Detecting Clickbait with Classical and Transformer-Based NLP

Rex Sheikh
CSPB 3832 – Natural Language Processing

December 5, 2025

Abstract

Detecting clickbait has a clear usage in flagging low-value content disguised by carefully deployed language. I trained Naive Bayes, logistic regression, and linear SVM models before fine-tuning BERT and DistilBERT to compare classical frequency-driven pipelines against contextual transformers. Three datasets were split deterministically (80/20 training/testing) so that every model saw the same evidence.

Results can be interpreted from several perspectives including data-focused and model-focues. The balanced Kaggle data led the classical models resulting in 0.95 accuracy, with logistic regression at 0.964 accuracy and BERT reaching 0.99 accuracy and recall. Train2, with only about 20% clickbait, hurt the performance of classical models. Webis landed between the two, resulting in 0.816 accuracy with classical models and roughly 0.857 with transformers. Taking a model-based perspective, performance can be ranked from worst to best in the following order: Naive Bayes, Logistic Regression, Linear SVM, DistilBERT, BERT.

Introduction

Misrepresentation, misinformation, and qualitatively bad information is prevalent on the modern internet. Clickbait headlines are crafted to capture attention first. Their content is often of dubious value and can at times be purely misleading. I framed the project around the idea that we can use a suite of NLP techniques to label whether a headline reads like news or clickbait. Applications to mitigate clickbait effects are discussed in the future work section.

The experiments include classical and transformer-based models to not only move toward the overall project goal of label classification but also to illustrate some of the strengths and weaknesses of each model. Linear models highlight which words, punctuation marks, or structure choices correlate with either label, while transformers encourage a more contextual read that incorporates interactions between tokens. Training on curated datasets, keeping splits controlled, and calculating metrics to include precision, recall, F1, ROC-AUC,

PR-AUC creates a reliable environment for comparing the strengths and weaknesses of each model.

Related Work

There are several notable and well-cited publications that cover this binary classification or relate to the training and testing pipeline in some way. First, Chakraborty et al.'s Stop Clickbait browser extension showed some pattern-based strategies for flagging clickbait headlines by scanning for cue phrases ("This is why," "You won't believe"), abnormal punctuation density, and exaggerated adjectives before the user clicks a link, then showing an inline warning or suppressing the click entirely in the browser. Potthast et al. expanded insight into this area with the Webis Clickbait Corpus, comprised of 195k examples with additional media; they recruited raters to rate clickbait strength and ran benchmark baselines to demonstrate how the corpus could drive textual and multimodal detection studies.

The models used in this project also have a rich history of well-cited publications. Joachims formalized the use of TF–IDF features with linear SVMs for web classification. BERT and DistilBERT are popular transformer based contextual models that have proven value for classification once fine-tuned [5, 6]. Overall, these references helped to outline the progression of these models from classical approaches to large pre-trained language models and informed the implementation choices in this project.

Data

The Kaggle dataset provides 31,997 headline/label pairs in CSV format, evenly split between clickbait and news, making it an ideal for quickly spotting discriminative language features. Train2 has roughly the same file structure but the data is quite imbalanced. Only about 20% of the 32k rows are clickbait. This dataset was kept to illustrate how different models handle imbalanced data. The Webis Clickbait Corpus provides 195,389 entries in JSONL files split between instances and truth. I extracted headlines and truth labels by unique identification numbers but the majority of this corpus went unused. The scope of this data, though, offered some ideas for future work.

The data required some preprocessing before entering the formal training and testing pipeling. Baseline experiments include minimal preprocessing with only casing and white space removal. From there, ablation studies for stopwords and common words are applied to asses not only effects in metrics but to extract the most informative tokens for each model. Each dataset is shuffled with a fixed seed (42) and split 80/20 to allow for model comparison.

Methodology

The classical pipeline starts with a deterministic shuffle, an 80/20 split, and word tokenization. Stopwords plus a short list of common yet intuitively uninformative words can be optionally removed from training and testing via CLI flags. Other flags include '–punct-signals' to add exclamation/question/emoji ratios, '–struct-features' to inject token/character-

length statistics, '–ignore-terms' to strip overused words like "headline," '–min-df-high' to tighten vocabulary pruning, and '–class-weight' to tell the estimator to pay extra attention to minority labels. The following experiments only utilized the stopwords and ignore-terms flags to keep scope limited. Other flag combinations and experiments are left to future work.

To better understand the results, we must first introduce each model in detail. The Naive Bayes baseline assumes conditional independence between tokens and can be described generatively: each token in a headline votes for the class that best explained it during training, adding to that class's likelihood score, and the model multiplies those likelihoods together before finally normalizing with the class priors. For a headline represented by word counts x, the argmax classifier computes

$$\hat{y} = \arg\max_{y \in \{0,1\}} P(y) \prod_i P(x_i \mid y), \tag{1}$$

which states for each possible class label y, multiply the class prior $P(y)$ (which captures how common the label is overall) by the likelihood of each observed token $P(x_i \mid y)$. The argmax functions picks whichever class produces the largest product. So, this model maps directly to frequency counts. It is especially weak when a test set contains a previously unseen token and cannot capture contextual clues.

Logistic regression improves upon this by learning a weight vector w where x is the TF-IDF feature for a given headline (an improvement in tokenization from pure frequency counts). The sigmoid outputs calibrated probabilities to generate labels.

The linear SVM, trained via 'SGDClassifier', searches for a hyperplane that maximizes the margin while tolerating hinge-loss violations:

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C \sum_i \max\bigl(0, 1 - y_i(w^\top x_i + b)\bigr), \tag{2}$$

where labels are remapped to $y_i \in \{-1, +1\}$. The same TF–IDF and structural features drive the classifier.

Transformer models begin with 'bert-base-uncased' and 'distilbert-base-uncased' checkpoints from Hugging Face. I fine-tuned them for two to three epochs with maximum sequence lengths of 64 (DistilBERT) and 128 (BERT). Each transformer block computes self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \tag{3}$$

allowing the model to contextualize every token against all others. This does very well to capture word and word phrase dependence common in natural language.

Evaluation stays consistent regardless of model: accuracy, precision, recall, and F1 are reported per class alongside macro and micro averages, while ROC-AUC and PR-AUC summarize ranking quality. Confusion matrices provide an immediate sense of how often the system mistakes news for clickbait (false positives) or overlooks genuine clickbait (false negatives). Learned coefficients are also sorted to print the strongest clickbait vs. news identifiers, making the models interpretable even when they disagree.

Results

On Kaggle, Naive Bayes already clears 0.95 accuracy thanks to easily separable tokens like "you," "this," and "your" dominating clickbait predictions while names such as "Breitbart" or verbs like "says" anchor news. Logistic regression lifts accuracy to 0.964 with balanced per-class precision and recall, and toggling the optional feature sets barely shifts the metrics because the dataset is evenly split. Linear SVM mirrors those numbers and confirms that the discriminative hyperplane is almost trivial in this domain. Transformers mainly validate the ceiling: BERT posts 0.99 accuracy/recall and DistilBERT trails by only a few tenths.

Train2 exposes the limits of frequency-based models with imbalanced data. The logistic baseline yields 0.823 accuracy yet captures only 0.224s. Enabling structural and punctuation features nudges recall to 0.633 and raises the positive F1 score from 0.336 to 0.491, but accuracy drops to 0.738 due to additional false positives. The Webis corpus tells a similar story though with milder imbalance: classical models hover around 0.81 accuracy. Top identifiers shift toward listicle cues like "things," "ways," and "quiz."

Transformers showed better performance where classical models struggled. On Train2, BERT delivers 0.825 accuracy with 0.576 precision and 0.458 recall on the clickbait class, while DistilBERT trades some recall (0.392) for slightly better precision (0.623). On Webis, both checkpoints sit near 0.85 accuracy with recall above 0.65, confirming that contextual modeling plus subword tokenization gives better contextual understanding and results.

Discussion

Vocabulary differences alone flag nearly every Kaggle headline, so lightweight models suffice and provide great results. It can be argued that "you" or "this" rank highest in clickbait cues matches intuition. Imperative and second-person structures dominate clickbait headlines, while datelines and proper nouns are more prevalent in news headlines.

For Train2 and Webis, data imbalance is the biggest detractor to performance. Adding class weights, structural signals, and more aggressive pruning helps but classical models still conflate subtle clickbait with news because the majority class dictates the separating hyperplane. Transformers, even when fine-tuned for only a few epochs, capture word order and phrase templates ("You won't believe...") that linear bag-of-words features ignore.

Conclusion and Future Work

Taking a data-focused perspective, Kaggle remains the benchmark where vocabulary features dominate. Train2 functions as a stress test for imbalance. Webis lands in between the two. In terms of models, Naive Bayes is a quick and efficient tool with supringly good results with balanced data. Logistic regression and linear SVMs provide good accuracy on balanced corpora too. Transformers extend recall on harder datasets at the expense of training time.

Looking ahead, I want to fuse headline labeling with article-body validation, particularly for Webis where full articles, teasers, and media metadata are readily available like in

the Webis corpora. Modeling engagement intensity or clickbait sub-types could push beyond binary classification. Finally, the CLI already exposes feature-ablation switches and class weighting, so automating experiments with different combinations can surface which linguistic signals most directly impact performance metrics.

References

[1] Abhijnan Chakraborty, Rhia Perry, Saptarshi Ghosh, and Niloy Ganguly. "Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media." In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2016.

[2] Thorsten Joachims. "Text Categorization with Support Vector Machines: Learning with Many Relevant Features." In *European Conference on Machine Learning*, 1998.

[3] Kaggle. "Clickbait Challenge Dataset." https://www.kaggle.com/clickbait-dataset, accessed 2025.

[4] Martin Potthast, Matthias Hagen, and Benno Stein. "The Webis Clickbait Corpus." In *Journal of Data and Information Quality*, 2018.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of NAACL-HLT*, 2019.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter." arXiv preprint arXiv:1910.01108, 2019.

[7]
- Use of AI:
- genmetrics.py: using ChatGPT, I uploaded raw printouts with the following prompt: "I'd like to consolidate metrics in two ways. First, create bar charts that show classical models (naive bayes, logistic regression, and SVM) without stopwords removed and with stopwords removed. Second, create scatter plots for each model for macro-F1." The file `gen_metrics.py` uses regex to extract metrics from text files.
- Equations 1, 2, and 3 in this report. Again using ChatGPT, I gave the prompt: "Give equations for naive bayes, linear SVM, and self-attention in an NLP context in LaTeX."
- General debugging efforts in both classical and transformer experiments.

Appendix: Figures

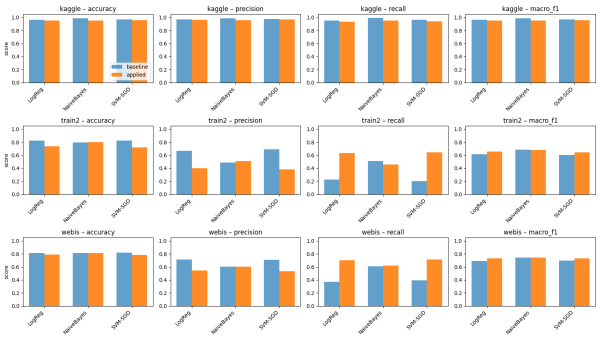

Figure 1: Baseline vs. feature-augmented classical metrics across datasets.

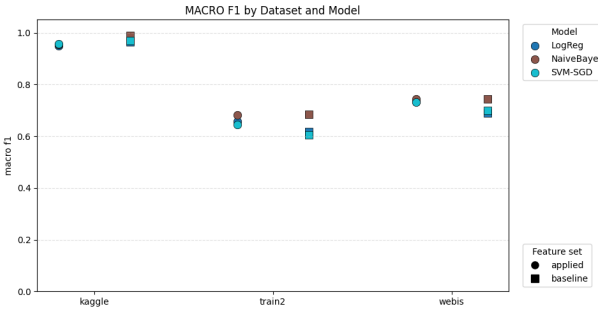

Figure 2: Macro-F1 scatter highlighting dataset-level performance for all models.

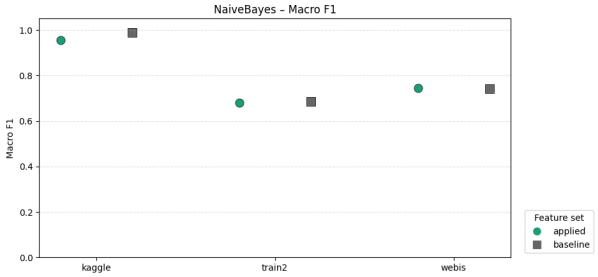

Figure 3: Naive Bayes macro-F1 scatter segmented by dataset and feature set.

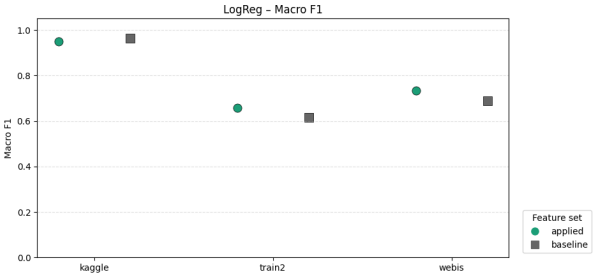

Figure 4: Logistic regression macro-F1 scatter segmented by dataset and feature set.

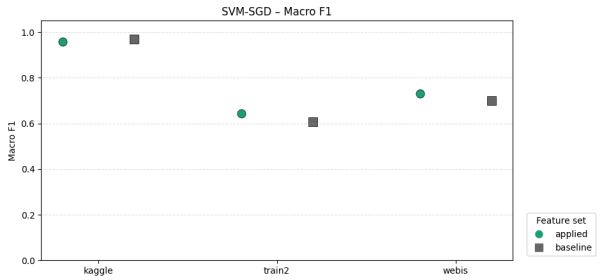

Figure 5: Linear SVM (SGD) macro-F1 scatter segmented by dataset and feature set.

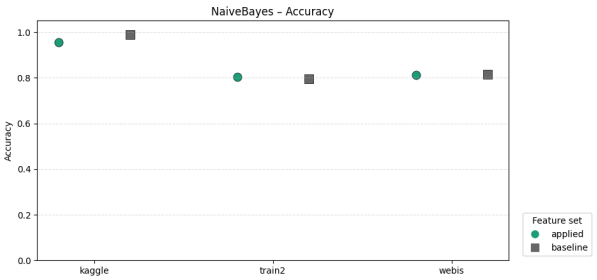

Figure 6: Naive Bayes accuracy scatter segmented by dataset and feature set.

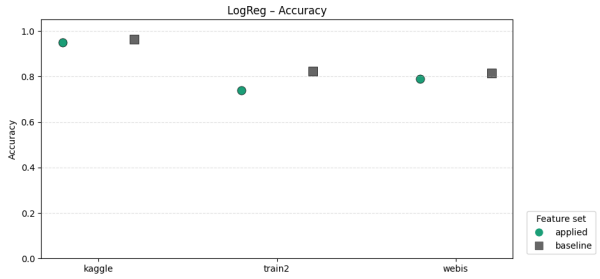

Figure 7: Logistic regression accuracy scatter segmented by dataset and feature set.

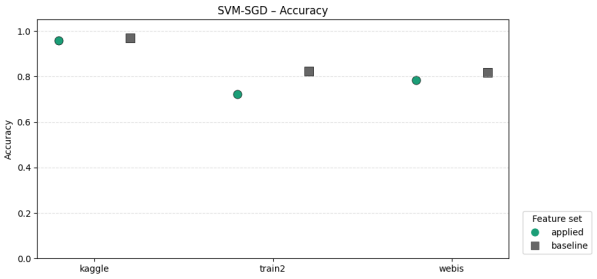

Figure 8: Linear SVM (SGD) accuracy scatter segmented by dataset and feature set.

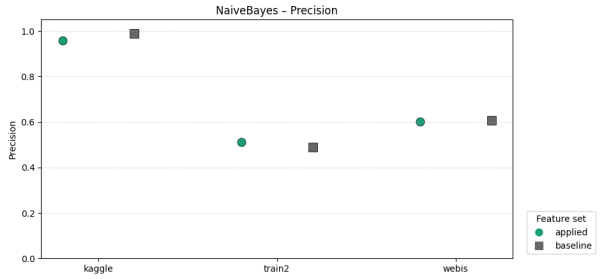

Figure 9: Naive Bayes precision scatter segmented by dataset and feature set.

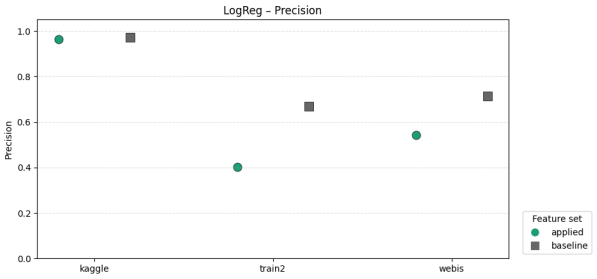

Figure 10: Logistic regression precision scatter segmented by dataset and feature set.

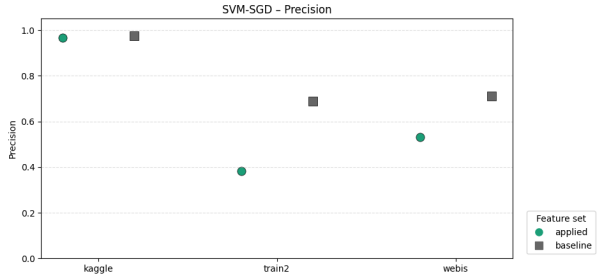

Figure 11: Linear SVM (SGD) precision scatter segmented by dataset and feature set.

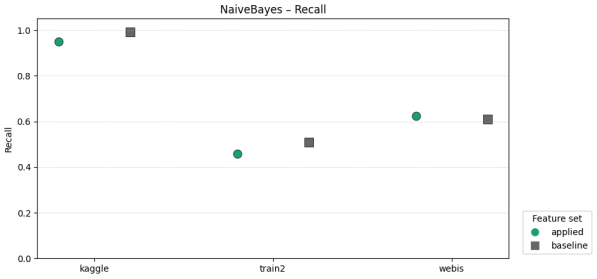

Figure 12: Naive Bayes recall scatter segmented by dataset and feature set.

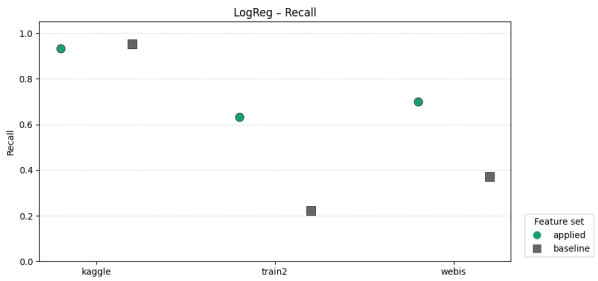

Figure 13: Logistic regression recall scatter segmented by dataset and feature set.

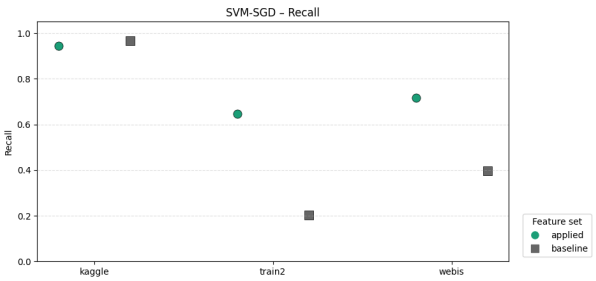

Figure 14: Linear SVM (SGD) recall scatter segmented by dataset and feature set.