Combining Sentiment and Sentence Type to Aid in Classification

Min Gyu Woo
Data Science and Artificial Intelligence
University of Waterloo
mgwoo@uwaterloo.ca

Abstract

Pokharel and Bhatta [1] state that studies in sentiment analysis (positive/negative) and types of sentences already exist but are typically done separately on well-structured datasets (text-corpus). The novelty of the paper chosen discusses the use of both sentiment and sentence type on a non-structured dataset: in this case, YouTube comments. The importance of classifying YouTube comments is for the benefit of the content creators. Classifying comments to gain intuition on whether their videos are gaining traction or notoriety is much easier compared to searching in a pool of thousands of comments. The proposed solution is to pre-process the data, model with a variety of choices, and compare each model using cross validation and F1 scores to determine the optimal model. The paper will be summarized and verified.

1 Introduction

YouTube is dominating in the video streaming platform industry. With videos reaching millions of views over a span of days, it is home to many content creators (YouTubers). In order to thrive as a YouTuber, the content creator must be able to adapt to the needs of the audience. Without views, YouTubers cannot survive the competitive content creator market. Thus, one way of measuring the potential success of future videos is through comments. Audience members provide comments on most YouTube videos. Through these comments, YouTubers can gain intuition on whether they should keep the same format of the video or adjust accordingly. Reading through comments may work when the view count is low, but for videos with over a million views, it is infeasible to look for constructive criticism within the comment section. The paper chosen suggests an alternative efficient way by classifying comments based off sentiment and sentence structure. The categories are Negative, Positive, Interrogative, Imperative, Corrective, and Miscellaneous.

There are many studies that stem around sentiment analysis. These include Twitter sentiment analysis [2], YouTube polarity trend analysis [3], and user comment sentiment analysis on YouTube [4] mentioned in the paper. However, the author of the paper chosen argues that there is a lack of analysis through classification of sentence types. Being able to introduce sentence types increases the number of categories instead of simply positive and negative. The challenges of including sentence types as mentioned in the paper are: non-standard language, spelling errors, unformatted texts, and trivial comments.

Some considered solutions include categorizing comments based on the vocabulary, and extracting the features and using neural networks. However, both have an issue when it comes to a comment falling into multiple categories. For example, a comment such as ”How did you get n+1 as a solution? Otherwise, great video!” has the potential to be a interrogative (first sentence) or a positive (second sentence) comment. It is difficult to explain the value of a comment when it falls under more than one category.

The approach of the paper is as follows: pre-process the raw YouTube comments, extract features, train on well-known supervised learning algorithms, apply hyperparameter tuning, and compare the cross validation score and F1 scores for the best model.

2 Related Works

There have been various studies in regard to classification of text data. Similar to the paper's dataset, Porche et al. [5] uses SVM and Naive Bayes to classify comments on coding tutorial videos as "content concerns" and "miscellaneous". Again, this is limited to only two broad categories. Taboada et al. [6] use the method of categorizing based off vocabulary. In this paper, words from a dictionary were manually labeled as negative to positive sentiment from -5 to 5. The limitations here is the manual labelling which is not scalable. In the chosen paper, this step is done using supervised learning. In another study, Siersdorfer et al. [7] uses linear SVM to measure the positive/negative values of the comments with respect to a thesaurus. The paper chosen claims that their accuracy is higher than the study's 0.72 even without the use of a thesaurus. Krishna et al. [3] classifies sentiment on a different dataset (IMBD) using Naive Bayes. The issue here is, Naive Bayes requires the words to be categorized. I.e. if a new word appears, the model will fail. Bhuiyan et al. [4] uses the sentiment on comments to return a relevant video. Again, it is limited to only two categories. Maas et al. [8] approache the classification using both the semantic (unsupervised model) and sentiment side of sentences (supervised model). However, the result of their study only include the sentiment and did not include the sentence type. Now looking at classifying sentence types, Madden et al. [9] did not classify the sentences but instead provide 10 categories and 58 subcategories to work with. Cheung et al. [10] did classify their Webclopedia data but did not include any sentiment classification. Using more advanced neural networks, Kim [11] uses different variants of convolution neural networks to classify sentiment on different datasets. Hassan et al. [12] uses a recurrent neural network instead. The limitations of both is the fact that they only can classify sentiment. The work that is most similar to the chosen paper is by Khoo et al. [13]. The study contains 14 different classes of sentences and tests on multiple models such as Naive Bayes, Decision Tree, and SVM. The limitation here is the chosen dataset: it is well-structured (as compared to YouTube comments) since it is based off emails.

3 Problem Formulation

Below will be detailed explanation on the approach of the chosen paper.

3.1 Data Collection

YouTube comment data is not easily accessed publicly. The YouTube Data API must be used to access the comments on each video. Then, the comments can be extracted into a csv file using scraping techniques. The YouTube comments will not be scraped using the method given in the paper (in order to preserve the original dataset). The rest of the report will assume the data is freely available. After the scraping process, the authors of the paper manually label the dataset into the following categories: Positive, Negative, Interrogative, Imperative, Corrective and Miscellaneous. Positive refers to comments that indicate that the video was well made and that the content creator is on the right track for future viewership. Negative refers to comments that show dislike of the video. Interrogative comments include questions that the viewers want answered by the content creators. Imperative comments show the needs and wants of the viewers. Corrective refers to comments that point out errors made in the video. Miscellaneous comments refer to the remaining uncategorized comments. Figure 1 shows a summary of the categories and Figure 2 shows the data per category. Note that there is an imbalance of data.

In order to address the issue of a comment that is categorized in more than one class, the study chose to subjectively choose the topic which would best help the content creator. The example they gave was "Your solution is not practical. Can you suggest another one?" It is both a negative and interrogative comment. However, the content creator would benefit more greatly from the interrogative comment as fixing the issue would lead to more viewership. Another solution to this issue will be addressed later on in this reproduced paper.

3.2 Data Pre-processing

One of the issues with the YouTube comments dataset is that it is unstructured. Thus, pre-processing the data is a crucial step before training. The paper suggests the following issues: non-standard language, spelling errors, unformatted texts, and trivial comments.

Non-standard language refers to slang and improper forms of words. An example of such is the word ”pic”. ”pic” is short for ”picture”. These two words mean the same however the model would treat those two words as separate.

Spelling errors are common in social media platforms as the setting is casual. Unless the spelling mistake is corrected, the machine will take spelling errors as unique inputs (similar case as non-standard language).

Unformatted text refers to comments related to computer codes. As the YouTube comments are from tutorial videos, when commenters ask for help from a coding video, often it would include a snippet of the code they want assistance on. As this should not affect the model, they need to be removed. Another issue would be the amount of unique token variables that will be assigned if code is included. This causes unnecessarily large vocabulary list when the words are converted to numbers in the next steps.

Trivial comments are comments that are not useful to the YouTubers. Anything from commenters chatting with one another, self-promotion comments, and random spam comments.

The paper suggests the following to address the issues above by removing: URLs, new line characters, punctuation, integers, emojis, spelling errors and stopwords. They also include lowercasing and lemmatizing.

Lemmatizing is used to group words of similar meaning together. Stopwords are a list of words that are frequently used but have no meaning. Examples of which are ”is”, ”a”, ”are”, etc ... Note however that some stop words may be useful in the classification. The example given in the paper is ”not” and ”no”. Although these are common stopwords, the negative words can be useful in categorizing the comments (as negative class). Thus, the author’s created a custom stopword list. Figure 3 shows the ignored stopwords.

Removing the mentioned issues will allow the model to focus on the important words rather than taking up extra space. Lowercasing and lemmatizing allows the model to group similar words together. For example, ”Children” and ”Child” would be grouped together as they have similar meaning (lemmatizing would turn ”children” into ”child”). After lowercasing, ”Child” and ”child” would be considered one word.

3.3 Converting Text into Features

Now that the text is pre-processed, the words must be converted to some number for the machine learning models to take as input. As mentioned in the paper, the authors used document frequency vectorizer and tf-idf vectorizer.

Class	Content
Positive	appraisals, appreciations
Negative	scoldings, not able to do what is told in the video
Interrogative	all type of questions, queries, asking for something, sentences starting with modal/auxiliary verbs
Imperative	requests, commands, expectations
Corrective	amendments, improvements, mistakes, remedy
Miscellaneous	remaining types, promotions, chitchat

Figure 1: Summary of the chosen categories [1]

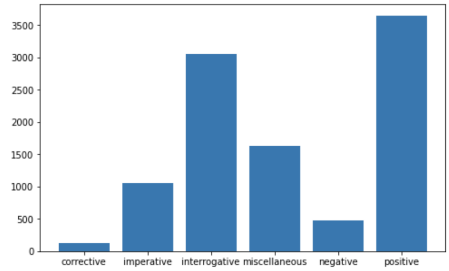

Figure 2: Number of comments in each category [1]

Class	Ignored Stopwords
Positive	NA
Negative	no, not
Interrogative	how, what, which, who, whom, why, do, is, does, are, was, were, will, am, are, could, would, should, can, did, does, do, had, have
Imperative	could, would, should, can
Corrective	NA
Miscellaneous	NA

Figure 3: Ignored stopwords by class
[1]

Document Frequency (df) The main idea of document frequency is the higher the frequency, the more important the word is. Every unique instance of a word within each comment is considered.

$$df = \frac{n}{N}$$

where df is calculated by the ratio of n number of times a word has appeared in N total comments

If a term appeared in less than or equal to 5 comments, they were ignored as they cannot distinguish the classes by it's low frequency. If a term appeared in the majority of the comments they were ignored as well for similar reasons but with high frequency. In total 2210 terms were derived and scaled using min-max scalar.

$$MinMaxScaler(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This keeps the data in a sufficient range for the model to handle.

Term Frequency–Inverse Document Frequency (tf-idf) The difference between df and tf-idf is that tf-idf is able to consider rare terms.

$$tf\_idf = tf * \log \frac{1}{df}$$

where tf is the number of times a word appears in a comment (no longer unique instance over comments like df and df is the same as before)

For the paper, the author's included unigram and bigrams (one and two character strings) which lead to 4304 features.

3.4 Model Selections

A variety of popular machine learning models were used to determine the optimal approach. The models used were: Linear Support Vector Classification (Linear SVC), Logistic Regression, Multinomial Naive Bayes (Multinomial NB), Random Forest Classifier, and Decision Tree Classifier. For hyperparameter tuning, a variety of hyperparameters were compared using Grid Search.

4 Experiments

For each of the grid searches, 10-fold stratified cross-validation was applied. The stratification is used since the data is unbalanced. The measure to determine the optimal parameters was mean_test_score. The tables below include at most the 5 highest mean_test_score due to the sheer number of parameters.

4.1 Document Frequency

Linear Support Vector Classification (Linear SVC) The parameter of interest are the regularization parameter $C = 0, 0.1, 1, 10$ and the choice of kernel = linear, sigmoid, rbf, poly. This is with fixed γ = scale.

Highest	mean_test_score	C	kernel
1	0.847 125	1.00	linear
2	0.845 417	10.00	sigmoid
3	0.841 417	1.00	sigmoid
4	0.839 333	0.10	linear
5	0.837 958	10.00	rbf

Table 1: Top 5 hyperparameter choices for Linear SVC

Looking at Table [1], the best parameters for Linear SVC are $C = 1.0$ and kernel = linear.

Logistic Regression The parameters of interest are the type of penalty = l1, l2, elasticnet, none, the regularization parameter $C = 100, 10, 1.0, 0.1, 0.01$, and type of solver = newton-cg, lbfgs, liblinear.

Highest	mean_test_score	C	penalty	solver
1	0.866 792	1.00	l2	lbfgs
2	0.866 792	1.00	l2	newton-cg
3	0.859 167	10.00	l2	liblinear
4	0.856 333	10.00	l2	lbfgs
5	0.856 333	10.00	l2	newton-cg

Table 2: Top 5 hyperparameter choices for Logistic Regression

Looking at Table [2], the best parameters for Logistic Regression are $C = 1.0$, penalty = l2 and solver = lbfgs

Multinomial Naive Bayes (Multinomial NB) The parameters of interest are the smooth parameter $\alpha = 0, 0.5, 1$ and choice to fit the prior or not (True, False).

Highest	mean_test_score	α	fit_prior
1	0.834 375	1.0	True
2	0.834 250	0.5	True
3	0.808 417	1.0	False
4	0.796 833	0.5	False
5	0.792 083	0.0	True

Table 3: Top 5 hyperparameter choices for Multinomial NB

Looking at Table [3], the best parameters for Multinomial NB are $\alpha = 1.0$, and fit_prior = True.

Random Forest Classifier The parameters of interest are the number of trees n_estimators = 10, 100, 1000, the criterion = gini, entropy, and the number of features to consider for the best split, max_features = sqrt, log2.

Looking at Table [4], the best parameters for Random Forest Classifier are criterion = gini, max_features = sqrt, and n_estimators = 1000.

Highest	mean_test_score	criterion	max_features	n_estimators
1	0.856 167	gini	sqrt	1000
2	0.855 208	gini	sqrt	100
3	0.854 125	entropy	sqrt	1000
4	0.852 833	entropy	sqrt	100
5	0.850 333	entropy	log2	1000

Table 4: Top 5 hyperparameter choices for Random Forest Classifier

Decision Tree Classifier The parameters of interest are the criterion = gini, entropy, the number of features to consider for the best split (max_features = None, sqrt, log2, auto), the maximum depth of the tree (max_depth = None, $2, 4, 6, 8, 10, 12$), minimum samples required to split a node (min_samples_split = $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$), and the minimum samples required to be a leaf (min_samples_leaf = $1, 2, 3, 4, 5$).

Highest	mean_test_score	criterion	max_depth	max_features	min_samples_leaf	min_samples_split
1	0.823 542	entropy	None	None	1	8
2	0.823 542	entropy	None	None	1	9
3	0.823 333	gini	None	None	1	8
4	0.822 958	gini	None	None	1	5
5	0.822 833	entropy	None	None	1	7

Table 5: Top 5 hyperparameter choices for Decision Tree Classifier

Looking at Table [5], the best parameters for Decision Tree Classifier are criterion = entropy, max_depth = None, max_features = None, min_samples_leaf = 1 and min_samples_split = 8.

4.2 Term Frequency-Inverse Document Frequency

Linear Support Vector Classification (Linear SVC) The parameters of interest are the regularization parameter $C = 0, 0.1, 1.0, 10.0$ and the choice of kernel = linear, sigmoid, rbf, poly. This is with fixed γ = scale.

Highest	mean_test_score	C	kernel
1	0.862 125	10.0	rbf
2	0.862 125	50.0	rbf
3	0.862 000	1.0	linear
4	0.858 833	1.0	sigmoid
5	0.856 125	1.0	rbf

Table 6: Top 5 hyperparameter choices for Linear SVC

Looking at Table [6], the best parameters for Linear SVC are $C = 10.0$ and kernel = rbf.

Logistic Regression The parameters of interest are the type of penalty = l1, l2, elasticnet, none, the regularization parameter $C = 100, 10, 1.0, 0.1, 0.01$, and type of solver = newton-cg, lbfgs, liblinear.

Looking at Table [7], the best parameters for Logistic Regression are $C = 10.00$, penalty = l2 and solver = liblinear

Multinomial Naive Bayes (Multinomial NB) The parameters of interest are the smooth parameter $\alpha = 0, 0.5, 1$ and choice to fit the prior or not (True, False).

Looking at Table [8], the best parameters for Multinomial NB are $\alpha = 1.0$, and fit_prior = False.

Highest	mean_test_score	C	penalty	solver
1	0.866 167	10.00	l2	liblinear
2	0.865 958	10.00	l2	lbfgs
3	0.865 875	10.00	l2	newton-cg
4	0.861 833	10.00	l1	liblinear
5	0.860 708	100.00	l2	lbfgs

Table 7: Top 5 hyperparameter choices for Logistic Regression

Highest	mean_test_score	α	fit_prior
1	0.814 208	1.0	False
2	0.807 917	0.5	False
3	0.804 583	0.5	True
4	0.792 333	1.0	True
5	0.785 833	0.0	True

Table 8: Top 5 hyperparameter choices for Multinomial NB

Random Forest Classifier The parameters of interest are the number of trees n_estimators $= 10, 100, 1000$, the criterion = gini, entropy, and the number of features to consider for the best split, max_features = sqrt, log2.

Highest	mean_test_score	criterion	max_features	n_estimators
1	0.855 000	gini	sqrt	1000
2	0.854 167	gini	sqrt	100
3	0.853 875	entropy	sqrt	1000
4	0.853 125	entropy	sqrt	100
5	0.848 708	gini	log2	1000

Table 9: Top 5 hyperparameter choices for Random Forest Classifier

Looking at Table [9], the best parameters for Random Forest Classifier are criterion $=$ gini, max_features $=$ sqrt, and n_estimators $= 1000$.

Decision Tree Classifier The parameters of interest are the criterion = gini, entropy, the number of features to consider for the best split (max_features = sqrt, log2, auto), the maximum depth of the tree (max_depth $= 2, 4, .., 12$), minimum samples required to split a node (min_samples_split $= 1, 2, 3, ..., 10$), and the minimum samples required to be a leaf (min_samples_leaf $= 1, 2, .., 5$).

Looking at Table [10], the best parameters for Decision Tree Classifier are criterion $=$ gini, max_depth $= 12$, max_features $=$ sqrt, min_samples_leaf $= 2$ and min_samples_split $= 5$.

5 Main Results

For both approaches, some optimal parameters matched with the ones mentioned in the paper. Since, the optimal values of the top 5 tend to provide very similar mean_test_scores there will be slight discrepancy between which parameter is "optimal". In Table 11 and Table 12 the scores were calculated based off the optimal parameters that were replicated.

Highest	mean_test_score	criterion	max_depth	max_features	min_samples_leaf	min_samples_split
1	0.811 250	gini	None	None	1	2
2	0.810 667	entropy	None	None	1	8
3	0.810 417	entropy	None	None	1	6
4	0.810 125	entropy	None	None	1	2
5	0.810 125	entropy	None	None	1	3

Table 10: Top 5 hyperparameter choices for Decision Tree Classifier

Model Name	Cross Validation Score	Cross Validation Score Paper	F1-Score	F1-Score Paper
Linear SVC	0.847	0.83	0.866	0.86
Logistic Regression	0.866	0.85	0.871	0.87
Multinomial NB	0.834	0.79	0.849	0.84
Random Forest Classifier	0.856	0.83	0.850	0.85
Decision Tree Classifier	0.823	0.80	0.835	0.83

Table 11: Final scoring results for df

Overall, the study is reproducible. As mentioned in the study there is a key issue with the dataset: having comments that are more than one sentence which refer to two or more categories. In order to address the problem, the idea of subjectively deciding which comment is more important to the content creators was used in the paper. However, this only assists in the labelling process. The machine learning model will not be able to differentiate what is "important" to the content creators. Rather, for example a comment has a strong positive word like "amazing", the model will lean towards the positive class. A solution to this is to break each comment into sentences and quantify the sentiment or sentence type. Then, combine the rankings to learn what is important. This approach is similar to Taboada et al. [6] where they ranked negative to positive sentiment from -5 to 5. Using the quantitative measure including the original features, the model may learn what is important to content creators (more subjective) instead of over-valuing strong words (like "amazing" mentioned before). For example a comment that is positive and imperative may be more likely to be predicted as impertive since the imperative comment may be more important to the content creators than the positive comment .

6 Conclusion

Pokharel and Bhatta [1] provided a great start to combining sentiment analysis and sentence type. As mentioned in their paper, this allows YouTubers to easily view the general trend of comments and can easily address any concerns with this model. With further improvement, the idea can be extended to businesses who also utilize reviews in order to match customer satisfaction and grow their business. In future studies the use of more intricate categories along with the idea of ranking the sentiment types could be used to further improve the results.

Model Name	Cross Validation Score	Cross Validation Score Paper	F1-Score	F1-Score Paper
Linear SVC	0.862	0.84	0.866	0.86
Logistic Regression	0.866	0.84	0.867	0.86
Multinomial NB	0.814	0.76	0.821	0.82
Random Forest Classifier	0.855	0.83	0.849	0.84
Decision Tree Classifier	0.811	0.80	0.819	0.81

Table 12: Final scoring results for tf-idf

References

[1] Rhitabrat Pokharel and Dixit Bhatta. "Classifying YouTube Comments Based on Sentiment and Type of Sentence". *arXiv preprint arXiv:2111.01908* (2021) (cit. on pp. 1, 3, 4, 8).

[2] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. "Sentiment analysis of twitter data". In: *Proceedings of the workshop on language in social media (LSM 2011)*. 2011, pp. 30–38 (cit. on p. 1).

[3] Amar Krishna, Joseph Zambreno, and Sandeep Krishnan. "Polarity trend analysis of public sentiment on youtube". In: *Proceedings of the 19th international conference on management of data*. 2013, pp. 125–128 (cit. on pp. 1, 2).

[4] Hanif Bhuiyan, Jinat Ara, Rajon Bardhan, and Md Rashedul Islam. "Retrieving YouTube video by sentiment analysis on user comment". In: *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. IEEE. 2017, pp. 474–478 (cit. on pp. 1, 2).

[5] Elizabeth Poché, Nishant Jha, Grant Williams, Jazmine Staten, Miles Vesper, and Anas Mahmoud. "Analyzing user comments on YouTube coding tutorial videos". In: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE. 2017, pp. 196–206 (cit. on p. 2).

[6] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. "Lexicon-based methods for sentiment analysis". *Computational linguistics*, vol. 37, no. 2 (2011), pp. 267–307 (cit. on pp. 2, 8).

[7] Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. "How useful are your comments? Analyzing and predicting YouTube comments and comment ratings". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 891–900 (cit. on p. 2).

[8] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. "Learning word vectors for sentiment analysis". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 2011, pp. 142–150 (cit. on p. 2).

[9] Amy Madden, Ian Ruthven, and David McMenemy. "A classification scheme for content analyses of YouTube video comments". *Journal of documentation* (2013) (cit. on p. 2).

[10] Zhalaing Cheung, Khanh Linh Phan, Ashesh Mahidadia, and Achim Hoffmann. "Feature extraction for learning to classify questions". In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2004, pp. 1069–1075 (cit. on p. 2).

[11] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751 (cit. on p. 2).

[12] Abdalraouf Hassan and Ausif Mahmood. "Deep learning for sentence classification". In: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE. 2017, pp. 1–5 (cit. on p. 2).

[13] Anthony Khoo, Yuval Marom, and David Albrecht. "Experiments with sentence classification". In: *Proceedings of the Australasian Language Technology Workshop 2006*. 2006, pp. 18–25 (cit. on p. 2).