Table of Contents

List of Figures

List of Tables

Summary

The open-source compliance team at Blackberry QNX is responsible for determining open-source licenses since it is used to comply with the legal terms of theses licenses before a product can be released into the public. This is done through a software pipeline that gets the source code information from a package found in a Jenkins server where comment blocks are then extracted and then predicted upon by a machine learning model called fastText to determine the specific license a comment block is referring to. These comment blocks along with any license information is then inserted into a MongoDB database. The problem that is being faced is that there are many false positives found in the database where certain comment blocks have absolutely no information about licensing, which are considered invalid licenses.

This report aims to provide the decision process and analysis of a machine learning experiment to determine whether machine learning can be used to accurately identify whether a comment block found in source code contains open-source license information. This is a binary classification natural language processing task since the potential prediction of a model is that given a comment block (which is text), determine if it contains license information or not. The two approaches taken were unsupervised and supervised learning.

Unsupervised learning was initially used since the dataset that will be used to train models may contain misclassifications because the data in the database is not actively monitored. Since this type of learning does not use labels, it can bypass this problem. One approach that was used was a K-means clustering algorithm that would attempt to cluster the comments into groups. This approach did not cluster groups into the groups wanted since the ideal number of groups found using the elbow method was about 54. Another approach that was used was an autoencoder that would attempt to replicate comments that contained license information. From testing, it was clear that the model was not able to generalize on a comment containing a license compared to ones that didn't. This can be due to the various open-source licenses that can be found in software.

Supervised learning was then used where the data was initially cleaned up as much as possible and then preprocessed. Three different models that may perform well on this NLP task was used. Hyperparameter tuning was conducted either through Cross Validation or automatically if the model allowed it. Precision and recall along with a fast prediction time were used to determine how well a model performed and after testing on a training and test dataset, it was determining that the fastText model performed quite well and satisfied the constraints and criteria placed. Future training and testing should be used to further validate the obtained results and placing the model in a live production environment is a way to do so.

1.0 Introduction

Artificial intelligence is a form of intelligence that is shown by computers that is initially programmed by humans. This differs from the intelligence humans and animals display which is called natural intelligence where the individual has a natural interest to its environment and actively pursues it. Artificial intelligence is an active field of research in computer science and is an umbrella term that covers a wide spectrum of topics. Subsets of this field that has seen a boom in interest due to recent advancements is machine learning and natural language processing.

Machine learning (ML) is an attempt to get machines to improve on a specific objective which they are assigned to do overtime through data. More formally, “Machine learning is the study of computer algorithms that improve automatically through experience” (Wikipedia, n.d.). A combination of such algorithms creates something called a machine learning model. Underneath this model is a mathematical model that is developed through the data that is being passed into it. This data helps the underlying model update parameters that are most suited for the dataset that it is seeing. The machine learning model can then make decisions or predictions on future data samples without the explicit programming found in traditional programs. Machine learning has been used in many different applications such as in sentiment analysis, email filtering and even in autonomous vehicles.

Natural language processing (NLP) deals with the interaction between machines and natural language. It aims to create programs that can analyze and process large amounts of natural language data (Wikipedia, n.d.). In the past, NLP was largely rule-based where explicit rules were defined for things such as determining grammar. The recent boom that occurred for the NLP field is largely due to the ability for machine learning algorithms to help perform natural language processing tasks. This is done through the accumulation of a large “corpus” which is just a set of documents such as tweets from Twitter or paragraphs from a story. These datasets help these ML models understand the sophisticated language underneath and can perform tasks significantly better than rule-based algorithms. NLP has been used in language translation, speech recognition and natural language generation.

There are 2 broad categories within machine learning that will be investigated in this report. These include:

- Supervised Learning
- Unsupervised Learning

1.1 Supervised Learning

Supervised learning is a broad category within machine learning that deals with mapping inputs to outputs to create input-output pairings. The inputs called features and outputs called labels used to develop a machine learning algorithm is called a training dataset. A supervised learning algorithm would analyze both the features and labels of the dataset and would learn to produce an appropriate mapping function that would map the given inputs to the corresponding output. Ideally, the supervised learning model can then infer the output of given input data that it has not seen before. This requires the model to generalize on the data that it is seeing during training to correctly determine unseen data. An example of a supervised learning model's output can be seen in Figure 1.

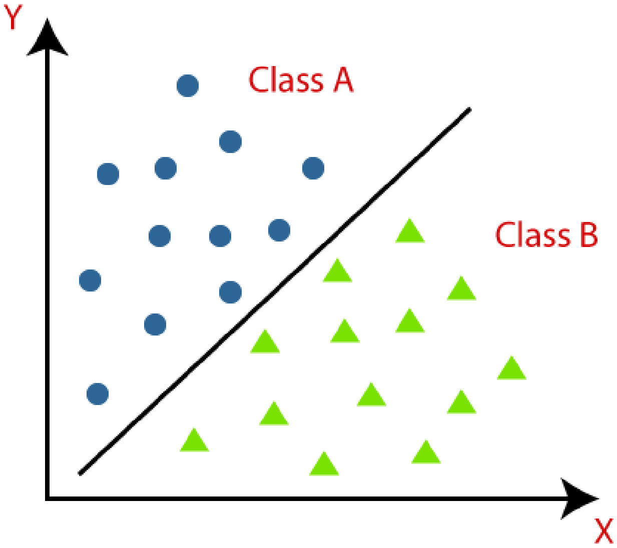

Figure 1: A visualization of a supervised learning algorithm's output (Javatpoint, n.d.)

Supervised learning can be broken down into regression and classification. Problems that involve regression attempt to predict a continuous-valued output such as the housing prices in Ottawa or the value of a stock the next trading day. Classification problems try to predict a discrete number of values such as whether a picture is a picture of a dog or of a cat which is often called binary classification (2 distinct outputs). Another type of classification is multi-label classification where there are more than 2 outputs such as predicting the type of cancer an individual has.

A common supervised learning model is linear regression. Linear regression attempts to find the line of best fit on a set of datapoints. It is used for regression since it tries to model a target value based on specific input values which are independent variables. Once a linear function has been found, new predicted outputs can then be obtained given new inputs.

Another common supervised learning model is called logistic regression. It uses regression to predict a continues probability, ranging from 0 to 1 of a data sample belonging to a specific category or class. Based on that probability, the sample is then classified which makes this model ultimately a classification algorithm. Internally, the model attempts to find coefficients for the features found in the training set which are then multiplied by values of the inputs and then passed into a sigmoid function to obtain the probability. It is known for its use in binary classification but can be used in multi-variable classification using the multinomial logistic regression model.

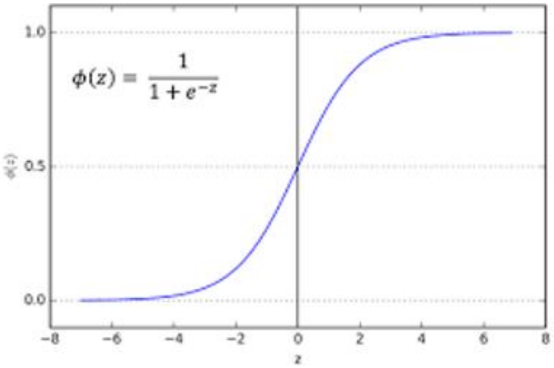

Figure 2: The sigmoid function that is used within the logistic regression algorithm (Quora, n.d.)

The Naive Bayes classifier is a supervised machine learning algorithm that leverages Bayes' Theorem to make predictions and classifications. Bayes' Theorem is the basis of a branch of statistics called Bayesian Statistics where prior knowledge is considered before predicting new probabilities. The equation that defines Bayes' Theorem is shown in Eqn. (1).

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{1}$$

where P(A|B) is the probability that event A will happen given that event B already happened, P(B|A) is the probability that event B will happen given that event A already happened, P(A) is the probability that event A will happen and P(B) is the probability that event B will happen. This classifier assumes that data points are all independent events which is not always the case. The Naive Bayes classifier is often used in text classification such as in email spam filters where it is used to determine whether certain words indicate that an email is spam or not.

fastText is an open-source, lightweight library that is often used for text classification which was created by Facebook's AI Research lab. It transforms text into continuous vectors that can be used for any NLP related task. The model contains complex logic that allows it to train on a very large dataset very quickly and make fast predictions, hence the name. The fastText model can also be used to create word embeddings as well and posses the ability to automatically optimize its hyperparameters (parameters set before training) so that it is fine tuned to the dataset that it is trained on.

1.2 Unsupervised Learning

Unsupervised learning is how we find patterns and structure in data. Clustering is the most well-known unsupervised learning technique where the program learns the inherent structure in unlabeled data by grouping them in clusters. This differs from supervised learning techniques where each data record possesses a label, inherently teaching the machine learning model what is right from wrong. That luxury does not exist when data is unlabeled. Labelling data is a very tedious process that requires a lot of time and effort. In addition, data may be "dirty" where the labels placed on the data is incorrect and this may be due to human error or changes in the task requirement which makes the label invalid. Unsupervised learning is a way to obtain useful information when one does not know what to obtain. Examples of uses of unsupervised learning is social networks clustering topics in their news feed, consumer sites clustering users for recommendations and search engines to group similar objects in one cluster.

K-means clustering is a clustering algorithm that falls under unsupervised learning that intends to separate data so that similar data is in the same group. The 'k' refers to the number of groups or "clusters" that are expected to be found in the dataset. The 'means' refers to the average distance of the data to each cluster center, which is also known as a centroid, which is maximized. It is an iterative approach that follows the following steps that can be seen in Figure 3:

1. Place k random centroids for the initial clusters.
2. Assign data samples to the nearest centroid.
3. Update the centroids based on the newly assigned data samples.
4. Repeat steps 2 and 3 until convergence which is when the points do not move between clusters and the centroids stabilize.

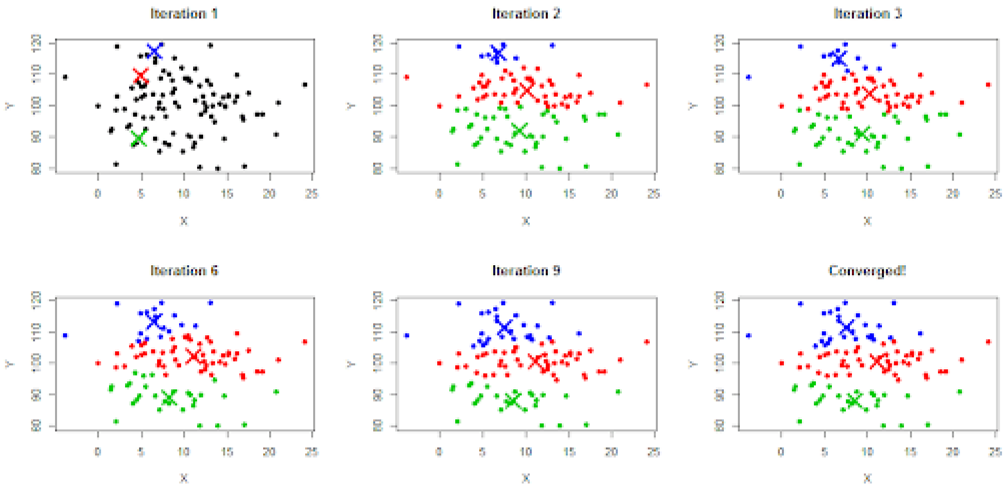

Figure 3: K-means clustering over consecutive iterations until convergence (Stack Overflow, n.d.)

Good clustering from the K-means clustering algorithm results in tight clusters, meaning that the samples of each cluster are close/bunched together. Inertia is a measurement that describes how spread out the clusters are and is simply the distance from each sample to the centroid of its cluster. Ultimately, the goal is to have low inertia and the least number of clusters. Since k is chosen prior to running the algorithm, there are different ways to determine the optimal k for a specific dataset. One way is through the elbow graph where the number of clusters is graphed against the inertia for that cluster. The 'elbow' of the graph is considered to be the most optimal k as seen in Figure 4.

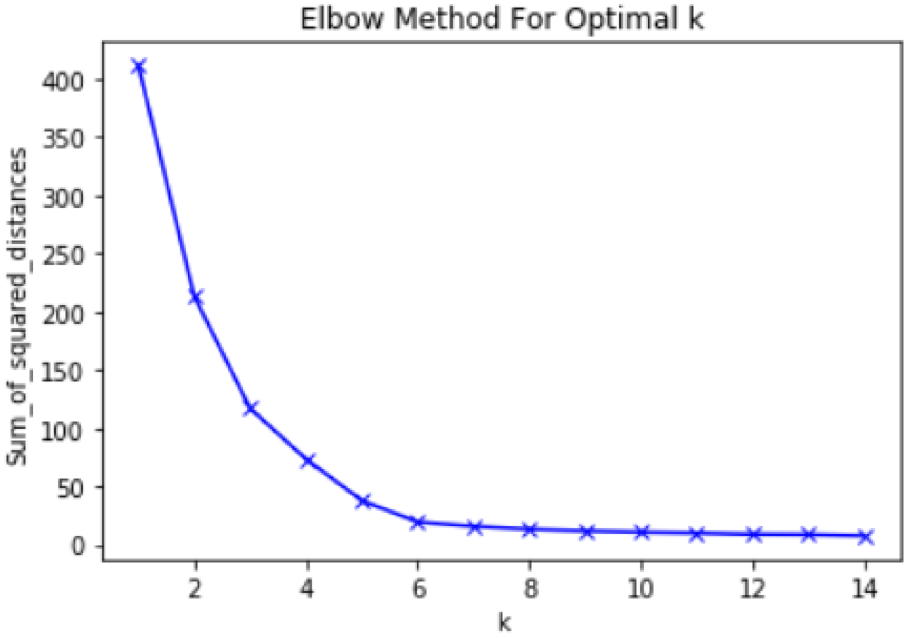

Figure 4: The number of clusters graphed against the sum of squared distances (inertia) (Medium, n.d.)

Anomaly detection is the process of determining whether a datapoint is an unexpected record in the dataset which differs greatly from the rest of the dataset (the norm). Anomaly detection mainly falls under unsupervised learning since anomaly detection is often applied to unlabelled data. Anomaly detection is based on the assumption that anomalies occur rarely in data and that their features (the data that define the datapoint) is different from the considered norm of the dataset. An autoencoder is an unsupervised learning technique that can be used for anomaly detection. Essentially, the objective of this model is to recreate the inputs passed into it, creating a representation of the input through the inherent structure of the input. For example, an autoencoder can attempt to recreate an image passed into it as its output. This model possesses a measurement called the reconstruction error which measures how well it has recreated the input data. This can be used for anomaly detection where the "normal" data will possess a low reconstruction error but anomalies (which differ from the norm) will possess a high reconstruction error since the model was unable to recreate it based of the norms that it has learned from the rest of the normal data. This model is specifically a form of something called a neural network that won't be described further in this report.

1.3 Data Preprocessing in Natural Language Processing

Before data can be passed into a machine learning model, it should be preprocessed. Preprocessing data allows one to possess a high quality of data which would allow the model to learn much more efficiently and tends to improve its overall performance. Generally, preprocessing can include removing null/invalid data points, normalization which makes data points possess the same scale so that each feature possesses equal importance, and feature selection where the most important features are selected for the model to train on. In NLP, preprocessing (also called cleaning) is a very crucial step since not only all the text in a data record is considered useful but that machine learning models can only understand numbers, not letters or words.

There are many common tasks in cleaning data for an NLP tasks. Common tasks include noise removal where text is stripped of its formatting such as HTML tags after web scrapping a website. Tokenization is another task done during preprocessing where text is broken down into individual words/letters. This is used to create the vocabulary of the dataset which is used for further preprocessing to help determine the features of the dataset. Another common task is normalization which consists of many other smaller tasks. These include lowercasing, stop word removal (removing words such as ‘the’, ‘a’ etc.), stemming/lemmatization and other related tasks.

Stemming and lemmatization are ways to convert a word to its root form. For example, the words ‘playing’, ‘plays’, and ‘played’ can all be converted to the root form ‘play’. These techniques are done to decrease the size of the vocabulary that will be seen by the machine learning model and is used in many different applications such as web search where results for the word ‘play’ should also bring up results for ‘playing’. The main difference between stemming and lemmatization is the way they go about getting the root form of a word. Stemming is a crude way of getting the root forms of words since it just removes the suffixes which may result in words that are not actually words. For example, the words ‘trouble’, ‘troubling’, and ‘troubled’ may create the root form ‘troubl’ which is not a word. On the other hand, lemmatization uses an external corpus and parts-of-speech identifier to obtain the root form of a word which will be an actual word. The downside to this is that lemmatization is much slower than stemming since lemmatization requires additional resources. Either technique can be used depending on the NLP task at hand and the time and processing requirements.

Word embedding is a process within NLP where words or phrases from a vocabulary are mapped to a set of vectors which consist of numbers. Essentially, it is the representation of a word or phrase as a numeric vector, enabling us to compare and contrast how words are used and identify words that occur in similar contexts. The idea behind word embeddings is a theory known as distributional hypothesis which states that words that co-occur in the same contexts tend to have similar meanings. With word embeddings, we map words that exists with the same context to similar places in our vector space. This can be seen through the cosine distance between two vectors where words with similar contexts have a small cosine distance. The literal values of a word’s embedding usually will not have any actual meaning but encoded in these vectors is latent information about how the words are used.

Bag-of-words is a language model that only cares about word count. It initially creates a feature dictionary which will consist of every unique token (word) within the training corpus and a corresponding index. The feature vector (word embedding) is created by going through each document in the corpus and assigning the count for each word at the specific index described by the feature dictionary. This method allows the conversion of all the documents into vectors which can be passed into a machine learning model. The language model is great when it comes to making predictions concerning topic or sentiment of a text but it is not great at text prediction since the word order of the documents is lost and the probability of the following word is always just the most frequently used words.

Tf-idf (term frequency inverse document frequency) is another method of creating vectors from text. Tf-idf is a numerical statistic used to indicate how important a word is to each document in a collection of documents (corpus). When tf-idf is applied to a corpus, each word gets a tf-idf score for each document. The higher the tf-idf score is, the more important the term is to the corresponding document. Tf-idf relies on 2 metrics. Term frequency which is the same as the bag-of-words word count where it is a tally of the word in the document. Inverse document frequency is how often a word appears in an overall corpus. The intuition of the method is that words that appear more frequently in the corpus gives less insight into the topic or meaning of the document which should be deprioritized. Tf-idf is used in ranking results in a search engine, text summarization and building smarter chatbots.

1.4 Objective

The objective is to determine whether machine learning can be used to detect if a given comment block found in a source repository contains an open-source license or not. This is a natural language processing task since comment blocks are a type of text and both supervised and unsupervised learning will be used to attempt this task.

2.0 Problem Definition

Open-source software is software that consists of source code that people can inspect, modify and enhance (opensource.com, n.d.). This differs from other forms of software called proprietary or closed-source software where an individual, team or organization that created the software possesses complete/exclusive control over it. In both open-source and closed-source software, the users of the software must comply with the legal terms found in the licenses of the software.

Open-source software licenses can differ greatly between each other. Many of the most popular licenses such as the MIT license gives permission to the user to distribute, modify and use the software for commercial and private use. The caveat is that the user most provide a copy of the license and a copyright notice when they ship their software. In addition, the user is not provided any form of warranty and cannot hold the creator(s) liable in case anything happens with the software. Other open-source licenses like the GNU General Public License v3.0 are called copyleft licenses since the user must also keep the same license when the software is redistributed, with or without changes. This means that the user must provide the same freedom to their users that they received from using this software such as distribution, modification and commercial use. The GNU General Public License v3.0 also contains conditions to disclose source which means to also provide the source code for the software that is being distributed and state the changes that were made to the original source files.

Failure to comply with open-source software called a ‘breach of a license’ can open the user or organization to lawsuits from the original creators of the software. An example of this was in 2009, when the Software Freedom Law Center (SFLC) made a copyright lawsuit against several major consumer electronic companies such as Best Buy, Samsung, Bosch and JVC for violating the GNU General Public License (GPL) (Network World, Inc, n.d.). These companies failed to release the code and their respective modifications to the users/customers of the product containing GPL-licensed code. This lawsuit had major financial consequences such as Best Buy losing 1.4 billion dollars from their market cap which otherwise could have been avoided if they simply had complied with the GPL license. This goes to show how important open-source software compliance is and the serious repercussions it can have on a company for breaching such licenses. One might believe that simply not using open-source software can relieve companies of the problems mentioned but this is easier said than done. Open-source software is a vital aspect of software that is used by many major technologies such as the Internet, Linux operating system and Apache Web servers. Keeping software open source allows all individuals to contribute to the software which keeps the code up-to-date and helps progress software forward than if it were proprietary. Open-source licensing creates innovation through collaboration (BigCommerce, n.d.).

2.1 Open Source at Blackberry QNX

The open-source compliance team at Blackberry QNX is responsible for creating compliance artifacts for QNX projects before being released to the public. The QNX operating system is a Unix distribution that consists of millions of lines of code that includes a lot of open-source software that requires compliance. Compliance artifacts consists of the required compliance necessary to comply with all the open-source licenses found in a project. At QNX, there is the NOTICE file and the COMPLIANCE file. In the NOTICE file of a project, a summary of the open-source compliance information for the software is provided. In the COMPLIANCE file, an enumeration of all the open source licenses found in the product is provided. Since it is very difficult to keep track of all the open-source software used in a product during its development, the open-source compliance team has created a machine learning pipeline where given the specific QNX product, determine all the open-source software licenses found in it

2.2 Machine Learning Pipeline

The pipeline consists of the several different steps. Firstly, a product manager will login to a React.js front-facing web application where they will be able to select multiple artifacts that were created by their project's Jenkins build. This data is then sent through a Node.js server to a Jenkins job that begins the actual process of determining the compliance information of the selected artifacts. Firstly, the packages need to be downloaded to the Jenkins job's workspace for future steps. Once downloading is complete, the packages then need to be extracted. The extraction process decompresses and extracts all files to their original form. Packages may contain many compressed files within themselves such as a zip file inside a zip file inside a tar.gz file for example which becomes a recursive step. Then the analysis process occurs where ELF files located within the artifacts are extracted to obtain a source-to-binary mapping of the package. This is where all the source code/files that were used to create the package is found. Once the source files are found, the pipeline goes through each source file to extracts all the comment blocks. These comment blocks that may possess open-source licenses are then inserted into a MongoDB database. After determining all the comment blocks found in all the source code, a fastText model is then used to predict what license that specific comment block is talking about which can then be used to determine compliance information.

2.3 Invalids Found in MongoDB Database

When a comment block is found in a source file, it is inserted into a MongoDB database to produce traceability of where the specific comment block was found so that if the comment block is encountered again, there is no need to use the fastText model to repredict the comment block. It also creates a dataset that can be used to further train the fastText model to make it more accurate. In addition, even though the fastText model is quite accurate, it still makes mistakes which can be costly if it goes unnoticed. That is why after all the predictions by the ML model is complete, manual verifications are still conducted on all comment blocks to determine whether the correct license was predicted or if the comment block wasn't an indication of a license in the first place which is considered an 'invalid' license.

Invalid licenses currently plague the database. It was found that over 20% of the database contains invalid licenses which provides no use whatsoever since obtaining compliance information for valid licenses are all that matters when creating the compliance artifacts for QNX products. Also, invalid licenses take precious time away from compliance officers who must manually verify hundreds or even thousands of licenses found in a single product. In the QNX SDP 7.1 release, there were over 6000 open-source licenses that needed to be verified, where over 20% of them were invalid licenses. As well, invalids cannot be used to train the ML models and can creep into datasets used for training, validation and testing which makes the data dirty and can ruin the effectiveness of the model.

Invalid licenses can come in different forms. There are comment blocks that have no indication of a license such as a comment block talking about a particular method or the text contains words such as 'restriction' or 'license' that could have been an indicator of a license but turned out not to be one. Other comment blocks could have contained valid license information but was altered by a developer/programmer enough so that in does not follow the SPDX rules and guidelines (SPDX is standard format used to talk about license and copyright information in software), thus making the license invalid.

The problem that needs to be addressed is how to significantly reduce the number of invalid licenses creeping into the MongoDB database. Currently, a deterministic method is used to determine if a comment block is a license by using a regex that contains specific words that could be indicators of an open-source license which still brings in a large portion of invalid licenses. A new solution is required to tackle this problem to reduce invalids which has the potential to reduce 20% of the costs from manual verifications, provide cleaner data for the ML models to train upon and provide even more accurate results

2.4 Constraints and Criteria

The new solution must make significant improvements over the current regex method to prove to be viable. Firstly, there needs to be significant reduction in the number of false positives and false negatives when predicting comment blocks. To explain false positives and false negatives, a 1 will indicate that a comment block contains a license while a 0 indicates it does not. A false positive is when something is predicted to be a 1 (a valid license) but, it is not (supposed to be 0). A false negative is when a method predicts that a comment block is 0 but it is really a 1. False negatives need to be reduced because missing a license can have consequences and reducing false positives gets rid of invalids that otherwise would be cluttering the database. Good metrics that can be used measure a false positive and false negative rate are precision and recall. Precision is the number of correct labels among labels predicted and recall is the number of labels that successfully were predicted among all real labels. Possessing a high precision means a low false positive rate while possessing high recall means a low false negative rate. Figure 5 showcases the differences of precision and recall in a visual format.

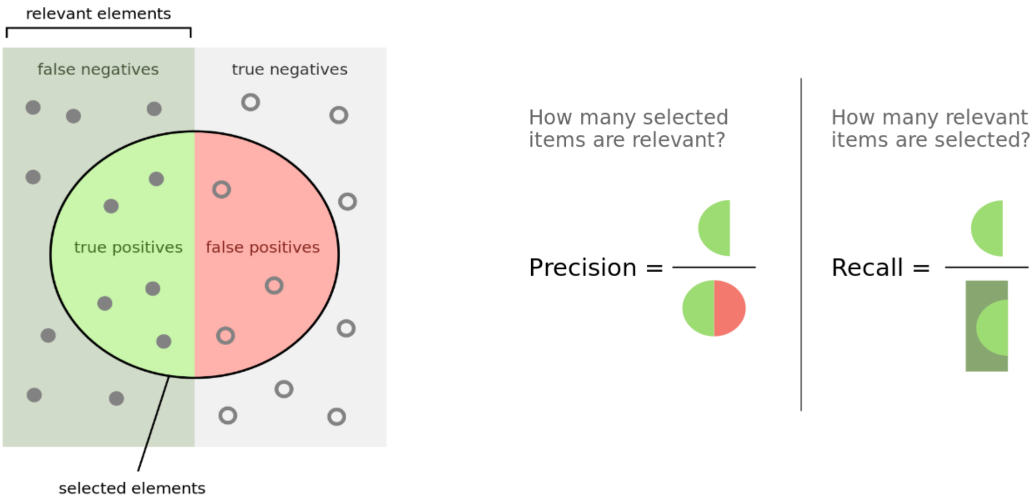

Figure 5: A visual representation of precision and recall metrics (Wikipedia, n.d.)

The solution must also be able to predict comment blocks very quickly since there could be thousands of licenses in a QNX project and even more when it comes to the individual comment blocks. A slow solution would mean less time for the manual verifications since determining compliance information for a product is done close to the release date.

Therefore, the proposed solution must be able to predict comment blocks to determine whether it is a license with a high precision, high recall and a fast prediction time.

3.0 Technical Progress

3.1 Machine Learning as a Solution

Since a deterministic method like the current solution of using regex is simply not viable, a probabilistic approach will be taken like the problem of predicting a specific license where machine learning was used. The machine learning experiment conducted is to determine whether machine learning is a feasible solution to determine if a comment block contains a license or not. Since this is an NLP binary classification problem, the intuitive solution is to specifically use supervised learning. The problem is that since the MongoDB database contains dirty data that has accumulated over the years, some of the labels for the comment blocks are misclassified, thus can confuse a supervised ML model and it would take significant time to clean this data up since there are thousands of unique records and a limited amount of time to conduct the experiment. Unsupervised learning can help avoid this problem since it does not require any labels to begin with.

Before any machine learning can be done, the data must be obtained. Firstly, the MongoDB database contains over 150 thousand records pertaining to comment blocks and their respective validity but only about 23 thousand records are unique. This makes sense since comment blocks that contain licenses can be the same across multiple source files. In addition, a dataset pertaining to comment blocks was also found online where the dataset was separated between different comment block categories such as explanation of methods and licenses. Also, data was created programmatically, a process called synthetic data generation, where comment blocks were randomly generated which contained words that a license may have but these comment blocks were always labelled as invalid. These words come from the Ninka dataset. Therefore, the combined dataset will contain about 40 thousand unique records that is balanced where there is an even split between comment blocks that contain a license and ones that do not.

Once the dataset was created, the data was preprocessed by using common techniques such as lowercasing, noise removal etc. Stemming was used over lemmatization as per the constraint of possessing a fast method to complete this task since lemmatization takes significantly longer than stemming. In terms of creating vectors from comment blocks, the traditional tf-idf method was chosen over the bag-of-words model because the bag-of-words is essentially what is currently being done with the regex method. In addition, the tf-idf method tends to be used for industry-specific text and comment blocks for open-source licenses falls in that category. There are other methods for creating word embeddings like word2vec and document2vec but since the dataset is very small compared to datasets with millions of records, the advantages these word embeddings would have would be lost.

3.2 Unsupervised Learning as a Solution

Initially, the K-means clustering model was used to cluster the data. Ideally, the k value wanted would be 2 and specifically have the clusters of valid and invalid licenses. Since, this clustering algorithm is unaware of these details, it clusters purely off where the data lies in the vector space. The elbow method was used to determine the most optimal k. The k value and its respective inertia were graphed against each other up to a k value of 100. Figure 6 displays the elbow graph obtained. The elbow for this graph is about k=54 which can be made sense of since there are about 54 unique open-source licenses found in the dataset. The K-means clustering model cannot be used as a viable solution for the given problem since it cannot be used to cluster the dataset into 2 separate clusters optimally.

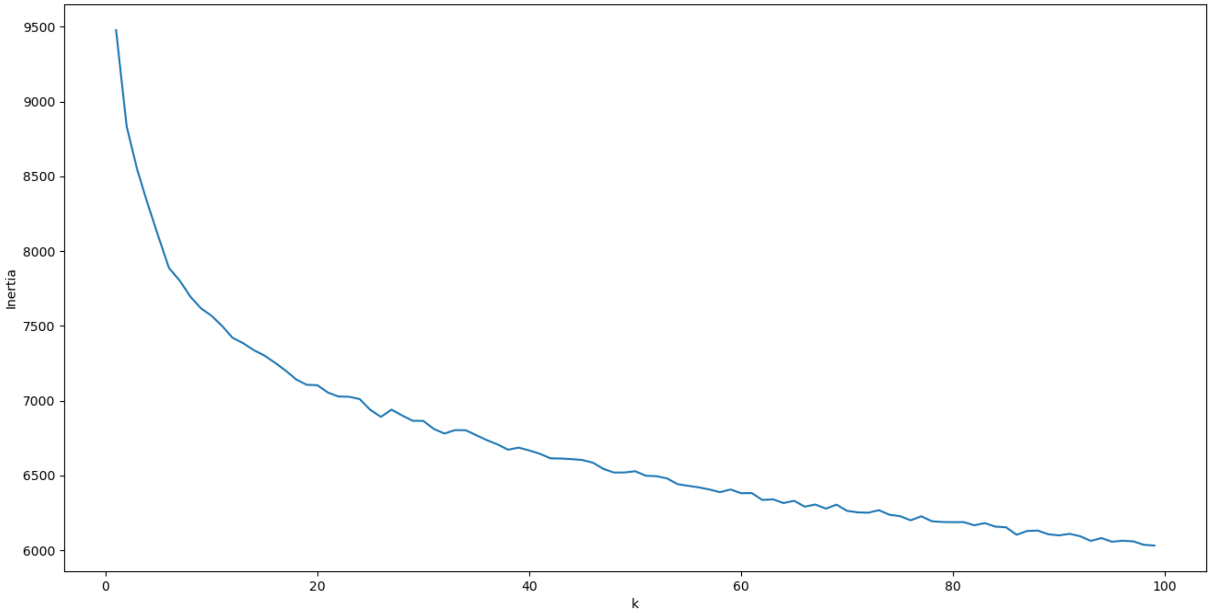

Figure 6: Elbow graph obtained from K-means clustering algorithm

The next unsupervised ML model used was an autoencoder. The idea here was to see if the model can detect an anomaly in the dataset which in this case was an invalid comment block. The dataset needed to be altered since it is currently was a balanced dataset, but this model requires the anomalies to be very rare. Multiple attempts were made to see if the autoencoder was able to pick out the invalids from the valid licenses, but the metrics obtained were subpar at best. Since the labels are not used for unsupervised learning models, specific curated datasets (different from the one used for training) were created that contained either only valid licenses or only invalid licenses. The model performed subpar on theses datasets obtaining a greatest accuracy, recall and precision of around 85% after hyperparameter tuning.

Unsupervised learning at this stage of the experiment seemed to show poor results and thus an attempt to use different supervised learning models to complete this task was conducted.

3.3 Supervised Learning as a Solution

Since unsupervised learning provided poor results, supervised learning was then looked at to see if there were any viable solutions. Since this approach was initially avoided due to ‘dirty’ data found in the MongoDB database, the data needed to be cleaned up. After some data exploration, certain blocks of records that were known to contain incorrect labels were discarded. Other records that were incorrect were notified to the open-source compliance officer to correct in the MongoDB database. Even though there may be some dirty data found in the dataset, the addition of the new dataset found online, and data created from synthetic data generation helps offset this problem. The continuation of the ML experiment needs to have this issue considered when comparing results.

Three different ML models were considered initially for the supervised learning portion of the experiment. They were the Naïve Bayes classifier, a logistic regression model and a fastText model. The fastText model was chosen since the model was already being used to classify specific licenses for comment blocks. The two traditional machine learning models were chosen since they are known to perform well on NLP tasks and were used to compare against the fastText model.

Since the dataset that is being used is relatively small, 3-fold cross-validation was used to train and tune hyperparameters for the logistic regression model. Cross validation is a technique where the training set can be used for both training and validation. The dataset is first split into k groups (folds), then one group is taken as the validation dataset and the remaining datasets become training. Once the model has trained, the model is then tested on the validation dataset and metrics are recorded. After this, another group becomes the validation dataset and the remaining become part of the training set. This is done k times until every group has been the validation dataset. The metrics recorded for each evaluation are then averaged out to see how well the model has performed. A method called Random Search Cross Validation was used to determine the best hyperparameters (parameters set before model training) to determine the best general hyperparameters to tune on. Then another technique called Grid Search Cross Validation was used to find the best values for these hyperparameters. The Naïve Bayes classifier did not undergo hyperparameter tuning since it did not have hyperparameters to tune upon (excluding smoothing parameter) but the fastText model had automatic hyperparameter tuning, so Random Search and Grid Search was not necessary to use. Multiple metrics were recorded including training accuracy, validation/test accuracy (data the model has not seen), precision, recall and f1-score. A confusion matrix was also created for each model to visually see the false positives and false negatives from the testing of each model on the test dataset. Table 1, Table 2, Table 3 and Table 4 record the classification various statistics obtained for each model. Figure 7, Figure 8 and Figure 9 showcases the confusion matrices.

Table 1: Metrics obtained from the fastText supervised learning model on the test dataset

	Precision	Recall	F1-Score	Support
Invalid Comment Blocks (0)	98%	96%	97%	3770
Valid Comment Blocks (1)	97%	98%	97%	4208
Accuracy	-	-	97%	7978
Macro Avg	97%	97%	97%	7978
Weighted Avg	97%	97%	97%	7978

Table 2: Metrics obtained from the Logistic Regression supervised learning model on the test dataset

	Precision	Recall	F1-Score	Support
Invalid Comment Blocks (0)	92%	95%	94%	3770
Valid Comment Blocks (1)	96%	93%	94%	4208
Accuracy	-	-	94%	7978
Macro Avg	94%	94%	94%	7978
Weighted Avg	94%	94%	94%	7978

Table 3: Metrics obtained from the Naïve Bayes supervised learning model on the test dataset

	Precision	Recall	F1-Score	Support
Invalid Comment Blocks (0)	91%	92%	91%	3770
Valid Comment Blocks (1)	92%	92%	92%	4208
Accuracy	-	-	92%	7978
Macro Avg	92%	92%	92%	7978
Weighted Avg	92%	92%	92%	7978

Table 4: Training and test accuracy of 3 supervised ML models

	fastText	LR	NB
Training Accuracy	96.8%	95.2%	92.7%
Test Accuracy	97.1%	93.8%	91.7%

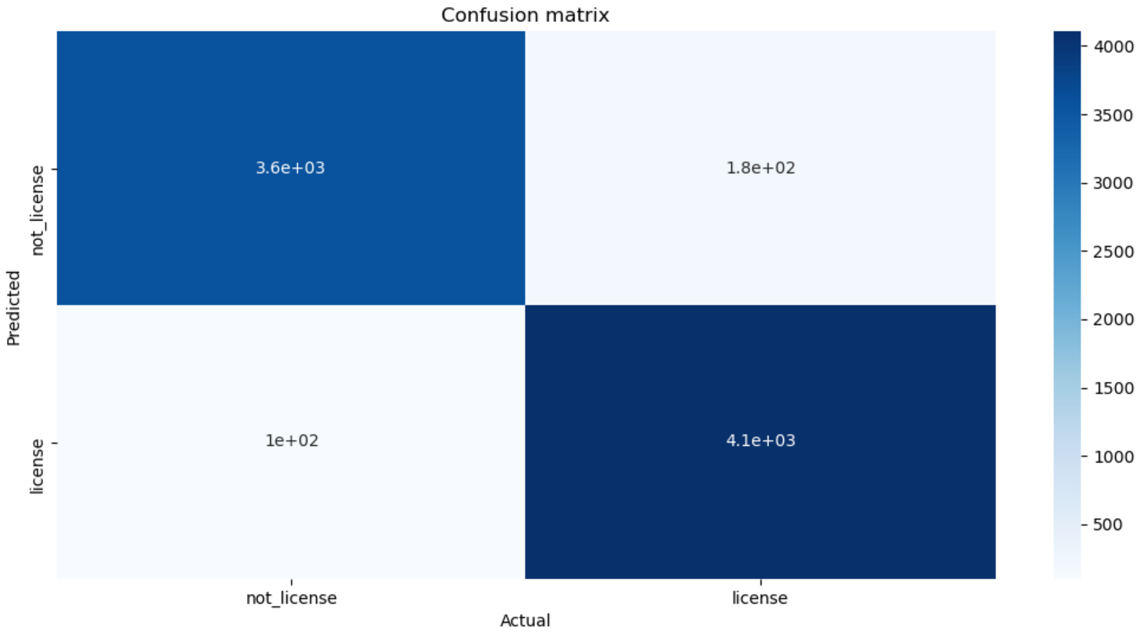

Figure 7: Confusion matrix of fastText model

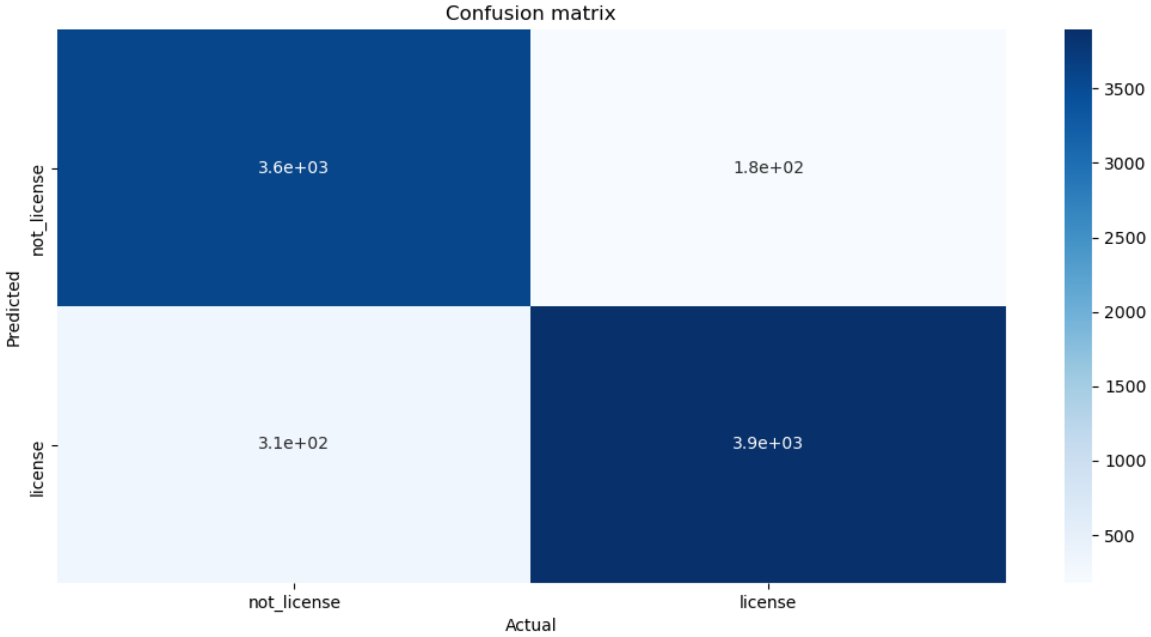

Figure 8: Confusion matrix of Logistic Regression model

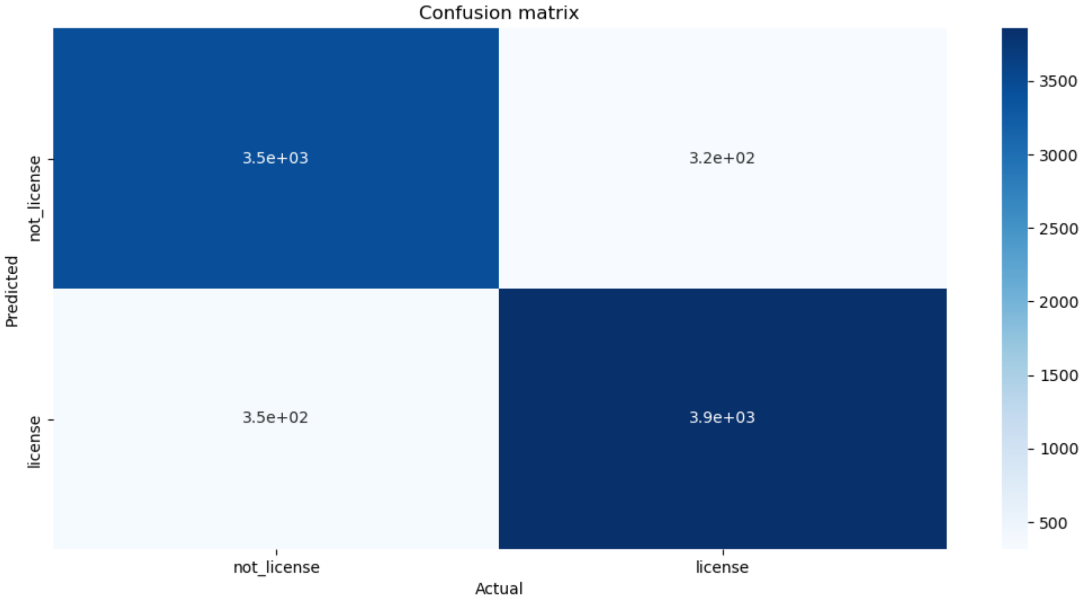

Figure 9: Confusion matrix of Naïve Bayes model

As seen by the support column from the tables, the test dataset was the same across all three models, thus the metrics can be compared directly against each other. From the data, the fastText model has outperformed the Logistic Regression and Naïve Bayes model by a considerable margin in all metrics. The confusion matrices show that the number of false positives is much smaller with the fastText than that of the two traditional models (100 misclassifications compared to 300+ misclassifications). The fastText model is also very fast in training and predictions, which satisfies the speed constraint.

3.4 Remaining Challenges

It is clear from the results collected that the fastText model is a viable solution for the problem that is being faced since it possesses a high precision, recall and f1-score. Considerations still need to be made about the metrics obtained because there is still a very high likelihood that the test dataset contains dirty data and the model may have learned to incorrectly classify certain comment blocks but there is a workaround to combat this.

The fastText model provides both a label and a confidence score whenever it tries to predict a comment block. This confidence score is the probability that the label that was predicted was correct. In production, there will be a specific threshold that the model's confidence score for a prediction must surpass if the model predicted 0 (an invalid license). All predictions that are 1 (valid licenses) will be considered for manual verifications regardless of confidence score but the threshold will hold for invalid license predictions. This will help minimize and eventually eliminate false negatives created by the model and create certainty that comment blocks that are discarded do not contain any indication of a license.

4.0 Conclusions

4.1 Conclusions

The problem that was faced was determining if machine learning can be used to predict whether a comment block contains a license to reduce and eventually eliminate the false positives found in the MongoDB database. Machine learning was used since it was a probabilistic method of tackling this problem as opposed to the deterministic method of using regex to complete this task.

Unsupervised learning was initially used to attempt this problem because there was a considerable amount of misclassified data in the database. Both the K-Means Clustering algorithm and autoencoder model produced poor results, thus supervised learning was then attempted. The data in the database was then cleaned up to the best that could have been done with the timeframe this experiment was conducted in. Additional data found either online or synthetically generated was used to offset some of the misclassifications in the dataset since there would be less misclassifications compared to correct labels. Statistics were then collected on three supervised machine learning models training and testing on the same datasets. Through the comparison of statistics, it was found that the fastText model clearly outperformed the other 2 models used in the experiment. This model satisfied the constraints and criteria that were set since it possessed a high precision and recall when tested on the test dataset and the predictions that it made on a comment block was very quick, thus a fast prediction time.

4.2 Next Steps / Recommendations

The remaining steps is to see how well this model behaves in production in terms of the number of the number of false positives that enter the database with the addition of the model to the machine learning pipeline. As well, the continuation of data cleanup efforts will help make the datasets for training and testing more fruitful and erase any doubts regarding the obtained metrics of a model. Finally, data still needs to be collected so that the model has more data to train on and can improve it even further.

Some recommendations to continue the ML experiment is to test out a greater variety of machine learning models such as Random Forest and deep learning models. This can provide more evidence to support the fastText model or can create a realization that another model can outperform fastText. Different text preprocessing techniques, specifically word embeddings, should also be tried out to see if those changes can impact the performance of a model

References

BigCommerce. (n.d.). *What is open source, and why is it important?* Retrieved August 11, 2020, from https://www.bigcommerce.ca/ecommerce-answers/what-open-source-and-why-it-important/#:~:text=Open%20source%20is%20a%20type,work%20based%20on%20the%20original

Javatpoint. (n.d.). *Classification Algorithm in Machine Learning*. Retrieved August 9, 2020, from https://www.javatpoint.com/classification-algorithm-in-machine-learning

Medium. (n.d.). *Tutorial: How to determine the optimal number of clusters for k-means clustering*. Retrieved August 11, 2020, from https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f

Network World, Inc. (n.d.). *Multiple consumer electronics companies hit with GPL lawsuit*. Retrieved August 11, 2020, from https://www.networkworld.com/article/2240678/multiple-consumer-electronics-companies-hit-with-gpl-lawsuit.html

opensource.com. (n.d.). *What is open source?* Retrieved August 11, 2020, from https://opensource.com/resources/what-open-source

Quora. (n.d.). *Sigmoid Function*. Retrieved August 15, 2020, from https://www.quora.com/How-does-a-sigmoid-function-map-any-data-points-into-the-range-0-1

Stack Overflow. (n.d.). *K-Means Clustering Algorithm*. Retrieved August 13, 2020, from https://stackoverflow.com/questions/60312401/when-using-the-k-means-clustering-algorithm-is-it-possible-to-have-a-set-of-dat

Wikipedia. (n.d.). *Machine Learning*. Retrieved August 8, 2020, from https://en.wikipedia.org/wiki/Machine_learning

Wikipedia. (n.d.). *Natural Language Processing*. Retrieved August 8, 2020, from https://en.wikipedia.org/wiki/Natural_language_processing

Wikipedia. (n.d.). *Precision and recall*. Retrieved August 10, 2020, from https://en.wikipedia.org/wiki/Precision_and_recall