Methods for Sentiment Analysis

Examples of NLP Techniques for Sentiment Analysis

Derek Kirsch[†]
student, MSDS, CU boulder

deki1596@colorado.edu

ABSTRACT

This paper provides examples of basic models for sentiment analysis as a method for classifying the emotional tone of textual data. Given the vast amount of textual content generated daily through social media platforms, automating sentiment measurement can offer valuable insights into public opinion and consumer behavior. While advanced methods like deep learning exist, this paper focuses on existing NLP (natural language processing) techniques such as Bag of Words (BoW), TF-IDF. And machine learning classifiers like Random Forests and Naive Bayes Classifiers. These approaches are chosen for their simplicity and efficiency as compared to DL (deep learning) Methods and SVM (Support Vector Machines). Also their accessibility on Python platform making them suitable for implementation in a project of this scope. The study utilizes pre-labeled datasets (labeled as positive, neutral, or negative sentiment) available on platforms like Kaggle, and use Python libraries such as NLTK and Scikit-learn for implementation. Implementation have resulted in a loose range of 50-80% average accuracy for datasets of about 3-5mb in size. Of particular note, Naive Bayes classifier implemented on either Bag of words or TF-IDF have shown a >80% prediction accuracy.

INTRODUCTION

Sentiment analysis is a method of classifying "*sentiment*" of a corpus of text. Webster definition of "*sentiment*" is a. **:** an attitude, thought, or judgment prompted by feeling, : b.: a specific view or notion : [1]. When prompted by a given context, human beings can express their emotion/opinion under that context, eg. text comments or social media comments online. Datasets with labeled measure of the "sentiment" of that person's comments will usually be labeled as 'positive' or 'negative' or 'neutral'. Several such labeled datasets exist in Dataset Repositories such as kaggle conveniently for this project. In this paper Sentiment Analysis is defined as classifying the emotional tone of textual data. Given the vast amount of textual content generated daily through social media platforms, automating sentiment measurement can offer valuable insights about the *positive/negative* "sentiment" of a body of text.

While advanced methods for sentiment analysis in DL (deep learning) exist, this paper focuses on existing natural language processing NLP techniques such as BoW (Bag of Words) and TF-IDF (Term Frequency-Inverse Document Frequency) for text representation and feature extraction to feed machine learning classifiers like Random Forests and Naive Bayes Classification. The methods are chosen for their simplicity and efficiency as compared to other ML/DL (deep learning) Methods, and their accessibility on Python platform.

The study utilizes labeled datasets (labeled as positive, neutral, or negative sentiment unless otherwise stated) available on platforms like Kaggle, and use Python libraries such as NLTK for BoW and TF-IDF and Scikit-learn for built-in Random Forests and Naive Bayes Classier for implementation. Datasets is split into Training and testing sets consist of looking at their accuracy score. Implementation have resulted in a loose range of 50-80% average accuracy.

Key findings include Naive Bayes implemented on either Bag of words or TF-IDF have shown a >80% prediction accuracy of sentiment, and datasets with less number of factors have better accuracy on average for all methods.

RELATED WORK

Sentiment representation techniques are found in the field of NLP (Natural Language Processing), namely **BoW** (Bag of Words), and **TF-IDF** (Term Frequency-Inverse Document Frequency) [4]. These methods transform data to feed into common **ML** (Machine Learning) techniques such as **Random Forest** classifiers an **Naive Bayes** to classify sentiment.

- There are other NLP techniques not considered here due to time and resource constraints. And the methods presented are chosen for their simplicity and readily available implementations in python libraries due to time and resource limitations.
- Examples of other techniques not touch upon are: Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA),Named Entity Recognition (NER), and Word Embeddings (Word2Vec, GloVe) .
- Due to high contemporary interest in this subject matter of NLP multitude of methods exist in literature. One class of methods are l*exicon based,*

here are some examples in literature: [6]

Some Lexicon based sentiment analysis methods:

1. WordNetAffect: Uses WordNet to determine sentiment scores.

2. SentiWordNet: A lexical resource that assigns sentiment scores to words.

3. VADER (Valence Aware Dictionary and sEntiment Reasoner): A rule-based model that uses a dictionary of words with sentiment scores.

4. AFINN: A dictionary-based approach that assigns sentiment scores to words.

5. OpinionFinder: Uses a dictionary-based approach to identify sentiment-bearing phrases.

6. LIWC (Linguistic Inquiry and Word Count): A dictionary-based approach that analyzes text for emotional and cognitive processes.

7. ANEW (Affective Norms for English Words): A dictionary-based approach that assigns sentiment scores to words.

Implemented NLP (Natural Language Processing) methods here are BoW and TF-IDF and are considered *corpus-based* methods.

METHODOLOGY

For the purpose of simplicity we limit sentiment analysis a classification problem using labeled training data (supervised learning) and mainly consider only the multiclass sentiment response with three factors *(positive, negative, or neutral)* There is one dataset with Binary classification (positive, negative) and one with four factors (positive, negative, neutral, irrelevant).

Ways of measuring sentiment: [8]

1. **Binary Sentiment:** Classify sentiment as positive or negative.

2. **Multi-Class Sentiment**: Classify sentiment into multiple categories (e.g., positive, negative, neutral).

3. **Regression-Based Sentiment**: Predict sentiment as a continuous value.

So, In the context of this proposal we mainly use the definition of sentiment as a factor with 3 possible values (positive, negative, neutral), this will be the standard response the implemented models of this paper will predict. Two of the datasets however have one with Binary sentiment and one with a 4-class sentiment factors (positive, negative, neutral, irrelevant)

We limit this paper to methods found in the field of NLP (natural language processing) which have current standard implementation in python libraries. For hypothesis testing of the modeling results we use base R and functions such as Shapiro.wilk test to test for nromality of each sample of 100 runds for each method applied to each dataset. Then once normality is confirmed use a pairwise welch T-test to compare differences of their averages.

Code for the implementation can be found at https://github.com/dak501/sentiment_analysis/.

Methods --

STEP 1. NLP Feature extraction/tokenezation):

1. **BoW**- bag of words – simple tokenization of each word as a vector dimension with its count as value.

2. **TF-IDF** (Term Frequency-Inverse Document Frequency) -- a statistical method used in natural language processing and information retrieval to evaluate the importance of words in a document based on their frequency and rarity across a collection of documents. [llama]

STEP 2. ML methods (supervised learning):

1. **Random forest -** machine learning algorithm that combines multiple decision trees.

2. **Naive bayes -** machine learning algorithm based on Bayes' theorem. 'Naive' in the sense of assuming independence of each predictor from each other.

Datasets: The following datasets from kaggle, due to limitations with computational power and memory requirements *if the dataset is too big a random sample subset of about 3-5mb is drawn from the larger datasets (about 20k rows or less):*

Data collection: We use collected datasets from kaggle [9] (. Large number of data set exist, we will prefer of course labeled training data with our preferred 3-value response (*positive, neutral, or negative*) although binary sentiment responses can be used (positive, negative) also. We take several datasets, which would have different contexts for sentiment analysis. This will provide a good mixture of data to do context-limited and general sentiment analysis.

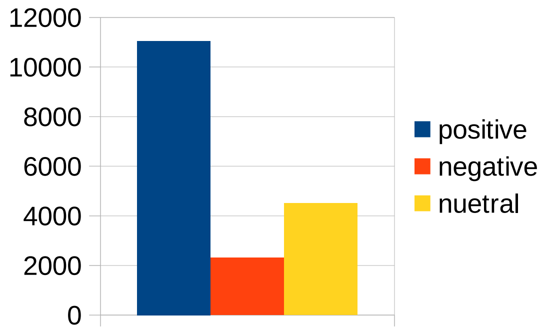
Here are the ff datasets and their sentiment distribution counts histograms:

Dataset 1 (Youtube dataset):

URL: https://www.kaggle.com/datasets/atifaliak/youtube-comments-dataset

Description: Youtube comments dataset.

Sentiment factor levels: *positive, neutral, negative*

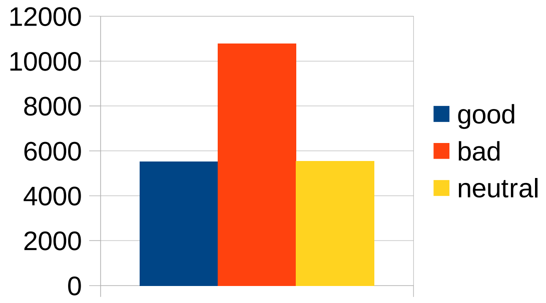


Dataset 2 (Chatgpt sentiment dataset):

URL: https://www.kaggle.com/datasets/charunisa/chatgpt-sentiment-analysis

Description: 10% random sample size of chatgpt dataset.

Sentiment factor levels: *bad, neutral, good*

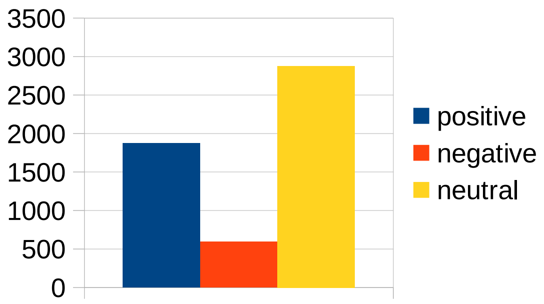


Dataset 3 (financial):

URL: https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis

Description: financial sentiment dataset.

Sentiment factor levels: *positive negative, neutral*

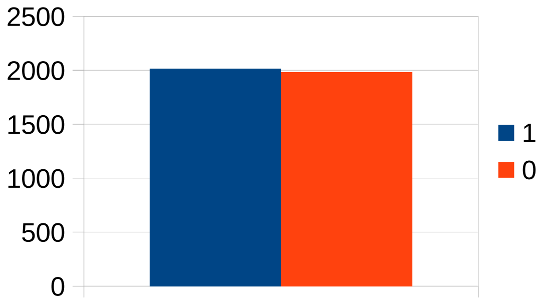


Dataset 4 (IMDB):

URL: https://www.kaggle.com/datasets/yasserh/imdb-movie-ratings-sentiment-analysis

Description: 10% random sample size of movie sentiment dataset.

Sentiment factor levels: 0, 1

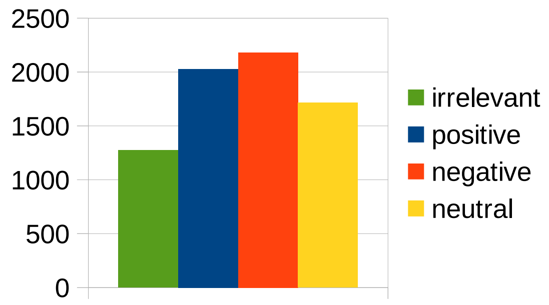


Dataset 5 (Twitter dataset)

URL: https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis

Description: twitter comment dataset

Sentiment factor levels: *positive, negative, neutral, irrelevant*



programming tools:

language: Python (for modeling), R (for statistics and graphing of results)

Python libraries: pandas, nltk, scikitlearn, re, langdetect

R libraries: R base, ggplot

Data Preprocessing: The collected data is essentially a table of at least two fields: Text and sentiment. Additional fields such as ID and context will be discarded. Only *text corpus* field and *sentiment values* are considered.

The text data field is cleaned and transformed to a form that can be accepted by the models. Of course another limiting factor we will have is to only consider English language comments any non-english data will be discarded.

Data Preprocessing Steps:

Cleaning (1)– Remove Special characters, drop missing values/duplicates

Cleaning (2)– Drop non-English comments, and drop english 'stop' words *

Normalization – convert text to standard form (ie. all lowercase or uppercase)

Lemmatization [10] – convert inflected words into standard root form**

Tokenization – split text into words and/or phrases (1-gram [words])

Vectorization – transformation of tokens into the vector space that can be processed by the ML method for prediction.

Bag of Words count vectorization and **TL_IDF** vectorization implemented successfully.

*article, common proposition words (ie, a, the, an, of, etc etc)
** eg. convert running, ran, or runs into root word: run (Provisionally)

modeling techniques:

BoW- bag of words -- Text documents are tokenized into individual words. A vocabulary is created by collecting all unique words from the corpus. Each document is represented as a numerical vector, where each element corresponds to the frequency of a word in the vocabulary.

Output: vector of tokens' count

- **TF-IDF (Term Frequency-Inverse Document Frequency)** -- a statistical method used to evaluate the importance of a word in a document based on its frequency and rarity across the entire corpus. It's a widely used technique in natural language processing (NLP) and information retrieval.

- Output: A vector of each tokens' TF-IDF scores

ML (Machine Learning) methods:

Random forest (ML) -- Random Forests is a popular ensemble learning algorithm used for classification and regression tasks. It combines multiple decision trees to improve the accuracy and robustness of predictions.

Naive Bayes – namely Multinomial Naive Bayes -- a popular supervised learning algorithm used for classification tasks. It's based on Bayes' theorem and assumes independence between features.

EVALUATION

As this project is a comparison of classification prediction models, we use the accuracy measure to analyze each recipe of models overall accuracy on each dataset.

To measure accuracy we devise 100 runs of the model recipes to each dataset. For each run each dataset is split 80/20 into training and testing data then each recipe is applied to the training/testing set and measured for accuracy for each recipe. Each run will get a different random 80/20 split. Accuracy is defined as the proportion of correctly predicted sentiment on the test set and is implemented by the sklearn.metrics library. For the large datasets of chatgpt and movies, a fraction of 0.1 random sample was used.

Table of average scores of 100 runs by dataset/recipe:

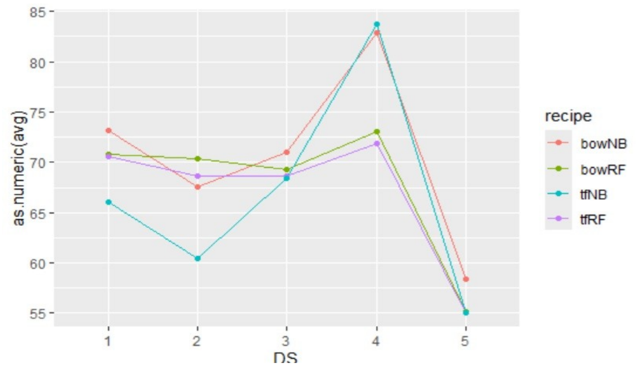
AVG	#factors	bow/RF	bow/NB	tf/RF	tf/NB
DS1 (youtube)	3	70.77	73.17	70.61	66.0
DS2 (chatgpt)	3	70.38	67.53	68.63	60.41
DS3 (financial)	3	69.28	70.98	68.61	68.43
DS4 (movies)	2	73.05	82.83	71.9	83.72
DS5 (twitter)	4	55.12	58.36	55.08	55.09

As we can see:

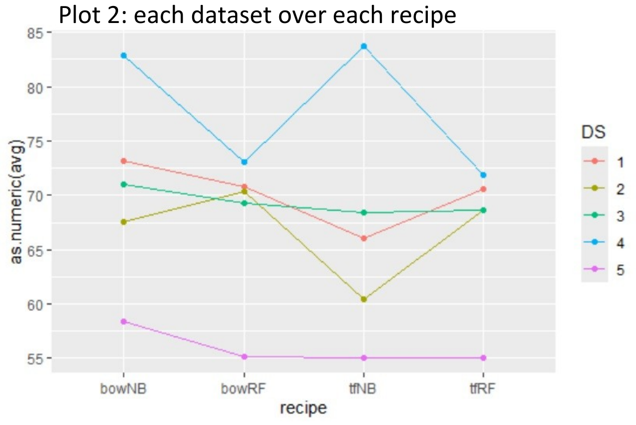
* NB for DS4 stands out as the highest results (82.83,83.72 for bow/NB and tf-idf/NB respectively)

* dataset 5 averages are much lowers than the rest of datasets at (55.12 58.36 55.08 55.09)

plot 1: each recipe over each dataset:



Plot 2: each dataset over each recipe



We can see from the two graphs that averages for dataset 5 are the lowest, most likely due to having 4 factors to predict. Another observation as noted before is that Naive Bayes see to have excellent (>80%) scores in regard to dataset #4 (movies) with binary factors. Also, there seems to be an uptick in the avg accuracy for bow/NB for all datasets except dataset 2.

We ran a shapiro-wilk test on all the test runs and have yielded a a p-value > 0.05 on all the runs, so the samples are likely from a normal distribution distribution for all the test runs.

See table below (shapiro-wilk test):

P-VALUE*	bow/RF	bow/NB	tf/RF	tf/NB
DS1 (youtube)	0.1795	0.6521	0.6814	0.0947
DS2 (chatgpt)	0.1581	0.5887	0.8558	0.9345
DS3 (financial)	0.8799	0.6213	0.6201	0.0685
DS4 (movies)	0.2034	0.1402	0.7594	0.3659
DS5 (twitter)	0.4933	0.5722	0.3854	0.0539

*all P-values > 0.05 so at a level of significance of 5% for each sample of runs we fail to reject the hypothesis that sample comes from a normal distribution.

With all run samples normal and assuming each generated samples to be independent of each other. We do a pairwise welch t-test to compare each run over datasets and over recipes to see if they are significantly different in performance. See tables below:

Recipe pair/dataset	1	2	3	4	5
bowRF – bowNB	T	T	T	T	T
bowRF – tfRF	F	T	T	T	F
bowRF – tfNB	T	T	T	T	F
bowNB - tfRF	T	T	T	T	T
bowNB - tfNB	T	T	T	T	T
tfRF - tfNB	T	T	F	T	F

*T - pair is statistically different (welch test, α=0.05), F - otherwise

From the table above we do not see any statistical difference for (bowRF,tfRF,tfNB) with essentially the same worst performance at ≈ 55.1% for dataset 5 (with 4 sentiment factors), also no difference for bowRF – tfRF for dataset 1, and likewise for tfRF – tfNB for dataset 3.

Dataset Pair/Recipe	bow/RF	bow/NB	tf/RF	tf/NB
1(youtube) – 2 (chatgpt)	T	T	T	T
1(youtube) – 3 (financial)	T	T	T	T
1(youtube) – 4 (movies)	T	T	T	T
1(youtube) – 5 (twitter)	T	T	T	T
2 (chatgpt) – 3 (financial)	T	T	F	T
2 (chatgpt) – 4 (movies)	T	T	T	T
2 (chatgpt) – 5 (twitter)	T	T	T	T
3 (financial) – 4 (movies)	T	T	T	T
3 (financial) – 5 (twitter)	T	T	T	T
4 (movies) – 5 (twitter)	T	T	T	T

Comapring datasets, all seem to be statistically different from each other with chatgpt and finacial dataset having statistically the same average for tfRF recipe as a particular anomaly.

DISCUSSION

The biggest takeaway from the results is that as we *reduce the number of sentiment factors the more effective the methods are.* Dataset 5 with 4 sentiment factors have results in the mid 50% area, while Dataset 5 have >80% accuracy results for Naive Bayes in particular. For Datasets 1, 2, and 3 with three sentiment factors accuracy is more or less around 70%.

Limitations, Challenges and alternatives:

As we can see from the result overall average prediction accuracy was 68.0%. To improve the general accuracy the best approach in my opinion is to implement a "bag of phrases" ie. tokenize phrases or n-grams (where a 2-gram is a a two word phrase). This would however increase dimensionality as it would be exponential in n.

The datasets used in this project is whittled down to a random sample of about 5mb each as that is the maximum that can fit on my laptop pc (ryzen 3/8gb memory). So for 2 of the datasets a fractional random sample was taken from the datasets in lieu of the full dataset. Also due to time-complexity constraint, **SVM** was dropped as a ML method from the proposal. Also **Keybert** was not implemented as an NLP preprocessing technique due to project time contraints.

Evaluation is the average of 100 runs of four accuracy scores of the following model implementations: BoW (Bag of Words)/RandomForest, TF-IDF/Random Forest, BoW/NaiveBayes, TF-IDF/NaiveBayes. All 4 methods were implemented to each of the 5 datasets and scored for average accuracy. Then each run is tested for normality (shapiro-wilk in R) and pairwise welch test of difference of means over method recipes and over datasets is performed (see EVALUATION).

We are only limited to labeled datasets. These datasets were taken from kaggle's already labeled and we rely on the veracity of the labeled data from kaggle, which is not assured. These labeled datasets are necessary as this paper's targeted NLP and ML techniques are classification (supervised learning). In the real world, data collected will usually be unlabeled and rely on clustering or other unsupervised learning for sentiment discovery. This will be out of the paper's scope and will only focus on the evaluation of the classification techniques outlined above. We will assume the correctness of the labeled data provided.

We will not consider Deep Learning Methods as implementing such tailored method would most likely be out of scope and limitations of time and resources of this project. Perhaps, asking a pretrained general purpose LLM such as llama, openAI, deepseek, etc. for its "analysis" on a samples of corpus of text asking whether its sentiment is positive, neutral, or negative as a potential data source if time and resources allow.

The methods also used tokens(words) as dimension and uses count vectorization (bow) and vector of scores (tf-idf) as a measure of weight or frequency to be passed to the ML models for prediction. So unlike lexicon-based methods, the methods implemented are agnostic in regards to any sentiment meaning or context found within the tokens themselves.

The ML methods does not perform any tuning of either implemented method (Random Forest and Naive Bayes). It implements the "vanilla" implementation with the usual defaults provided by python, except for Random Forest where I have set a run count n=10 instead of the default n=100 for efficiency.

The study also have focused only on accuracy score as the ultimate measure of effectiveness. Confusion matrix was not implemented and the study does not take into account of False Positives, False Negatives, and subsequent measures of recall and precision.

CONCLUSION

In conclusion, this paper is an exploration of sentiment analysis techniques using traditional natural language processing (NLP) methods such as Bag of Words (BoW) and TF-IDF for text representation, combined with machine learning classifiers like Random Forests and Naive Bayes. The study leverages labeled datasets from Kaggle to classify sentiments into three categories: *positive, negative, or neutral*. By focusing on multi-class sentiment classification (positive/negative/neutral), the paper implements these methods on text data across different domains, such as financial news, YouTube comments, and Twitter sentiments.

The methodology involved thorough preprocessing steps, including data cleaning, tokenization, lemmatization, and vectorization, ensuring that the data fed into the models will be consistent and relevant. The evaluation will used average accuracy score over a sample of 100 runs of each model recipe to provide insights into model performance on various datasets, with an different random 80/20 training-testing split approach for each of the run.

While the paper acknowledges limitations such as reliance on existing labeled datasets and the exclusion of more complex ML methods like SVM and deep learning methods due to resource constraints, we hope to successfully demonstrate the effectiveness of classical NLP techniques in sentiment analysis. In evaluation of the implemented methods in this paper, of note is the >80% accuracy measured for Naive Bayes for the dataset with only two sentiment factors. Also, increasing sentiment factors to 4 lead to a decrease in average accuracy preformance for all methods (around ~55% for dataset 5).

Sentiment analysis has had a large uptick in interest in recent years [11] largly in Deep Learning and LLM based methods. Those methods are out of the capabilities of the hardware available for these methods. But hopefully the efficient corpus-based ML methods (Random Forest and Naive Bayes) with NLP methods (Bag of Words vectorization, TF-IDF [Term Frequency-Invrse Document Frequency]) have provided the reader with good insight into sentiment analysis.

As one possible direction of future research, the “Sentiment Analysis” methods outlined here might also be easily changed to some sort of fraud detection as the methods are agnostic to content. This can be done by changing the response variable from expressing “truth” rather than “sentiment”. However keep in mind the limitation as the methods are supervised learning (classification) and would need a valid corpus of pre-classified data to base model training on.

Overall, I hope this study will contribute valuable insights into the application of NLP methods for sentiment analysis, offering solid foundation for further learning in the field of machine learning and natural language processiing.

REFERENCES

[1] https://www.merriam-webster.com/dictionary/sentiment

[6][8] llama LLM (meta messenger)

[4] https://www.kaggle.com/code/drisrarahmad/youtube-comment-dataset-nlp-ipynb

[7] https://www.sciencedirect.com/search?qs=sentiment%20analysis%20machine%20leaning

[9] https://www.kaggle.com/search?q=sentiment+analysis+in%3Adatasets)

[10] https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[11] https://www.sciencedirect.com/search?qs=sentiment%20analysis

PROJECT CODE:

https://github.com/dak501/sentiment_analysis/