Classifying Text as Female or Male using Word2Vec and Google News Corpus

Jonas Harrison
Department of Mathematics and Computer Science
Lawrence Technological University
Southfield, Michigan, USA
jharriso2@ltu.edu

***Abstract*—Classification of text as female or male has many real-world uses. Using Word2Vec in conjunction with the Google News Corpus we are able to classify text as female or male. Prepossessing, numerical representation and classification are all techniques used. 5 classification methods are used, naive bayes, k-nearest neighbors, logistic regression, MLP, and decision trees. Despite all 5 classification methods ending up with similar scores, differences in the accuracy of female and male classifications leads us to our final model being k-nearest neighbors. In general, it is important to carefully evaluate the performance of different classification methods and choose the one that provides the most accurate results for the specific task at hand.**

***Index Terms*—Word Embeddings, Data Preprocessing, Gender Classification**

I. Introduction

Which classification method most accurately defines text as female or male? Classification of text as female or male is an important task in many industries. Stores may want to use the information to better advertise to their customer base, hospitals may use the information to separate medicine based on gender, and search engines may use the information to serve more accurate results to their users. To classify the text a data scientist has multiple classification methods to choose from. However, these methods are each different and are all better or worse relative to one another based on the type of data being evaluated, the amount of data being evaluated, and how the data is being evaluated. We use word embeddings from the Google News corpus in conjunction with these classification methods to define text as female or male. Word embeddings are a numerical representation of a word. These numerical representations can then be used to decipher the semantic meaning of a word. To create a word embedding we can use a two-layer neural network known as Word2Vec. Word2Vec is a natural language processing technique created by Google in 2013, it learns associations between words from a large dataset. After the model has been trained on the dataset it can successfully identify words within a corpus with similar semantic and syntactic qualities. [1] If you take two semantically identical words such as computer and machine, the cosine of their vector representations would be close to being zero. The closer the cosine of two numerically represented words comes to reaching zero the more semantic meaning those two words share. (Shown in Figure 1) To answer our proposed question, we use word2vec in conjunction with multiple classification methods in order to see which of the classification methods most accurately classifies text as female or male.

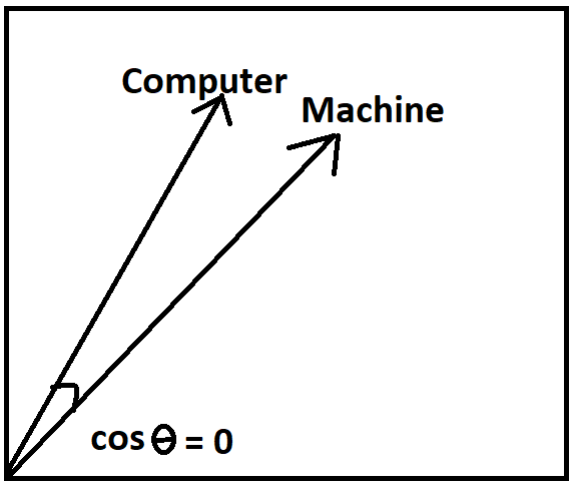

Fig. 1. Cosine of two similar words

II. Related Work

Many works and papers have been written on the topic of gender classification in text. We will be leveraging them to help us complete the project. While not the direct topic of this study classification of text as female or male can help when deciding if a dataset uses abusive language. [3] We also know that word embeddings are more likely to contain statistical gender biases so we must be careful when using them.[7] This is due to language being a living breathing entity, changing every day. However, this paper will more be about what classification method most accurately classifies text as female or male rather than if that classification itself is biased.

III. Methods and Design

A. Dataset description

The dataset used for this study comes from the study Collecting a Large-Scale Gender Bias Dataset for Coreference Resolution and Machine Translation.[8]

B. Data Preprocessing

To preprocess my text we write a function that uses regex to remove whitespace, punctuation, or anything with digits from the input. We then tokenized the input into sentences, then tokenized that into word tokens. Next, we loop through the word tokens and remove stopwords, convert them to lowercase and then remove tokens smaller than 3 characters. After that we call our preprocessing function twice, passing in Xtrain first and Xtest second. This leaves us with a list of lists with each list inside the list containing the word tokens. We then loop through the list of lists and within each list we join the words together with whitespace. Finally, we turn our preprocessed Xtrain and Xtest into data frames.

	0
0	case person understands message tongues previo...
1	enrolled self help student meant worked school
2	experienced techniques gray agent seemingly se...
3	author makes actuarial evaluation study clinic...
4	better person environment lead lower perceived...
...	...
10706	series also featured back stories involving ph...
10707	doctor comes wakes patient dying sleeping warn...
10708	change prompted qualified female electronics t...
10709	subsequently sentenced life prison robbed equi...
10710	dancer julia rourke wins first world title cou...

Fig. 2. Xtrain after being preprocessed

	0
0	elderly farmer played victor rebengiuc lost wi...
1	danys also asked include soviet ideas movie sc...
2	condie years born third marriage point laurenc...
3	sitting news desk turns correspondent seems la...
4	reasonable person would conclude must seen thu...
...	...
7136	person helped gangi reddy daughter night marri...
7137	during experiment professor mingles blood frag...
7138	author lucy brien described song book
7139	larita delighted teacher suggests enters schol...
7140	secretary village council generally helped gra...

Fig. 3. Xtest after being preprocessed

C. Representing Xtrain and Xtest Numerically

To represent Xtrain and Xtest Numerically we create a count vectorizer. We then use that vectorizer to fit() or data frames. From fit() we are able to calculate the features. Then we use a temp variable to get the vocabulary from fit(). Once we have the vocabulary, we fit transform() our Xtrain and transform() our Xtest. Next using the vocabulary we create a list with each element of the list being a key from the vocabulary. Finally, we create new data frames containing our fit transform() for Xtrain, and transform() for Xtest, while each column for the data frames contains a key from their vocabularies.

	case	person	understands	message	tongues	previously	spoken	unknown	language	enrolled	...	nabieva	kiryashov	ulrich	renews	phantom
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10706	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
10707	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
10708	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
10709	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
10710	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

Fig. 4. Xtrain numeric representation

	elderly	farmer	played	victor	rebengiuc	lost	wife	flood	danys	asked	...	abiyev	outpost	gangi	reddy	indu	mingles	fragments	lilith	larita
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7136	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
7137	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
7138	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
7139	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
7140	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

Fig. 5. Xtest numeric representation

D. Check Dataset Against Google News

After the dataset has been converted into its numeric representation, we then import the Google News Corpus. Once it has been successfully imported we can now check to see which words in our original dataset are represented in the Google News Corpus. To do so we write a function that takes in an input, creates an empty data frame, loops through the input, and for each element in our input, loop through our vocabulary and see if the current word is represented in the Google news Corpus. If it is, we can add it to our sentence. After looping through an input, we append a data frame containing our sentence into the empty data frame we created at the beginning of the function.

```
def text2Vect(text):

    #transform input into numeric
    conv = vectorizer.transform(text)

    #create dataframe to hold transformed input
    countVecText = pd.DataFrame(conv.toarray(), columns=vectorizer.get_feature_names())

    #dataframe to hold sentences
    sent_dataFrame = pd.DataFrame()

    #loop through input
    for i in range(countVecText.shape[0]):

        #init sentence with all zeros
        sentence = np.zeros(300)

        #loop through the vocabulary
        for word in wordsVocab[countVecText.iloc[i,:].values>=1]:

            #if a word is in the google news corpus add it to the sentence
            if word in googleNewsWE.key_to_index.keys():
                sentence = sentence + googleNewsWE[word]

        #append sentence to dataframe
        sent_dataFrame = sent_dataFrame.append(pd.DataFrame([sentence]))

    return sent_dataFrame
```

Fig. 6. Checking Xtrain against the Google News Corpus

E. Model building

For this study 5 classification methods are used.

```
def text2Vect1(text1):

    #transform input into numeric
    conv1 = vectorizer1.transform(text1)

    #create dataframe to hold transformed input
    countVecText1 = pd.DataFrame(conv1.toarray(), columns=vectorizer1.get_feature_names())

    #dataframe to hold sentences
    sent_dataFrame1 = pd.DataFrame()

    #loop through input
    for i in range(countVecText1.shape[0]):

        #init sentence with all zeros
        sentence1 = np.zeros(300)

        #loop through the vocabulary
        for word in wordsVocab1[countVecText1.iloc[i,:].values>=1]:

            #if a word is in the google news corpus add it to the sentence
            if word in googleNewsWE.key_to_index.keys():
                sentence1 = sentence1 + googleNewsWE[word]

        #append sentence to dataframe
        sent_dataFrame1 = sent_dataFrame1.append(pd.DataFrame([sentence1]))

    return sent_dataFrame1
```

Fig. 7. Checking Xtest against the Google News Corpus

1) Naive Bayes: Naive Bayes is a set of supervised learning algorithms that implement Bayes' theorem that assumes conditional independence between all pairs of features to be given the value of the class variable. [9] We will be using Gaussian Naive Bayes which is defined in the equation below.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)\gamma \qquad (1)$$

2) K-Nearest Neighbors 5: K-Nearest Neighbors works by using a predefined number of samples near the new point and predict the label of the samples. The number of samples is passed in as a parameter to the classifier. Optimally choosing the number of samples is almost always dependent on the data, however, a good rule of thumb is that a larger K value will make the bounds less distinct.[10]

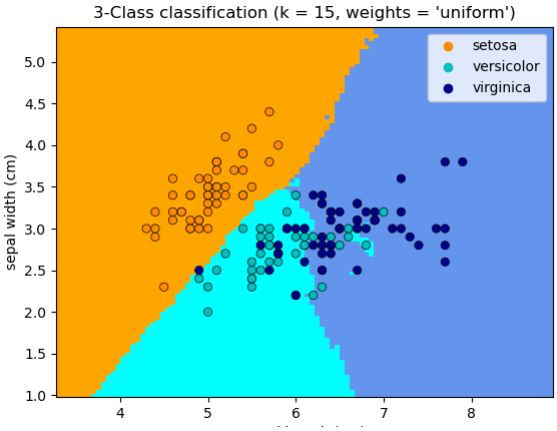

Fig. 8. Sample usage of Nearest Neighbors classification [10]

3) Logistic Regression: Logistic Regression is a classification method used to predict a binary outcome based on a set of individual variables, the outcome being female or male and our individual variables being the features.

4) MLP: The Multilayer Perceptron classification method as its name implies connects multiple layers in a directed graph. MLP is a deep learning technique.

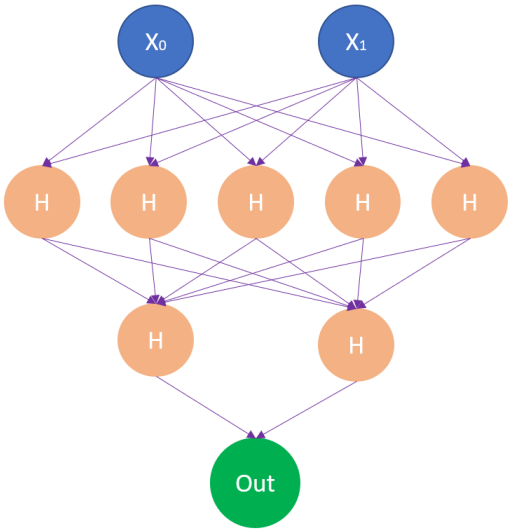

Fig. 9. Example of an MLP neural network [11]

5) Decision Tree: A decision tree uses simple decision rules inferred from the features. The more nodes the tree has the deeper its learning. [12]

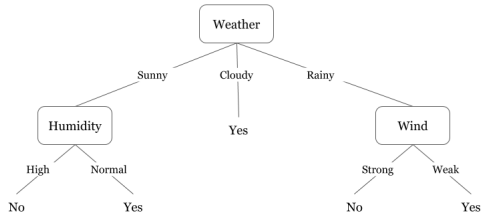

Fig. 10. Example of a decision tree

IV. Results and Evaluation

Shown in Figures 11 - 15 are the classification reports for each classification method we applied to our data.

1) Naive Bayes: As we can see, Naive Bayes gives us the highest accuracy score, tied with KNN. However, there is a large disparity in the f1 scores for female and male. Therefore this model will not be selected as our final classifier.

2) K-Nearest Neighbors 5: As we can see, K-Nearest Neighbors gives us the highest accuracy score, tied with Naive Bayes. Unlike our Naive Bayes classifier, the f1 scores for

Naive Bayes

	precision	recall	f1-score	support
female	0.53	0.60	0.56	3628
male	0.52	0.44	0.48	3513
accuracy			0.52	7141
macro avg	0.52	0.52	0.52	7141
weighted avg	0.52	0.52	0.52	7141

Fig. 11. Classification report for Naive Bayes

female and male are close in value. Therefore this model will be selected as our final classifier.

K-Nearest Neighbors 5

	precision	recall	f1-score	support
female	0.53	0.56	0.54	3628
male	0.51	0.48	0.50	3513
accuracy			0.52	7141
macro avg	0.52	0.52	0.52	7141
weighted avg	0.52	0.52	0.52	7141

Fig. 12. Classification report for K-Nearest Neighbors 5

3) Logistic Regression: Logistic Regression gives us a similar accuracy score to our highest-ranking models, naive bayes and KNN, being tied with MLP for accuracy. While logistic regression is tied with MLP in accuracy scores, we can say that logistic regression does worse than MLP for the same reason Naive Bayes does worse than KNN- a large difference in the accuracy of female and male classifications. Logistic regression does not have the highest accuracy, therefore this model will be selected as our final classifier.

Logistic Regression

	precision	recall	f1-score	support
female	0.52	0.45	0.48	3628
male	0.50	0.57	0.53	3513
accuracy			0.51	7141
macro avg	0.51	0.51	0.51	7141
weighted avg	0.51	0.51	0.51	7141

Fig. 13. Classification report for Logistic Regression

4) MLP: Multilevel Perceptron gives us a similar accuracy score to our highest-ranking models, Naive Bayes and KNN, being tied with logistic regression for accuracy. While MLP is tied with logistic regression in accuracy scores, we can say that MLP does better than MLP for the same reason KNN does better than naive bayes- the absence of a large difference in the accuracy of female and male classifications. MLP does not have the highest accuracy, therefore this model will be selected as our final classifier.

5) Decision Tree: Decision tree gives us the worst accuracy score out of all of our classification methods. It only comes in at .50. While it does have the most consistent deviation between the female and male classification f1. The low accuracy prevents it from being our final model.

MLP

	precision	recall	f1-score	support
female	0.52	0.48	0.50	3628
male	0.50	0.55	0.52	3513
accuracy			0.51	7141
macro avg	0.51	0.51	0.51	7141
weighted avg	0.51	0.51	0.51	7141

Fig. 14. Classification report for MLP

Decision Tree

	precision	recall	f1-score	support
female	0.50	0.49	0.50	3628
male	0.49	0.51	0.50	3513
accuracy			0.50	7141
macro avg	0.50	0.50	0.50	7141
weighted avg	0.50	0.50	0.50	7141

Fig. 15. Classification report for Decision Tree

V. Conclusion

To wrap things up the main takeaways from this study are a few things. First, K-Nearest Neighbors with 5 neighbors did the best out of all 5 of our classification methods. Secondly the accuracy score of our KNN model could have potentially been increased had normalization been applied before passing the data off to our models. And lastly, with current training, it appears that key words such as "video game" potentially hold more weight when determining gender than actual gendered words.

References

[1] https://en.wikipedia.org/wiki/Word2vec
[2] Multiple Authors, Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings, 2016
[3] J. Park, J. Shin, P. Fung, Reducing Gender Bias in Abusive Language Detection, 2018
[4] Multiple Authors, Mitigating Gender Bias in Natural Language Processing: Literature Review, NA
[5] A. Field, Y. Tsvetkov, Unsupervised Discovery of Implicit Gender Bias, 2020
[6] E. Dinan, A. Fan, L. Wu, J. Weston, D. Keila, A. Williams, Multi-Dimensional Gender Bias Classification, 2020
[7] D. Jergens, S. Mohammad, P. Turney, K. Holyak, SemEval-2012 Task 2: Measuring Degrees of Relational Similarity, 2012
[8] S. Levy, K. Lazar, G. Stanvosky, Collecting a Large-Scale Gender Bias Dataset for Coreference Resolution and Machine Translation, 2021
[9] https://scikit-learn.org/stable/modules/naivebayes.html
[10] https://scikit-learn.org/stable/modules/neighbors.html
[11] https://scikit-learn.org/stable/modules/generated/sklearn.linearmodel.LogisticRegression
[12] https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perceptron-classifier-mlpclassifier/mlpclassifier
[13] https://scikit-learn.org/stable/modules/tree.html