIME’s explanations are document specific, which means to which extent a word contributes to the classification is evaluated within the scope of an individual document. However, the PMI score for every word in a document, the score determines the order of words in a document, is a global observation, which
describes the association of a word with a classification within the scope of the overall corpus. Although the Kentall's tau is adopted to mitigate the discrepancy by comparing the relative orders of the words in LIME list and the baseline, PMI-ordered list, the evaluation is not a perfect one.

Another limitation merged through the research is that the methodologies of different interpretation models are not the same. Therefore, the performance of an interpretation model might be better on certain types of text classifier and worse on others. In another word, if we find the most suitable interpretation model for a text classifier after applying the methodology introduced in this paper, the best explainer might be another model if the underlying text classifier is changed into another type. As the conclusion for one experiment is hard to extend to others, extra experiments are expected in practice.

6. Future works

As discussed in the limitation section, some issues remain unsettled through the research process. These issues majorly derive from two aspects: the discrepancy between the exposures of interpretation model’s output and the baseline for performance evaluation, and the hidden relationship between interpretation model’s performance and the combination of underlying machine learning algorithms and explainers.

To find a more accurate baseline for the performance evaluation, future works should go for either a document-specific metric that measure how much a word in the document contributes to the document’s classification or interpreter model that is not document-specific, measuring every word’s contribution to the classification within a global scope.

For the issues emerged from different suitability of classifier and interpreters, two approaches are considered as well. On the one hand, we could conduct experiments under
the same methodology with different classifiers and interpreters. For example, we could try to use interpret a Naïve Bayes classifier with a LIME model. With this approach, multiple trials are expected to achieve the best interpreter. On the other hand, we could analyze the property of classifiers at first and choose corresponding interpreter based on the classifier’s property. For example, a linear interpreter is more possible to perform better on a linear classifier.

Apparently, it’ll take more detailed works to concrete and implement these two perspectives of future work ideas. However, if these works can be proceeded, the accuracy and efficiency of the interpreter’s evaluation will be improved to a promising extent, contributing to the extension of evaluation methodologies in this paper to real-life user cases.
References


Appendix

Top 50 words ranked by the PMI with a spam label

<table>
<thead>
<tr>
<th>PMI</th>
<th>Word</th>
<th>PMI</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6894</td>
<td>out</td>
<td>5.2610</td>
<td>can</td>
</tr>
<tr>
<td>6.4800</td>
<td>this</td>
<td>5.2509</td>
<td>http</td>
</tr>
<tr>
<td>6.3563</td>
<td>on</td>
<td>5.1768</td>
<td>https</td>
</tr>
<tr>
<td>6.2643</td>
<td>my</td>
<td>5.1203</td>
<td>in</td>
</tr>
<tr>
<td>6.2419</td>
<td>and</td>
<td>5.1086</td>
<td>guys</td>
</tr>
<tr>
<td>6.2072</td>
<td>to</td>
<td>5.0479</td>
<td>have</td>
</tr>
<tr>
<td>6.1834</td>
<td>Check</td>
<td>5.0354</td>
<td>m</td>
</tr>
<tr>
<td>6.0550</td>
<td>a</td>
<td>5.0226</td>
<td>money</td>
</tr>
<tr>
<td>6.0458</td>
<td>I</td>
<td>5.0226</td>
<td>is</td>
</tr>
<tr>
<td>5.9938</td>
<td>you</td>
<td>5.0097</td>
<td>i</td>
</tr>
<tr>
<td>5.9338</td>
<td>video</td>
<td>4.9966</td>
<td>just</td>
</tr>
<tr>
<td>5.9338</td>
<td>com</td>
<td>4.9834</td>
<td>your</td>
</tr>
<tr>
<td>5.8863</td>
<td>the</td>
<td>4.9834</td>
<td>t</td>
</tr>
<tr>
<td>5.8863</td>
<td>check</td>
<td>4.9564</td>
<td>so</td>
</tr>
<tr>
<td>5.8644</td>
<td>YouTube</td>
<td>4.9286</td>
<td>will</td>
</tr>
<tr>
<td>5.6631</td>
<td>me</td>
<td>4.9144</td>
<td>You</td>
</tr>
<tr>
<td>5.6495</td>
<td>it</td>
<td>4.9144</td>
<td>Please</td>
</tr>
<tr>
<td>5.6495</td>
<td>channel</td>
<td>4.9000</td>
<td>up</td>
</tr>
<tr>
<td>5.5179</td>
<td>of</td>
<td>4.8706</td>
<td>music</td>
</tr>
<tr>
<td>5.4942</td>
<td>subscribe</td>
<td>4.8706</td>
<td>from</td>
</tr>
<tr>
<td>5.4942</td>
<td>for</td>
<td>4.8706</td>
<td>Hey</td>
</tr>
<tr>
<td>5.3573</td>
<td>www</td>
<td>4.8556</td>
<td>make</td>
</tr>
<tr>
<td>5.3573</td>
<td>if</td>
<td>4.8556</td>
<td>be</td>
</tr>
<tr>
<td>5.3480</td>
<td>please</td>
<td>4.8556</td>
<td>at</td>
</tr>
<tr>
<td>5.3006</td>
<td>like</td>
<td>4.8248</td>
<td>s</td>
</tr>
</tbody>
</table>