Stereotype and Unconscious Bias in Large Datasets

Elliot Silva, Atul Gandhi, Eileen Cho, SunJoo Park

***Abstract*— As social movements strive to raise awareness of the impact of gender norms and stereotypes, it becomes increasingly important to think consciously of how we speak and write about others, particularly in professional settings. In this paper, we combine insights from behavioral science with deep learning methods to tackle the task of identifying words communicating unconscious, yet potentially harmful, gender biases. We apply both naive Bag of Words approaches with semi-supervised clustering methods, as well as sophisticated architectures such as LSTM and BERT, in order to produce contextualized embeddings indicating whether a word is being used in a biased way. Our work will help improve automated bias detection systems.**

I. Introduction

Language reflects culture, and biased language perpetuates stereotypes and reinforces social norms that affect entire groups of people. This is true across all cross-sections of society, but in particular, language used in professional settings can affect evaluations of colleagues, hiring decisions, and ultimately impact career progression [1]. Consider the following examples:

She is difficult to work with.
He always produces great work.
We feel the final result was not satisfactory.

Studies have shown that women are more likely to be evaluated on their interpersonal skills and friendliness, whereas men are more likely to be evaluated on the quality of their work [2]. As we see in the first example above, women can often have their ability to work with others commented on in a negative way, while an objective assessment of the quality of their work is overlooked. Conversely, as in the second example, men are more likely to have the quality of their work evaluated, and often have positive words attributed to them in ways that encode that they or their work is inherently good. A bias-free evaluation of an employee would provide objective criterion as to whether the employee did or did not meet requirements, rather than encode inherent positive or negative qualities to the reviewer, or express vague feelings of the reviewer, as in the last example. Often, reviewers use this type of biased language subconsciously, without even thinking of the larger implications of reinforcing gender norms and double standards.

FairFrame is a company whose primary goal is to reduce unconscious bias, by providing a technology platform wherein reviewers can upload employee evaluations and receive feedback on words which may encode gender bias, as in the above examples. Their current algorithm uses a string-matching algorithm which compares words of a document against an internal list of words encoding gender bias, as identified by a team of linguists and behavioral research scientists. However, often these words can be used in contexts that don't connote any form of bias. Consider the following examples:

She successfully completed a difficult task.
He put a great deal of time into understanding our business needs.
We feel glad to be working with Amy.

In the first example, *difficult* is being used to describe a task, rather than a person. In the second example, *great* has a different word sense compared to the biased example seen earlier. The last example could be argued either way, however FairFrame ultimately deemed this example non-biased, as it harmlessly expresses that a company appreciates working with that particular freelancer.

For our project, we train a model to be able to discern between these various instances of biased or non-biased contexts. In doing so, the model ultimately learns to apply a linguistic or semantic context to each word, to determine whether it is being used in a biased way. As shown in the last example above, there is often ambiguity which would be difficult for even a human to resolve (hence "unconscious" bias), however our modeling is based on rules and heuristics set forth by FairFrame. Ultimately, our work can be used to reduce the number of words falsely flagged by their algorithm, thus optimizing efficiency of use and increasing user trust in their product.

II. Related Works

A. Language Models on Learning Gender Bias

Previous research into analyzing and detecting biased language in computational linguistics commonly relied heavily upon human annotations for labels, or rule-based models based on expert knowledge [3], [4]. These methods yield results easily interpretable by humans, but are extremely labor intensive to produce. Other studies in Natural Language Processing (NLP) analyzing pre-trained word embeddings such as GloVe or ELMo have found that the systemic gender bias in the original textual data was passed on to the learned embeddings [5], prompting research into creating gender neutral word and contextual embeddings [6]. These new embeddings can be used for subsequent NLP tasks with the guarantee that gender bias has been diminished (e.g., embeddings for professions like *doctor* or *nurse* will lose their male or female connotations, respectively), but will not produce actionable results to help humans reduce their bias in everday settings. Our project combines insights from behavorial linguistics with novel NLP methods to produce

innovations in the space of computer-assisted efforts to reduce gender bias in human language.

B. Semi-Supervised Learning

Much of the work done in language modeling used supervised learning models. Indeed in this task, an unsupervised approach would not guarantee the model to learn information we care about. However, our dataset of 110K performance reviews (discussed below) came to us without indication of whether flagged words were used in biased contexts. Manual annotation of the entire dataset would have been impossible, since the complexity and nuance of the task prohibited large-scale outsourcing of the labeling. To reconcile this, we utilized a semi-supervised learning approach, shown in previous works to improve results of unsupervised NLP models [7], [8].

III. Data

Our dataset had two main components. The first was a set of ∼110K public reviews scraped from various sites such as Freelancer and 99Designs; the second was a list of words identified as containing gender bias, as determined by the behavioral linguistics team at FairFrame. We split these reviews into separate sentences and filtered to include only sentences containing the flagged words. To account for sentences with multiple instances of flagged words, we generated copies for each instance, noting the position of the flagged word. This resulted in a total of ∼110K sentences. For the purposes of this task, we refer to each of these sentences as a review.

A. Manual Annotations

In order to enable a semi-supervised learning approach, we annotated a small subset (4.8K) of the data. We randomly selected reviews and labeled them according to heuristics explained by FairFrame. We refer to flagged words being used in a gender-biased context as True Positive (class 1) and those used in a non-biased context as False Positive (class 0). After annotation, we discovered our dataset was extremely imbalanced, with 87% labeled as true positive and only 13% labeled as false positives. Further, the top five most frequently occurring words in our dataset accounted for over half of the entire dataset. As seen in Figure 1, these words largely contributed to the class imbalance. To resolve this we employed upsampling: we resampled instances from class 0 to be represented in our dataset at the same frequency as instances from class 1.

B. Pre-processing

We used the NLTK (Natural Language ToolKit) [9] package to tokenize our reviews. After tokenization, we observed that only a few sentences had more than 30 tokens, so we didn't include sentences of length longer than 30 words. Of the 4.8K manually annotated examples in our dataset, we split this into train (80%), validation (10%), and test (10%). All of our unlabeled examples were included only in the training set. Thus, we used data from both `train_labeled` and `train_unlabeled` to train our model in a semi-supervised setting, with `validation` and `test` fully labeled to assist evaluation of our results.

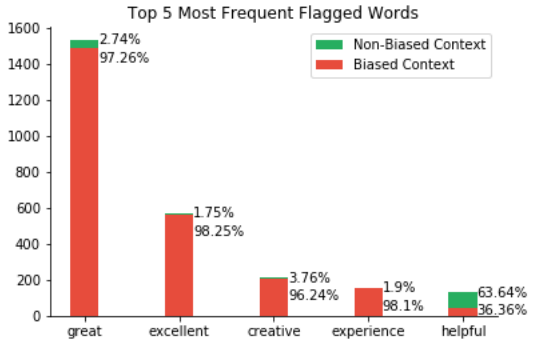


Fig. 1: Breakdown for top 5 most frequent flagged words in dataset

IV. Modeling

Our overall approach consisted of training embedding representations of sentences containing flagged words, and clustering these embeddings into two separate groups. Our goal was for embeddings of sentences containing flagged words in a biased context to be clustered separately from sentences containing flagged words in a non-problematic context.

A. Baseline - Bag of Words Model

1) Fully-unsupervised model: A bag-of-words model is a naive approach, wherein a document is represented as a collection of its words, thus disregarding grammar and word order. Despite its simplicity, it is frequently used in NLP tasks as it is easily interpretable, computationally light and often produces surprisingly good results. We therefore selected this model as our baseline.

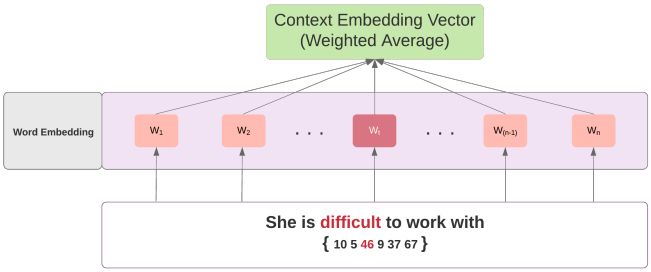


Fig. 2: Architecture of our baseline model

Figure 2 shows the architecture of our baseline neural network model. The tokenized input is first passed through a word embedding layer which trains encoded vectors for each word. These embeddings do not take the surrounding context into account. Then, in order to obtain a single contextual representation for the flagged word, we take a weighted average of the embeddings for every word in the review, assigning a weight of `w` to the embedding of the flagged word and a weight of 1 to the other embeddings. The optimal value

of this parameter, `w`, is then estimated using hyperparameter tuning. Once we obtain contextualized representations for each review, we apply the unsupervised K-means algorithm to cluster them into two groups: biased contexts and non-biased contexts.

$$k_i = \arg\min_k ||x_i - \mu_k||^2 \tag{1}$$

K-means algorithm uses Equation 1 to assign clusters to each embedding. The k_i in the equation refers to the cluster assignment (class 0 or class 1) of training example x_i, and μ_k refers to the centroid of cluster k. We assign x_i to the cluster k whose centroid it is closest to. At the end of each iteration, the model updates the cluster centroid to the midpoint of examples assigned to that cluster, using Equation 2 (where C_k refers to the count of points belonging to cluster k):

$$\mu_k = \frac{1}{C_k} \sum_{x \in k} x \tag{2}$$

In our first model, the initial values of the word embeddings were randomly generated. We also experimented with using pre-trained GloVe embeddings for our task [10]. Since GloVe vectors have been pre-trained on a large corpora of written text, we believed these may have contained more information than randomly initialized embeddings. Hence, we replaced our word embedding layer with a frozen GloVe embedding layer to build a second variation of this model. As a third and final variation of this baseline model, we allowed the model to fine-tune the GloVe embeddings so that it could learn better contextual representations from our dataset.

2) Semi-supervised model:

$$\ell_i = \sum_k w_{ik} ||x_i - \mu_k||^2 \tag{3}$$

In the previous baseline model, we used clustering loss as our loss function to optimize the weights for all the layers in our neural network. The cluster loss (Equation 3) for each point (embedding) is calculated as the distance between the point and the center of its assigned (nearest) cluster. The parameter $w_{ik} = 1$, when point $x_i \in k$, and 0 otherwise. However, even though reducing this loss ensures that we get compact clusters, there is no way to guarantee that the clusters formed are the clusters based on biased and non-biased contexts. To avoid this, we decided to modify our cluster assignments such that all the reviews present in the `train_labeled` dataset are always assigned to the annotated cluster regardless of the distance to its center. This helps the model produce more meaningful clusters. Additionally, we also modified our loss function by multiplying the cluster loss for the `train_labeled` with a parameter `lambda` to increase the weights of the loss produced by those reviews. This helps align cluster centroids more closely with embeddings of the labeled set.

We repeated our experiments on all previous models by making this change to the cluster assignments and cluster loss function.

B. LSTM Model

So far all the models we have built estimated the context of a review using a weighted average of the context-free embeddings of the words in that review. This is a naive approach which does not reflect the way humans interpret linguistic context from written sentences. To address this issue, we implemented RNNs that would inherently "learn" the grammar and generate contextualized embeddings for every word in the sequence.

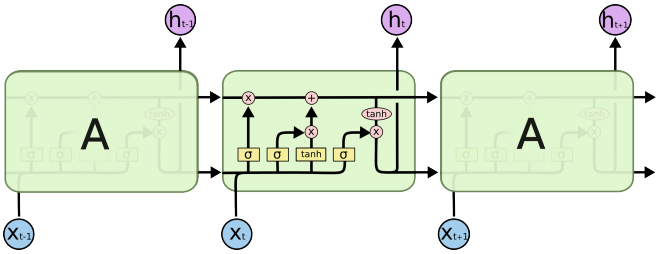


Fig. 3: A typical LSTM cell, src: [11]

RNNs, or recurrent neural networks, are a class of neural networks which allow previous outputs to be used as inputs while having hidden states. The hidden state gives an advantage over traditional neural networks in that information can now flow between two input tokens. But this causes the problem of vanishing gradients, i.e. while learning many layers, the model struggles to tune the initial layers correctly. To address this problem, a new type of RNNs called LSTM (Long Short Term Memory) has been developed. We used LSTMs for this exact reason.

LSTMs have an additional state called the "cell state", through which the model can make adjustments to the flow of information. This provides the model with more flexibility in terms of selectively "remembering" and/or "forgetting" information from the previous states. Figure 3 shows the architecture of a typical LSTM cell. Such a model has been used in previous research to produce contextual embeddings for words in a sequence.

Since LSTM is a directional model and takes inputs sequentially, we used a bi-directional LSTM, which processes text in both left-to-right and right-to-left directions, so that the new embedding for each word would contain information from all the surrounding words in the sequence.

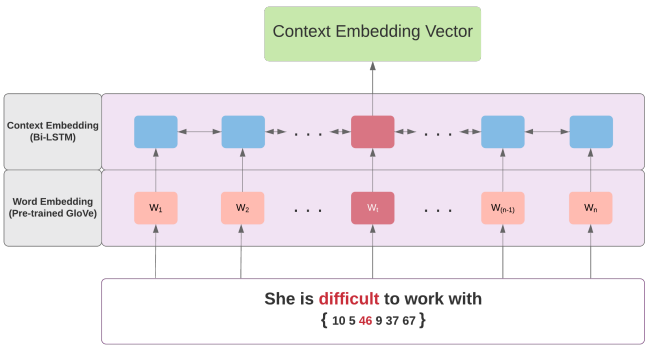


Fig. 4: Architecture of our LSTM model

Figure 4 shows the architecture of our LSTM model. We used the pre-trained GloVe embeddings for each word as our initial input to the Bi-LSTM layer. The output of the Bi-LSTM layer provided contextual embeddings for each word in our sequence. Then, we selected the contextualized embedding of our flagged word and applied the K-means clustering algorithm to group these embeddings into two clusters, biased context and non-biased context.

Our prior models struggled with finding meaningful embedding representations and clustering them at the same time, and instead focused on minimizing the distances within clusters, due to our choice of loss function. Hence, we decided to train this model in two phases, with the first phase being training the contextual embeddings and the second being finding optimal cluster centers.

We set up the first phase as a fully-supervised classification task by applying a linear layer at the end of the LSTM layer. The key objective of this sub-task was to train our models to produce contextualized embeddings for our flagged words. For this task, we only used the smaller dataset of labeled reviews, `train_labeled`. Then, in the second phase, we again applied the semi-supervised K-means clustering algorithm to find two cluster centers which could be used to divide these embeddings into clusters of biased contexts vs. non-biased contexts. We used the complete dataset of labeled and unlabeled reviews for this sub-task. To ensure that our clusters have "meanings", we forced the reviews in the labeled set to be part of the corresponding cluster irrespective of the distance from its center. We also performed hyperparameter tuning on our model to find the ideal values for hyperparameters such as hidden size, number of hidden layers, etc.

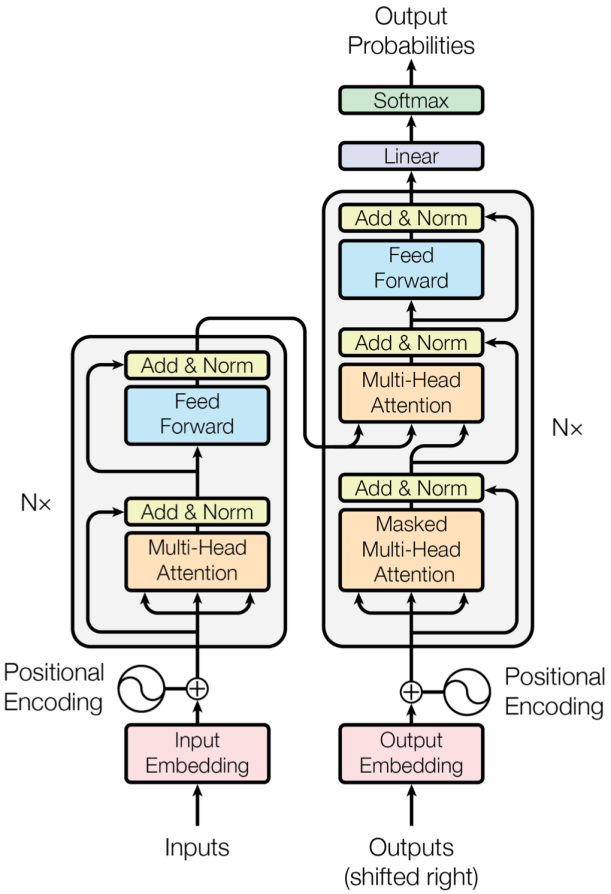


Fig. 5: The Transformer model architecture, src: [12]

C. Pre-Trained BERT Model

Various researchers have shown the significance of transfer learning in the field of NLP, i.e. training a neural network model on a known task, such as Language Modeling, then fine-tuning the pre-trained model on a specific task. We attempted to make use of the capabilities provided by the state-of-the-art BERT model[12].

BERT, or Bidirectional Encoder Representations from Transformers, is a revolutionary open-sourced model developed by researchers at Google in 2018. It has outperformed all traditional models in a variety of different NLP tasks such as Question Answering, Natural Language Inference, etc. Part of the reason for BERT's success is that it is pre-trained on a large corpus of unlabeled text which includes the entire Wikipedia.

One of the major innovations brought by BERT is applying Transformers (as shown in Figure 5) to Language Modeling. All the previous efforts in the field have been based around looking at the context of a sequence in a particular direction (left-to-right or a combination of left-to-right and right-to-left), but the Transformer's unique attention model allows it to learn the context by reading the entire sequence at once. Thus, it is considered bi-directional, although it would be more accurate to call it non-directional. The paper's results show that such a model can have a deeper sense of language context and flow than single-direction language models.

BERT also introduced a novel approach called "Masked LM" in which BERT masks out some words from the input sequence and then conditions each word from the vocabulary to predict the masked word based on the context provided by other non-masked words in the sequence. Another technique introduced by BERT is "Next Sentence Prediction", wherein BERT tries to learn the relationships between pairs of consecutive sentences.

Our final model used BERT to generate the contextualized embeddings instead of the LSTM. We used the BERT-Base-Cased and BERT-Large-Cased models, both in similar fashion as the LSTM model described in section IV-B. The BERT-Base model has 12 layers, uses a hidden size of 768, and a 12-headed attention to train 110M parameters, while the BERT-Large model has 24 layers, uses a hidden size of 1024, and a 16-headed attention to train 340M parameters. The models were first trained to generate a single embedding for our flagged word using context from the words surrounding it. Since the dimension of the hidden size was too high, we added a linear layer after BERT to reduce the embedding dimension to 20 and 50 respectively. These contextual embeddings were trained as part of a classification problem using the `train_labeled` set.

Then, as before, we used the pre-trained model to generate embeddings for the `train` set (including unlabeled reviews) and then applied K-Means algorithm to find two optimal cluster centers that would be used to distinguish between the cluster of embeddings of biased context vs that of the unbiased context.

V. Results

For our evaluation metric, we used F1 score, so as to achieve a balance of both precision and recall in terms of which words our model predicts are or are not biased.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

We list the F1 score from each model in Table I. Our best model, BERT_Large, produced a F1 score of 93.57 in Phase 1. Similarly, and BERT_Base and LSTM models also performed better in Phase 1.

Model		F1 Score
Unsupervised	Random Initialization	<u>70.29</u>
	Pre-trained GloVe Frozen	68.39
	Pre-trained GloVe Unfrozen	69.60
Semi-Supervised	Random Initialization	78.22
	Pre-trained GloVe Frozen	72.21
	Pre-trained GloVe Unfrozen	<u>81.28</u>
LSTM	Phase1	<u>82.66</u>
	Phase2	80.78
BERT_Base	Phase1	<u>85.02</u>
	Phase2	79.25
BERT_Large	Phase1	<u>93.57</u>
	Phase2	77.22

For each model, the best score is underlined

TABLE I: F1 Score on Validation Set

F1 scores on test set is presented in the table II. Here, LSTM worked better than BERT_Large model in both phases.

Model		F1 Score
LSTM	Phase1	<u>93.05</u>
	Phase2	79.81
BERT_Large	Phase1	<u>88.94</u>
	Phase2	76.90

TABLE II: F1 Score on Test Set

Below, Table III shows the hyperparameters we used for the best model from each bag of words based model.

Model	Embed_size	Upweight	λ	n
Unsupervised				
Random Initialization	256	10	N/A	N/A
GloVe Frozen	256	25	N/A	0
GloVe Unfrozen	256	10	N/A	2
Semi-Supervised				
Random Initialization	128	25	1	N/A
GloVe Frozen	128	25	1	0
GloVe Unfrozen	128	25	1	3

λ: weighting of two losses, n: the number of unfrozen epochs

TABLE III: Best Hyper-parameters

For LSTM, the best hyperparameters are: number of unfrozen epochs = 1, number of hidden layers = 3, hidden size = 128, and dropout = 0. BERT is a pretrained model, so we did not perform hyperparameter tuning.

For visualizing the embeddings of the flagged words (including information from surrounding context), we used the UMAP (Uniform Manifold Approximation and Projection) technique [13], which is the state-of-the-art dimension reduction method. The key difference between UMAP and other methods of dimension reduction such as t-SNE (t-Distributed Stochastic Neighbor Embedding) is that UMAP preserves both local and most of the global structure in the data, and is also computationally more efficient. We used UMAP to reduce our 200-dimensional embeddings to 3-dimensional embeddings, which we can easily visualize.

Figure 6 shows the 3-dimensional embedding representation of our best model, BERT-Large, after Phase 1 on the `train_labeled` dataset. Corresponding plots for other models are presented in the Appendix (Figure 8).

Figure 7 shows UMAP projections of embeddings from BERT-Large after Phase 2 on validation set. The centroids of each cluster are represented here by a star. Similar plots for our other models are presented in the Appendix (Figure 9).

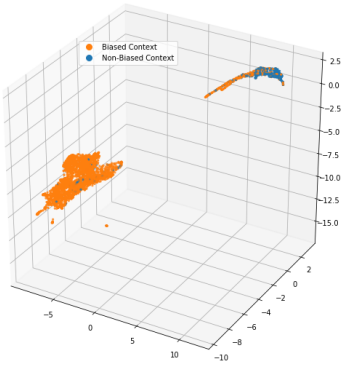


Fig. 6: BERT-Large Embeddings Phase 1

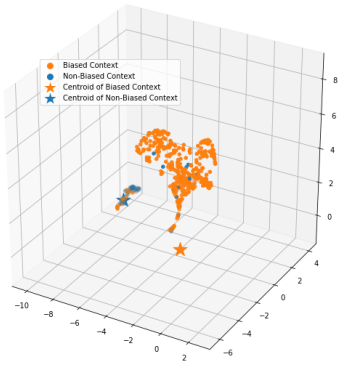


Fig. 7: BERT-Large Embeddings Phase 2

VI. Discussion

As stated previously, we observed that our models performed better in Phase 1 than in Phase 2. Initially, we thought that this may be because our model was unable to learn embeddings that could be separated into clusters during Phase 1. However, as seen in our UMAP figures, the embeddings do appear to be separable. There are many possible reasons for this result. Since Phase 1 was only trained over the smaller sized `train_labeled` dataset, and Phase 2 was trained on both the `train_labeled` and the much larger `train_unlabeled` dataset, the `train_labeled` might not have been a perfect instance of the data generating distribution. Another possibility is that in Phase 1, there is a linear layer that the model could exploit to obtain a higher score, but in Phase 2, the embedding weights are frozen, and the model will be able to identify centroids for the two clusters while producing a lower F1 score. We see this in the result of LSTM on the test set. Although this model was able to achieve an extremely high F1 score (93.05), when we looked more closely at the model's predictions, we found that all the reviews had been assigned to class 1. In Phase 2, the F1 score was not as high (79.81) but the model was able to correctly assign some reviews to cluster 0.

For our baseline models, the upweight parameter w was a crucial component. Since our main goal was to find the contextualized embedding for the flagged word given the words around it, this parameter controlled how much information got passed on to this flagged word's embedding from the other words. That is, having a large value would be closer to the original context-free embedding of the word, while a smaller value would incorporate more from the embedding values of the surrounding words. Thus, we tried to find the optimal value of this parameter that would perform best on the validation set.

Due to the small number of labeled reviews, our models were prone to overfitting on the training set. Hence, we had to be very cautious during training especially with making our model more complex. For example, the two BERT models seemed to overfit on the training set after only 1 epoch. Along the same vein, we observed that the LSTM model performed best with the smallest hidden size we experimented with.

Example	Human	Machine
Let's be honest, if we as customers knew exactly what we wanted and how to do it we' d like	0	1
Made commitments for deliverable without understanding it.	1	0
Excellent designer with a very strong feel for aesthetics and mechanical function.	1	0

TABLE IV: Misclassified Examples

Additionally, since there was an imbalance between the number of reviews in each class in our dataset, our models often assigned every review to class 1. As expected, when using the upsampled set, we observed that the models were better able to identify reviews belonging to class 0. That lead us to conclude that if we have more annotated reviews belonging to class 0, then the model would perform better. In fact, in general, having more annotated reviews with more variation in the text (in terms of style, language, vocab etc) would certainly help avoid overfitting on the training set. However, getting more annotations is itself a tough task because of the ambiguous nature of the task (identifying "unconscious" bias). Oftentimes, it is difficult to actually get a consensus on the true label for a given review. This became increasingly evident when we looked at the reviews which were "misclassified" by our model. One could argue that a lot of these reviews should actually belong to the other class. Some of the misclassified examples by our model are shown in the table IV.

In the first sentence, the model assigned the review to class 1, but this should actually be a false positive, as *honest* was not being used in a biased context. In the next two sentences however, due to the somewhat ambiguous nature of the sentence, although the model and human assignments disagree, the model had actually correctly identified these sentences true positive and false positive, respectively.

Lastly, we had assumed that assigning higher weights to the cluster loss associated with the `train_labeled` set should help our model find the cluster centers close to the actual optimal cluster center points. However, we observed that this was not in fact the case: assigning equal weights to the losses of both `train_labeled` and `train_unlabeled` sets (i.e. $\lambda = 1$) returned a higher F1 score during hyperparameter tuning.

VII. Conclusion

In this paper, we applied deep learning methods to train contextual embeddings for flagged words to evaluate whether they are used in a way that communicates unconscious gender bias. We successfully applied a semi-supervised learning approach to utilize a small subset of labeled data. Ultimately,

cutting edge models like BERT were able to simulate bias detection in our dataset comparable to human performance.

VIII. Future Work

Our current model is trained on publicly viewable reviews of freelancers. However, this is a quite different from FairFrame's most common use case as a platform companies would utilize to detect gender bias in employee evaluations. Obtaining a more diverse dataset with language more similar to that of a typical company employee evaluation would help generate models more closely aligned to FairFrame's needs. Another point of improvement would be adapting our current architecture to produce contextual representations for sentences with phrases found in a biased context, as opposed to just individual words. Additionally, although the results we obtained from fine-tuning BERT are quite good, it is prone to overfitting. Therefore, further work into modifying our current architecture with transformers, GRU's, highway networks, etc., as well as incorporating certain contextual features such as part-of-speech tags and named entity representations may help improve our model performance while avoiding overfitting.

IX. Acknowledgments

We would like to thank our advisors Amy Auton-Smith and Isak Nti Asare from FairFrame, as well as the NYU Capstone advisors for their help and support.

References

[1] Monica Rubini and Michela Menegatti. "Linguistic Bias in Personnel Selection". In: *Journal of Language and Social Psychology* 27.2 (2008), pp. 168–181. DOI: 10.1177/0261927X07313653. eprint: https://doi.org/10.1177/0261927X07313653. URL: https://doi.org/10.1177/0261927X07313653.

[2] Michela Menegatti and Monica Rubini. *Gender Bias and Sexism in Language*. Sept. 2019. URL: https://oxfordre.com/communication/communication/view/10.1093/acrefore/9780190228613.001.0001/acrefore-9780190228613-e-470?source=post_page----------------------------.

[3] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. "Linguistic Models for Analyzing and Detecting Biased Language". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1650–1659. URL: https://www.aclweb.org/anthology/P13-1162.

[4] Rachel Rudinger et al. "Gender Bias in Coreference Resolution". In: *CoRR* abs/1804.09301 (2018). arXiv: 1804.09301. URL: http://arxiv.org/abs/1804.09301.

[5] Jieyu Zhao et al. *Learning Gender-Neutral Word Embeddings*. 2018. arXiv: 1809.01496 [cs.CL].

[6] Jieyu Zhao et al. *Gender Bias in Contextualized Word Embeddings*. 2019. arXiv: 1904.03310 [cs.CL].

[7] Eric Bair. "Semi-supervised clustering methods". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.5 (July 2013), pp. 349–361. ISSN: 1939-5108. DOI: 10.1002/wics.1270. URL: http://dx.doi.org/10.1002/wics.1270.

[8] Joseph Turian, Lev Ratinov, and Yoshua Bengio. "Word Representations: A Simple and General Method for Semi-supervised Learning". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL '10. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 384–394. URL: http://dl.acm.org/citation.cfm?id=1858681.1858721.

[9] Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. ETMTNLP '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 63–70. DOI: 10.3115/1118108.1118117. URL: https://doi.org/10.3115/1118108.1118117.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

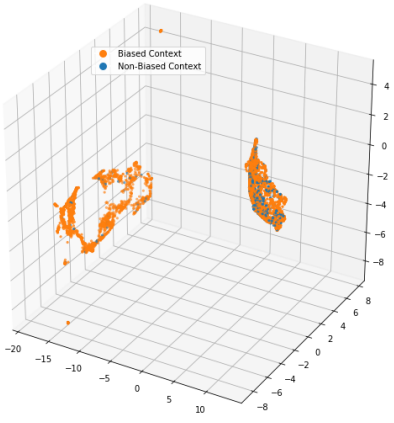
[11] *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[12] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

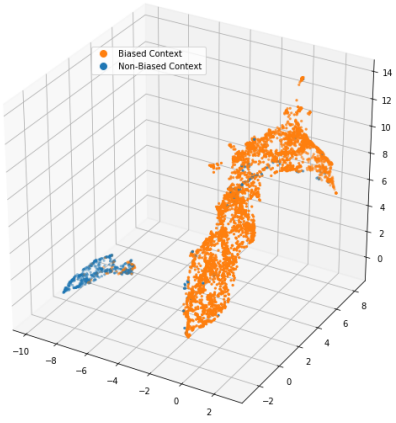
[13] John Healy Leland McInnes and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

APPENDIX

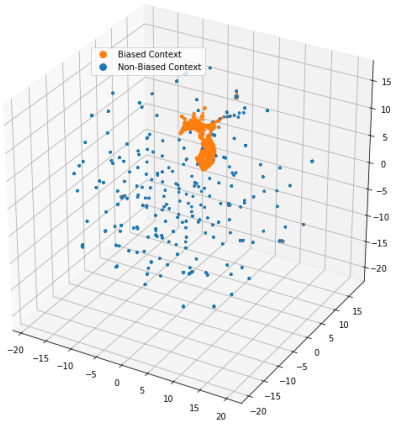
A. *UMAP Projections*



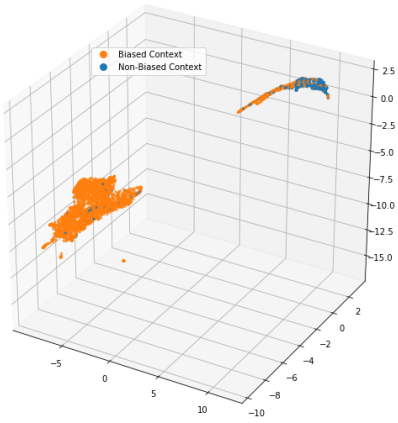
(a) Unsupervised Weighted Embeddings



(b) Semi-Supervised Weighted Embeddings

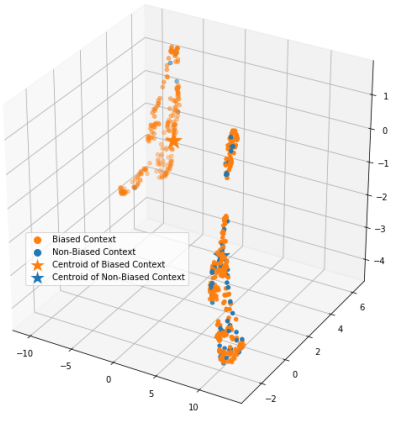


(c) LSTM Embeddings

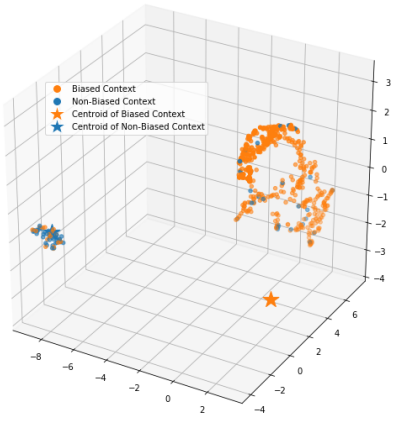


(d) BERT Embeddings

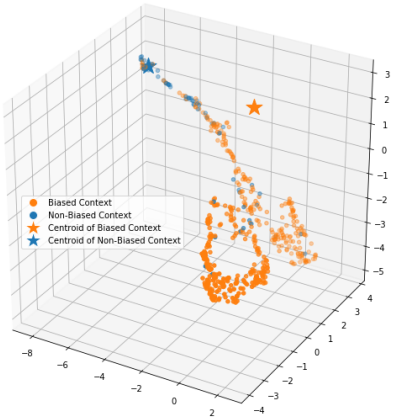
Fig. 8: UMAP Projection On Labeled Training Set



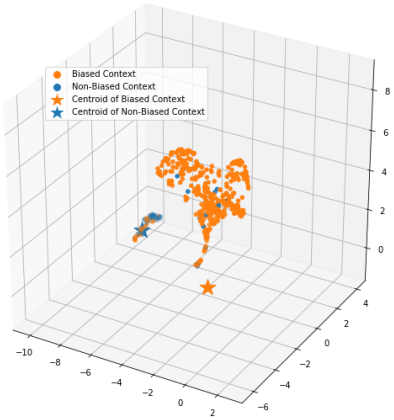
(a) Unsupervised Weighted Embeddings



(b) Semi-Supervised Weighted Embeddings



(c) LSTM Embeddings



(d) BERT Embeddings

Fig. 9: UMAP Projection On Validation Set