Bias in Artificial Intelligence Project

Finlay Boyle
Department of Computer Science
Durham University
Durham, United Kingdom
finlay.boyle@durham.ac.uk

***Abstract*—Natural language processing (NLP) combines artificial intelligence and linguistics to process and extract meaning from spoken or written natural language. A subset of NLP is sentiment analysis which is the processing of language and structure of a text to determine the emotions, and their strength, within the text. In this project I will explore the effect of bias on sentiment polarity across demographic groups specifically relating to gender identity terms for males and females. The project focuses on the effect of reducing sentiment polarity within the word embedding vectors themselves to affect downstream applications related to valence prediction.**

***Index Terms*—embeddings, fairness, NLP**

I. Project Proposal

Sentiment is a concept that is closely related to the positive and negative emotions conveyed in speech and text. Sentiment polarity is an important part of NLP to explore in relation to bias because machine learning is used heavily for automated moderation tasks such as removing toxic comments within forum boards, social media, and news websites. This could unintentional filter the narrative of conversations [1] online by misinterpreting the sentiment of a sentence which could have social consequences in relation to important discussions, especially those taking place around the world at the moment, if minority voices are filtered out and suppressed by machine learning algorithms if they are biased against minority groups and believe the comments are toxic when they are truly not. This is not limited to gendered phrases but it also expands to the sentiment encoded within these algorithms about African-American names versus European-American names which is also examined in [2].

The project that I am going to develop will be focused on reducing sentiment polarity between demographic identities in word embeddings directly using a technique called adversarial learning as explored by [2] which I will loosely follow throughout my project.

To begin, I will need a set of word embedding vectors. I have chosen to use a pre-trained model provided by the Python library gensim [3] which has been trained on the Google News corpus[1] using the word2vec technique described in [4] which was recommended by the paper that I am following. This will form the basis for any debiasing which will transform the vector space of word embeddings.

As detailed in [2] the first concrete task will be to determine a positive and negative sentiment subspace within the vector space of word embeddings. To do this, I will use the sentiment lexicons[2] provided by [5] to determine these subspaces. Then, using a technique called Principal Component Analysis which defines orthonormal component axes using eigenvectors in the directions explaining the most variance within the vector subspace [6]. I will examine the explained variances within the subspaces and compute the directional sentiment vector connecting these which I will explore further in the progress report.

Then, I will be able to train the adversarial debiaser to reduce the sentiment polarity nested within the word embeddings. This is the primary section of the project which I will develop in much more depth in the progress section of this report where I will implement this using mini-batch stochastic gradient descent.

The next task will be to evaluate the effect of the debiasing by using the SemEval-2018 Task 1: Affect in Tweets data set[3] [7] to train a Support Vector Regression (SVR) model to predict the valence - a numeric measure of sentiment polarity - of text. Once this has been trained, I will finally evaluate the debiasing by using the Equity Evaluation Corpus[4] [8] which consists of a variety of sentence templates using different noun phrases specific to each gender[5]. The purpose of this will be to compare the difference in the effect that using the biased and the debiased word embeddings for the training of the SVR model has on the valence bias.

My project will be coded in Python 3.8.2 using gensim 4.0.1, scikit-learn 0.23.2, numpy 1.19.2, matplotlib, pandas 1.1.3 and scipy 1.5.3. I will implement the adversarial debiasing machine learning algorithm directly and use the SVR model provided by scikit-learn for evaluation purposes. This project requires quite a few data sets, each data set download link is included as a footnote.

II. Progress Report

A. Data Analysis

My project uses a combination of different data sets. I will focus on analysing the SemEval-2018 Task 1: Affect in Tweets valence training data[3] as it is used to train the model for evaluation of the debiaser. This data set is a collection of

[1] downloaded via gensim https://code.google.com/archive/p/word2vec/

[2] http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

[3] http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/SemEval2018-Task1-all-data.zip

[4] http://saifmohammad.com/WebPages/Biases-SA.html

[5] e.g. she/he, her/him, my aunt/my uncle etc

tweets collected via the Twitter API during June 2017 and July 2017 [7]. Each tweet has been assigned a valence score from 0 to 1. Valence is a numeric measure closely related to the intensity of sentiment within some text. The columns provided are ID, Tweet, Affect Dimension, and Intensity Score. There are 1181 records provided.

The initial file comes as a text file where the columns are tab separated. I chose to convert this to a CSV during my initial data analysis so that it can easily be manipulated via pandas using the native CSV support.

As this is the valence data training set, the 'Affect Dimension' of all of the records is 'valence'. This is because the data set is part of a larger collection that was used for a competition[6]. This column does not provide any information so it was removed from the data set. The mean valence was 0.500, the standard deviation was 0.209, and the maximum and minimum values were 0.0170 and 0.968 respectively (all to 3 significant figures).

Finally, I transformed the tweets into vectors using the word embeddings. To do this I use the mean of the word vectors for each word in the tweets (if they are present in the word embeddings) to compute an averaged sentence vector for each tweet. Then, I store each vector component of this average vector as a feature in the data frame (it introduces an additional 300 features). Then I drop the ID column and the initial Tweet column as they are no longer relevant to the processing of the text as it has now been fully vectorised and can be used to train my machine learning model. This completes the transformation of the training set. The testing set (for validation only rather than evaluation of the bias) followed the exact same routine.

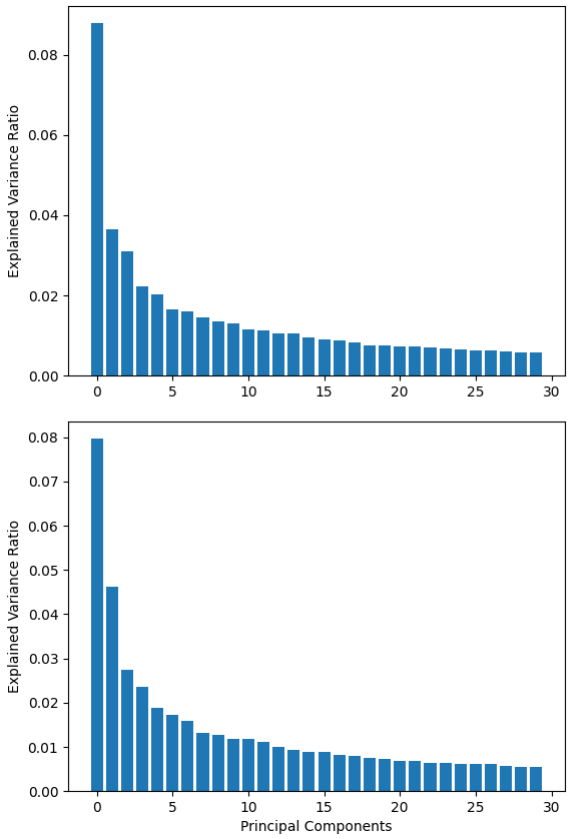

Fig. 1. The bar charts show the explained variance attributed to the top 30 principal components of each subspace. Both have significant variance attributed to the first component. **Top**: PCA of positive sentiment subspace. **Bottom**: PCA of negative sentiment subspace.

B. Principal Component Analysis

Prior to debiasing I needed to compute the *directional sentiment vector* detailed in [2] which connects the positive and negative subspaces as determined by the sentiment lexicons provided by [5]. To find this vector I perform Principal Component Analysis on each subspace which is the process of finding orthonormal axes in the directions of the most significant uncorrelated variances [6] within a vector space.

As evidenced by Fig. 1, it is evident that the first principal component contains significantly more explained variance than the other components. The shape is similar to that found in Figure 2 of [5]. However, it was unclear what was on the y-axis in the paper and the chart had an unusual shape in that positive sentiment graph is not monotone decreasing. My results differ slightly from the original paper because we are using slightly different variations of the word embeddings.

To compute the directional sentiment vector I took the difference between the most significant principal component of the positive subspace and the most significant principal component of the negative subspace. To validate that this vector was suitable to encode the sentiment, I measure the accuracy and precision of classifying the sentiment lexicon words by projecting their vectors from the word embeddings on to the directional sentiment vector. If it is negative then I classify the word as having negative sentiment and if it is positive I classify it as having positive sentiment. This produces an accuracy of 88.2% and a precision of 90.2%[7] suggesting that the directional sentiment vector is appropriate.

C. Adversarial Debiaser

After having computed the directional sentiment vector I use this to create the debiaser. The debiaser is a machine learning algorithm with two loss functions: the standard loss and the adversarial loss. This is important as while the standard objective focuses on reducing the sentiment polarity of a word vector the competing adversarial objective attempts to keep the word vector in a neighbourhood of the original position within the subspace. This is important to ensure that reducing sentiment does not distort the meaning of the word.

This technique was originally presented in [9] which defines two separate vectors of weights $\mathbf{W}$, the weights associated with reducing the sentiment, and $\mathbf{U}$, the weights associated with the adversarial objective. First, I define $\mathbf{y}$ as the original

[6]https://competitions.codalab.org/competitions/17751

[7]compared to 91.3% precision in the original paper, accuracy was not reported in the paper

word vector, $\mathbf{k}$ as the directional sentiment vector and $\hat{\mathbf{y}}$ as the debiased word vector calculated as:

$$\hat{\mathbf{y}} := \mathbf{y} - \mathbf{W}\mathbf{W}^\mathbf{t}\mathbf{y} \tag{1}$$

Then, the standard loss function, L_P, and the adversary loss function, L_A, defined as:

$$L_P := (\mathbf{y} - \hat{\mathbf{y}})^2 = (\mathbf{W}\mathbf{W}^\mathbf{t}\mathbf{y})^2 \tag{2}$$

$$L_A := (\mathbf{k}^\mathbf{t}\mathbf{y} - \mathbf{U}^\mathbf{t}\hat{\mathbf{y}})^2 \tag{3}$$

These are defined in [2]. The square in L_P is representing mean-square distance so L_P is equivalently defined as:

$$L_P := \frac{1}{m}\sum_{i=1}^{m}([\mathbf{W}\mathbf{W}^\mathbf{t}\mathbf{y}]_i)^2 \tag{4}$$

where m is the number of features (i.e. 300) and $[\mathbf{P}]_i$ is the i[th] component of the vector $\mathbf{P}$. The objective to minimise is defined[8] as:

$$\nabla_\mathbf{W} L_P - \text{proj}_{\nabla_\mathbf{W} L_A}\nabla_\mathbf{W} L_P - \alpha\nabla_\mathbf{W} L_A \tag{5}$$

where $\nabla_\mathbf{P} Q$ is the gradient of Q (a scalar field) with respect to $\mathbf{P}$ (a vector), $\text{proj}_\mathbf{P}\mathbf{R}$ is the vector projection of $\mathbf{R}$ on to $\mathbf{P}$ and α is a hyperparameter. I have calculated[9] the j[th] component of the gradients and these are:

$$[\nabla_\mathbf{W} L_P]_j = \frac{2\mathbf{y}_j}{m}\sum_{i=1}^{m}\mathbf{W}_i[\mathbf{W}\mathbf{W}^\mathbf{t}\mathbf{y}]_i \tag{6}$$

$$[\nabla_\mathbf{W} L_A]_j = 2(\mathbf{k}.\mathbf{y} - \mathbf{U}.\hat{\mathbf{y}})(\mathbf{W}.\mathbf{y})(\mathbf{U}_j + \mathbf{y}_j) \tag{7}$$

and finally, $\nabla_\mathbf{U} L_A$ is used to modify U:

$$[\nabla_\mathbf{U} L_A]_j = 2(\mathbf{U}.\hat{\mathbf{y}} - \mathbf{k}.\mathbf{y})\hat{\mathbf{y}}_j. \tag{8}$$

To optimise the weights according to the objective using the loss functions I have implemented mini-batch stochastic gradient descent which uses an implementation of the Adam Optimiser [10] optimisation algorithm to update the weight according to the gradients. I implemented both Adam and the mini-batch gradient descent directly in Python using numpy for speed. In my project I trained the adversarial debiaser on 40000 batches of 1000 word vectors from the word embeddings as recommended by the original paper.

D. Valence Regression Results

Once the debiaser has been trained, I debiased the entire word embedding vector space using equation (1) with the trained weights. I then use the SemEval-2018 Task 1: Affect in Tweets valence training data [7] to train a Support Vector Regression (SVR) model [11] from scikit-learn [12]. To do this, I transform the valence data set into the valence score and the sentences in to averaged vector form.

Before evaluating the effect of the debiaser on the sentiment bias I present my findings on the accuracy of the model before and after debiasing of the word embeddings by using the SemEval-2018 Gold-Standard Test Set from [7]. In [5] the Pearson's Correlation Coefficient (PCC) is used to determine the success of the model and achieve scores of 42% before debiasing and 43% after debiasing. I use the same metrics and before debiasing the SVR model has a PCC of 47.2% and after it has a PCC of 46.7% which indicates that there was a very slight decrease in the accuracy of the model. However, this is a small trade-off for the improvements in the bias reduction.

I evaluated the model on the EEC [8] by calculating the valence for sentences differing only by the gender associated with the noun phrase in the sentence template. I take the difference between the male valence and the female valence. If it is positive then it is biased in favour of males, if it is negative then it is biased in favour of females.

Table I compares the valence bias before and after debiasing of the word embeddings. The purpose of debiasing is for the mean difference in the valence to be closer to 0. This would indicate that there is less sentiment difference between the same sentences with different gender-associated noun phrases as well as less variance between words. It uses the same notation introduced in [8] where **F↑-↓M** means the sentence valence delta was in favour of females and **F↓-↑M** means the sentence valence delta was in favour of males.

Prior to debiasing there is a clear bias with the majority of sentence pairs being awarded a higher valence when the noun phrase is associated with a female descriptor. There is also a significant portion in favour of males too. The model is initially biased to predict that emotions and sentiment are more strongly conveyed when the person involved is female. The purpose of the debiasing in this case is to attempt to equalise the strength of emotion between two equal sentences.

The debiasing is successful in reducing the mean valence difference in both male and female biased sentences and it leads to a significant reduction in sentiment bias in favour of females by 45.1% and it also decreases the sentiment bias in favour of males by 14.2%. These results are weaker than those from [2] but are still significant improvements. The difference in these values can be explained by using different data sets. They use the Glove word embeddings which are trained on a different text corpus [13] instead of the Google News trained word embeddings. The implementation of the algorithms almost certainly differs too as well as their processing of the lexicons and calculation of the directional sentiment vector. Ultimately, the process is stochastic in nature too and so it would not be possible to replicate their results exactly.

TABLE I
VALENCE EVALUATION METRICS

Valence Metrics	Biased		Debiased	
	F↑-↓M	**F↓-↑M**	**F↑-↓M**	**F↓-↑M**
Biased Pairs	1192	248	864	576
Mean Difference	2.33e−2	1.07e−2	1.28e−2	9.18e−3
Variance	2.69e−4	1.47e−5	1.24e−4	3.44e−5

[8]there was a disagreement between the objective functions in [2] and [9], I chose to agree with the latter

[9]due to space limitations the proofs have been omitted

REFERENCES

[1] R. Binns, M. Veale, M. V. Kleek, and N. Shadbolt, "Like trainer, like bot? inheritance of bias in algorithmic content moderation," *CoRR*, vol. abs/1707.01477, 2017. [Online]. Available: http://arxiv.org/abs/1707.01477

[2] C. Sweeney and M. Najafian, "Reducing sentiment polarity for demographic attributes in word embeddings using adversarial learning," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT* '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 359–368. [Online]. Available: https://doi.org/10.1145/3351095.3372837

[3] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

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

[5] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 168–177. [Online]. Available: https://doi.org/10.1145/1014052.1014073

[6] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[7] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "SemEval-2018 task 1: Affect in tweets," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1–17. [Online]. Available: https://www.aclweb.org/anthology/S18-1001

[8] S. Kiritchenko and S. M. Mohammad, "Examining gender and race bias in two hundred sentiment analysis systems," *CoRR*, vol. abs/1805.04508, 2018. [Online]. Available: http://arxiv.org/abs/1805.04508

[9] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 335–340. [Online]. Available: https://doi.org/10.1145/3278721.3278779

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik *et al.*, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, 01 2014, pp. 1532–1543.