Embeddings for Classifying Racist Tweets

Raymond Ye Lee [*1] **Cindy Xuan** [*1]

1. Introduction

The racial dynamics surrounding anti-Asian racism in the United States are complex for a number of reasons including the pandemic, state tensions, rhetoric by elected officials (15; 3), and ignorance (5). Though Tweets are an arguably biased representation of public sentiment, they nevertheless serve as a barometer more open and representative than traditional media outlets. Our project centers around exploring word embeddings—wherein documents (here, tweets) are represented in a vector space—in order to understand their effectiveness in representing tweets, and how these representations might be used for predicting racial animus.

Using a hand-annotated dataset of tweets (20), we constructed word embeddings with latent semantic analysis (9), word2vec (11), and GloVe (16). Then, using these obtained embeddings as input, we built k-nearest neighbors, logistic regression, support vector machine, and neural network classifiers to predict the racist character of the tweet as denoted by the annotation.

2. Related work

The most related paper is Ziems et al. (2020), which uses the same dataset for a portion of their analyses in classifying tweets. However, our approach differs substantively: embeddings are obtained by Ziems et al. (2020) via BERT and GloVE, whereas we also use latent semantic indexing and word2vec, both of which have been used fruitfully as benchmarks for text classification (12).

Moreover, embeddings comprise only one of three sets features used as input into their classifiers; they also look at linguistic features and hashtag. On the other hand, we are much more interested in the embeddings per se, and accordingly provide discussion of our embedding spaces. Ziems et al. (2020), on the other hand, devotes little attention to the actual space in which the resulting embeddings live. The set of classification algorithms we train also differ from that in Ziems et al. (2020); whereas Ziems et al. (2020) uses random forest in addition to logistic regression and support vector machine, we include k-nearest neighbors and feed-forward neural networks, motivated by the properties afforded by each.

In addition to Ziems et al. (2020), other papers have also looked at anti-Asian sentiment using Twitter data during the pandemic; in one example, the authors use a transformers-based approach to detect hateful sentiment (18). More broadly, there is a rich literature on sentiment analysis of Twitter text data (7), including using machine learning techniques (14) together with embeddings (8). However, our work differs in that we are interested in a particular type of speech on Twitter (those centered around hate), and the comparison of particular embeddings (i.e., a certain type of feature).

3. Dataset and Features

We used a dataset containing 2,319 tweets occurring over a three-month period from January 15, 2020 to April 17, 2020 (20). Over 30 million English-language tweets were obtained through keyword searches in the dataset at large; we analyze a subset hand-annotated as expressing either hate against Asians (labeled "Hate"), hate against non-Asians (labeled "Non-Asian Aggression"), Neutral, or "Counterhate".Table 1 provides the class distribution.

Class	Count	Share
Hate	678	0.29
Non-Asian Aggression	321	0.14
Neutral	961	0.41
Counterhate	359	0.16
Total	**2,319**	**1.00**

Table 1. Distribution of classes in data.

We applied the same pre-processing procedure to all tweets in the data, which involved, in order: lowering case; removing hashtags [1], URLs, and emojis; keeping only alphabetic symbols, lemmatizing using spaCy (2), tokenizing, remov-

[*]Equal contribution [1]Management Science and Engineering Department, Stanford University. Correspondence to: Raymond Ye Lee and Cindy Xuan <ryelee@stanford.edu, cindyx@stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Hashtags can contain valuable information about sentiment, and so we cache these separately but do not analyze them for this project

ing tokens considered stop words in both spaCy's and Natural Language Toolkit's (1) English-language dictionaries, removing strings of length two or less, and removing tokens that appear only once.

This leaves us with 2,312 tweets (7 tweets consisted entirely of URLs, hashtags, stop words, and/or mentions). We further split these tweets into a training set and a test set according to a 90-10 split.

The embedding for a given tweet is the element-wise mean of the word embedding for each constituent token. Since our dataset is rather modest and because the smallest pre-trained embedding available for both GloVe and word2vec is 100, we opt for an embedding size of 100 across all embedding algorithms. We then normalize across examples each of these 100 features in order to obtain the design matrix we use as input. We also use pre-trained embeddings we describe in Section . For tokens idiosyncratic in our dataset, we assign it the mean vector across all vectors in the pre-trained embedding to align with the distribution of vectors in the pre-trained embedding space.

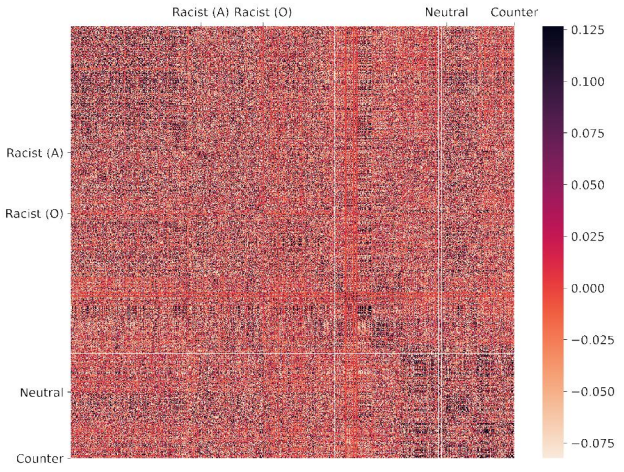

Figure 1. Heatmap of cosine similarities between embeddings of tweets in the training set. For visualization, similarities are winsorized at the 10th and 90th percentiles. Columns and rows ordered so that axes ticks denote the last index of a tweet from the labeled category. Racist (A) denotes anti-Asian while Racist (N) denotes non-Asian. Ordering is arbitrary within a given category.

4. Methods

4.1. Embeddings

4.1.1. Latent semantic indexing

We use latent semantic indexing (also known as latent semantic analysis, LSA, and LSI) as a first approach for representing our tweets in vector space. Let $A \in R^{m \times n}$ be a term-document matrix so that the (i, j)th row contains the number of occurrences of term (word) i in document j (implying that there are m words in the universe and n documents of interest). Taking the singular value decomposition $A = U\Sigma V^T$, note that our matrix A can be thought of as being composed of different linear combinations of the columns of U with coefficients given by SV^T. We can think of the ith column of U as our vector representation of the ith word.[2] As U contains the eigenvectors of AA^T, this representation is based on co-occurrence of terms between documents. Following previous work, we use cosine similarity (distance) as our similarity (distance) metric (9).

From Figure 1, we see that despite being a rather naive embedding, comparisons in the vector space constructed by latent semantic indexing can reveal similarities between objects. In particular, we see that tweets that express racism against Asians tend to be similar to each other in this space and relatively less similar to those from other categories. This is also the case for tweets expressing counter-hate. We also see bands of (dis)similarity throughout, some of which are correlated with category membership, suggesting that sans any modeling, the vector space that latent semantic indexing constructs can be useful despite being relatively less sophisticated.

[2]In principle, the rows of V^T provide tweet-level embeddings. However, we use the word representation from U for fair comparison against GloVe and word2vec.

4.1.2. Word2vec

Relying only on global co-occurrence of words in documents elides context, motivating motivates word2vec (11), which constructs word vectors by first initializing an embedding for every word in some fixed dictionary, and then adjusting these vectors as we iterate through each position in some text so that the probability computed from the similarity of their vector representations of seeing some word given adjacent words in a rolling window (or vice-versa, predicting the adjacent words taking the "center" word as input) is maximized using a shallow feed-forward neural network. The first flavor is dubbed continuous bag of words (CBOW); the second, skip-gram.

For this project, we use the skip-gram variation since empirically it works well for smaller datasets, "represents well even rare words or phrases" (especially important given that tweet text data is often rife with internet colloquialisms), and aligns with the pre-trained embeddings we use (10). Also for fairness of comparison with the pre-trained embeddings, we use negative sampling; at a high level, for a given context-center token pair, we randomly sample words to compute our conditional probability described above, instead of computing exhaustively over the entire vocabulary.

4.1.3. GLOVE

GloVe ("Global Vectors") combines the idea of leveraging global co-occurences (as in LSA) as well as the context (as in word2vec) to form an embedding that captures both global statistics and local statistics in a corpus (16). That is, similar to LSA, we look at co-occurences of words. However, unlike LSA and more like word2vec, we look at co-occurrences within a given window (e.g., how often two words appear within q words of each other in a given document).

We then compute, for a given corpus with V words, the co-occurrence matrix $X \in R^{V \times V}$. For given words indexed i, k, $P_{ik} = X_{ik}/X_i$ denotes the probability of seeing words indexed i and k together. The ratio P_{ik}/P_{jk} gives us useful global statistics: given two words indexed i and j, if a third word k is very similar to i but irrelevant to j, then the ratio is high; it's low if k is very similar to j but irrelevant to i; is close to 1 if k is roughly equally related or unrelated to either word. GloVe then constructs an embedding by iteratively optimizing a loss based on this ratio; see Pennington et al. (2014) for more details.

4.1.4. VISUALIZING WORD2VEC AND GLOVE

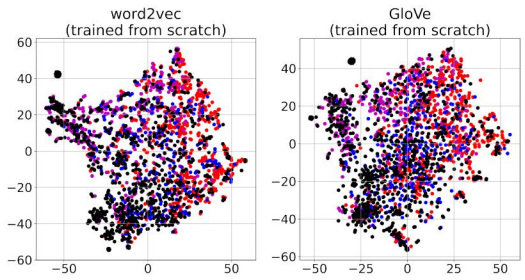

Figure 2. 100-dimensional word2vec and GloVe embeddings in 2-dimensional t-SNE space for tweets in the training set. Hate, Counterhate, Non-Asian Aggression, and Neutral tweets colored red, magenta, blue, and black, respectively.

As we saw with in Figure 1, a heatmap of cosine similarities between tweet embeddings constructed from word embeddings given by latent semantic indexing revealed a clustering structure sans additional supervision. We do a similar visualization exercise here using t-SNE, which is a stochastic dimensionality reduction algorithm that, at a high level for the sake of space, models a probability distribution for each point over neighbors in each of the original space as well as the reduced space and minimizes their Kullback-Leibler divergence.

We see that here, too, clustering is evident though not obviously linearly separable, providing motivation for the non-linear classifiers we implement.

4.1.5. BIAS IN EMBEDDINGS

Despite observable clustering, we might suspect that our embeddings are biased from a relatively small and idiosyncratic sample. Indeed, we find that the closest neighbors to the token "china" in our LSA embedding include "beat", "bomb", and "hitler". Word2vec returns more innocuous results, including "restaurant" and "takeout", while the nearest neighbors in GloVe include "danger" and "factory", reflecting the negative bias of the text in the data.

Analogy task results are similarly bizarre. For a vector obtained via the operation $v = v_{\text{america}} - v_{\text{american}} + v_{\text{china}}$, the closest vectors are "american", "america", and "jew" for LSA, word2vec, and GloVe, respectively, suggesting that our embeddings should not be thought of as reprentative of English language text in general.

4.2. Classification algorithms

As noted above, we constructed classifiers using k-nearest neighbors, logistic regression, support vector machine, and feed-forward neural networks.

4.2.1. k-NEAREST NEIGHBORS

The k-nearest neighbors algorithm simple: the prediction for a query is determined by plurality vote of its k nearest neighbors. It also has the benefit of being non-parametric which, given that our data points live in R^{100}, seems like a suitable benchmark. A variation treats the distance betwen neighbors non-uniformly by according each neighbor a weight inversely proportional to the Euclidean distance. In our experiments, we choose both k and this weighting decision through cross-validation.

4.2.2. MULTINOMIAL LOGISTIC REGRESSION

As another benchmark, we might wonder if a linear discriminative classifier might suffice for our problem. Since we have more than just two classes, we use multinomial flavor of logistic regression (MNL). For K classes, MNL is equivalent to running $K-1$ independent binary logistic regression models, with each outcome K taking a turn as the "pivot". That is, we compute for each $k \in \{1, 2, \ldots, K-1\}$:

$$\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = k)} = f(k, i) \tag{1}$$

where $f(X_i, k)$ is the predictor function associated with assigning observation i to category k defined as

$$f(\mathbf{X}_i, k) = \boldsymbol{\beta}_k \cdot \mathbf{X}_i$$

where β_k is a vector of weights (or regression coefficients) corresponding to outcome k so that:

$$\Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot X_i}}. \quad (2)$$

We add an L2-regularization term to avoid non-convergence of gradient descent in the case of perfect separability, and to prevent overfitting to certain features. The penalty parameter λ is chosen via cross-validation.

4.2.3. Support vector machine

As a bridge between linear and non-linear classifiers depending on the kernel used, we construct a support vector machine classifier. Let w, x be points and $\phi(x)$ represent the kernel function that maps points in the input space into a kernel space. Denote $\phi(w)^T\phi(x) = K(w, x)$. The primal problem that SVM solves in order to obtain the optimal hyperplane is

$$\phi^*(w) = \arg\min_{\phi(w)} \frac{1}{2}\phi(w)^2, \quad (3)$$

$$\text{such that} \quad y_i(\phi(w)^T\phi(x_i) + b) \geq 1 \quad (4)$$

for bias term b. Manipulating the dual, we obtain the classifier:

$$g(x) = \text{sign}\left(\sum_i y_i\alpha_i\phi(x_i) \cdot \phi(x) + b\right) \quad (5)$$

$$= \text{sign}\left(\sum_i y_i\alpha_i K(x_i, x) + b\right). \quad (6)$$

where the values of α_i are obtained from the dual problem.

Despite being in principle a linear classifier, we use a kernel function so that the optimal hyperplane lives in a potentially high-dimensional kernel space so that it corresponds to a non-linear separation in the original space. Since the radial basis function kernel can be universally approximating (17) and so might serve us well in the embedding space, we cross-validate for the choice of kernel as well as a L2-regularization penalty term.

4.2.4. Feed-forward neural network

We also implement a simple feed-forward neural network, tuning for both the number of affine transformations (i.e., fully-connected layers) as well as the inclusion of dropout layers (wherein some outputs from the preceding affine transformation are removed to prevent overfitting) and batch normalization layers (wherein features are normalized across examples before, in our case, being fed into the next fully-connected layer to stabilize the learning process (e.g., prevent backpropagating extreme gradient values) (4). For all hidden layers, we heuristically use rectified linear unit activations, which takes the element-wise maximum between the activation value and 0 (Karpathy).

Similar to our change from binary logistic regression to multinomial logistic regression, our output layer is a softmax function—which first exponentiates each activation output by the previous hidden layer, then normalizes by the sum of all exponentiated scores—to provide probability scores for each of the four classes, and our loss function is categorical cross-entropy rather than binary.

We tune for the number of hidden layers (limiting the depth at three due to computational constraints), the number of neurons in each dense layer, and the dropout rate by cross-validation. Following CS 231n lecture notes, we considered a batch size of 32 but ultimately opted for 10 since that worked well in our experiments and so that the algorithm take more steps per epoch (with the caveat that those steps may result from a noisier gradient than for larger batches).

5. Results

As we saw in Section 4.1, there's evidence that our word embeddings, even for more common tokens less idiosyncratic to Twitter, are likely not representative of English text at large. Then since our tweet-level embeddings are constructed directly from the word-level embeddings, this bias will also affect our tweet-level embeddings and thus be consequential for classification. To mitigate this, we also use pre-trained embeddings: those trained on Wikipedia for word2vec and GloVe, and Twitter for GloVe, since text on Wikipedia in general may be different in general that on Twitter corpus, whereas the pre-trained Twitter embeddings may be more suitable to our setting (16; 19). We use the 100-dimensional version of these embeddings given our dataset size.

We ultimately construct one classifier for each embedding-classification algorithm pair, and we noted in Section 4.2 the cross-validated hyperparameters for each classification algorithm. Given the size of our dataset, we chose 5-fold cross-validation so that each fold contained approximate 400 examples.

Our metrics of choice included the accuracy (the share of predictions that were correct, the AUROC (the area under the curve formed by shifting the classifier threshold and plotting the resulting values of true positive rate against false positive rate, where 0.5 indicates no discriminatory power and 1 indicates a perfect classifier), and the F1-score (the harmonic mean of precision and recall): $F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$ where precision is the share of true positives out of all predicted positives, and recall is the share of true positives of all predicted positives. We used the weighted F1-score so that scores are calculated for

each label and then their average is weighted by the number of true instances for each label. Similarly, the accuracy is calculated as the average accuracy across classes.

Our results are as follows:

Embedding	Acc. (Tr)	Acc. (Te)	AUC (Tr)	AUC (Te)	F1 (Tr)	F1 (Te)
LSA	0.995	0.459	0.999	0.690	0.995	0.425
w2v	0.997	0.472	1.000	0.685	0.997	0.434
w2v (W)	0.995	0.532	1.000	0.721	0.995	0.518
GloVe	0.995	0.541	1.000	0.749	0.995	0.512
GloVe (T)	0.994	0.524	1.000	0.725	0.994	0.488
GloVe (W)	0.993	0.515	1.000	0.739	0.993	0.481

Table 2. Results for k-nearest neighbors . **(Tr)** denotes training data. (W) indicates pre-trained on Wikipedia data. (T) indicates pre-trained on Twitter data. **(Te)** denotes test data.

Embedding	Acc. (Tr)	Acc. (Te)	AUC (Tr)	AUC (Te)	F1 (Tr)	F1 (Te)
LSA	0.581	0.511	0.816	0.749	0.535	0.446
w2v	0.541	0.498	0.761	0.735	0.474	0.431
w2v (W)	0.574	0.584	0.811	0.783	0.535	0.537
GloVe	0.557	0.528	0.791	0.769	0.509	0.468
GloVe (T)	0.593	0.571	0.814	0.790	0.568	0.539
GloVe (W)	0.553	0.532	0.791	0.770	0.488	0.465

Table 3. Results for multinomial logistic regression.

Embedding	Acc. (Tr)	Acc. (Te)	F1 (Tr)	F1 (Te)
lsa	0.594	0.532	0.546	0.468
w2v	0.543	0.494	0.448	0.407
w2v-pre	0.563	0.537	0.470	0.444
GloVe	0.540	0.515	0.446	0.426
GloVe (T)	0.558	0.511	0.464	0.419
GloVe (W)	0.550	0.532	0.456	0.440

Table 4. Results for support vector machine.

We observe that k-nearest neighbors overfits extremely, but performs poorly compared to other algorithms on the test data. Surprisingly, even with the radial basis function kernel, support vector machine doesn't uniformly outperform either multinomial logistic regression or k-nearest neighbors. However, looking at the confusion matrices for the test data, we find that both MNL and SVM are biased towards predicting the dominant class (Neural, 0.41 of examples) and second-most dominant class (Hate, 0.29 of examples).

Our neural network classifiers, on the other hand, also display overfitting on the training data, but also performs mostly consistently best in terms of both the AUC and the F1 score. Looking at the confusion matrix , it also performs the best in terms of AUC, and does well in terms of F1.

Tuned	Embedding	Acc. (Tr)	Acc. (Te)	AUC (Tr)	AUC (Te)	F1 (Tr)	F1 (Te)
No	LSA	0.725	0.537	0.924	0.802	0.705	0.506
No	w2v	0.535	0.511	0.785	0.760	0.510	0.487
No	w2v (W)	0.674	0.550	0.892	0.782	0.677	0.549
No	GloVe	0.643	0.528	0.878	0.803	0.607	0.471
No	GloVe (T)	0.751	0.571	0.936	0.801	0.748	0.567
No	GloVe (W)	0.748	0.550	0.931	0.800	0.748	0.548
Yes	LSA	0.897	0.511	0.980	0.777	0.896	0.503
Yes	w2v	0.553	0.498	0.815	0.776	0.491	0.431
Yes	w2v (W)	0.848	0.550	0.974	0.783	0.848	0.536
Yes	GloVe	0.726	0.545	0.920	0.808	0.712	0.524
Yes	GloVe (T)	0.893	0.558	0.981	0.798	0.892	0.539
Yes	GloVe (W)	0.855	0.532	0.973	0.768	0.853	0.506

Table 5. Results for feed-forward neural network.

Looking at the confusion matrices, we find that while it also has a bias towards the two most dominant classes, the extent to which this is the case is substantially less than for the other three algorithms. Further, whereas the dominant-class bias the other three algorithms exhibited also precluded prediction of the remaining classes, this was not the case for the neural network classifiers.

We also observe that the choice of embedding matters; in general, the pre-trained GloVe embedding using Twitter data seems to perform well, which makes sense given both its linguistic proximity to our own data, and also its size. However, even within the same model class we observe differences between embeddings: SVM tended to predict the dominant class for the LSA and word2vec embeddings, but the second-dominant class for the GloVe embeddings. There were substantive differences between embeddings for the neural network classifer: whereas the LSA and word2vec embedding biased predictions toward the dominant classes, GloVe resulted in predictions that were more balanced across classes, which may stem from the richer information it encodes relative to LSA and word2vec as described in Section 4.1.

6. Conclusion

A primary challenge for classification in this setting was class imbalance, as evidenced by the poor performance of the non-neural network classifiers, which tended to predict the dominant classes.

Interestingly, we found that there was substantive heterogeneity within model classes across embeddings, and similarities across model classes for the same embedding. Previous experimental work has shown that the space occupied by word embedding vectors can display interesting geometric properties; for example, occupying a narrow cone rather than spanning the space (13). Considering the known ties of k-NN, SVM, and MNL to the geometry of the space, one avenue of future work is to explore both how this might affects the performance, and why a feed-forward neural network might tend to fare better.

Contributions

Both Raymond and Cindy learned about embeddings. Cindy conducted an experiments-centered literature review, found the data, did exploratory data analyses, conducted experiments with GloVe and neural networks, generated report tables, and conducted error analysis. Raymond conducted a methods-centered literature review, cleaned the data, did the experiments involving latent semantic indexing, word2vec, and GloVe, conducted exploratory analysis in the embedding space, and built pipelines for generating input data and non-neural network classification algorithm results. The time spent was roughly equal. Both contributed equally to writing this report.

References

[1] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

[2] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

[3] Hswen, Y., Xu, X., Hing, A., Hawkins, J. B., Brownstein, J. S., and Gee, G. C. (2021). Association of "#covid19" Versus "#chinesevirus" With Anti-Asian Sentiments on Twitter: March 9-23, 2020. *American Journal of Public Health*, 111(5):956–964.

[4] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*.

[5] KABC (2021). Arrest made after 70-year-old woman badly beaten, allegedly called anti-Asian slurs on Metro bus in Eagle Rock.

[Karpathy] Karpathy, A. CS231n Convolutional Neural Networks for Visual Recognition.

[7] Kouloumpis, E., Wilson, T., and Moore, J. (2011). Twitter Sentiment Analysis: The Good the Bad and the OMG! *Proceedings of the International AAAI Conference on Web and Social Media*, 5(1).

[8] Madisetty, S. and Desarkar, M. S. (2018). A Neural Network-Based Ensemble Approach for Spam Detection in Twitter. *IEEE Transactions on Computational Social Systems*, 5(4):973–984.

[9] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

[10] Mikolov, T. (2013). de-obfuscated Python + question.

[11] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*.

[12] Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

[13] Mimno, D. and Thompson, L. (2017). The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878, Copenhagen, Denmark. Association for Computational Linguistics.

[14] Neethu, M. S. and Rajasree, R. (2013). Sentiment analysis in twitter using machine learning techniques. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–5.

[15] Nguyen, C., Carroll, J., and Kevin, N. (2020). COVID-19 Community Spread Didn't Start in a Nail Salon: Salon Owners.

[16] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[17] Que, Q. and Belkin, M. (2016). Back to the Future: Radial Basis Function Networks Revisited. In *Artificial Intelligence and Statistics*, pages 1375–1383. PMLR.

[18] Vidgen, B., Hale, S., Guest, E., Margetts, H., Broniatowski, D., Waseem, Z., Botelho, A., Hall, M., and Tromble, R. (2020). Detecting East Asian Prejudice on Social Media. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 162–172, Online. Association for Computational Linguistics.

[19] Yamada, I., Asai, A., Sakuma, J., Shindo, H., Takeda, H., Takefuji, Y., and Matsumoto, Y. (2020). Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. *arXiv:1812.06280 [cs]*.

[20] Ziems, C., He, B., Soni, S., and Kumar, S. (2020). Racism is a Virus: Anti-Asian Hate and Counterhate in Social Media during the COVID-19 Crisis. *arXiv:2005.12423 [physics]*.