Revealing Inherent Gender Biases in Using Word Embeddings for Sentiment Analysis

Matthew Plotnikov Libbins[1], *Seamus Bornhauser*[1,2]

[1]School of Visualization for Boomers
[2]School of Visualization for Zoomers
{matplotlib,seaborn}@viz.edu

Abstract

Word embeddings or **word vectors** are high dimensional mathematical representations of words that are ubiquitous in their application in many natural language processing (NLP) tasks. While using word embeddings effectively improves the accuracy of many tasks, because many of the recent word embeddings are extracted from complex black-box machine learning models (known as "neural networks"), understanding and unpacking the *meaning* of these embeddings is a challenging task. Furthermore, because these embeddings are learned from real world data, many of the biases of the world become part of these embeddings. Many of these biases are further exacerbated depending on how the machine learning models are trained. As a result, any downstream task that uses word embeddings becomes prone to these biases leading to unethical decisions and predictions. In this work, we use word embeddings extracted from 4 popular machine learning models (Word2Vec, GLOVE, ELMo, and BERT), and use them to create four different sentiment analysis systems. We then conduct an experiment using "IMDB reviews" to check and compare how these sentiment analysis systems perform based on whether the review has been written by male or a female person. While the gender inherently does not have a sentiment, we find striking differences in the way these models change their predictions as a result of our controlled experiment. We conduct our experiment on reviews from both "hollywood" movies and "bollywood" movies, and find that 3 out of four models (Word2Vec, BERT, GLOVE) tend to predict a negative sentiment if the review is signed by a female. The model ELMo on the other hand favors female names. In contrast, a model not trained on word embeddings, while lower in accuracy, tends to not get affected by the gender of the reviewer. Our results inform us about the way existing models behave, and serve as a starting point in understanding how unwanted biases can affect the technology that we make and use.

Index Terms: word embeddings, gender biases, explainable AI, visualization, matplotlib, seaborn

1. Introduction

1.1. Word Embeddings

What is 1+1-1? 1. What is 3-2+1? 2. What is six + seven? 13. So far so good. What is ***king - man + woman?*** Hmm. That is a tricky one. What if I told you it is ***queen?***?

When someone asks us to perform a mathematical operation on *six* and *seven*, we form a mental image of six being 6, and seven being 7. But we are not used to creating a mathematical image of our daily life vocabulary such as *man*, *woman*, and perhaps after *Meghan Markle*'s interview, the word *queen* :).

Our computers, however, only understand numbers and math. Our computers have the ability to convert a 6 to *0101* and 7 to *0111*, and perform any mathematical operation. However, even understanding to interpret what 'six' or 'seven' is, is hard for our computers. This means if we want our computers to be *intelligent*, and be able to understand our words, there needs to be a way to convert a word to a mathematical representation. These representations are what are typically known as *word embeddings*.

If you have ever used *Alexa, Siri, or Google Home,* chances are you have benefited from word embeddings as many of these systems use different types of embeddings to answer our questions.

Now the question is how do we create a word embedding? If everyone in the the world used flower language, and the language had only 3 words in the vocabulary say "sentosa", "virginica", and "versicolor", then creating an embedding would be easy i.e. "sentosa" gets mapped to '[1,0,0]", "versicolor" gets mapped to "[0,1,0]", and "virginica" gets mapped to "[0,0,1]". (Remember one-hot encodings from PS2?).

Unfortunately, three words are not enough. The languages of the world are filled with words. English alone has about a million words. What this means is, if we were to use *one-hot encodings* as above to represent words, we would need a vector of 1 million with all values being zero except one value for each word. While that is possible, this approach is highly inefficient and somewhat infeasible. Where would you store such long vectors? How would you process them? Can we do better?

This is where neural networks come in. Neural networks are machine learning models, that capture patterns, and relationships between vast amounts of data. Many times, these are relationships that even humans could not have captured.

To explain what a neural network does, let us think of an example. Think of a dataset where there are only three variables: *HEIGHT*, *WEIGHT*, and *HEALTH STATUS*. Let's say we had to predict the *HEALTH STATUS* (*HEALTHY* or *UNHEALTHY*) of a person based on *HEIGHT*, and *WEIGHT*. If we were to create a simple machine learning model, we would use *HEIGHT*, and *WEIGHT* as *features* or independent variables for our model, whereas *health condition (HEALTHY/UNHEALTHY)* would be our dependent variable. Can we do better? What if we were to *create* a new feature called the *BMI* such that

$$BMI = \frac{WEIGHT}{HEIGHT^2} \tag{1}$$

Suddenly, you have a new arguably useful feature to be used in your model. What if I told you that there is a new feature that may also be useful for your model say for example:

$$f = \frac{(BMI)^\pi}{e^{WEIGHT}} \tag{2}$$

Now what is this feature? Is there a name for it? Not really. But so what? There needs **not** be a name for it. While this feature is not *interpretable*, that does not mean it is not useful.

A neural network does a very similar thing. Some people call neural networks *feature generators*. They *transform* your data to create useful features. Neural networks, however, use extremely complex mathematical operations to create feature vectors. While, this makes these vectors non-interpretable, they are highly efficient, and useful.

In the context of words, these feature vectors are also called word embeddings. For each word, these vectors are basically a bunch of numbers that represent that word, or the features of that word. The embedding of the word depends on the machine learning model. Like multiplication and division, different machine learning models represent words differently using their own mathematical operations. Few of the most common models for word embeddings are *Word2Vec*[1], *GLOVE*[2], *ELMo*[3], and *BERT*[4]. A word embedding for the word "king", for example, from the *ELMo* model is as follows:

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

If we had to represent these numbers as a heatmap, they would look as shown in figure 1.

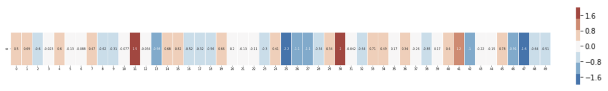

Figure 1: *Word vector representation of the word "king" using ELMo.*

Let us also look at other words such as "man", and "woman" as shown in figure 2.

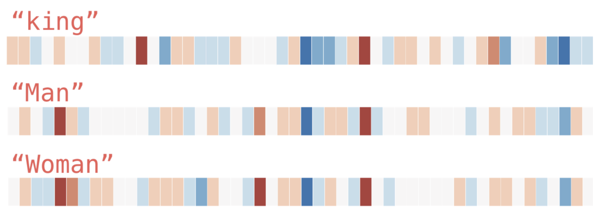

Figure 2: *Word vector representation of the words "king", "man", and "woman" using ELMo.*

If you examine figure 2 carefully, you would notice that the embeddings for *man* and *woman* are very similar to each other, more than either of them are to the embedding for *king*.

To make things even more interesting, in figure 3, we have the embeddings for even more words.

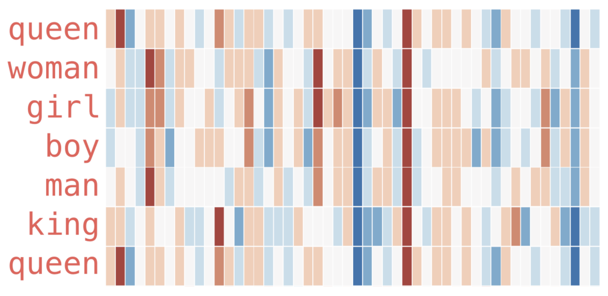

Figure 3: *Word vector representation of different words using ELMo.*

Here are a few things to notice:

- There's a straight red column through all of these different words. They're similar along that dimension (and we don't know what each dimensions codes for).
- You can see how "woman" and "girl" are similar to each other in a lot of places. The same with "man" and "boy".
- The "boy" and "girl" also have places where they are similar to each other, but different from "woman" or "man". Could these be coding for a vague conception of youth? That is possible.
- There are clear places where "king" and "queen" are similar to each other and distinct from all the others. Could these be coding for a vague concept of royalty?

In essence, these representations of words are not just compact (unlike one-hot encodings), but are also meaningful as they form clusters around similar words, or words in similar contexts. Visualizing these clusters is very hard as each embedding has anywhere between 200 to 2000 values, and we, unfortunately, live in a 3 dimensional world, and can hardly even visualize 4 dimensions. This is one of the main reasons why it is so hard to explain the meaning of each of the features in the word embeddings. That said, there are ways to reduce the dimensionality of an embedding to retain only important features. Some of these techniques are *Principal Component Analysis*[5] or *t-distributed stochastic neighbor embedding* (t-SNE)[6]. The details of these methods are beyond the scope of this paper, and not quite important.

Nevertheless, in figure 4, we represent a scatter of 400 different words in the english language by choosing 2 of the *important* features using the t-SNE technique. We provide this dataset of 400 words (which includes english letters) with their two important dimensions as a supplementary ".csv" document for reproducibility of the plot.

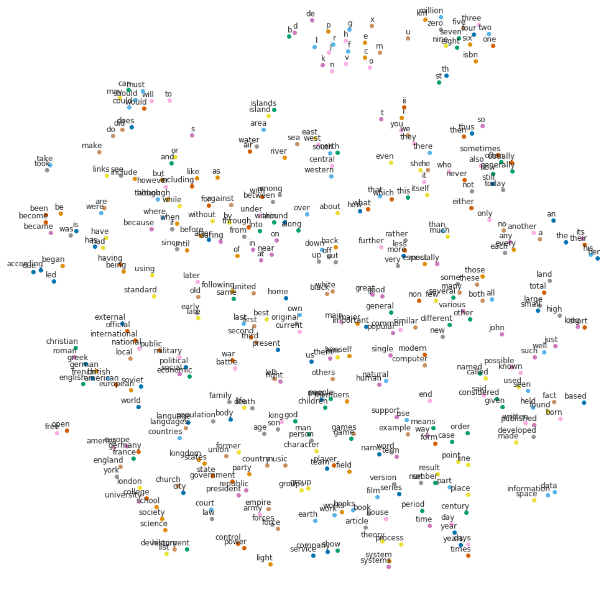

Figure 4: *The scatter plot of 400 different English words plotted against 2 dimensions extracted using the t-SNE technique given GLOVE embeddings. The plot was created using Python's Matplotlib library. Similar context words such as english letters, or words representing nationalities are seen to cluster together.*

1.2. Word Embeddings for Sentiment Analysis

Now the question is: why would you need these embeddings at all? That is because they make downstream tasks such as *sentiment analysis*, *machine translation*, *speech recognition*, and many other NLP tasks much easier and quicker to handle. *Sentiment Analysis* is the task of assigning a *positive*, *negative*, or *neutral* "valence" to a piece of text. It is extremely useful in analyzing customer reviews, movie reviews, social media data, and other tasks that involve textual data. In most of these tasks, a given piece of text is first processed to remove *stop words* (eg. "the", "a", etc.), and the text is then segmented into words. Each word is then passed through a trained neural network to extract its word embedding. The word embeddings of a given text are then concatenated to represent each text. Because different texts may have different lengths, these texts are brought to same length by padding zeroes. Finally this long "sentence embedding" is passed to a machine learning model to train sentiment analysis system.

For the purposes of this paper we used 5 different types of methods to create 10 sentiment analysis systems for IMDB movie reivews. Five of these systems were created using reviews from "Hollywood", and five of these systems were created using reviews from "Bollywood". "Hollywood" and Bollywood" represent the industries in our study. Four of the systems in each industry were created using word embeddings of popular models (Word2Vec, GLOVE, ELMo, and BERT), and one of the systems was created without using word embeddings (i.e. using traditional machine learning models). The features for the sentiment analysis models were the word embeddings, whereas the dependent variable was the review. For our purposes, we standardized all the reviews to fall between -1 and 1 where values < 0 meant the model predicted the review as "negative", values > 0 signified a "positive" review, and a value of 0 meant a "neutral" review.

The performance accuracy of the different sentiment analysis systems is given in table 1.

Model	Acc. (Hollywood)	Acc. (Bollywood)
GLOVE	78%	83%
ELMo	89%	84%
BERT	88%	92%
Word2Vec	96%	95%
No Embedding	75%	77%

Table 1: *This table shows the accuracy of different sentiment analysis systems trained in 5 different ways for the two industries.*

1.3. Bias in Word Embeddings for Sentiment Analysis

Given the accuracy statistics in table 1, you could simply choose the model created using "Word2Vec" as it performs the best across both the industries. But what if you found out that while "Word2Vec" performs the best overall, it's also most likely to assign a more positive sentiment to the sentence *"The main character is a man"* than to the sentence *"The main character is a woman"*? Would you reconsider your decision?

Maybe your goal is to just select a few good movies for yourself to watch next. In that case, maybe it's not of utmost importance to you to consider the gender biases. Maybe you are just implementing something for you class project, in which case, it is also not absolutely essential to account for the gender bias. However, what if you were hiring and paying actors and actresses according to their average movie review ratings, as assessed by the sentiment analysis model? That would certainly be problematic. With increasing usage of machine learning models in assessing decisions, this is not a stretch, and is very much possible. This leads us to investigate the robustness of these models against gender biases.

2. Methodology

2.1. Retrieving Test Review Samples

After the sentiment analysis systems were trained in 5 different ways for each industry, we created our test set of IMDB reviews from "hollywood", and "bollywood" movies. We did so, by creating an *out-of-sample* dataset of 1000 reviews from each industry. These reviews were balanced in terms of their actual valence i.e. 500 reviews within each industry were positive, and 500 reviews were negative. The actual distribution of the reviews from positive and negative categories was kept identical by choice.

2.2. Retrieving Gendered Names

After choosing the test samples, we created a list of 400 first names chosen from the common male and female names from the United States Social Security Administration (USSSA). We controlled for the way we created this list as described below. We chose a list of American names to be used for "hollywood reviews".

We also created a list of 400 Indian first names chosen from LinkedIn profiles to be used for "bollywood reviews"

2.2.1. Top common names

Our first list of 200 names was created by choosing the top most frequent male names, and top most frequent female names from the USSSA database. 100 of these names were male, and 100 of these names were female.

2.2.2. Top common names by matched counts

Because the top 100 male names do not have the same distribution as the top 100 female names, this may introduce some bias in our analysis. Therefore, we controlled for names by pairing most frequent male and female names with the same counts. This resulted in a paired dataset of names matched by count.

2.3. Experimental Setup

Finally, we conducted our experiment by appending to each of the reviews, **"reviewed by______ "** where the blank is filled in with a name. For each review, we replaced the blank with the 100 male names and 100 female names. Each time the review was passed to a sentiment analysis system to predict a sentiment/valence. This resulted for each review 100 sentiment values for male, and 100 sentiment values for female. We then computed the average sentiment values for male, and average sentiment values for female for each review. Finally, we computed the differences in the predicted average valences for each review for male names and female names, and plotted the distribution of those differences. For each review, the difference D was computed as:

$$D = \mu_f - \mu_m \tag{3}$$

where μ_f represents the **average rating as predicted by**

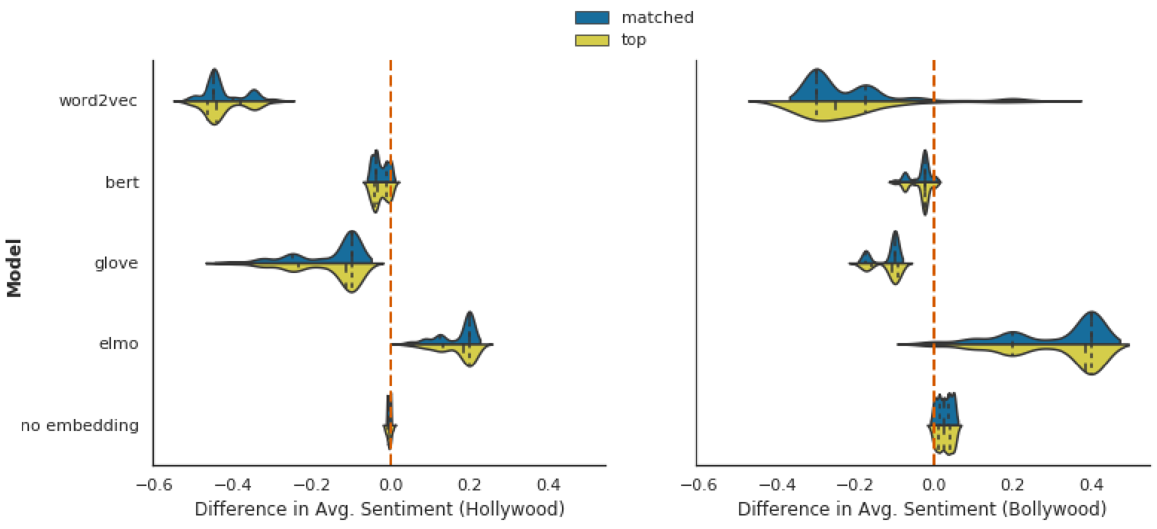

Figure 5: *The figure shows the two violin plots associated with the two industries. Each violin plot contains the kernel density estimation of the difference in average sentiments across different models. Each distribution is further split by whether the names were most frequent or were matched by counts. Note that this plot is depicting the "differences" in the average sentiment for each review rather than the actual valence values. As such values greater than 0 depict if the model prefers female names for assigning a relatively positive valence whereas values less than 0 depict if the model prefers male names.*

the system for the review with female names, and μ_m represents the **average rating as predicted by the system for the review with male names**.

You must note that nothing else in the review changed except for the **"reviewed by _______"** line. As such, we should expect the sentiment analysis models to perform exactly the same for both male names and female names.

Think of it this way. If we have a review that says: *"Really liked the movie, reviewed by Shahan"* versus *"Really liked the movie, reviewed by Bedoor"*, where "Shahan" is a majority male name, and "Bedoor" is a majority female name, the sentiment analysis system should make no distinction in who it was reviewed by, and should assign exactly the same valence value to both the reviews, regardless of the gender of the name. That is precisely what we are testing in this paper as to whether these sentiment analysis systems are independent of such biases.

Note that we conduct experiments for "hollywood", and "bollywood" separately. We do this to check if these biases are just limited to America or can also be seen in other places. "Bollywood" was chosen as it is the other big movie industry with the most number of reviews on IMDB aside from "hollywood".

3. Results

From our results, we find that there are in fact inherent gender biases in the sentiment analysis systems that use word embeddings. We conducted statistical analysis for the difference of means in ratings as predicted by the sentiment analysis system using the t-test, and all the results were statistically significant with $p < .001$[1].

The violin-plots in figure 5 show the distribution in differences of average sentiment scores. We show results for four word embeddings, as well as a model (no embedding) that doesn't use a word embedding.

Analyzing the difference in sentiment for a model with no embedding is a good check that confirms that the sentiment associated with the names is not coming from the small IMDB supervised dataset, but rather is introduced by the pretrained-embeddings, and hence the machine learning models. We can also see that different embeddings lead to different system outcomes, demonstrating that the choice of embedding becomes a key factor in downstream tasks.

From the plots, we can see that while model with no embedding has the least accuracy in terms of performance, it also incurs the least bias. This shows that there may be a trade-off between accuracy and bias. We note that the density of the plots does not change with the type of names being used ("top" vs. "matched") except for the Word2Vec model. In terms of the word embedding models, all models except ELMo prefer male names in comparison to female names, whereas ELMo seems to prefer female names. The distribution plots for hollywood and bollywood differ in terms of variance. However, the results remain the same which makes our analysis and claims stronger.

4. Conclusions

In this work we uncover the inherent gender biases introduced in sentiment analysis systems created using word embeddings from popular machine learning based models. We show that while these models perform really well in terms of accuracy, almost all of these models carry heavy biases that can yield unethical recommendations and predictions in downstream tasks. Our results inform us about the way existing models behave, and serve as a starting point to trigger discussions in understanding how unwanted biases can affect the technology that we make and use.

[1]These analysis are present in an imaginary supplementary document.

5. Acknowledgements

Illustrations for word embeddings were retrieved from here. The story of the paper was inspired by the similar work of Google AI Ethics team the blog for which can be found here.

6. References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[3] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," *arXiv preprint arXiv:1705.00108*, 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[6] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.