Application of Logistic Regression in Sarcasm Detection

Data 410 Final Report

Daniel Krasnov, Keiran Malott, Ross Cooper

2023-04-13

Contents

Introduction **2**
- Data Collection Method . . . 2
- Variable Description . . . 2
- Data Preprocessing . . . 3

Feature Extraction Methods **3**
- TF-IDF . . . 3
- Word2Vec . . . 4
- GloVe . . . 4

Evaluation Metrics **5**

Regression Analysis **6**
- Variable Selection . . . 6
  - TF-IDF . . . 6
  - Word2Vec . . . 7
  - GloVe . . . 7
- Fitting, Evaluations, and Violations . . . 8
  - TF-IDF . . . 8
  - Word2Vec . . . 10
  - GloVe . . . 12
- Other Findings . . . 14

Conclusion **14**
- Model Comparison . . . 14
- Limitations . . . 15
- Future Work . . . 15

References **15**

Introduction

Reddit is an American social news website that hosts discussion boards where users can share, comment and vote on various posts. These posts are housed in subreddits which are communities on Reddit focused on a specific topic.

When writing comments on Reddit, users will often write /s at the end of their post to indicate their comment is sarcastic. This, coupled with Reddit's web scrapping Python API, provides a self labeled dataset of sarcastic comments.

The goal of our analysis will be to use the /s as a binary indicator of a comment being sarcastic and fit a Logistic regression model using various feature extraction methods. We can then explore this model's efficacy and optimize it for prediction.

Data Collection Method

On the subreddit *dataisbeautiful* one user posted the following figure:

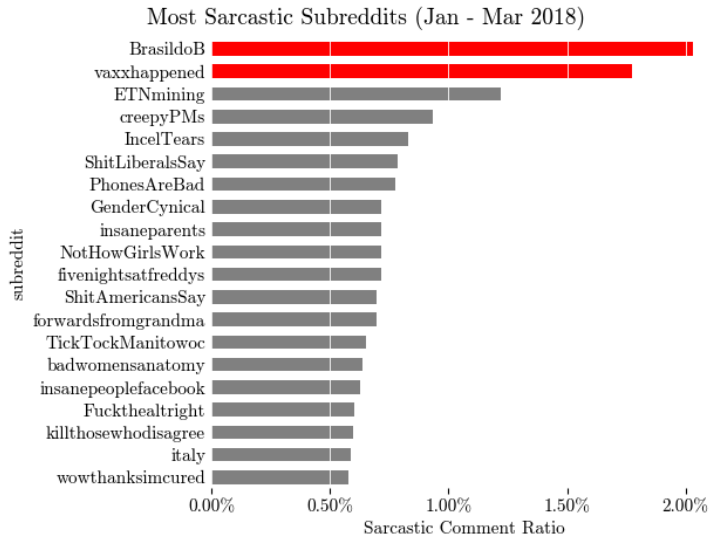

Figure 1

Visualization of the most sarcastic subreddits between January and March of 2018 (Most sarcastic subreddits, 2018).

We began by scrapping the top 10,000 posts from each of the above subreddits. We found that all the subreddits had approximately a 1:100 ratio for sarcastic to non-sarcastic comments. We constructed our first dataset by sampling from all the above subreddits however, we found the data to be too 0 heavy and no model specification could learn an underlying relationship between words and sarcasm. We then attempted to fit models to various ratios of sarcastic to non-sarcastic comments. We found that Logistic regression began to perform reasonably well at a ratio of 1:1 sarcastic to non-sarcastic. We also found that models tended to perform far better if all comments came from a single subreddit as opposed to multiple. As per our preliminary results, we opted for a single subreddit at a ratio of 1:1 sarcastic to non-sarcastic comments. NotHowGirlsWork was found to have the largest count of Sarcastic comments at 321 therefore, we selected this subreddit for our dataset.

Variable Description

Our dataset is constructed as follows:

Variable Name	Data Type
Body	String
Sarcastic	Binary Integer

Figure 2

Table of dataset structure.

Where Body is the raw comment string and Sarcastic is 1 when /s is present in the comment and 0 when it is not.

Data Preprocessing

In *Natural Language Processing* (NLP) there are various text preprocessing steps that are common to employ (De Ville & Bawa, 2021):

- Punctuation, whitespace, and number removal - any punctuation characters such as !, @, #, etc. as well as empty space and numbers are removed.
- Stopword Removal - removal of words that fail to provide much contextual information, e.g., articles such as 'a' or 'the'.
- Stemming - identifying roots in *tokens*, individual words, and truncating them to their root, e.g., fishing and fisher transformed to fish.

In our dataset we first removed the /s from every sarcastic comment and performed the above preprocessing steps.

Feature Extraction Methods

In order to use text as data in a Logistic regression, we must numerically encode our strings. There are a plethora of feature extraction methods in NLP. For our analysis we compare TF-IDF, Word2Vec, and GloVe.

TF-IDF

Term frequency inverse document frequency (TFIDF) is a heuristic to identify term importance (Silge & Robinson, 2017). It calculate the frequency with which a term appears and adjusts it for its rarity. Rare terms are given increased values and common terms are given decreased values (De Ville & Bawa, 2021).

TFIDF is given by

$$\text{TFIDF}(t) = \text{TF}(t) \times \text{IDF}(t)$$

where

$$\text{TF}(t) = \frac{\text{\# of times term t appears in a document}}{\text{\# of terms in the document}}$$

and

$$\text{IDF}(t) = \ln\left(\frac{\text{\# of documents}}{\text{\# of documents where t appears}}\right)$$

In our analysis a document is a Reddit comment. After being preprocessed, the text of each comment is separated into tokens and has its TFIDF calculated. From there the TFIDF values are placed in a *Document Term Matrix* (DTM). This matrix has document ids as rows and tokens as columns. It is therefore a sparse matrix where entries are the TFIDF scores for corresponding tokens.

The DTM acts as the design matrix for our Logistic Regression model:

	husband	lost	potenti	surviv	two		will	common	famous
5	0	0	0	0	0	0.0000000	0	0	
6	0	0	0	0	0	0.2352558	0	0	
7	0	0	0	0	0	0.0000000	0	0	
8	0	0	0	0	0	0.0000000	0	0	
9	0	0	0	0	0	0.0000000	0	0	
10	0	0	0	0	0	0.0000000	0	0	

Figure 3 A truncated example a TFIDF DTM.

Word2Vec

Word2Vec is a group of predictive models for learning vector representations of words from raw text. Word2Vec uses either the *continuous Bag-of-Words architecture* (CBOW) or the *continuous Skip-Gram architecture* (Skip-Gram) to compute the continuous vector representation of words. Both CBOW and Skip-Gram use shallow neural networks to achieve this, but CBOW predicts words based on the context and Skip-Gram predicts surrounding words given the current word (Mikolov, 2013).

Each word is represented as a vector, and words that share common context are close together in vector space (Wei, 2018). Document vectors are representations of documents (Reddit comments) in vector space. A document vector can be constructed by summing the the word vectors from a common document and then standardizing them (Wijffels, 2021). The design matrix for logistic regression can be constructed with the rows of the matrix as the document vectors. The resulting design matrix therefore has one row per Reddit comment and is as follows:

-0.1090544	-0.3175069	0.0737101	0.2276453	-0.1243591
-0.0836128	-0.3167736	0.0872730	0.2301574	-0.1224159
-0.0933614	-0.3087860	0.0640571	0.2600178	-0.1476609
-0.0967188	-0.3358471	0.0920563	0.2071156	-0.0971877
-0.0937867	-0.3155456	0.0770679	0.2261149	-0.1111886
-0.0579854	-0.3277485	0.1235739	0.2106898	-0.1247984

Figure 4 A truncated example of a Word2Vec design matrix

GloVe

Global vectors for word representation (GloVe) is an unsupervised learning algorithm which creates a vector representation for words by aggregating word co-occurrences from a corpus. The resulting co-occurrence matrix X contains elements X_{ij} representing how often word i appears in the context of word j (Selivanov, Bickel, & Wang, 2020).

Next, soft constraints for each word pair are defined by:

$$w_i^T w_j + b_i + b_j = log(X_{ij})$$

where w_i is the vector for the main word, w_j is the vector for the context word j, and b_i and b_j are scalar biases for the main and context words. Finally, a cost function is defined:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log\ X_{ij})^2$$

Here f is a weighting function chosen by the GloVe authors to prevent solely learning on extremely common word pairs (Selivanov et al., 2020):

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_max}\right)\alpha & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases}$$

To create the design matrix below, a vocabulary of the words in the corpus was created. Since this method creates a co-occurrence matrix, we prune all words which appear less than five times to reduce bias from less common words (Selivanov et al., 2020). From there we constructed a term-co-occurrence matrix and factorized it via the GloVe algorithm. The resulting matrix consists of word vectors as rows, which are added together to create sentence vectors that are used to train the model:

	sarcasm	X2	X3	X4	X5	X6
say	1	-0.3375972	-0.3314272	0.3183125	-0.8590263	-0.7803252
dont	0	0.3233637	-1.9377928	-3.3353427	-1.9327384	1.1647931
guy	1	0.5388376	1.7035687	1.4645455	-0.4911736	1.9058781
can	0	-0.4656894	-0.6457838	0.2223314	-0.4875888	-0.4863418
enough	1	0.5373132	0.3530400	-2.3006922	-1.3824662	0.3701053
dont.1	1	-0.2615503	1.0240006	-0.6365101	-1.2134062	-0.6013752

Figure 5 A truncated example of a GloVe design matrix

Evaluation Metrics

Model performance is assessed based on classification performance. In *sentiment analysis* the most common metrics to tune model for performance are Precision, Recall, and F1 Score (Rachman, 2020). Given a confusion matrix:

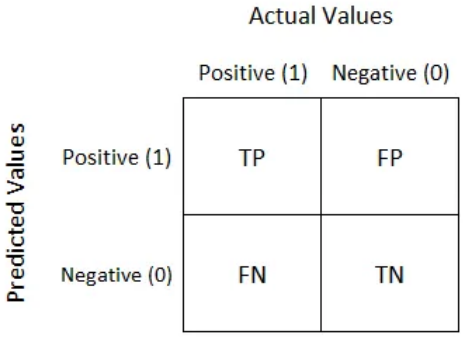

Figure 6 Example of a confusion matrix. TP stands for true positive, FP stands for false positive, FN stands for false negative, and TN stands for true negative.

Precision, Recall, and F1 Score are given by (Kyriakides & Margaritis, 2019):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}} \qquad \text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Recall is a measure of how well we are doing at predicting the true positive class. For our analysis a true positive is correctly predicting a comment is sarcastic. Precision will measure how many positives we classify correctly out of the true and false positives. A good model will be one that achieves a balance of these two

metrics. The F1 score is the harmonic mean of Precision and Recall and so we will look to maximize the F1 Score while maintaining a balanced Precision and Recall.

Regression Analysis

This analysis is a comparison of logistic regression model performance when using 3 types of feature extraction. For each feature extraction we fit a base model. We then perform *Principal Component Analysis* (PCA) to reduce dimensionality and deal with multicollinearity. Finally, we investigate *LASSO* as a means for variable selection.

After model fitting we perform weighting on all 3 model types and decide a best model for each feature extraction method. Weighting is done by multiplying each predictor by w if a comment is sarcastic and by $1 - w$ if a comment is not sarcastic. This is done for every w starting from 0.01 to 0.99 in increments of 0.01. We record testing and training metrics foe each model and discuss our optimal selection.

We hypothesis that models using GloVe as the feature extraction method will perform similarly to Word2Vec and TF-IDF will perform the worst. TF-IDF essentially numerically encodes text based on rarity. We think this is too simple an approach to extract important sarcastic words. Word2Vec captures context and GloVe interprets word co-occurrences which we believe could both be suitable strategies to capture sarcastic structure in text.

Variable Selection

TF-IDF

After performing text preprocessing and TFIDF calculations, the resulting DTM was 642×2074. This matrix has far too many columns compared to rows and so, some dimensionality reduction was required. One way to do so is to filter away unimportant terms. This can be decided by the percentiles of the TFIDFs. For a given percentile we can exclude columns of the DTM based on whether or not their values fall within that percentile. To decide what percentile to use as our threshold we fit various models at every 5th percentile starting at the 5th percentile and ending at the 95th percentile. We then record the AIC and opt to keep the DTM that produced the lowest AIC model.

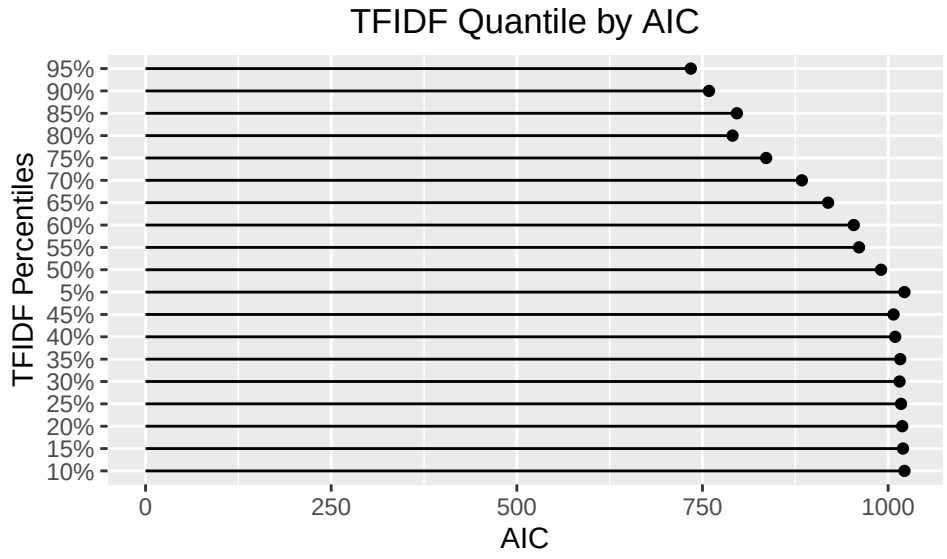

Figure 7 Plot showing TFIDF Percentiles on the y-axis and the corresponding AIC received from fitting a model to a DTM that excluded all TFIDFs below that percentile

The model corresponding to the lowest AIC had a DTM filtered to disclude TFIDF values below the 95th percentile resulting in a 512×74 matrix.

Word2Vec

Before fitting the base model or creating the design matrix, Word2Vec requires the user to specify the dimensions of the word vectors to be created and whether the CBOW or Skip-gram architecture should be used. The Skip-gram architecture has been shown to have higher semantic accuracy than the CBOW architecture (Mikolov, 2013) so we believe it will preform better for predicting sarcasm, and have chosen to use it over the CBOW architecture when fitting the Word2Vec model. To decide on an appropriate vector dimension, the Word2Vec model was fitted with a range of dimensions from 1 to 200 in increments of 1. For each dimension of the Word2Vec model fitted, a logistic regression was fitted on that model and the testing F1 score was calculated. However, the testing F1 score of the model varies between the same vector dimension because Skip-Gram determines the vector representation of the text stochastically. This means the predictors used for logistic regression and any evaluation metric, such as testing F1 score, are stochastically determined too. Therefore, to determine the optimal vector dimension, each dimension was fitted 5 times and a graph of the testing F1 score vs Word2Vec vector dimensions was produced (Figure 8). We have chosen Testing F1 score to determine the optimal vector dimension because it is the primary metric used to evaluate the predictive performance of our fitted models.

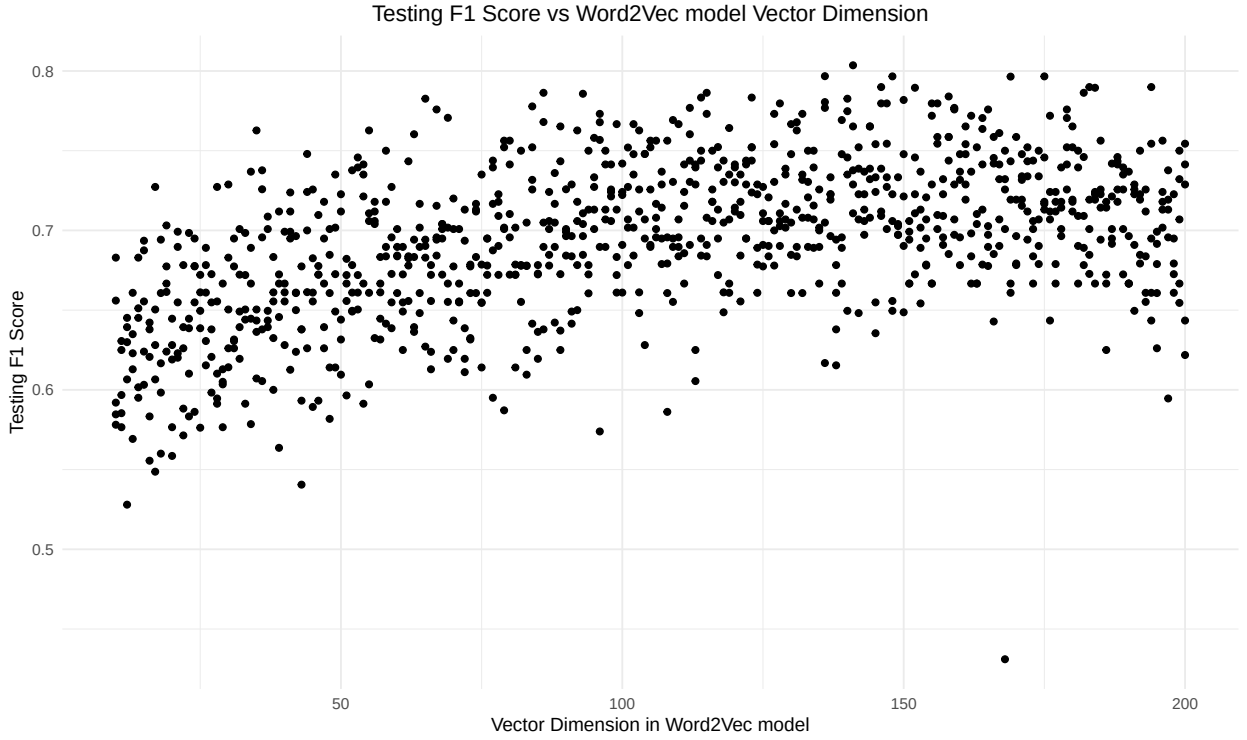

Figure 8 Plot show Testing F1 Score on the y-axis and Vector Dimension used in the Word2Vec model on the x-axis.

GloVe

Before fitting word vectors and creating the design matrix, we must select a dimension for word vectors. To do this we ran a for-loop to create models with vector dimensions of 1 up to 200. In each loop, we created a model using sentence vectors of that length for 5 iterations. Since the GloVe algorithm is an unsupervised model, each creation of the word vectors will be different. From there, we checked the F1 score given by

the testing set and graphed them against vector dimensions. From there we were able to determine that a vector length of 150 is the best length for word vectors in this dataset. However, this graph shows a lot of variability indicating that the best dimension relies heavily on the model generated.

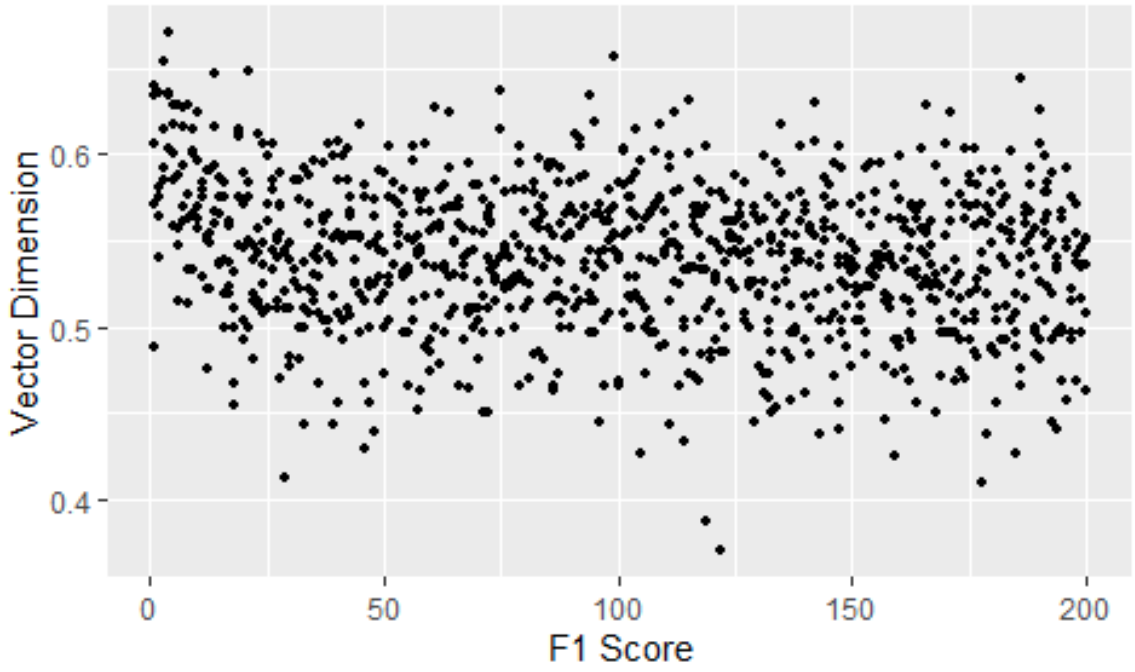

Figure 9 Plot showing dimension selection for the GloVe model.

Fitting, Evaluations, and Violations

TF-IDF

For the base model we take the minimum AIC model found during variable selection. This model has a 14% reduction in deviance and an AIC of 734. R fails to estimate several predictor coefficients and outputs them of NA. This suggests investigation of multicollinearity is needed. After examining the model's VIFs and the correlation between predictors it is was found that 6 variables have VIfs that are over 10 and several predictors are perfectly correlated. To deal with multicollinearity PCA and LASSO are explored.

For PCA we kept a cumulative proportion of up to 90% which resulted in using 24 principal components. Fitting our model to the data the AIC was 17135 and the deviance increase by a factor of 24. Clearly the model does not fit the data well. However, no VIFs were found to be over 10 so the multicollinearity was removed.

Next we employed cross validation to fit a LASSO model. An optimal λ of 0.025 was selected and the resulting model produced a 34.55% reduction in deviance.

We now move onto optimal weight selection. We seek to achieve the best balance of Precision, Recall, and F1 Score.

Figure 10 Plot showing weightings used in model fitting and the resulting test metrics for TFIDF models.

Examining the above graphs the best weightings are 0.75 for the base model, 0.62 for the PCA modek, and 0.55 for the LASSO model.

With model selection finished we may now compare all models and pick the best TFIDF model.

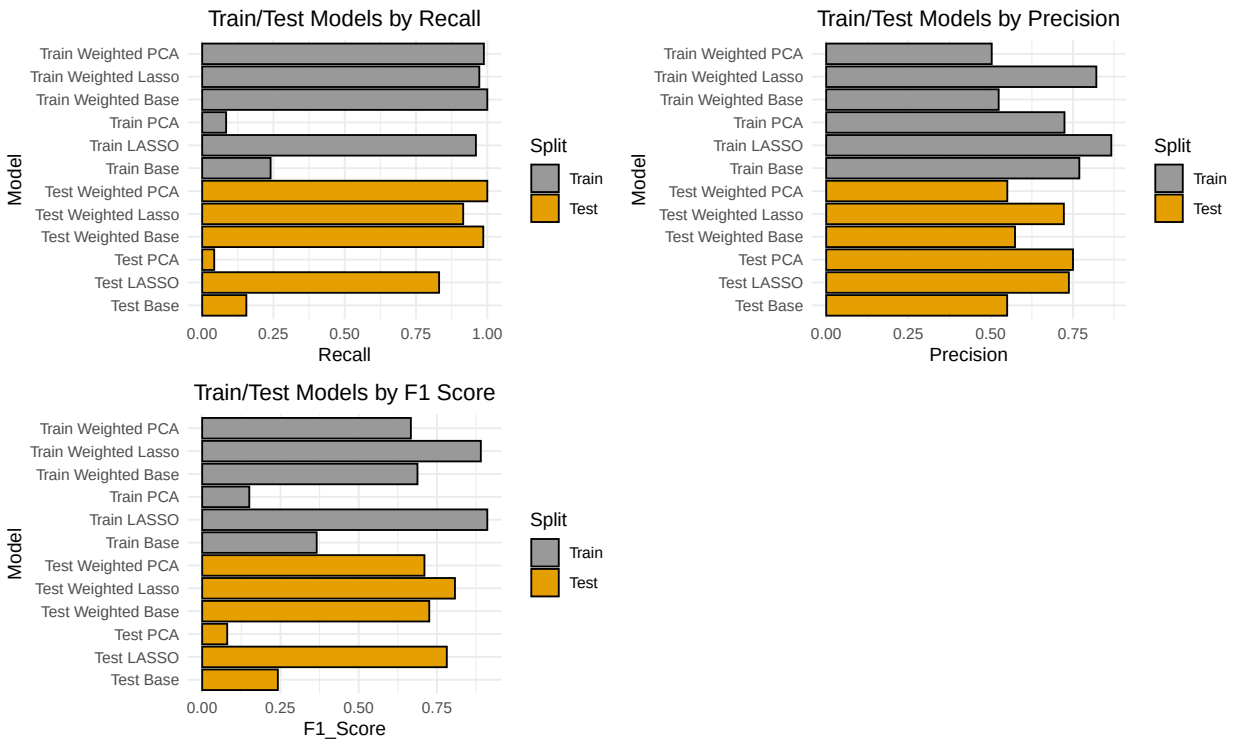

Figure 11 Bar graphs showing train and testing metrics for all models using TFIDF feature extraction.

Weighted LASSO is the superior model. While it has lower Recall than Weighted PCA and Base, it does the best in F1 Score which indicates it is the most balanced model. Both Weighted PCA and Weighted base have a poor Precision and F1 Scores.

Word2Vec

First, a baseline logistic regression model was fitted from the Word2Vec model using the Skip-Gram architecture and a vector dimension of 140. This baseline model showed a significant percent decrease in deviance over the null model at 52.03%. However, of the 140 variables used to fit the model 118 had VIFs greater than 10, indicating serious multicollinearity issues within the model. Next, PCA was used to reduce the dimensionality and multicollinearity of the baseline model. We opted to select the first 55 principal components of the model to be used for logistic regression, which accounted for 90% of the variation in the model. The PCA model showed a lesser percentage decrease in deviance than the baseline model at 26.46%, but this decrease is still significant. Although, the fit of this model was slightly worse than the baseline, none of the variables used to fit the PCA model had VIF greater than 10, indicating the multicollinearity issues in the baseline model have been addressed. We opted to use LASSO as an alternative method to reduce the dimensionality of the model while hopefully retaining or improving the performance of the baseline model. This involved first using 5 fold cross-validation to select an optimal shrinkage parameter (λ) for LASSO, and then fitting a logistic regression with the optimal λ found. LASSO resulted in 108 of the variables being shrunk to zero. The fitted LASSO model had a 16.11% decrease in deviance. We thought the prediction of the models could be further improved by weighting each of the 3 previously fitted models. The optimal weights will be found for each model by examining the graph of w vs Testing F1 Score, Precision, and Recall and selecting the w that achieves a balance of F1 Score, Precision, and Recall.

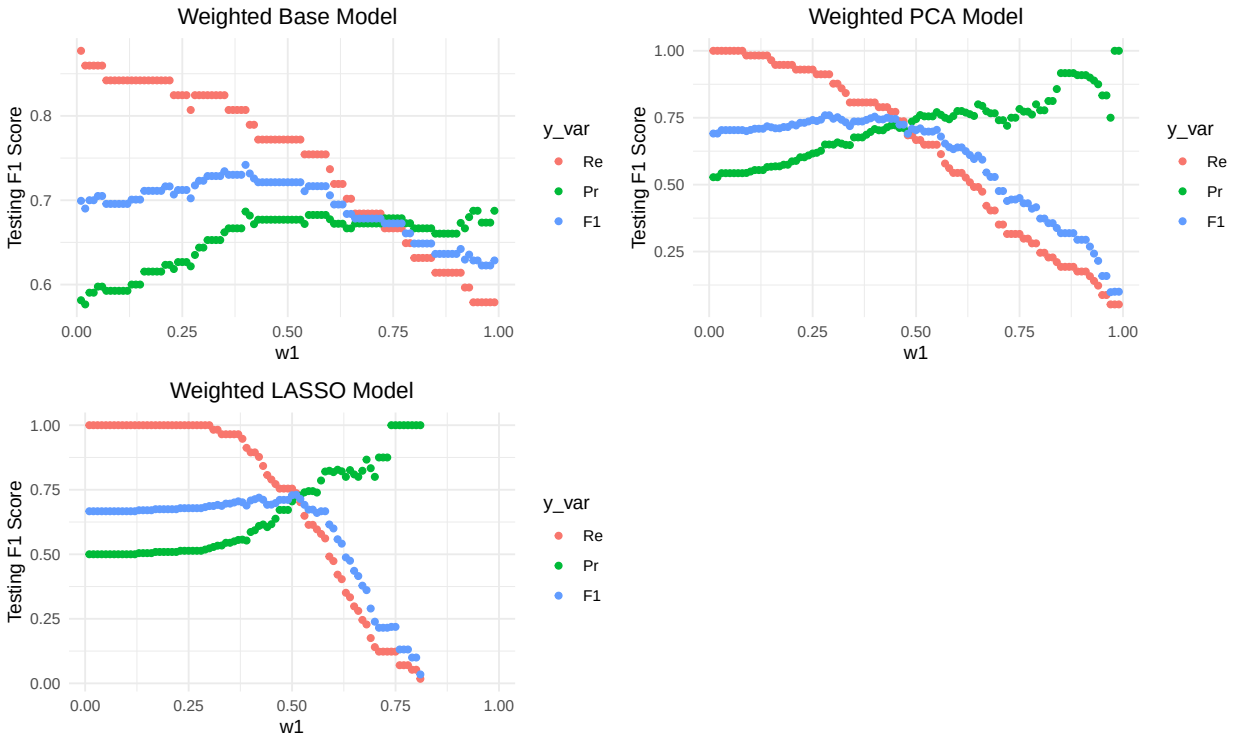

Figure 12 Plot showing weightings used in model fitting and the resulting test metrics for Word2Vec models.

Starting with the baseline model, the optimal weights were found to be 0.4 for the sarcastic class and 0.6 for the non-sarcastic class by examining Figure 12. This model has a significant decrease in deviance at 51.6% however it suffered from the same multicollinearity issues of the unweighted baseline model with 118 variables having VIFs greater than 10. For the weighted PCA model, the optimal weights were found to be 0.28 for the sarcastic class and 0.72 for the non-sarcastic class by examining Figure 12. The weighted PCA model has a 25.38% decrease in deviance which is less than the baseline model. Similar to the unweighted PCA model the weighted PCA model had no variables with VIFs greater than 10. For the weighted LASSO model, the same optimal lambda parameter from the unweighted model was used and optimal weights were found to be 0.51 for the sarcastic class and 0.49 for the non-sarcastic class by examining Figure 12. The weighted LASSO model had a 16.12% decrease in deviance. The testing and training metrics for all 6 models are presented in Figure 12. The weighted PCA model preformed the best prediction of sarcastic comments as evidenced by it having the highest testing F1 score.

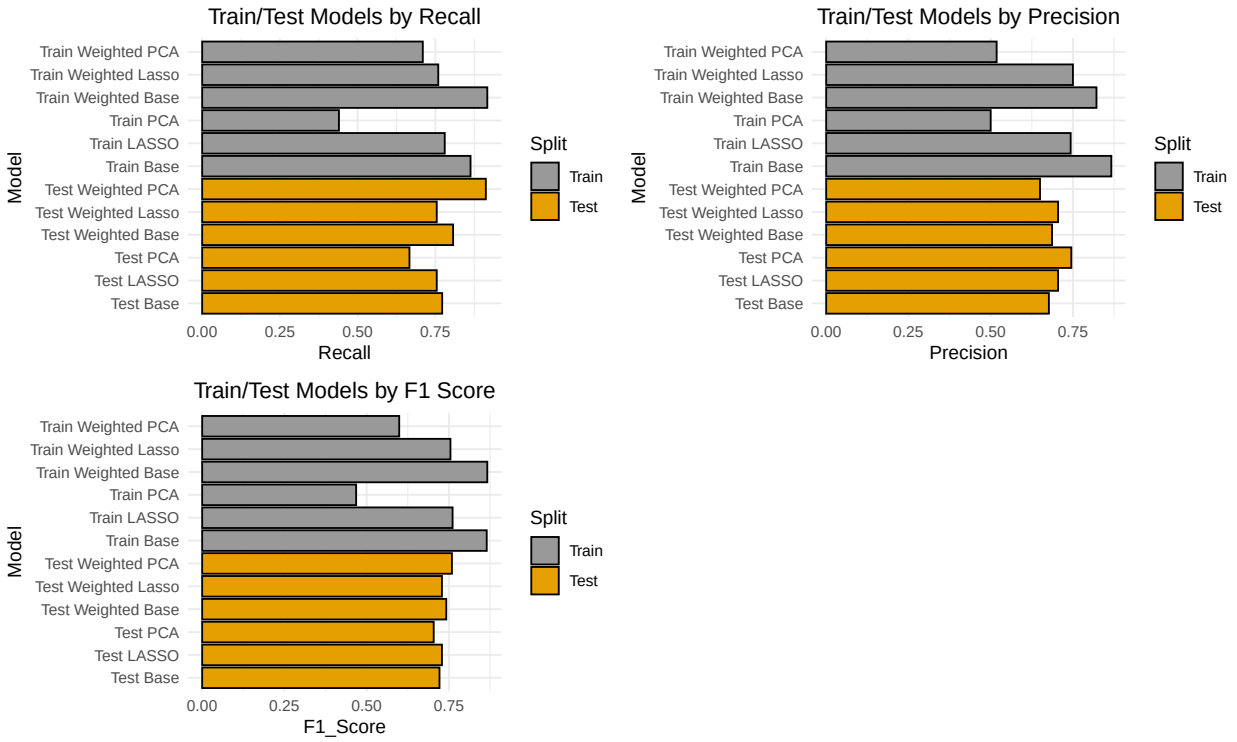

Figure 13 Bar graphs showing train and testing metrics for all models using Word2Vec feature extraction.

GloVe

Once we determined the best length of word vectors for the GloVe model, we fit a base model using the sentence vectors with a vector dimension of 150. This baseline model resulted in a F1 Score of 0.61 with a Recall of 0.59, a reasonable baseline model to improve on. However, we found most of the 150 variables had a VIF > 10, indicating multicollinearity issues in the model. Next, we used PCA to configure a new model to eliminate multicollinearity and reduce dimensionality. This model was regressed on the first 70 principal components, which account for 90% of the variation in the model. We found that this model performed very similarly to the baseline model, however it had a smaller F1 score indicating a lower sarcasm prediction rate. The variables used to fit the PCA model had VIFs < 10, showing that the multicollinearity had been addressed. Next, we chose to create a LASSO model as an alternate method to reduce dimensionality. This is conducted in the same way as in Word2Vec using 5 fold cross-validation. This model improved on the baseline model barely with a slightly larger F1 score and recall, but precision decreased indicating there was not a significant improvement in the model.

Since the F1 score of each of these models were all between 0.5 and 0.6 we thought the prediction of the models could be improved by weighing each of the 3 previously fitted models. The optimal weights for each model were found by plotting w_1 against Testing F1 Score, Recall, and Precision. We found that Precision remained relatively constant throughout each model while the F1 Score and Recall increased as w_1 increased. Next, we selected a w_1 value of 0.85 as this is the best weighting where Recall and F1 Score plateaued as the model tended not to improve significantly past that point. Starting with the baseline model we found that there was an improvement over the previous models as F1 Score and Recall are slightly larger. Additionally, there was a significant decrease in the amount of VIFs > 10, indicating a large decrease in the multicollinearity. The weighted PCA model was found to have an optimal w_1 value of 0.88 and performed much better than the previous models with an F1 Score of 0.68. The Recall of this model also increased significantly, indicating that more sarcastic comments were predicted correctly. Similar to the unweighted PCA model, no VIFs were greater than 10. Finally, we created a weighted LASSO model with a w_1 value of 0.8 as this model plateaued to a Recall of 1 much sooner than the rest. This resulted in the highest F1 Score of 0.72, and a Recall of 0.99.

This indicates that the weighted LASSO model performed the best at predicting sarcastic comments as seen by a significantly larger F1 Score.

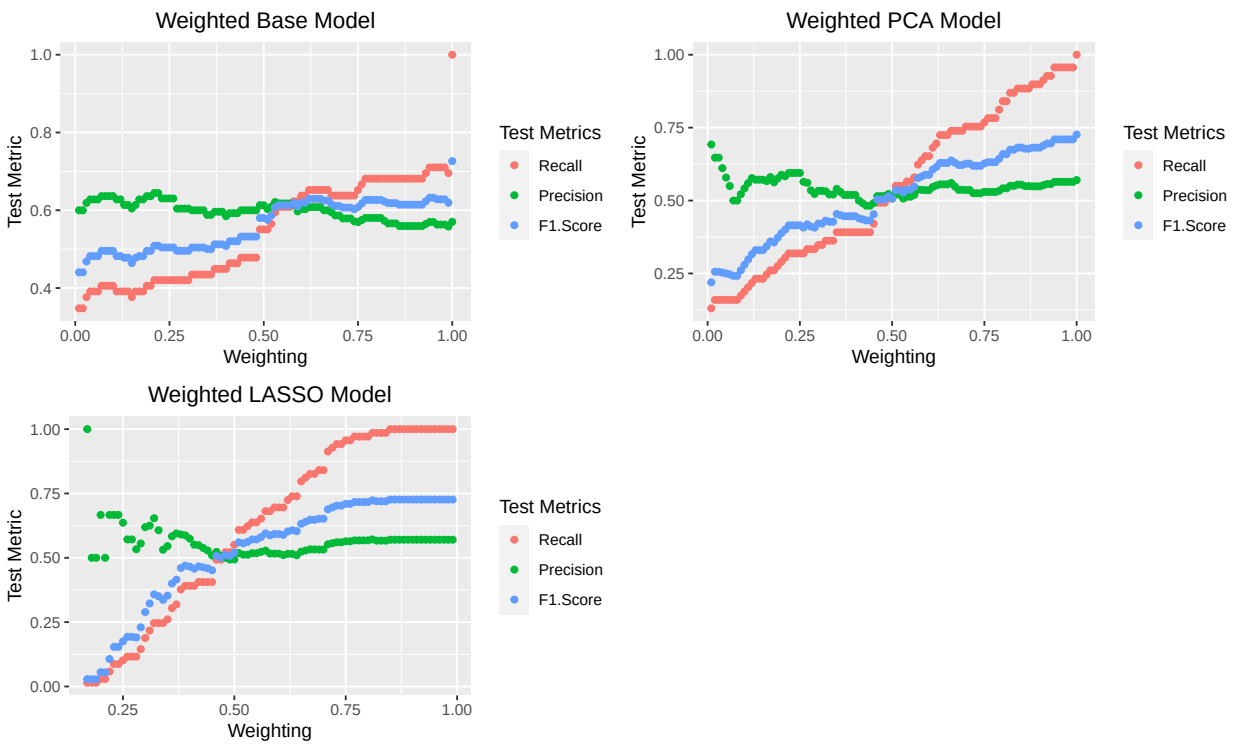

Figure 14 Plot showing weightings used in model fitting and the resulting test metrics for GloVe models.

These graphs show the test metrics for the Base, PCA, and LASSO GloVe models as the weighting factor w_1 increases.

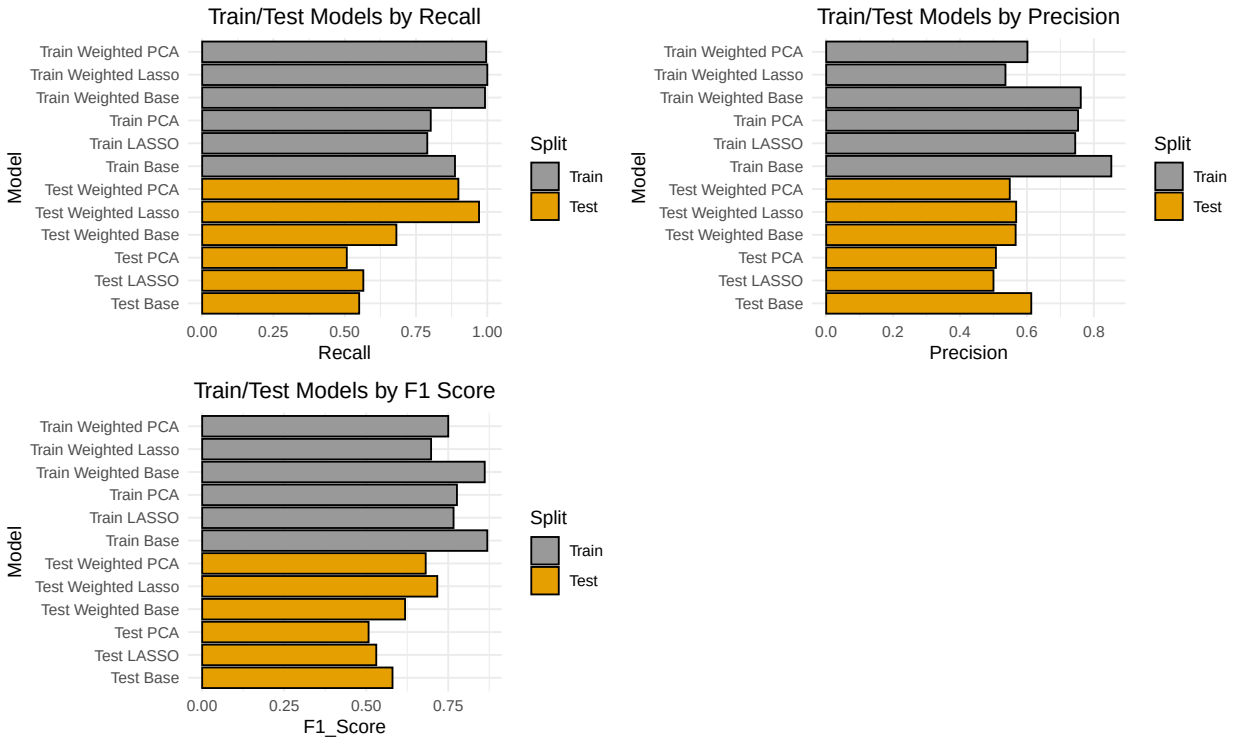

Figure 15 Bar graphs showing train and testing metrics for all models using GloVe feature extraction.

Other Findings

As weighted LASSO was found to be the best TF-IDF model we have access to the set of words that were not shrunk to 0. This provides us we insight as to which words predict best for sarcasm in the subreddit Not How Girls Work. The words are:

tate	written	companionship	reduct	final
remov	report	nowaday	gold	realis
asexu	sister	dump	iron	page
pictur	puberti	gal	act	broke
ignor	crime	cop	polic	pedo
jail	depend			

The results are quite interesting. Not How Girls Work is a subreddit about making fun of those who seemingly unaware of why women act the way they do and we see terms related to sexuality, relationships, and crime. Notably we also see *tate* a highly controversial figure for his views on women.

Conclusion

Model Comparison

	GloVe	TFIDF	Word2Vec
F1	0.723	0.807	0.759
Pr	0.571	0.722	0.650

	GloVe	TFIDF	Word2Vec
Recall	0.986	0.915	0.912

The best overall model was surprisingly the TFIDF model. It had the highest F1 score indicating it is the best model when it came to classifying sarcasm. The GloVe model did have better Recall than all other models but its Precision was considerably lower indicating it does not handle non-sarcasitc comments well. Finally Word2Vec performed somewhere in between these models. It had a Recall that was on par with the TFIDF however its lower Precision gave it a lower F1 Score. It was seen that LASSO was the strongest variable selection method and all models greatly benefited from weighting.

Limitations

The largest issue with this analysis is the dataset. Logistic regression is not equipped to handle such 0 heavy data and without us artificially constructing the ratio of sarcastic to non-sarcastic comments, this analysis likely would not work.

As a note, since GloVe uses unsupervised learning and Word2Vec uses a neural network there is extreme variance in model fitting. For these two models we got considerably different results each pass through. As seen in dimensionality selection for Glove, there is a lot of variability for each dimension. This further speaks to TFIDF's benefits as it is simpler and consistent.

Future Work

There is much room for improvement in using all 3 feature extraction methods with Logistic regression. Decision thresholding is where the probability of predicting a class as 1 or 0 is changed from being greater than or less than 0.5 respectively. It is a common approach when using logistic regression as a classification algorithm and could have improved our results.

Using the GloVe feature extraction method, we could have collected test metrics for several dimensions using each model. For the Word2Vec model we could have additionally used the CBOW architecture in addition Skip-Gram to assess the performance of this architecture.

References

De Ville, B., Bawa, G. S., Wiley Online Library Frontlist All English Titles 2021, Wiley Online Library, & O'Reilly for Higher Education. (2021). Text as data: Computational methods of understanding written expression using SAS. Wiley. https://doi.org/10.1002/9781119487142

Kyriakides, G., Margaritis, K. G., & O'Reilly for Higher Education. (2019). Hands-on ensemble learning with python: Build highly optimized ensemble machine learning models using scikit-learn and keras. Packt Publishing, Limited.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Most sarcastic subreddits [OC]. Reddit. (2018). Retrieved April 13, 2023, from https://www.reddit.com/r/dataisbeautiful/comments/9q7meu/most_sarcastic_subreddits_oc/

Narkhede, S. (2021, June 15). Understanding confusion matrix. Medium. Retrieved April 13, 2023, from https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Rachman, F. H. (2020, October). Twitter sentiment analysis of Covid-19 using term weighting TF-IDF and logistic regresion. In 2020 6th Information Technology International Seminar (ITIS) (pp. 238-242). IEEE.

Selivanov, D., Bickel, M., & Wang, Q. (2020). Package 'text2vec'.

Silge, J., Robinson, D. (2017). Text Mining with R: A Tidy Approach. United States: O'Reilly Media.

Wei, D., Bhardwaj, A., & Wei, J. (2018). Deep Learning Essentials.

Wijffels, J. (2021) word2vec: Distributed Representations of Words. https://cran.r-project.org/web/packages/word2vec/word2vec.pdf