Introduction to Data Mining Project Report

Cpt\_S 315

Washington State University

Toxic Comment Classification

Created by

Ben Kaufmann

Daniel Semenko

April 2022

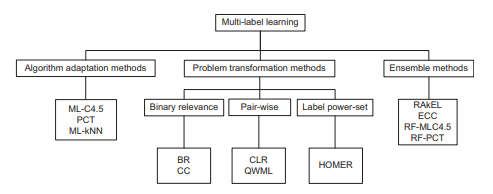
**1.0 Introduction**

DISCLAIMER FROM THE DATA SOURCE: The dataset contains text that may be considered profane, vulgar, or offensive.

**1.1 Motivations and Questions**

Discussing topics in any sort of online format can be challenging, for many different reasons. The threat of conversational toxicity is an issue that can lead people to no longer feel comfortable, out of the fear of aggravation and abuse online. This is a common problem in online video games and forums, where the ability to “hide behind your screen” allows malicious users to leave these comments, without risk of repercussion. The goal of this project is to use Machine Learning algorithms to train a model to identify toxic behavior in comments. This could be used to discourage those hostile users from posting potentially toxic or hurtful messages, allowing positive, civil discourse to occur.

The goal of this project is to implement a few different algorithms – specifically – to find the best approach to classifying toxic comments. Through the project, the authors will evaluate and analyze the effectiveness of the different algorithms. These different algorithms all stemmed from the type of problem presented in this project – the comments having multiple “labels”. Also known as Multi-Label Classification, it allows the authors to attempt many different methods (see Fig. 1), as Multi-Label Classification has many different approaches to solve it. These can include Binary Classification, Multi-Class Classification, Ensemble methods, in addition to classic algorithms being adjusted to the multi-label task – including kNN, Decision Trees, Neural Networks, and more (Madjarov, Kocev, Gjorgjevikj, & Džeroski, 2012). These were the starting point of many of our questions for this project. What is the best approach to classifying toxic behavior? Are there correlations between labels, and why do those correlations exist? How would the authors’ approaches change if just classifying as toxic vs non-toxic – single-label classification?



**Fig. 1.** Multi-label learning methods

The decision to choose this topic, and therefore this project, stems from both authors’ familiarity with toxic comments and hostile online environments. Both play video games and engage in online forums, where these types of hostile interactions are all too common. In some situations, these online platforms struggle to facilitate online conversations, leading to the removal of user comments and text chats. The authors hope that with the implementation of algorithms such as the one the authors explore, their usage can lead to improvements in online conversation with more productive and respectful discussion.

**1.2 The Task**

The authors ran into many challenges throughout the duration of the project, the first coming on just the first day! This project takes data from the *Jigsaw Toxic Comment Classification Challenge*, hosted by Kaggle, in which the data provided is a dataset of comments from Wikipedia’s talk page edits. The very first problem was the formatting of the provided csv files, as during the download, the delimiting (occurring on commas) didn’t work as intended. Therefore, the authors had to work with the format more to figure out how best to fix it. Another problem was the inclusion of apostrophes in cleaned comments, as it would conflict with stop words.

The main aspect of the project revolved around testing different multi-label classification methods against each other. To get there, there were a few steps before. First, the authors determined the definition of the problem, and how they would go about solving and evaluating them. The problem being the challenge found [here](https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) on Kaggle – build a model capable of detecting different types of toxicity, which predicts the probability of each type of toxicity for each comment, trained using a highly-imbalanced training data set. The authors could then evaluate the performances of the different methods by comparing the accuracies of each, and the CPU computation time. The next step was to import and clean up the data – the preprocessing step. Here, the authors make the data as nice and clean as possible to make the next steps easier. At this point, the data is prepared for the different multi-label classification methods. Once those methods have been successfully implemented and run, the authors can evaluate the effectiveness of the algorithms, by comparing their accuracies and compute times.

When it comes to multi-label classification, there are many possible approaches to solving them. These can include Problem Transformation Methods, Adapted Algorithms, Learning Paradigms, Multi-Label Stream classification, and Statistics and evaluation methods, among others (Wikipedia, 2022). Out of those, the author’s decided to use two approaches – problem transformation methods and adapted algorithms. Within problem transformation, the authors will use the Binary Relevance method, and the label powerset transformation. And the authors will test an adapted version of k-nearest neighbors – the ML-kNN algorithm.

**1.3 Brief Result Overview**

[Briefly summarize results]

**2.0 Data Mining Task**

**2.1 Task Details**

Participants of the Toxic Comment Classification challenge, the authors included, are provided with many Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity in which the authors must classify comments are:

* toxic
* severe\_toxic
* obscene
* threat
* insult
* identity\_hate

The data provided for the Kaggle competition are given to participants in a couple of different files.

* train.csv – the training set, containing comments with their binary labels
* test.csv – the test set, of which the authors predict toxicity probabilities of comments

The training data has eight columns, including the comment id, comment text, and the six toxicity labels (see Fig. 2).

Graphical user interface, text, application

Description automatically generated

**Fig. 2.** Five random rows of the training data

The testing data only has two columns, the comment id and the comment text (see Fig.3).

Graphical user interface, text, application, email

Description automatically generated

**Fig. 3.** Five random rows of the testing data

**2.2 Authors’ Questions**

As stated before, the authors set out to answer a multitude of questions, revolving around the effectiveness of different multi-label classification algorithms.

**2.3 Key Challenges**

**- downloading csv’s, delimiting sometimes didn’t work**

**- stop words and apostrophes**

**- rewriting TF-IDF**

**3.0 Technical Approach**

**3.1 Algorithm**

3.1.1 Exploratory Data Analysis

The focus of this step is to learn about our data, figuring out what questions to ask, how the data might have to be manipulated, what the answers tell the authors, among others.

To get there, the authors first had to load the data, using a pandas data frame, to look at all the aspects of information about the data. To get actual useful information, the authors first had to collect basic information on the input data, including information such as the number of rows and columns in the data, and the labels of each column. From here, the authors could look at more important information, such as the number of comments in each toxicity category, the ratio of clean comments to comments with labels, and comments with multiple labels (see Fig. 4).

A screenshot of a computer

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated Chart

Description automatically generated

**Fig. 4.** Exploratory Data Analysis

3.1.2 Data Pre-Processing

While this step isn’t a difficult or interesting step, it is very important. For the models to be trained well, the data must be taken from the raw data (in which comments may include special characters, HTML tags, punctuation, etc.) into TF\_IDF vectors with no stop words and stemmed phrases.

First, the data must be cleaned. In this step, the authors clean comments one-by-one, by setting each comment to lower case and removing non-alphabetic/non-numeric characters. They do so by creating a cleanComment() function, which can be applied to the pandas data frame.

Next, the data gets stop words removed. Stop words are any words in our stop list which are filtered out before processing the data, to save space and time in processing this large data set. The authors had done a similar step in a previous homework assignment by hand but decided to use an existing library containing common English stop words. This library is the Natural Language Toolkit for python, which the authors imported for this step. The authors referenced helpful steps to do this [here](https://www.geeksforgeeks.org/removing-stop-words-nltk-python/) (GeeksforGeeks, 2021). Then, the authors created a removeStopWords() function which, using NLTK functions, was then applied to the pandas data frame.

The next step is stemming. “Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words ‘chocolates’, ‘chocolatey’, ‘choco’ to the root word, ‘chocolate’ and ‘retrieval’, ‘retrieved’, ‘retrieves’ reduce to the stem ‘retrieve’” (GeeksforGeeks, 2021). Again, the authors referenced helpful steps found [here](https://www.geeksforgeeks.org/python-stemming-words-with-nltk/) for using NLTK again, to stem. Like before, the authors created a stemComment() function which, using NLTK functions, could then be applied to the data.

Finally, the last step of pre-processing is creating TF-IDF vectors. This is done using the sklearn python library, and the TfidfVectorizer function found within the feature\_extraction.text sub library of sklearn. The authors want to calculate TF-IDF scores for every word in every comment, to show which words are significant to each other. Doing so allows the creation of TF-IDF vectors, which will be used in our Multi-Label Classification methods.

3.1.3 Multi-Label Classification

At this point, the data has been pre-processed, and is ready to be evaluated by the three methods the authors chose to analyze.

The first method is Binary Relevance. This method ended up being the simplest, by treating each label as a separate single classification problem. The key assumption being that there is no correlation among different labels. Clearly, this is an oversight in the algorithm (that the authors are aware of), as some comment labels often correlated with others (See Fig. 4). [include info about the actual code].

Next was Label Powerset Transformation. This transformation method would change a multi-label problem into a multi-class problem with 1 multi-class classifier trained on the label combinations found in the training data (Szymański & Kajdanowicz, 2017). Unlike with Binary Relevance, this method does consider possible correlations between different labels. The problem with this method is the computational complexity, as when the number of classes increases, the number of distinct label combinations grows exponentially. [include info about the actual code].

Finally, was the ML-kNN Algorithm. “In detail, for each unseen instance, its K nearest neighbors in the training set are firstly identified. After that, based on statistical information gained from the label sets of these neighboring instances, i.e., the number of neighboring instances belonging to each possible class, maximum a posteriori (MAP) principle is utilized to determine the label set for the unseen instance” (Zhang & Zhou, 2007). [include info about the actual code].

3.1.4 Evaluation

Finally, the last step of the code is to evaluate the different methods. This is simply done by comparing the accuracies and the computational times of the methods.

**3.2 Addressing Challenges**

**4.0 Evaluation Methodology**

**4.1 Input Data**

**4.2 Evaluation Metrics**

**5.0 Results and Discussion**

**5.1 Results**

**5.2 Discussion**

**6.0 Lessons Learned**

**7.0 Acknowledgements**

[bibliography goes here]

Appendix 1 Python Code Modules

[python code goes here]