Content-Page View Relationship

Mayank Gupta[1,*,†], Shivesh Gupta[2,*,†]

[1]United States of America
[2]United States of America

Abstract

This study examines how webpage content influences page views using data from New York Times articles in August 2013. Leveraging web traffic and textual data, we employ advanced NLP techniques to extract features and analyze their impact on viewership. Our research extends prior work on content virality, focusing on page views. Through constructed features like author popularity and sentiment analysis, we develop a predictive regression model to elucidate content-viewer dynamics.

Keywords

NLP, dense word vector, online content, viewership

1. Introduction

In today's digital economy, both individuals and companies are interested in attracting users to their websites in order to earn ad revenue. While many factors might motivate a user to visit a particular page, certainly one important factor is that webpage's content. This paper explores the relationship between the content of a webpage and the number of page views it receives by constructing a unique dataset of all articles published by the New York Times (NYT) during August 2013. This dataset is built from two major components: the NYT's internal web traffic data and webpage content data parsed from the NYT website.

Typically, a study such as ours tends to be very difficult to conduct. Accurate viewership measurements are either private or unavailable[1]. Even in cases where page views are publicly, for example Youtube or various other video sharing sites, feature extraction of the content is far too challenging given the tools available to us today[2]. Fortunately, our access to the NYT's internal web traffic data allows us to exactly measure the number of page views an article receives. The web traffic data is rather rich and also includes internal meta-data that we use to build various control features. Moreover, since we are working with mostly textual data, we are able to take advantage of recent advancements in machine learning and statistical NLP to do feature extraction on article text.

A similar study by Berger and Milkman (2012) [1] examines the relationship between content and word-of-mouth virality. They find that the emotional content of an NYT article is predictive of its virality. Using simple measures of an article's sentiment and emotionality, Berger and Milkman show that positive articles are more likely to show up on the New York Times "Most-Emailed" list. They also show that articles that evoke high physiological positive or negative arousal (such as awe or anger) tend to be more viral than articles that evoke deactivating emotions (sadness). We build on this study in two ways: first, we relate the content of an article to the number of page views it receives rather than its virality[3]. Second, we employ more sophisticated machine learning feature extraction techniques than those used by Berger and Milkman.

Proceedings of RTA-CSIT, May 2025, Tirana, Albania
*Corresponding author.
†These authors contributed equally.
0009-0003-7400-9542 (M. Gupta); 0009-0007-9309-2010 (S. Gupta)

[1]While precise viewership data tends to be not available openly, oftentimes researchers use related observables, such as Facebook likes, or number of Tweets and Retweets
[2]Though this is quickly changing with advances in computing power and development of better and better machine learning methods
[3]Which is arguably more important to companies, since word-of-mouth virality is usually a means to increase page views

Recent Literature in Viewership Prediction

Since Berger and Milkman (2012), the landscape of content virality research has evolved significantly. Recent studies have used deep neural networks, transformer-based language models, and cross-platform analytics to better capture user engagement. For instance, Zhao et al. (2019) used BERT embeddings to predict article sharing likelihoods, and Sharma et al. (2021) leveraged attention-based models for viewership forecasting. These studies underscore the growing power of semantic understanding in predicting content popularity.

2. Data

2.1. NYT Internal Web Traffic Data

Our NYT internal web traffic dataset is a record of all individual user activity on the NYT website covering the period of April 3 to October 31, 2013. Each time a user[4] moves from one page to another on the NYT website, this activity is captured as an individual JSON object. After cleaning up the url data to ensure each url mapped uniquely to a particular piece of content, we were left with a total of 6,682[5] URLs. We then parse all the web data for the month of August and the first week of September, counting the number of impressions each URL received. In order to make an apples-to-apples comparison between articles, we only count the number of page views received in the 7 days immediately following publication, since an article that has been out longer should have more page views in expectation. Given the tendency for the viewership of an article to drop off sharply soon after publication (as recency is an important factor in news readership), our 7-day measure generally represents the vast majority (well above 90%) of total page views that an article receives[6]. Even after all this subsampling, our data still consists of 248,161,455 page views[7]. The distribution of page views is highly skewed with very heavy tails. After applying a log transformation (as seen in Figure 1), our distribution looks considerably more normal.

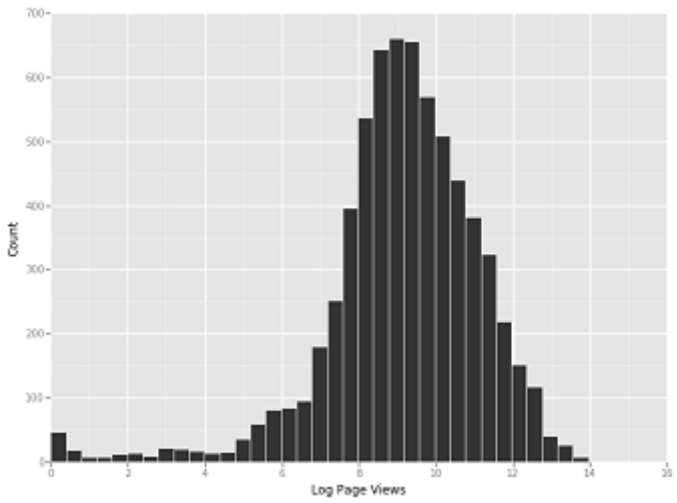

Figure 1: Histogram of Articles by Log of Page Views

[4]In this case, a “user" is uniquely defined by a device/browser id. So, while the same person might have multiple devices or may use multiple browsers, the NYT backend treats each device/browser combination as a unique “user" even though in reality its all the same person. In some cases, we are able to link various id’s together if the person happens to register an official user account on the NYT website and then logs into her account from multiple devices/browsers.

[5]If we had just included a few more URLs, we could have had 6,867 observations!

[6]at least for a reasonable stretch of time

[7]Though one video by PSY completely crushes this number

2.2. Parsed NYT Webpage Content Data

Unfortunately, the NYT internal web traffic data does not contain the actual content displayed on each webpage, which is a very important aspect of our project. Luckily, all this content is freely hosted on the NYT website! In addition to extracting the raw text data, we checked for the presence of additional non-textual content such as pictures or videos in each articles HTML content. We created indicator variables that denote the presence of such content within an article.

3. Constructed Features

Using our collected NYT article data, as well as some additional data from secondary sources, we construct the features that will be fed into our predictive regression model. These features include the Flesch reading ease (Figure 2), the estimated gender of the author(s), the popularity of the author(s), variables indicating the section the article appeared in and the article's content type, the sentiment of the article text, and the perplexity of the article text. We provide a full list of these features below, as well as the methodology used to extract them. Where appropriate, we include discussion of testing and validation of our features and our algorithms.

One can conceive a few competing hypotheses that relate the readership of content to the ease with which people can read it. Maybe more complicated pieces of text are more engaging, and are more likely to be read. On the other hand, perhaps pieces of text that are easier to read will be consumed by more people. In order to capture relationships such as these in our data, we calculate the Flesch reading ease. The Flesch reading ease is a metric developed by Flesch in 1948 [2]. The score indicates how difficult a piece of English text is to understand. Lower scores correspond to more difficult passages, with 120.0 being the highest attainable score. The formula for calculating a passage's Flesch reading ease is

$$206.835 - 1.015\left(\frac{\#\text{ words}}{\#\text{ sentences}}\right) - 84.6\left(\frac{\#\text{ syllables}}{\#\text{ words}}\right) \tag{1}$$

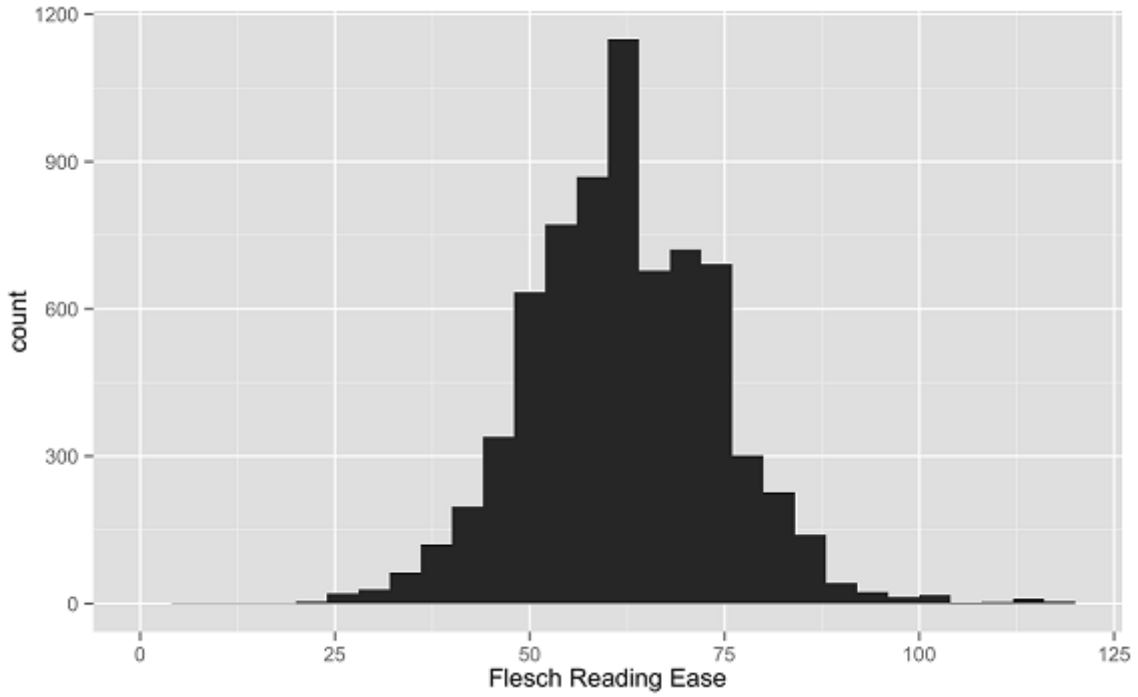

Figure 2: Histogram of Articles by Flesch Reading Ease

We also want to include some measure of a particular author's popularity. It stands to reason that a new article by Paul Krugman or A.O. Scott should garner more readership than a blog post by an unknown graduate student enrolled in 6.867 at MIT!

In order to measure something that will serve as a decent proxy for popularity, we programmatically searched for every distinct author in our dataset on Bing and recorded the number of search results that were returned by the query. In cases where a particular article has more than one distinct author, we calculate an "effective" popularity by simply averaging number of search results over all article authors. The distribution of log(number of Bing search results) is found in Figure 3.

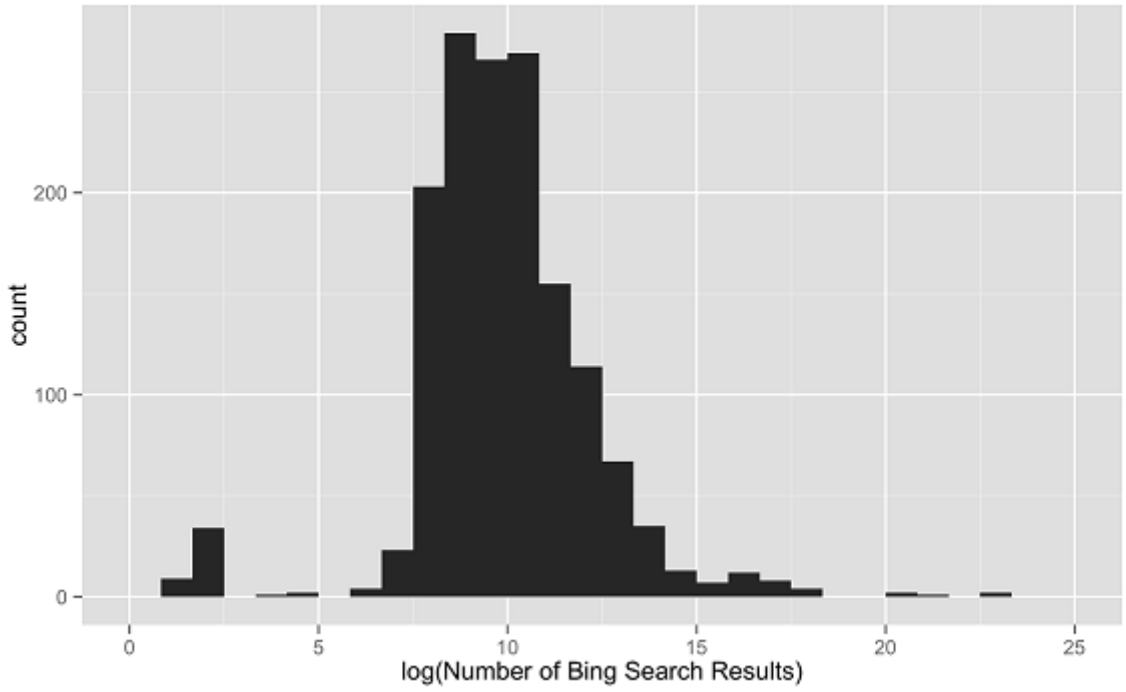

Figure 3: Histogram of Articles by log(Bing Search Results)

While we certainly don't think that an author's gender has a causal impact on the readership on an article, we believe that this feature allows us to control for some latent unobserved heterogeneity. For each (set of) author(s), we record the most likely gender of the author. In cases where the gender of the author is unclear (e.g., Robin) or there are likely multiple authors with different genders (e.g., The New York Times Staff), we record a third gender value, "ambiguous / unknown." Our gender data is gathered by cross-referencing the first names of all of the authors in our dataset against U.S. Social Security Administration baby name data from 1935 to 1997.

We also include a number of dummy variables, indicating the material type (e.g., 'News' or 'Obituary'), publishing desk (e.g., 'Weekend' or 'Real Estate'), article type ('Blog post' or 'Article'), section (e.g., 'Movies' or 'World'), and the day of week and time of day that the article was published. The hypothesis driving the decision to include these variables is that certain types of content (e.g., political news or international affairs) may be more widely read than local material (such as real estate) or less popular sections of the NY Times (e.g., the sports section). We also suspect that publishing an article on certain days of the week (for example, weekends) or at particular times of day (such as lunch hour) may correspond to higher levels of readership.

We also build features that attempt to capture the article sentiment and the article text perplexity. Since the design and computation of these features was considerably more complex and our algorithms required some amount of validation, we discuss these two features in separate subsections.

3.1. Article Sentiment

In order to measure article sentiment, we use a Naives Bayes text classification algorithm, as described in Rennie et al (2003) [3]. We assume that each article in our corpus can belong to one of three classes - 'negative' sentiment, 'neutral' sentiment, or 'positive' sentiment, which we will denote as C_k. The Naive Bayes model assumes that the likelihood of observing a given article $\mathbf{x} = (x_1, ..., x_n)$, where x_i is the number of times that word i appears in the article, is

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}, \tag{2}$$

where p_{ki} is the probability of word w_i conditional on a document belonging to class K. Applying a log transformation to this expression, we can compute $\log(p(\mathbf{x}|C_k))$ as:

$$\log(p(\mathbf{x}|C_k)) = \log(p(C_k)) + \sum_{i=1}^{n} x_i \cdot \log(p_{ki}). \tag{3}$$

to classify a given article, we simply compute $p(\mathbf{x}|C_k))$ for each class, and select the class with the highest log-likelihood.

We coded up a basic implementation of the Naive Bayes algorithm, drawing heavy inspiration from Greg Lamp's 2014 python tutorial on Naive Bayes [4]. In order to get the probabilities $p(C_k)$ and p_{ki}, we needed some labeled training data. In order to obtain these labels, we selected a random subset of 200 articles from our dataset and created a task on Amazon Mechanical Turk. Each Turker was asked to score the sentiment toward the subject of the article in question. Scoring was done on a scale from -2 to +2, with -2 being extremely negative and +2 being extremely positive. In order to make sure these scores were relatively robust, we recorded 5 scores for every article from 5 different Turkers and calculated the average sentiment score. We classified any article having an average score greater than 0.5 as 'positive'. Any article with an average sentiment less than -0.5 was classified as 'negative.' Any other articles were classified as 'neutral.' Ultimately, our labels were 66% neutral, 14.5% negative, and 19.5% positive. This is unsurprising, as a newspaper such as the New York Times likely strives for neutrality when reporting on most topics.

We wanted to measure how our Naive Bayes implementation did compared to an off-the-shelf implementation of the same algorithm. In order to do so, we trained NLTK's multinomial Naive Bayes classifier [5] on the same training data, and then compared the predicted sentiment between the two articles on a 1,000 article subset of our data.

Overall, we find 89.4% agreement between the two algorithms. Alarmingly, however, the NLTK implementation seems to predict neutral an overwhelming percentage of the time (99.8%). This warrants further investigation, and may be due to small differences in implementation, or peculiarities in the sample of 1,000 articles we chose to compare the two algorithms. In any case, the predictions of our algorithm are in the same neighborhood as the NLTK implementation and are of comparable, if not better, quality. As a result, we feel relatively comfortable moving forward using our sentiment labels.

3.2. Article Perplexity

In order to determine the perplexity score, we first need to build some language model that gives us the probabilities of each word. While perplexity typically is a measure of how well a probability distribution can predict a sample, in our context, we interpret perplexity essentially as a measure of article "uniqueness". The argument here is that if our language model can't predict the language used in article very well, then the language used in the article is atypical relative to the corpus used to build the language model. Hence, given some language model, an article's perplexity is given by:

$$2^{-n \cdot \sum_{i=1}^{n} \log p(w_i)} \tag{4}$$

where n is the length of the article, and $p(w_i)$ is the probability of the i-th word in the article. We think that perplexity might have some predictive power since people generally have a preference for novelty.

If many news articles about the same story are all using highly similar language, an article that covers the story using atypical language is likely unique in some way or another which may drive people to read it more or less. For this section we generally follow the 6.864 Lecture Notes 2 and 3 [6].

As for our paper, we construct a couple of different language models. First, we build a simple bigram language to use as a baseline. We also build more sophisticated word vector based n-gram neural network language models.

We split our articles into training (70%), validation (15%), and test corpora (15%). In order to keep the size of our vocabulary relatively manageable, we ignore any case sensitivities. Furthermore, we only include a word if it appears at least 5 times. Words that don't make this cutoff are mapped to a generic "rare word" indicator. Lastly, we also map any numbers (that is numbers comprised of digits, not numbers written with words) to a generic "number" indicator. Ultimately, this leaves us with a vocabulary size $|V|$ of 29,359. To estimate a bigram model, we simply need to compute the counts in our training corpus. Specifically, the probability of some word w_i conditional on its preceding word w_{i-1} is given by:

$$p(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}} \tag{5}$$

However, its reasonable to expect that there might be bigrams in the development or test corpora that aren't observed in the training corpus. This is particularly likely given that for our vocabulary size, there are nearly 900 million unique bigrams and our training data contains just over 3 million observations. If this is the case, then any article with a bigram unobserved in the training corpus would be assigned a predicted likelihood of 0. Needless to say, this is very bad. In order to avoid this issue, we apply a technique called add-α smoothing. Add-α smoothing adds α to each cell in the probability table. Hence, after smoothing, no bigram, given a fixed vocabulary V, will ever have 0 probability. This changes our our estimated word probabilities to:

$$p(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i) + \alpha}{\text{count} + \alpha|V|} \tag{6}$$

We use our validation set to determine the optimal value of of α by seeing what value of α minimizes the negative average log-likelihood per word (NALL) of our validation corpus.

One criticism of standard n-gram language models is that they are rather sensitive to the training data. Hence, we build word vector n-gram neural network models to see if we can achieve better performance. By using dense word vectors rather than one-hot encodings to represent words, we're able to capture the underlying similarity of words and their meanings. In particular, we train both a bigram neural network language model and a trigram neural network language model.

We build 2 sets of article perplexity scores. One set is derived from the smoothed bigram model to serve as a base comparison. Our other set is derived from the most recently completed pass of our trigram NN language model.

4. Predictive Regression Model

Using our full set of features, we are now able to perform the regression task we originally had in mind, and attempt to determine how (if at all) content drives viewership. We regress log(article pageviews), $\mathbf{y}$, on our design matrix, Φ, which includes entries for each of our $k-1$ features, plus an intercept term (in this case, k = 102).. We estimate the feature weights using the closed form solution for OLS and ridge regression:

$$\beta = (\Phi^T\Phi + \lambda\mathbf{I})^{-1}\Phi^T\mathbf{y}, \tag{7}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix, and λ is our regularization parameter. Setting $\lambda = 0$ corresponds to OLS, whereas a non-zero value of λ corresponds to ridge regression. The motivation for performing ridge regression as opposed to OLS is to not overfit on our data, and the value of λ can be interpreted as the strength of our Bayesian prior on the feature weights being equal to 0 [7].

In order to choose an appropriate value of λ, we split our data into training, validation, and test sets. 90% of the data is allocated to the training set, 10% to the validation set, and 10% to the test set. Although we estimate β on the training data using the above closed-form solution, we cross-validate on our validation set for each λ, and choose the value of λ that produces the lowest MSE on our validation dataset. To ensure that we have not overfit β to our validation dataset, we also calculate the MSE on the test set as a final step. A comparison of the training, validation, and test MSEs for various values of λ is found in Figure 6.

We find that $\lambda = 100$ minimizes the MSE on our validation set. Table 1 displays the 20 feature weights with the largest magnitudes $\lambda = 100$. Two charts showing weights for the full set of features (excluding the intercept term) can be found in Figures 4 and 5.

Table 1
20 Most Significant Weights

Feature	Weight
intercept	8.964
log(Word Count)	0.741
Desk: Foreign	0.550
Desk: Travel	-0.465
Section: World	-0.402
Section: Opinion	0.283
Type: BlogPost	-0.282
Type of Material: Schedule	-0.267
Desk: None	-0.262
Section: Movies	-0.239
Time of Day: 12-17	0.232
Type of Material: Review	-0.218
Desk: National	0.211
Section: Health	0.210
Section: Books	0.201
Type of Material: Letter	-0.200
Type of Material: News	0.187
Section: Sports	-0.175
Type of Material: Op-Ed	0.170
Desk: BookReview	-0.154

It's worth taking the time to discuss Figure 6, which shows the training and holdout MSE for various values of λ, in slightly more depth. There are a few things in this plot worth discussing. First, note that the validation MSE is consistently higher than the training data MSE, which is consistently higher than the test data MSE. Given the (relatively) small size of our dataset (6,682 observations), this is likely due to the sampling we used to separate our data into training, validation, and test data. However, we don't expect this effect the validity of our cross validation.

Feature Scaling

To ensure meaningful comparisons between feature weights in the regression model, all continuous features—such as log word count, perplexity, reading ease, and author popularity—were standardized (zero mean, unit variance) before modeling. This step is essential when interpreting coefficient magnitudes, as unscaled inputs would bias the results by favoring variables with larger numerical ranges.

Another thing worth noting is that even on the training dataset, there exist non-zero values of λ that achieve a lower MSE than the OLS estimate of β. At first, this was surprising to our group, as conceptually OLS is often thought of as the linear regression method that minimizes MSE. However, it is important to note that OLS only holds this distinction amongst unbiased estimators. Hoerl and Kennard (1970) [8] prove the existence theorem for ridge regression, which claims the existence of some λ such that β_{ridge} produces a lower MSE than β_{OLS}.

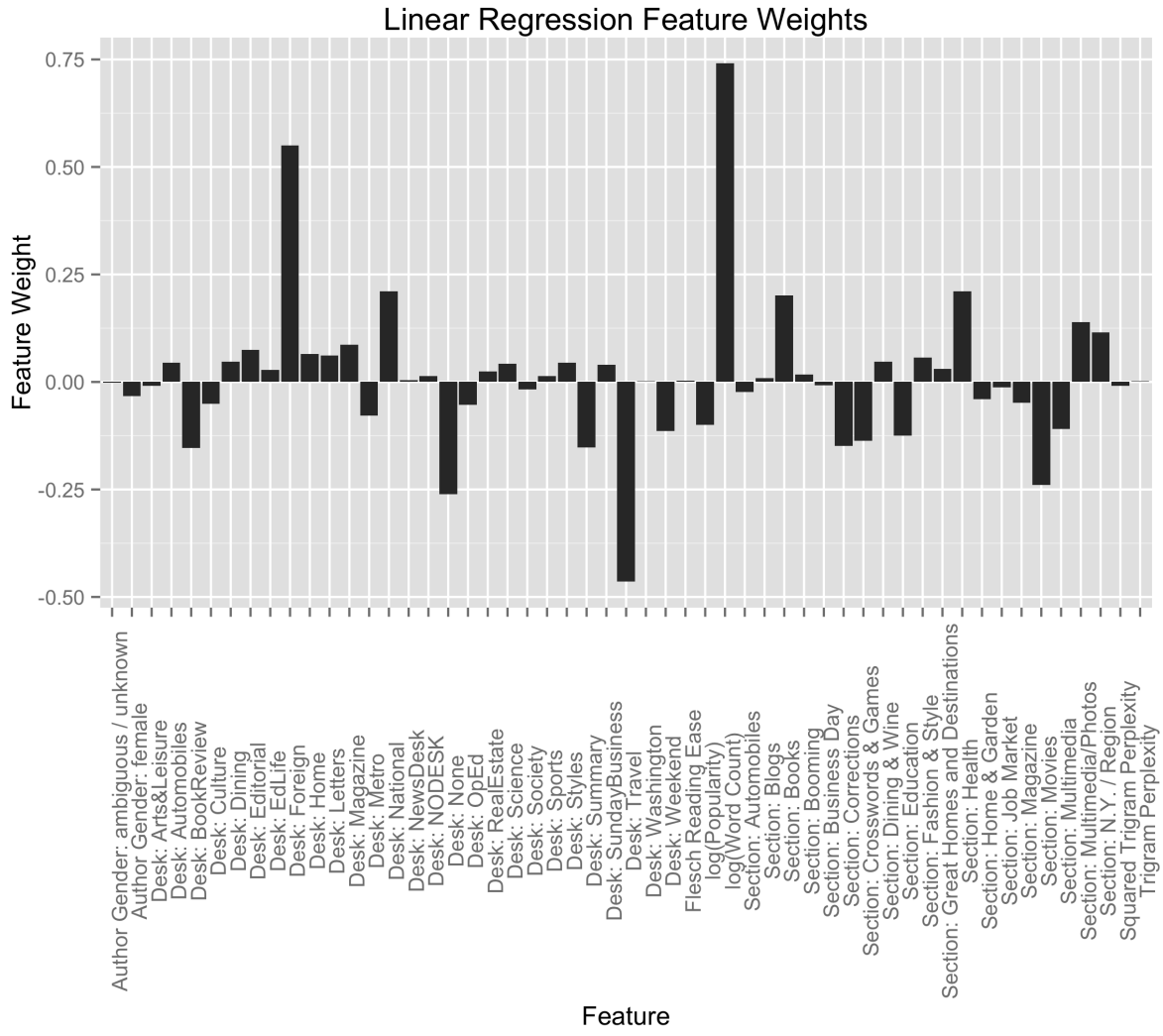

Figure 4: Linear Regression Feature Weights (excluding intercept term) (1 of 2)

Another way of framing this finding is through bias-variance tradeoff. Recall that the MSE can be written as a function of the bias and variance:

$$MSE = (\text{Bias})^2 + \text{Var}. \tag{8}$$

Clarifying the Intercept Term

While the intercept term (8.964) is mathematically the baseline prediction when all features are at reference or zero levels, its interpretation deserves more nuance. This high value likely captures latent factors not modeled explicitly, such as homepage exposure, existing subscriber behavior, or brand authority associated with the New York Times. In other words, even without standout features, articles may receive substantial traffic simply by being hosted on a trusted, high-traffic platform like NYT.
For some values of λ, ridge regression is able to lower the MSE by decreasing variance, but increase the bias from zero to some non-zero value. We believe the changes in MSE we observe in our dataset as we vary λ can be explained by this phenomenon.

In general, we can now interpret the strength of the feature weights produced by our regression to determine how predictive a particular feature is of readership. Note from Table 6 that most of the features with the most predictive power are not the text-based features. The intercept term in our

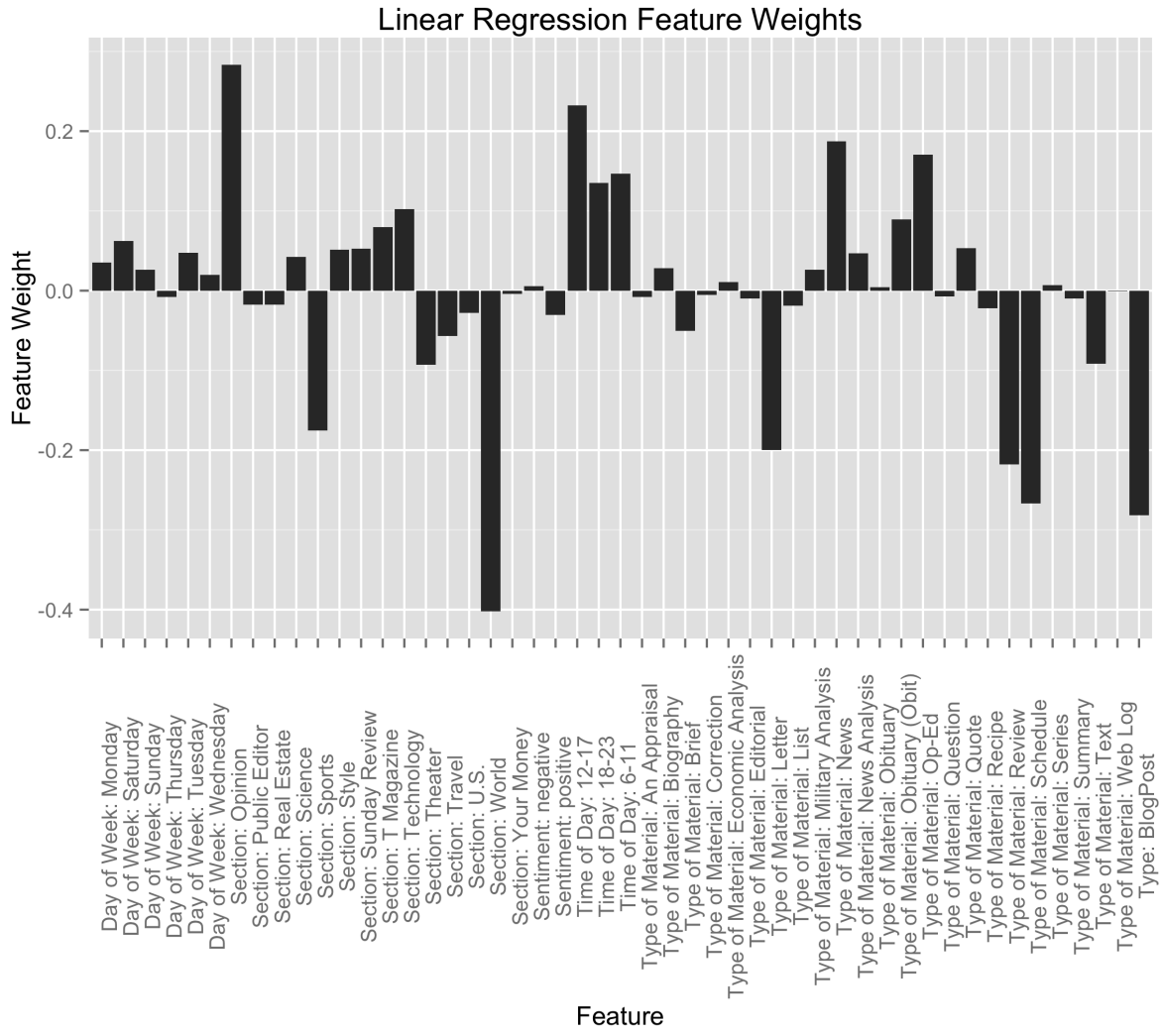

Figure 5: Linear Regression Feature Weights (excluding intercept term) (2 of 2)

regression is orders of magnitude larger than any other feature, implying that most of the NYTimes articles receive many pageviews in the baseline case. The strongest coefficients tend to be those that indicate the publication desk, section, and time of publication of the article. This suggests that the content of an article itself may not be as important as the context in which it is published. The one exception we see is that a higher word count is predictive of higher viewership. We suspect that what's going on here is correlative, rather than causal - the quality of longer pieces (e.g., NYTimes Magazine articles) is likely higher, thus driving more readership. But we don't expect that a website full of low-quality, 5,000 word pieces would be successful.

4.1. The Effect of Textual Features

Given the large amount of work we put into building numerous text-based features (such as the trigram neural network perplexity, the sentiment labels, and the Flesch reading ease) and the relatively low impact they seem to have had on our regression (based on feature weights), we want to specifically evaluate the impact of these features on our regression. Specifically, how much incremental reduction in MSE are we getting by including them? We first re-run the exact same regression specification, but instead of using the trigram perplexity calculated from a neural network, we instead use the simple bigram perplexity discussed earlier in this paper. We find that this actually leads to a reduction in

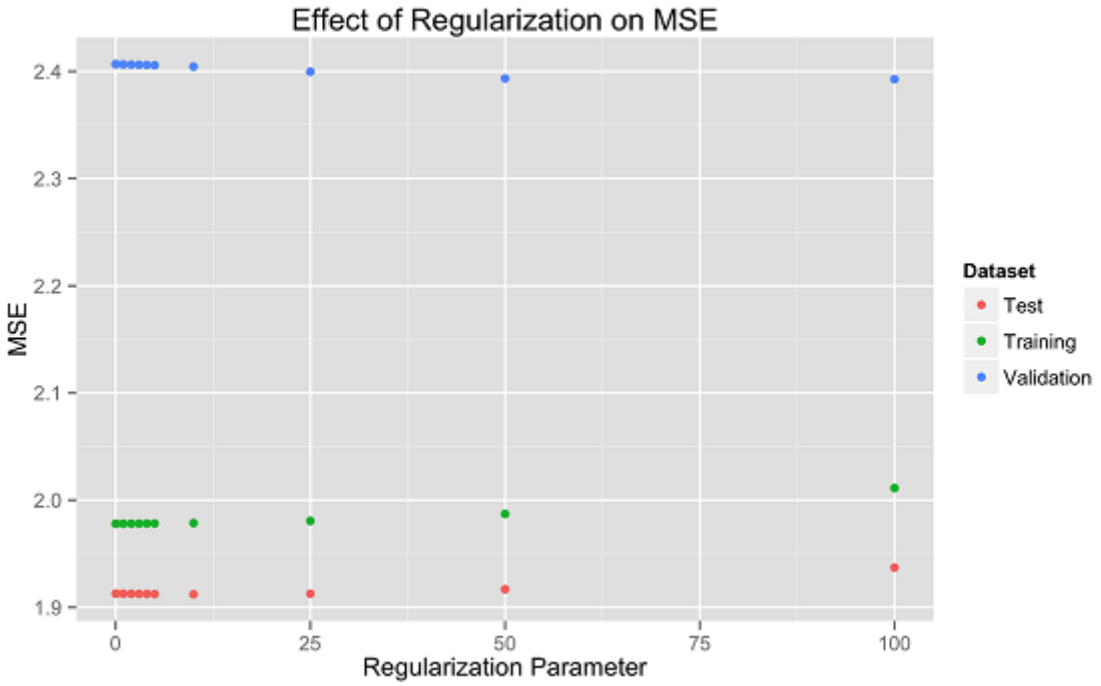

Figure 6: The effect of regularization on MSE

MSE, from 2.392 to 2.389. This small, but modest change suggests that there is currently almost no incremental value from using a neural network language model as opposed to a more straightforward language model.

As a next step, we ask if text features in general add much value to our model. Given that, in general, the magnitude of text feature weights is dwarfed by the magnitude of contextual feature weights, we might expect that text features do not add much. We find that removing the perplexity, sentiment, reading ease, and word count features from our regression leads to an increase in MSE, from 2.392 to 2.502 (Table 2). This result is encouraging, as it suggests that even if text features currently aren't contributing as much as we had hoped, they are doing something! Removing them from our model leads to a 4.6% increase in the MSE.

Table 2
Comparison of MSEs for different model specifications

	NN Trigram	Bigram	No Text
Best Val. MSE	2.392	2.389	2.502
Train. MSE	2.011	2.011	2.274
OLS Val. MSE	2.407	2.405	2.506

5. Future Work

Unfortunately, we were unable to implement all of several important features we initially wanted to include in our regression, due to time constraints and technical difficulties. Most notably, this paper currently lacks some form of topic modeling, which we expect to be a strong predictor of content viewership (at least in our context). We also suspect that the addition of features containing information about article headlines may provide some improvement.

There is also room for improvement in the design and extraction of our existing features. For example,

the perplexity score feature ultimately provided very little predictive power. This may be due to the issues in the quality of the underlying language model.

There is ample room to improve our sentiment analysis methodology. The training dataset generated using mechanical turk is relatively small (200 training articles). Because of this, there may exist many words that have a strong probability of appearing in an article conditional on sentiment that simply do not appear in our training set. With more time (and money!), we could label more data and improve the accuracy of our model. Furthermore, the model currently skips words that do not occur in the training corpus. An extension of our Naive Bayes implementation could use a method such as Laplace smoothing, so as to not simply ignore words we haven't seen in our training data.

In addition, the current discrepancies between our implementation of Naive Bayes and NLTK implementation of the algorithm, while not large in magnitude, are alarming. In the future, we hope to dig deeper into this discrepancy and identify the root cause.

Another avenue for potential improvement to our model is the application of basis expansion to our variables. This would allow us to include polynomial and interaction terms. In many cases, there is no good justification for assuming a feature is linearly related to the output feature. Hence, applying a polynomial expansion might uncover an entirely different relationship between our dependent variable and its covariates.

6. Conclusion

In this study, we investigated the factors influencing the readership of New York Times articles by integrating contextual and text-based features into regression models. Our findings reveal that contextual features, such as publication desk, section, and time of publication, are the most predictive of readership. Text-based features, including perplexity, sentiment, and reading ease, while contributing less overall, still provide incremental predictive power, particularly when combined with contextual features. Among text features, word count had the most significant impact, likely due to its correlation with in-depth, high-quality content.

Ridge regression outperformed ordinary least squares (OLS) by leveraging regularization to address overfitting, highlighting the importance of balancing bias and variance in predictive modeling. However, our experiments showed that despite significant effort in developing sophisticated textual features, such as neural network-based trigram perplexity, these models did not outperform simpler approaches. While Berger and Milkman (2012) emphasized emotional content as a driver of virality, our findings diverge by highlighting the primacy of publication context. This may reflect structural differences between virality (e.g., email shares) and page views (direct readership), or indicate that emotional features alone cannot fully explain user behavior in structured news platforms. Further integration of semantic emotion detection may help reconcile these views.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] J. Berger, K. L. Milkman, What makes online content viral?, Journal of marketing research 49 (2012) 192–205.

[2] R. Flesch, A new readability yardstick., Journal of applied psychology 32 (1948) 221.

[3] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, et al., Tackling the poor assumptions of naive bayes text classifiers, in: ICML, volume 3, Washington DC), 2003, pp. 616–623.

[4] G. Lamp, Naive bayes in python, http://blog.yhathq.com/posts/naive-bayes-in-python.html, ???? Accessed: 2015-12-06.

[5] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, O'Reilly Media, 2009.
[6] R. Barzilay, T. Jaakola, 6.864 lectures 2 and 3 notes (2015).
[7] G. Bresler, T. Hashimoto, Lecture notes on: regularization and bias-variance tradeoffs, 2015.
[8] A. E. Hoerl, R. W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics 12 (1970) 55–67.