Text Classification Using Classical NLP: A Comparative Study of Methods

Connor Casey, Alex Melnick,
Loren Moreira, Bogdan Sadikovic

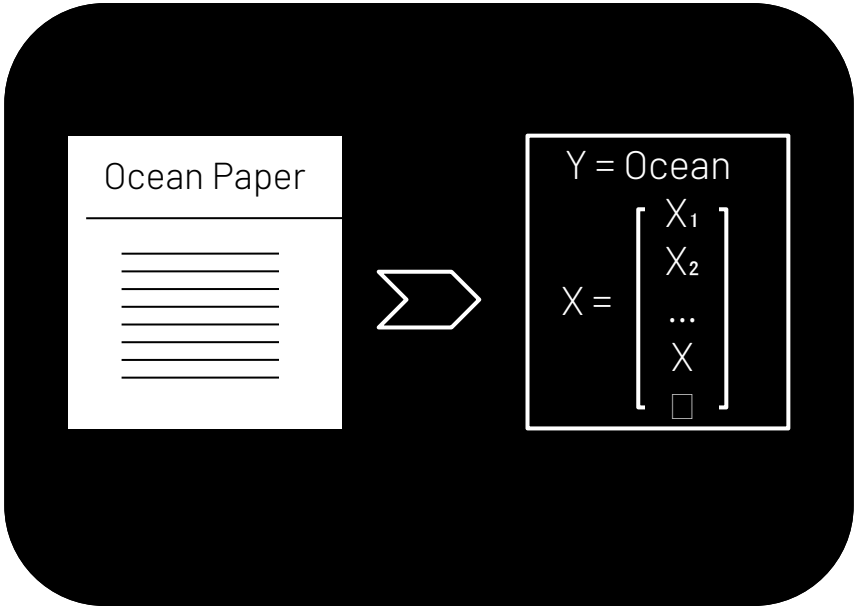
EC503: Learning From Data

Problem Definition: Supervised Text Classification

Labeled sets of text are **not usable** for Classical ML techniques

Natural Language Processing Techniques (NLP) must be utilized to produce feature vectors for a given text.

Ex: News Classification, Sentiment Analysis.

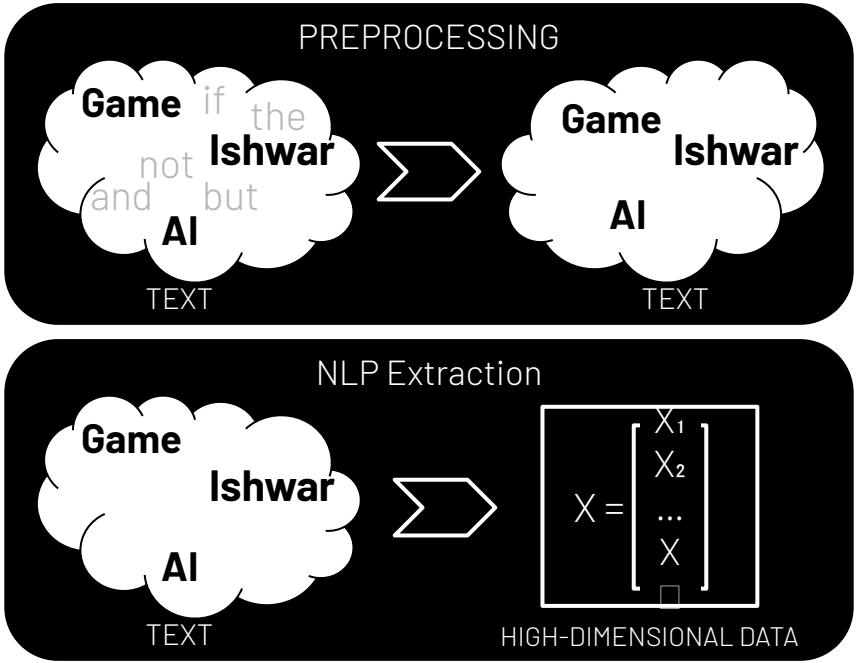


Common Properties/Roadblocks

Texts often contain thousands of unique words **leading to high dimensionality.**

Most are **stop words** that act as connectors, while important words are **keywords.**

High dimensionality can reduce model performance, **requiring preprocessing and feature selection.**

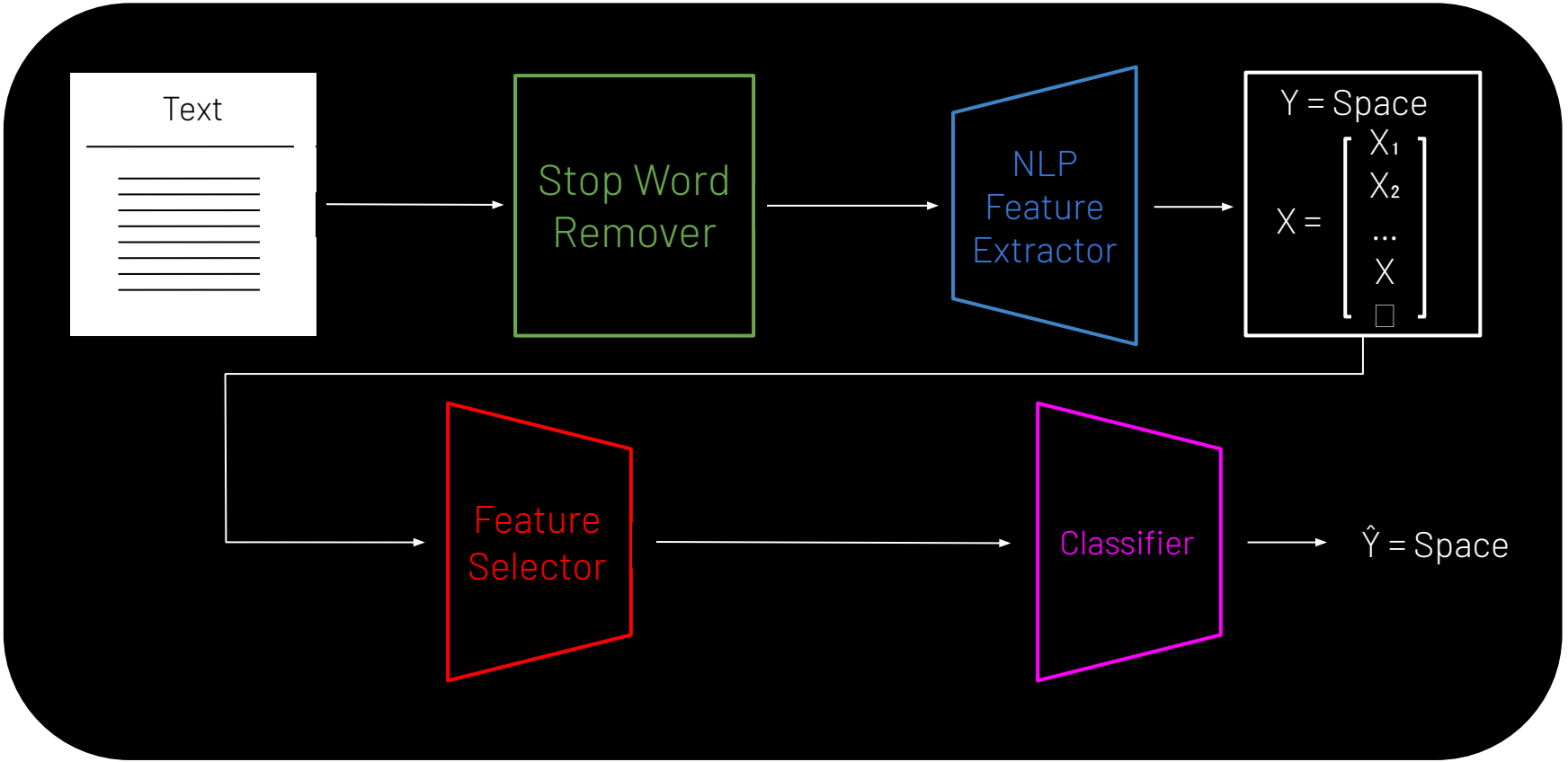


Datasets Used

20 Newsgroups Dataset: This dataset contains approximately **20,000 news articles** categorized across **20 distinct topics**, along with associated metadata. It has **a long history of use in classical text classification** tasks, making it a well-established benchmark for evaluating text classification models.

Clickbait Dataset: This **binary** classification dataset is made up of the headlines of approximately **16,000 articles**. It is designed to identify whether specific text classification models perform better in detecting clickbait content. **Useful for binary classification benchmark. This is our next step.**

Pipeline Design

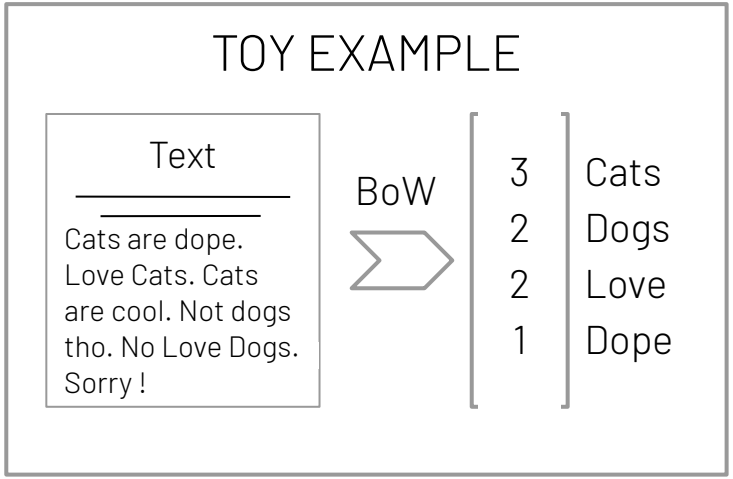


Bag of Words - NLP Method

BoW tokenizes the words in an article, **removing any stop words**, and creates a set of word collections featuring each article's word counts.

Creates a **Vocabulary** (set of words) of all important words in **Corpus** (set of texts)

Feature vectors are made using vocabulary, forming a **huge sparse dataset**.



Can be used on its own; For this project, we chose to **use it alongside TF-IDF**, to allow for optimized feature extraction from the dataset.

TF-IDF - NLP Method

TF-IDF **assigns higher weights to important words**.

Uses a **Corpus** and **Vocabulary** like BoW but different in **one key way**.

Words that are frequent in a text but rare across the **Corpus** are important.

For a corpus containing N texts and a vocabulary of J unique words:

$$\mathrm{TF}(\mathrm{word}_i, \mathrm{text}_m) = \frac{\#\text{ of times word}_i\text{ appears in text}_m}{\#\text{ of words in text}_m}$$

$$\mathrm{IDF}(\mathrm{word}_i) = \log\left(\frac{N}{1 + \#\text{ of texts word}_i\text{ appears in}}\right)$$

$$\mathrm{TFIDF}(\mathrm{word}_i, \mathrm{text}_m) = \mathrm{TF}(\mathrm{word}_i, \mathrm{text}_m) \times \mathrm{IDF}(\mathrm{word}_i)$$

TF measures how often a word appears in a text, while IDF measures how rare the word is across the **Corpus**. TF-IDF is the product of these and balances them. Outputs a **sparse dataset.**

Chi-Squared (χ^2) Feature Selection

Chi-Squared determines **whether there is a significant association** between two categorical variables (i.e., words and category).

In feature selection, the goal is to **measure how much a word is associated with a particular class**, identifying the most informative words.

The test compares **observed frequency** with **expected frequency**.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

O_i - Observed frequency for a word-newsgroup combination
E_i - Expected frequency for the same combination

Chi-Squared is **highly efficient** for high-dimensional datasets like natural language.

Mutual Information Feature Selection

Mutual Information (MI) **quantifies the amount of information one variable contains about another**.

MI **measures the dependency between a word and a class label**.

MI **captures non-linear relationships** between features and class labels.

The Mutual Information $I(X;Y)$ between a feature X and a class Y is:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

- $P(x,y)$: Joint probability of feature $X = x$ and class $Y = y$.
- $P(x)$: Marginal probability of $X = x$.
- $P(y)$: Marginal probability of $Y = y$.

However, it can be **computationally expensive for large datasets** with many features.

SVM

Suited for **sparse feature** set representations, as it mainly relies on the **support vectors which subset the feature set.**

Kernel Methods allow for non-linear decision boundaries. It's relevance in SVM allow us to test different kernel functions for a myriad of results.

These methods assist with **more accurate decision-making in higher dimensional space**, which is relevant in the context of text classification.

FEATURE EXTRACTED DATA

K > ~1000 WORDS

N > ~10000 ARTICLES

$$\begin{bmatrix} 5 & 0 & \dots & 0 & 0 & \dots & 0 & 8 \\ 4 & 0 & \dots & 2 & 0 & \dots & 6 & 0 \\ & & & \dots & & & & \\ 1 & 0 & \dots & 2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \end{bmatrix}$$

N x K

Our data is **sparse** because **key words are infrequent**, with each article containing only a **small subset** of the total vocabulary

Majority is ZERO

Random Forest

Robust to noise, **like unimportant words,** This also makes overfitting to noise highly unlikely.

Can "rank" the importance of individual features, which could provide insights as to which features are most important when it comes to decision making.

Random Forest **handles sparse high-dimensional features representations well**, which is crucial for working with TF-IDF and BoW.

Input: Test data

1. Predict and store the outcome of each randomly created decision trees (D Tree's) on given test data
2. Compute the total votes for individual class
3. Declare majority class as the final outcome class

Output: Final predicted class

Naive Bayesian

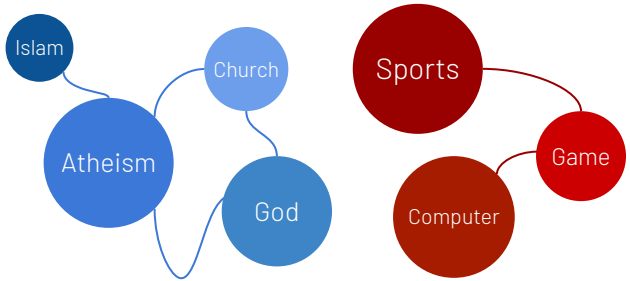
The Naive Bayesian algorithm **relates the conditional probabilities of events utilizing Bayes' Theorem.**

Assumes that features are conditionally independent of each other, **chooses class of highest posterior probability**, which is chosen by multiplying the prior probability by the probability of the class.

Great baseline algorithm. **Fails at a higher level because of its base assumption.**

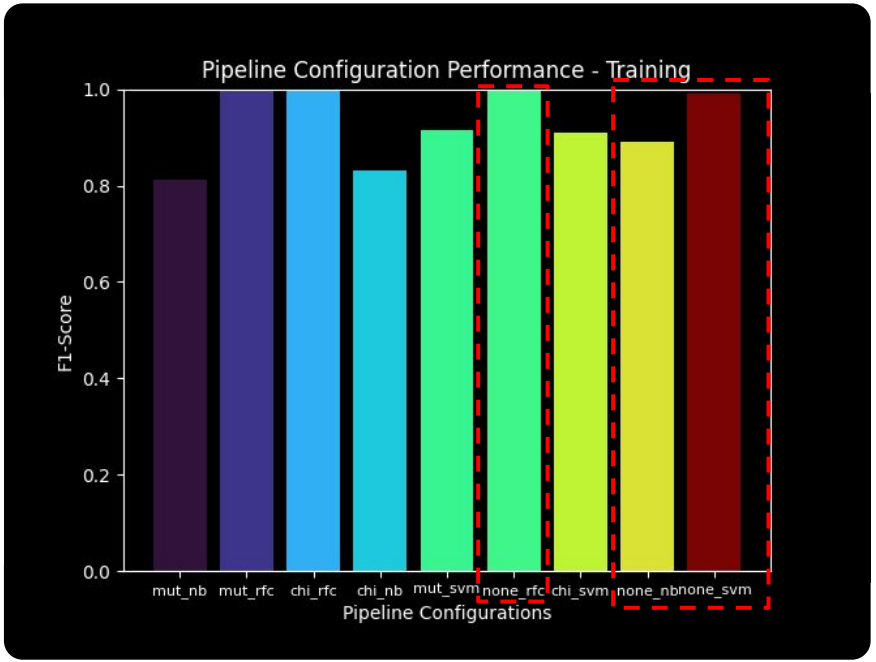
$$\frac{argmax}{k \in \{1, \dots K\}} P(C_k) \prod_{i=1}^{n} P(X_i | C_k)$$

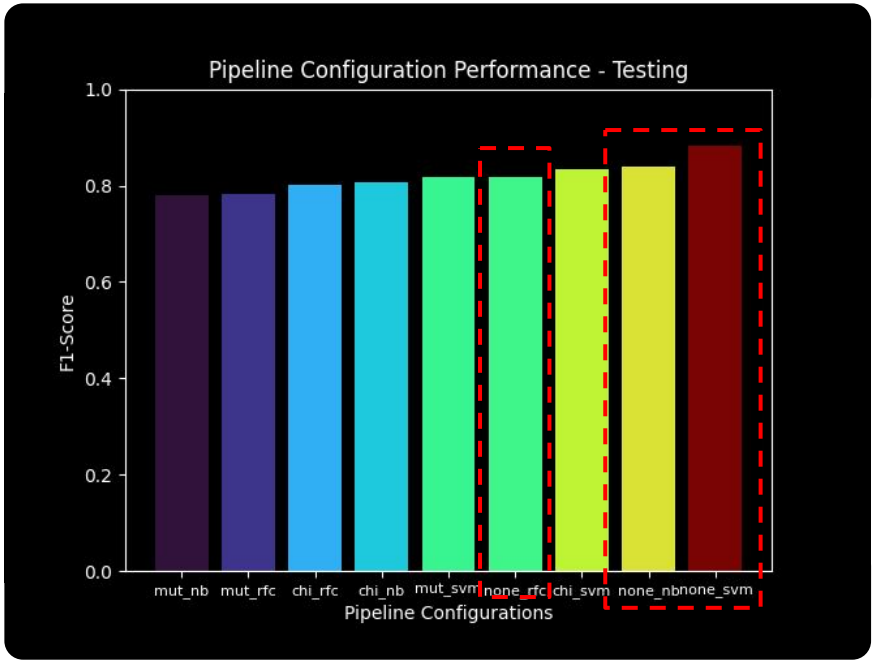
Not all Keywords are Independent



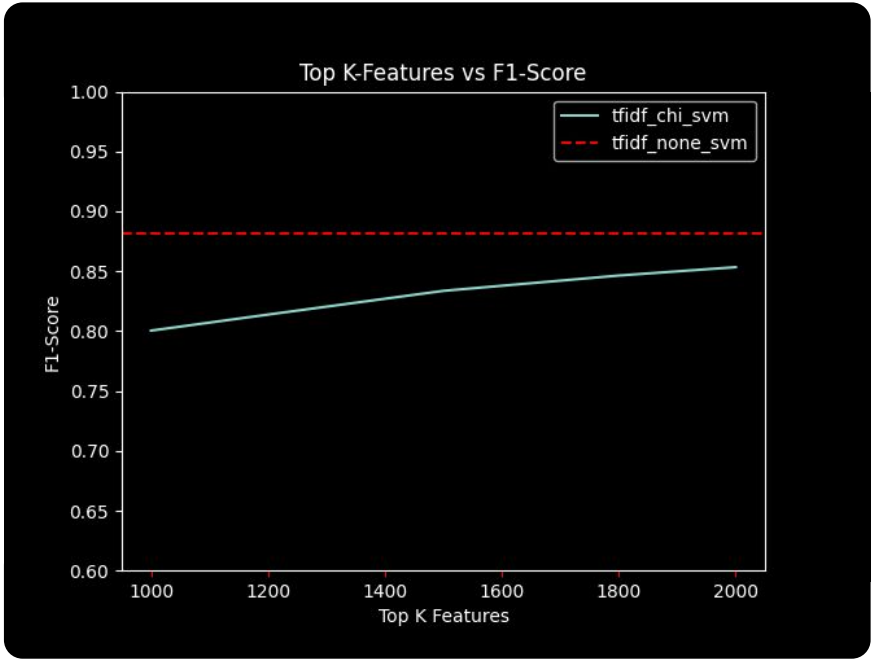
Best Performing Pipeline (20newsgroup)

5 Fold Cross-Validation with 80/20 Training/Test Split with TF-IDF





Feature Selection Tuning



Even the best feature selection techniques **fail to gain any accuracy**, but **both increased efficiency** significantly**.**

This is because TF-IDF is so **effective at identifying key features** (important words)

Any further attempts to reduce the dimensionality **remove helpful features** leading to a lower score.

THANK YOU