Words of Wisdom Unleashed: The Ultimate Guide to Language Dominance!

Cristian Andreoli

Jun 2023

Contents

Abstract

This programming assignment focuses on building a clickbait classifier using the Naive Bayes and logistic regression algorithms. The objective is to develop a browser capable of distinguishing clickbait headlines from regular headlines. Clickbait headlines are known for their deceptive and sensationalized nature, often misleading readers and not accurately reflecting the content they link to. For classification, the Naive Bayes and logistic regression algorithms are utilized.

1 Preliminaries

A dataset containing 32,000 headlines has been collected, evenly divided into the 'clickbait' and 'non-clickbait' classes. The dataset is split into training, validation, and test sets, comprising 24,000, 4,000, and 4,000 samples, respectively. The headlines are stored in text files, with each line representing one headline.

```
Counter({'the': 4782, 'you': 4169, 'for': 2373, 'and': 2119, 'your': 1942, 'are': 1555,
'that': 1545, 'this': 1434, 'with': 1185, 'will': 1040, 'from': 1015, 'what': 989, 'new':
954, 'about': 870, 'who': 826, 'people': 761, 'things': 747, 'how': 705, 'can': 674, 'which':
 668, 'know': 608, 'after': 601, 'make': 596, 'should': 482, 'these': 476, 'based': 470,
'have': 468, 'all': 452, 'actually': 443, 'over': 443, 'times': 423, 'their': 421, 'here':
401, 'was': 397, 'out': 393, 'like': 384, 'best': 381, 'more': 381, 'first': 379, 'one': 379,
 'most': 378, '2015': 376, 'life': 373, 'when': 349, 'world': 343, 'need': 324, 'his': 314,
'has': 311, 'year': 306, 'time': 305, 'just': 302, 'day': 293, 'dead': 293, 'her': 275,
'killed': 270, 'ever': 265, 'get': 265, 'dies': 255, 'every': 251, 'two': 247, 'everyone':
242, 'not': 240, 'real': 239, 'into': 236, 'man': 235, 'president': 235, 'too': 227, 'love':
227, 'zodiac': 227, 'women': 225, 'star': 216, 'says': 216, 'obama': 213, 'australian': 213,
'favorite': 212, 'british': 212, 'only': 209, 'wins': 208, 'were': 207, 'show': 206, 'ways':
206, 'kills': 205, 'sign': 203, 'off': 203, 'game': 202, 'now': 199, 'photos': 197, 'way':
196, 'halloween': 193, 'understand': 192, 'character': 191, 'christmas': 191, 'they': 190,
'tweets': 190, 'would': 186, 'pictures': 184, 'police': 183, 'but': 182, 'reasons': 181,
'had': 181, 'guess': 181, 'well': 179, 'really': 178, 'found': 178, 'movie': 177, 'may': 177,
 'china': 177, 'its': 176, 'video': 176, 'court': 176, 'questions': 175, 'never': 174, 'sex':
 173, 'years': 170, 'why': 169, 'iraq': 167, 'look': 166, 'being': 164, 'tell': 163,
'disney': 162, 'government': 158, 'try': 157, 'than': 157, 'week': 156, 'south': 156,
```


Figure 1: Most common vocabulary's words.

2 Build a vocabulary

Building a vocabulary is an important step in text processing and analysis. It involves creating a collection of unique words or tokens present in the text data. In the case of clickbait classification, we can build a vocabulary by considering the words present in the headlines.
To begin, we need to perform some pre-processing steps on the text data to clean it and prepare it for vocabulary construction. Firstly, we remove punctuation from the headlines to eliminate any unwanted characters that might interfere with our analysis. This step ensures that the focus is solely on the words themselves.
Next, we remove words that are less than three characters long. These short words are often common articles, prepositions, or other insignificant terms that do not contribute much to the overall meaning or context. By removing them, we reduce noise and focus on more informative words.
In addition to these pre-processing steps, we also explore different vocabulary sizes. By varying the vocabulary size, we can investigate the impact of different levels of word coverage on the performance of our clickbait classifier. Experimenting with various vocabulary sizes allows us to find a balance between having a sufficiently comprehensive vocabulary and avoiding excessive computational costs.
Let's have a look at the most common words into the vocabulary Figure 1.

3 Extract the features

One of the most common representations for text data is Bag of Words (BoW). It provides a simple yet effective approach to convert text into a numerical format that machine learning models can process. The BoW representation consists of building feature vectors that act as counters for the occurrences of words in the vocabulary.
By constructing the vocabulary from the text data, we create a set of unique words that serve as the basis for our feature vectors. Each word in the vocabulary becomes a feature, and its index within the feature vector represents its position. To create a feature vector for a given text document, we initialize a vector of zeros with a length equal to the size of the vocabulary. Then, for each word in the document, we increment the corresponding feature vector element (index) by one to keep track of the word's occurrence.
This approach allows us to represent a document as a sparse vector, where each element indicates the frequency of a specific word in the document. The resulting feature vectors will be used in next section as input for classification models like Naive Bayes or logistic regression.

4 Naive Bayes Classifier

Naïve Bayes classification is a popular and effective machine learning algorithm for text classification tasks. It is based on Bayes' theorem, which provides a probabilistic framework for making predictions given observed evidence. In the context of text classification, Naïve Bayes makes the assumption of feature independence, meaning that the presence or absence of a particular word in a document is considered independent of the presence or absence of other words. While this assumption may not hold true in reality, Naïve Bayes often performs well and is computationally efficient, making it a popular choice for text classification tasks.

4.1 Training

During the training phase, I experimented with different vocabulary sizes and approaches to further refine the clickbait classification model. I explored vocabulary sizes of 1000, 2000, 3000, 4000, 5000, and 10000 to evaluate their impact on classification performance.
For the initial approach, I used the vocabulary constructed earlier and the corresponding feature vectors obtained from the Bag of Words representation. This approach considered the occurrence of words in the headlines without any additional modifications Table 1.

Vocabulary Size	**Training Accuracy**	**Test Accuracy**
1000	94.55%	94.18%
2000	95.98%	95.43%
3000	96.66%	96.12%
4000	97.02%	96.48%
5000	97.27%	96.55%
10000	97.86%	96.98%

Table 1: Training and test accuracy for models with a vocabulary of different size.

For the vocabulary size of 1000, the model achieved a training accuracy of 94.55% and a test accuracy of 94.18%. Increasing the vocabulary size to 2000 resulted in improved performance, with a training accuracy of 95.98% and a test accuracy of 95.43%.
As the vocabulary size continued to increase, the model's accuracy further improved. For vocabulary sizes of 3000, 4000, and 5000, the training accuracies were 96.66%, 97.02%, and 97.27%, respectively. The corresponding test accuracies were 96.12%, 96.48%, and 96.55%.
Finally, for the largest vocabulary size of 10000, the model achieved a training accuracy of 97.86% and a test accuracy of 96.98%. These results indicate that a larger vocabulary size allows the model to capture more nuanced patterns in the clickbait headlines, resulting in improved classification performance.

To enhance the performance for the 10000-vocabulary scenario, I tried several additional techniques. Firstly, I incorporated a list of stopwords to filter out common words that typically do not carry much information for classification. This step aimed to reduce noise and focus on more meaningful terms.
Additionally, I explored the concept of stemming, which involves reducing words to their base or root form. This technique aimed to normalize words by removing affixes, enabling the model to capture the underlying semantic meaning regardless of variations in word forms.
Furthermore, I tested a hypothesis that numbers in the headlines could be represented as tokens to capture their significance. I categorized numbers into different ranges to create meaningful representations. For example, numbers between 0 and 100 were represented as "0_100," while numbers between 1980 and 2100 were represented as "1980_2100." Any other numerical value was represented as "_NUM." This approach aimed to capture the importance of numbers in clickbait headlines Table 2.

4.2 Function implementation

In the context of clickbait classification, it's understandable to question the effectiveness of using a conventional stopword list. Stopwords are commonly used words that are often removed from text data as they are considered to have little or no discriminatory power for classification tasks. However, in some cases, stopwords may indeed carry valuable information for distinguishing clickbait from non-clickbait headlines.
Instead of using a predefined stopwords list, you adopted a different approach by utilizing a function that penalizes words that appear approximately half of the time in clickbait and half in non-clickbait, while rewarding words that are predominantly present in one category.

$$|x - 0.5| \cdot 2 \tag{1}$$

where **x** represents the frequency of a word.
This approach allows for a more dynamic consideration of word importance based on their distribution across the clickbait and non-clickbait classes. By assigning higher weights to words that are more strongly associated with a particular class, your method prioritizes words that potentially carry discriminatory information for classification.
It's important to note that this approach assumes that words with imbalanced distributions are more informative for classification. Results are shown in Table 2

Table 2: Accuracy Results with Different Approaches

Approach	Training Accuracy	Test Accuracy
Stopwords + words<3	97.10%	95.50%
LinearFunction + words<3	97.95%	97.73%
LinearFunction + Numbers	97.98%	97.73%
LinearFunction + Stemming	95.67%	95.38%
LinearFunction + Stemming + Numbers	94.93%	94.70%
LinearFunction	98.06%	98.00%

Overall, these results demonstrate the impact of different text preprocessing techniques and feature extraction approaches on the accuracy of the classification model. The combination of techniques, such as stopwords removal, linear equations, stemming, and numerical treatment, can contribute to improved performance in distinguishing clickbait headlines from non-clickbait headlines. The highest accuracy was achieved when using the linear equation approach alone, highlighting its effectiveness in capturing the discriminatory patterns present in the data.

4.2.1 Non Linear Function

In addition to the linear equation approach discussed earlier, I further investigated the use of non-linear functions to assign weights to words based on their distribution in clickbait and non-clickbait headlines. By applying the non-linear function:

$$(x - 0.5)^n \cdot 2^n \tag{2}$$

where **x** represents the frequency of a word and **n** is a parameter controlling the non-linearity.
The choice of the parameter n in the non-linear functions allows for fine-tuning the level of non-linearity. A lower value of **n** results in a smoother transition between word frequencies, while a higher value amplifies the differences, making the classification more sensitive to specific words.
It's worth noting that the utility and effectiveness of these non-linear functions may vary depending on the dataset and the specific characteristics of the clickbait headlines.

These results show the training and test accuracy achieved by the classification model when using the non-linear functions with different values of **n**. As we can see, all three approaches demonstrate high accuracy on both the training and test sets, indicating their effectiveness in distinguishing between clickbait and non-clickbait headlines.

Table 3: Accuracy Results with Different Non-linear Functions (n)

n	Training Accuracy	Test Accuracy
2	98.08%	98.02%
4	98.09%	98.08%
12	98.10%	97.98%

5 Logistic Regression

In addition to the Naïve Bayes classifier, I also experimented with logistic regression as an alternative approach for clickbait classification. Logistic regression is a widely-used algorithm for binary classification tasks, and it can provide insights into the relationship between the features (words in this case) and the target variable (clickbait or non-clickbait). By training logistic regression models on the clickbait dataset, I aimed to leverage the algorithm's ability to learn and predict the probability of an input headline belonging to the clickbait class. This approach allows for a more nuanced understanding of the underlying patterns and associations within the data. I evaluated the performance of the logistic regression models using the same training and test datasets, calculating the training accuracy and test accuracy to assess their effectiveness in clickbait classification.
The logistic regression model was trained using a learning rate (lr) of 0.001, a lambda (λ) value of 0, and a total of 40,000 training steps.
The logistic regression model achieved a training accuracy of 90.30% and a test accuracy of 89.32%. These results indicate that the model has learned to classify clickbait headlines with a reasonable level of accuracy.
These results suggest that the Naïve Bayes classifier outperforms logistic regression in terms of accuracy for clickbait classification.

6 Precision-oriented scenario

In a "precision-oriented" scenario for the Naive Bayes classifier, the objective is to minimize the chance of false positives while classifying headlines as clickbait. This means focusing on improving precision and reducing the number of non-clickbait headlines incorrectly labeled as clickbait.
By analyzing the confusion matrix, precision, and recall together, we can evaluate the classifier's performance in minimizing false positives while maintaining an acceptable level of true positives. It allows us to assess the trade-off between precision and recall and identify the optimal bias that achieves the desired precision-oriented objective.
Let's analyze the model with non-linear equation with n=4, which is the best one both for training and test accuracy Figure 2 first line.

In order to have small chance of false positive, we change the bias b obtained during training. Low values of b would make the model more likely to output the negative class, reducing the number of false positive and increasing the number of false negative. We can observe the results in Figure 2 second line.
The model without changed bias achieved a FPR 0.02899. On the other hand, the model with changed bias had a FPR of 0.0055, indicating that it had a lower rate of incorrectly classifying negative instances as positive. This suggests that the model with changed bias was more conservative in its predictions, resulting in a lower number of false positives.

7 Analysis

By examining the most negative and positive words under two different scenarios, we can gain insights into the significance of certain terms. In the first scenario, where words with less than three characters are removed and a stopword list is applied, we can observe the impact of these preprocessing techniques. However, in the second scenario, where no words are removed and no lists are used, a function is employed to adjust the weight of the words. Interestingly, this approach reveals that the word "ll" holds great importance, potentially indicating a future contract form of "will". These observations suggest that the second approach, more flexible on different dataset, works better. It posses the property to identify stopwords, assigning a low weight. Figure 3

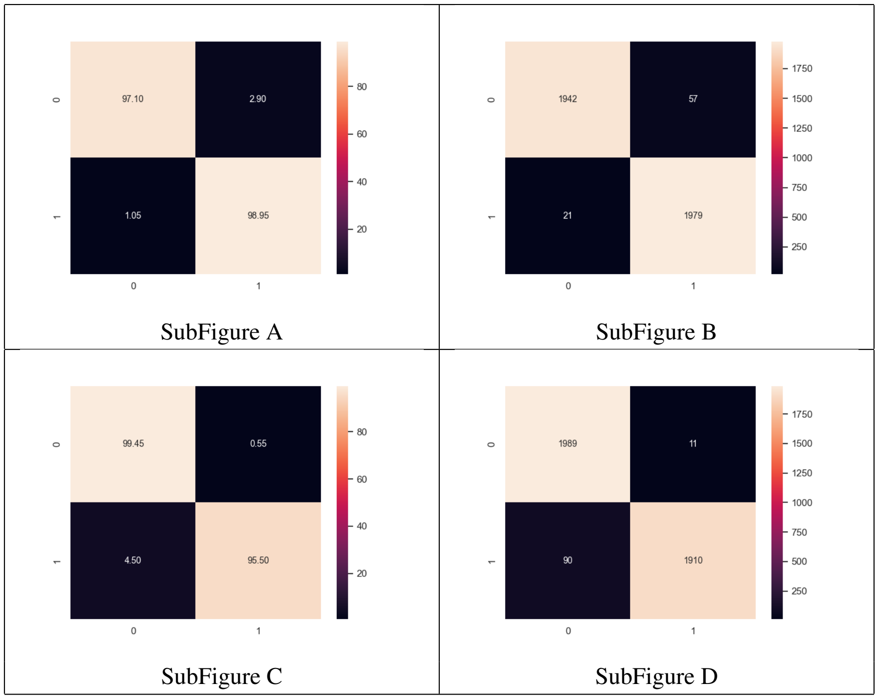

Figure 2: SubFigure A: Confusion matrix for non-linear model with n=4. Value in percentage. SubFigure B: Confusion matrix for non-linear model with n=4. SubFigure C: Confusion matrix for non-linear model with n=4 and adjusted bias. Value in percentage. SubFigure D: Confusion matrix for non-linear model with n=4 and adjusted bias.

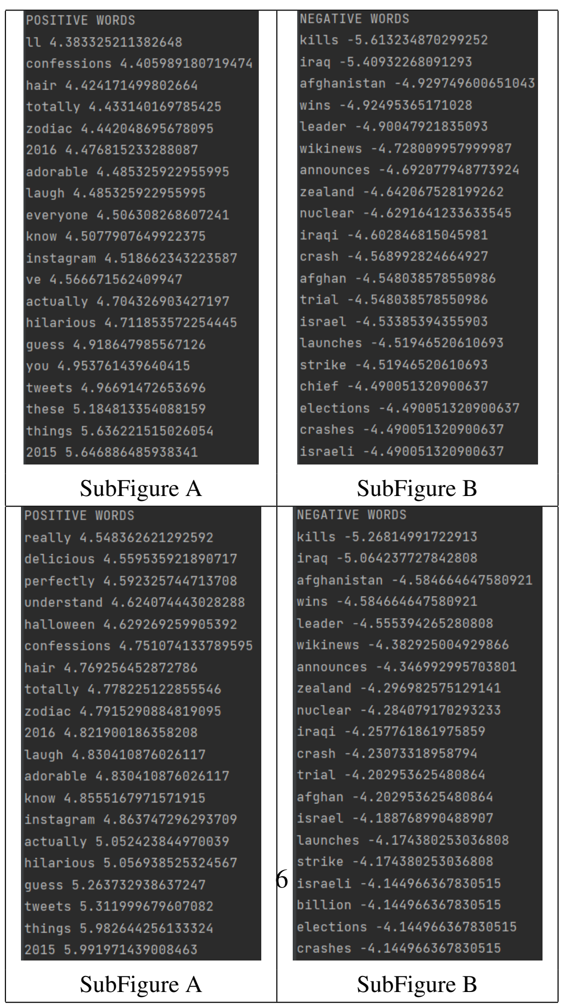

Figure 3: In the first row we have the positive and negative words for the case with the function. In the second row we have removed the words smaller then three character and used a stopwords list.

Following figure show the words for different scenario described in the 4.2 paragraph. Figure 4

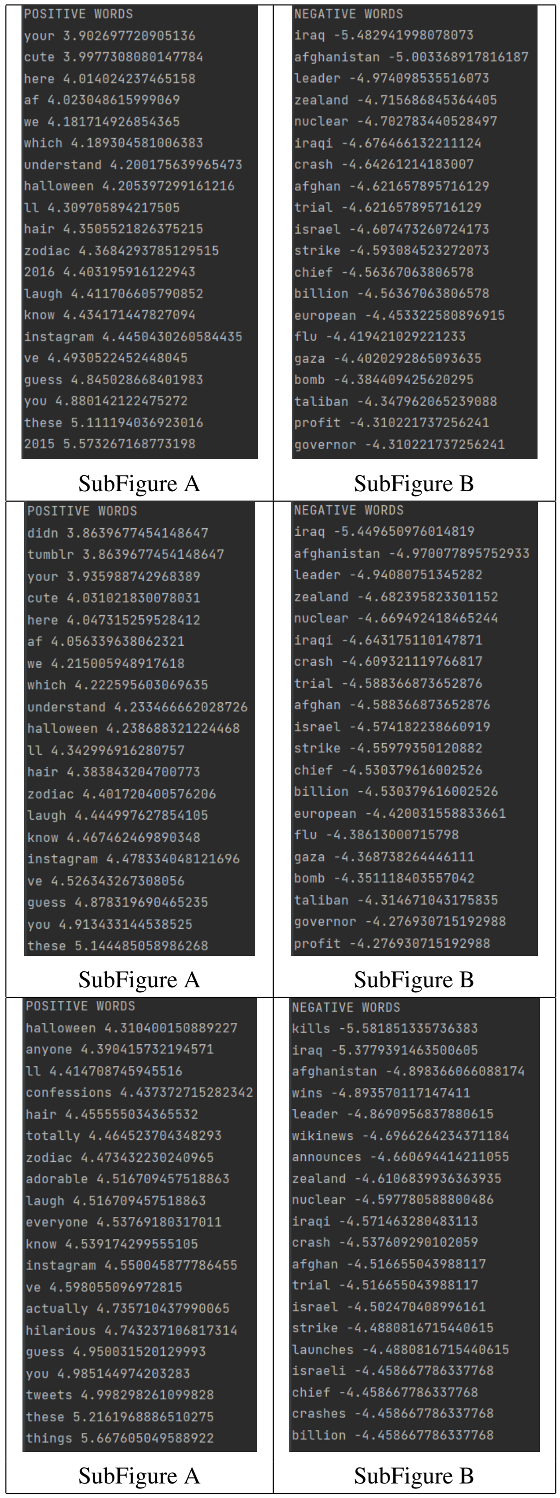

Figure 4: First row: Function with stemming. Second row: Function with stemming and grouped numbers. Third row: Function with grouped numbers.

8 Conclusion

Thanks to this groundbreaking work, I've come to a stunning revelation - my title is undeniably a clickbait masterpiece! With a touch of exaggeration and a hint of intrigue, it grabs attention like a magician's trick. It promises wonders and delivers excitement, just like a rollercoaster ride for the mind. So hold on tight and prepare to be captivated, because with this newfound knowledge, I can confidently say, my clickbait title reigns supreme in the realm of curiosity-inducing headlines!

9 Report

I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.