AUTOMATED DETECTION OF MISINFORMATION IN TWEETS ABOUT COVID-19*

A PREPRINT

Caitlin Moroney
Department of Mathematics and Statistics
American University
Washington, DC 20016
cm0246b@american.edu

December 1, 2020

ABSTRACT

Enter the text of your abstract here.

***K*eywords** COVID-19 · coronavirus · NLP · machine learning · misinformation

*This work is an extension of Boukouvalas et al. (2020) produced under the guidance of Dr. Zois Boukouvalas and Dr. Nathalie Japkowicz.

1 Introduction

Insert introduction text here.

- Misinformation is a big problem, especially on Twitter & social media
- Misinformation: unreliable vs reliable tweets
- Dataset: 560 tweets manually labeled as unreliable or not unreliable (refer to this class as reliable)
  - originally collected sample of 282,201 Twitter users from Canada, balanced for race, gender, and age by using Conditional Independence Coupling (CIC)
  - tweets were posted between January 1, 2020 and March 13, 2020
  - randomly undersampled 1,600 tweets with goal of creating balanced reliable/unreliable dataset
  - two experts reviewed tweets against set of rules (link to table in appendix); obtained 280 unreliable tweets
  - randomly undersampled the reliable class in order to have perfectly balanced dataset of 560 tweets total
- Topic modeling
- Word embeddings
- Latent variable methods: PCA & ICA
- Binary & one-class classification using SVM
- Implementation done in Python (footnote linking to code page)

The rest of the paper is organized as follows: Section 2 discusses our methodology; Section 3 presents the results of our experiments; Section 4 discusses our results and plans for future work.

2 Methodology

This paper explores a series of methods which seek to exploit linguistic features in raw text data in order to perform the automated detection of unreliable tweets. The experiments follow the same general framework with certain implementation details tweaked for each experiment. The first step in this framework is the application of NLP featurization methods to the raw tweet text. While different featurization methods are compared, all methods involve the use of NLP tools to represent the text with numeric features. Subsequently, latent variable methods are employed to reduce the dimensionality of the resulting $\mathbf{X}$ feature matrix (as well as to uncover latent variables). The latent variables are then used in the classification task. Finally, we evaluate the classification algorithms paired with the featurization methods with respect to performance and explainability. We use the typical performance metrics, including accuracy, F-score, precision, recall, and ROC-AUC. We use LIME (Ribeiro, Singh, and Guestrin 2016) to evaluate local explainability for non-interpretable methods. Furthermore, we present a new explainability framework for latent variable methods as well as a new explainability metric. Below, we explore in greater detail the methods used for featurization, latent variables, classification, and evaluation of explainability.

2.1 NLP featurization

In order to obtain features from the raw tweet text, we first employed standard preprocessing, to include removing stop words and some punctuation as well as lemmatizing words. Specifically, we removed special characters which were not punctuation (e.g., parentheses, question marks, exclamation points, periods, commas, hyphens, colons, and semicolons) or alphanumeric characters (i.e., letters or numbers), converted all text to lowercase, removed stop words, and lemmatized words using NLTK's WordNetLemmatizer aided by NLTK's part-of-speech tagger. After preprocessing, the two approaches we pursued involved (1) Bag-of-Words and (2) word embeddings, each followed by latent variable methods.

2.1.1 Bag-of-Words

A standard NLP featurization method is the Bag-of-Words model. This method can be described as quantifying textual data by representing each document in a dataset with a set of features which derive from the set of words used across all of the documents (Salton and McGill 1983). In other words, we create a dictionary from the unique terms used in a set of documents, and each word in the dictionary is associated with a feature in a matrix of documents (rows) by terms (columns). There are generally two methods for obtaining cell values for the document-term matrix, $\mathbf{X}$: binary indexing and weighted indexing (Salton and McGill 1983).

Binary indexing consists of scanning each document for the presence of each word in the dictionary. For the i^{th} document, if term j appears at least once in the document, cell $\mathbf{X}_{ij} = 1$; otherwise, $\mathbf{X}_{ij} = 0$.

Weighted indexing, on the other hand, accounts for the frequency of terms (Salton and McGill 1983). The most basic implementation of weighted indexing is count vectorization: for the i^{th} document, $\mathbf{X}_{ij}$ is equal to the number of times term j appears in the document. The corresponding formula is as follows:

$$\mathbf{X}_{ij} = \sum_{l=1}^{T_i} \mathbb{1}_{w_l = w_j}$$

where w_j is the term corresponding to column j in the document-term matrix, w_l is the l^{th} word in tweet i, and there are T_i words in tweet i. We refer to this method as the term frequency, or TF, method. Another common weighting scheme, term frequency-inverse document frequency (often referred to as TF-IDF), consists of a transformation on the TF Bag-of-Words matrix. We divide by the document frequency, which is the total number of documents in the dataset in which term j appears (Salton and McGill 1983). This can be represented as the following formula:

$$\mathbf{X}_{ij} = \frac{\sum_{l=1}^{T_i} \mathbb{1}_{w_l = w_j}}{\sum_{k=1}^{N} \mathbb{1}_{w_j \in C_k}}$$

where the only new term, C_k, is the set of words that appear in document k.

After constructing the Bag-of-Words matrix, we employ topic modeling to reduce the dimensionality of the Nxp matrix. This process is discussed in more detail below (see Section 2.2.3).

2.1.2 Word embeddings

> "You shall know a word by the company it keeps!"
> — (Firth 1957)

To create word embeddings, we obtain sparse vector representations which are subsequently transformed into dense vector representations via matrix decomposition. To first construct sparse vector representations, we use the word-word co-occurrence matrix (also known as the word-context matrix) which, unlike the Bag-of-Words method, is able to incorporate information about context from the raw text data. "In the study of selected words, ... an exhaustive collection of collocations must first be made. It will then be found that meaning by collocation will suggest a small number of groups of collocations for each word studied" (Firth 1957). We train the embeddings on the full set of 560 tweets. With the Bag-of-Words methods, we are able to encode information about words' presence in a given document; the co-occurrence approach improves upon this by allowing us to look at word usage with respect to the presence of other words. We construct a word-context matrix using the entire vocabulary for both target terms and context terms. In other words, the matrix is symmetric. We incorporate a number of hyperparameters related to the construction and subsequent transformation of this matrix, including context window size, the use of raw counts or variations on the Pointwise Mutual Information (PMI), and Laplace smoothing. We also include "<START>" and "<END>" tokens at the beginning and end of each tweet.

Context window size Window size refers to the number of tokens before and after a given target word that are considered context words. For example, a window size of three would mean that we would consider the three words preceding and following a target word as its context words. Different types of relations between words can be deduced from a word-context analysis: "syntagmatic" (or syntactic) relations and "paradigmatic" (i.e., semantic) relations (Firth 1957). According to some sources, a window size of four or larger captures semantic representations of the words, whereas smaller windows capture more syntactic representations (Jurafsky 2015; Church and Hanks 1989). In other words, when we restrict the context window to a width of less than four, we are capturing information about how each word interacts with the words immediately surrounding it; this allows us to form word embeddings that provide information about, for example, how words function grammatically in a phrase or clause. When we expand the context window beyond plus or minus four words, we can create embeddings that capture more information about the semantic meaning a word expresses. Also, note that larger context windows necessarily result in less sparse word-context matrices. Both categories of embeddings, broadly defined, provide useful information: "Meaning... is to be regarded as a complex of contextual relations, and phonetics, grammar, lexicography, and semantics each handles its own components of the complex in its appropriate context" (Firth 1957). Unfortunately, Church and Hanks (1989) note that Pointwise Mutual Information (discussed in further detail below) becomes unstable for window sizes less than five. For our purposes, the semantic meaning appears to capture more relevant information; consequently, we selected a window size of 15 for our experiments.

Weighting In constructing the word-word co-occurrence matrix $\mathbf{X}$, our goal is to encode contextual information into our representations of terms. We are interested in knowing which terms are used in which contexts, or which target words are used often with which context words. To that end, a simplistic approach is a count-based analysis, where $\mathbf{X}_{ij}$ is equal to the number of times the j^{th} context word occurs within the pre-established context window of the i^{th} target word. Importantly, there is no distinction between documents (in this case, tweets): we count how many times word i occurs near word j over all of the documents. This method is similar to the TF method for the Bag-of-Words model: weights simply represent word (co-occurrence) frequencies.[1] As with Bag-of-Words, for the word-context matrix, there are other weighting schemes available. Notably, Church and Hanks (1989) proposed a measure based on the concept of mutual information which captures word associations. Pointwise Mutual Information (PMI) is a popular alternative to raw co-occurrence counts because it allows us to compare the joint probability of words i and j with their marginal probabilities (Church and Hanks 1989). PMI is defined as follows:

$$PMI(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$$

where the numerator represents the probability of seeing word i with word j and the denominator is the product of the probability of seeing word i and the probability of seeing word j (Church and Hanks 1989). It is not uncommon to use a different log-base in defining PMI (Levy and Goldberg 2014). The PMI can be estimated by the counts for words i and j (separately) as well as the frequency of words i and j occurring together within a predefined window across the entire corpus. Church and Hanks (1989) suggest normalizing each of these values by the total number of words in the corpus, which equates to equation 10 from Levy and Goldberg (2014):

$$PMI(w_i, w_j) = \log \frac{c(w_i, w_j) \cdot |D|}{c(w_i) \cdot c(w_j)}$$

where $c(\cdot)$ is a count function, D is the "collection of observed words and context pairs" and, therefore, $|D|$ is the size of the vocabulary. One issue with using PMI is the fact that many of the possible word-context pairs will not be observed in the corpus, which results in $PMI(w_i, w_j) = \log 0 = -\infty$. One solution to this issue is to substitute 0 for $-\infty$ in these cases such that the PMI value for unobserved word-context pair w_i, w_j is equal to 0. We will refer to this method as PMI_0. This, however, results in an inconsistency in the word-context matrix because "observed but 'bad' (uncorrelated) word-context pairs have a negative matrix entry, while unobserved (hence, worse) ones have 0 in their corresponding cell" (Levy and Goldberg 2014). Thus, a common alternative to PMI is the positive Pointwise Mutual Information (PPMI), which is defined as follows:

$$PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0)$$

such that the word-context matrix is non-negative. Another method to address the issue of unobserved word-context pairs is discussed below. Note that for our experiments we found PMI_0 to work slightly better than or comparable to PPMI, while both mutual information metrics surpassed the raw counts approach. Therefore, our final ICA word embeddings were formed using PMI_0.

Laplace smoothing One approach to the problem of unobserved word-context pairs resulting in an ill-defined word-context matrix when using PMI consists of "smooth[ing] the [estimated] probabilities using, for instance, a Dirichlet prior by adding a small 'fake' count to the underlying counts matrix, rendering all word-context pairs observed" (Levy and Goldberg 2014). As the authors note, this smoothing technique adjusts the PMI word-context matrix such that there are no infinite values. It is common to use add-1 or add-2 Laplace smoothing in NLP applications (Jurafsky 2015). We compared using no smoothing, add-1 smoothing, and add-2 smoothing. Our results suggest that no smoothing produced better embeddings than incorporating Laplace smoothing.

Shifted vs unshifted PMI (or PPMI) Levy and Goldberg (2014) propose a novel word association metric based on their investigation of the Skip-Gram with Negative Sampling (SGNS) model from Mikolov et al. (2013).

$$SPPMI_k(w_i, w_j) = \max(PMI(w_i, w_j) - \log k, 0)$$

where k is a user-specified parameter which incorporates negative sampling (Levy and Goldberg 2014). We compared PPMI, SPPMI, PMI_0, and SPMI_0. We found that unshifted PMI_0 produced the best embeddings from our set of 560 tweets.

[1] In fact, we could reconceptualize the Bag-of-Words model as a word-context matrix where each document denotes a context; instead of each target word's context being defined by a moving window (which in turn derives from the pre-specified window width), the documents comprise static contexts.

Use of start & end tokens We add "<START>" and "<END>" tokens to the beginning and end of each tweet in order to capture information about which tokens (or words) collocate with the beginning or end of a tweet. For example, we might expect the "#" token to appear more often with the "<END>" token than other vocabulary words because hashtags are often included at the end of a tweet.

Latent variable methods are subsequently applied to the word embeddings in order to reduce the dimensionality. For more information on this process, please see Section 2.2.3.

Finally, we average over the word embeddings for the words in each tweet to obtain a single vector representation for each tweet:

$$\mathbf{v}_i = \frac{1}{T_i} \sum_{j=1}^{T_i} \mathbf{e}_j$$

where $\mathbf{v}_i$ is the vector representation for tweet i, $\mathbf{e}_j$ is the embedding for word j in tweet i, and there are T_i words in tweet i. Note that in our experiments $\mathbf{e}_j \in \mathbb{R}^{250}$ and, therefore, $\mathbf{v}_i \in \mathbb{R}^{250}$.

For more details on the performance of different word embedding hyperparameter combinations, see the Appendix.

- BERT embeddings from pre-trained model

2.2 Latent variable methods

In order to reduce the dimensionality of the data matrix $\mathbf{X}$, we employ latent variable methods. Specifically, we follow the methodology presented in Honkela, Hyvärinen, and Väyrynen (2010): we apply Principal Component Analysis (PCA) followed by Independent Component Analysis (ICA) to both the Bag-of-Words matrix and the word-context co-occurrence matrix.

2.2.1 Dimensionality reduction

In order to perform PCA, we rescale the data matrix and then apply Singular Value Decomposition (SVD). We first scale the data so that the columns have zero mean and unit variance. The SVD model is as follows:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values of $\mathbf{X}$, $\mathbf{U}$ is BLAH, and $\mathbf{V}^T$ is BLAHBLAH.

To achieve PCA, we can use truncated SVD. In essence, we perform SVD and keep only the columns of $\mathbf{U}$ and $\mathbf{V}^T$ that correspond to the largest k singular values in the diagonal matrix $\mathbf{\Sigma}$, where k is the desired order for the approximation of our initial data matrix. Therefore, if k is less than m, we have achieved dimensionality reduction. In our experiments, we use $k \in \{50, 100, 150, 200, 250, 500\}$.

2.2.2 Independent Component Analysis

ICA is one method to address the blind source separation problem (also referred to as the blind signal separation problem), which can be represented in matrix form as follows:

$$\mathbf{X} = \mathbf{A}\mathbf{S}$$

Honkela, Hyvärinen, and Väyrynen (2010) provide an intuitive explanation of the problem in the context of a real-world scenario:

> "... [T]he cocktail party problem is a classical blind signal separation task where heard sound signals are studied as the observed random variables. These are assumed to originate from a number of separate sources, in this case, the discussants in a cocktail party who are speaking at the same time. The heard signals are mixtures of the speech signals with different proportions depending on the relative distance of a listener to each sound source. The task is to separate the original speech signals from the observed mixtures."

In other words, ICA allows us to extract independent signals from the original data matrix. The columns of $\mathbf{X}$ are linear combinations (or mixtures) of the independent components. With the ICA decomposition, we obtain a mixing matrix, $\mathbf{S}$, which contains the weights for each of the independent components which together comprise each of the observed variables, and $\mathbf{A}$, where each column is a latent variable (or independent component).

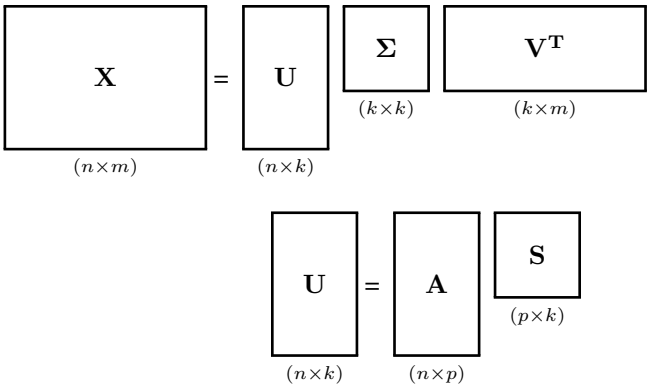

Figure 1: Truncated Singular Value Decomposition followed by Independent Componenet Analysis.

2.2.3 Latent variables pipeline

The pipeline involves performing SVD on the initial data matrix, $\mathbf{X}$, such that we can obtain the $\mathbf{U}$ matrix which contains the SVD feature vectors. This matrix is used as the input for ICA, so that we have $\mathbf{U} = \mathbf{AS}$. Then, $\mathbf{S}$ is the whitened mixing matrix, and $\mathbf{A}$ contains the ICA features. This process is represented visually in Figure 1. We use the estimated $\hat{\mathbf{A}}$ matrix as the input to our classification models.

The only difference between the topic modeling approach and the word embedding approach is the structure of the original data matrix $\mathbf{X}$ obtained from the chosen NLP featurization method. In one case, we obtain vector representations for documents; in the other, we obtain vector representations for terms.

For topic modeling, we start with a documents by terms matrix and perform dimensionality reduction via truncated SVD in order to obtain k latent variables. These SVD features are linear combinations of the original columns of the Bag-of-Words matrix, which are terms. In other words, each latent variable is a topic, comprised of a linear combination of words which can be ranked by their weights from the estimated $\hat{\mathbf{V}}$ matrix. We subsequently apply ICA to the estimated $\hat{\mathbf{U}}$ matrix, containing the SVD features, in order to obtain the k independent components. The resulting ICA features are linear combinations of SVD features.

When we start with the word-context matrix, we obtain word embeddings comprised of latent variables after first performing truncated SVD. These SVD features are linear combinations of the context words taken from the original co-occurrence matrix. However, we go a step further in order to obtain statistically independent latent variables by applying ICA to the estimated $\hat{\mathbf{U}}$ matrix. This allows us to obtain ICA word embeddings from the $\hat{\mathbf{A}}$ matrix, whose columns are linear combinations of the SVD features. The estimated $\hat{\mathbf{S}}$ matrix allows us to access the weights for these linear combinations.

2.3 Classification

- One-class SVM
- Binary SVM

2.4 Evaluation

In order to evaluate our experiments, including featurization methods and classification algorithms, we have identified three areas for comparison: performance, computational complexity, and explainability. To measure performance, we employ the standard suite of evaluation metrics, i.e., accuracy, F-score, precision, recall, and ROC-AUC. We report the macro-averaged versions of these scores. Computational complexity is easily captured by the amount of time taken for training and testing. Evaluating the explainability of one method versus another proves to be a more complicated task. For the experiments using word embeddings obtained from the pretrained BERT model, we use LIME (Ribeiro, Singh, and Guestrin 2016) to obtain local explanations for tweet predictions. For the ICA word embeddings, we propose a new framework to obtain global and local explanations for tweet predictions which derive from the ICA matrix decomposition. In order to then compare overall explainability for one method versus another, we have devised a metric which captures information from the LIME and ICA local explanations and aggregates these values to produce a single

number representing an explainability score. This allows us to compare two methods with respect to explainability in the same way that we might compare them in terms of accuracy or precision.

Our goal with assessing explainability is to determine whether the machine learning pipeline is making what we would consider intuitive decisions. In other words, when a human and the machine look at the same tweet, we are not only interested in knowing whether the machine can make the same classification as the human (which we can measure with accuracy), but we are also interested in knowing whether the machine and the human make the same judgment for similar reasons. With our data, this can be posed as a question of whether the algorithm labels a tweet as unreliable due to evidence that intersects with the rules-based labeling performed by a human team that produced the tweet labels for our dataset. These guidelines are summarized in Table 1 from Boukouvalas et al. (2020). This table has been reproduced in the appendix (see Table 5).

2.4.1 LIME

Explain LIME.

2.4.2 ICA

Explain ICA explainability.

Global

Local

2.4.3 Explainability metric

As mentioned above, our initial motivation in pursuing explainability was to provide an evaluation metric which would allow us to compare the overall explainability of one model and featurization method with that of another. The final result of this effort is a novel metric which allows us to incorporate the LIME (Ribeiro, Singh, and Guestrin 2016) and ICA explainability output into a single score which represents the overall explainability as an aggregation of local explanations.

Because our conception of explainability inherently relies on a comparison to human decision-making, our explainability metric takes an input which captures the rules our coders used to label our set of tweets. This input is in the form of a vocabulary list which comprises words that fall under the rules in Table 1 from Boukouvalas et al. (2020).[2] Our metric "rewards" the word embedding and algorithm pipeline for associating Table 1 words with the unreliable class. We produced two versions of the metric, one which penalizes the machine for associating Table 1 words with the reliable class, and one which does not include a penalty.

The formula which includes the penalty is as follows:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{j=1}^{T_i}\mathbb{1}_A(w_j) - \mathbb{1}_B(w_j)$$

where A is the set of words that the classifier associated with the correct class (e.g., tweet i is labeled as "reliable," and the classifier classified the tweet as "reliable") according to the LIME output for tweet i, B is the set of words that the classifier associated with the wrong class according to the LIME output for tweet i, there are T_i words in tweet i, and there are N tweets.

The following formula comprises the metric without no penalty for associating Table 1 words with the reliable class:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{j=1}^{T_i}\mathbb{1}_A(w_j)$$

where A is the set of words that the classifier associated with the correct class according to the LIME output for tweet i, there are T_i words in tweet i, and there are N tweets. This version of the metric essentially computes the percentage of Table 1 word occurrences that were correctly associated with the unreliable class per tweet and then computes an unweighted average over all of the tweets.

[2]The vocabulary list we used can be found in the Appendix 4.

Table 1: OCSVM Results

Dimensions	AUC	Accuracy	F1	Precision	Recall
50	0.764	0.668	0.630	0.695	0.668
100	0.750	0.671	0.629	0.709	0.671
150	0.730	0.654	0.598	0.693	0.654
200	0.725	0.636	0.579	0.668	0.636
250	0.731	0.648	0.586	0.691	0.648
500	0.688	0.630	0.539	0.684	0.630

Table 2: Isolation Forest Results

Dimensions	AUC	Accuracy	F1	Precision	Recall
50	0.643	0.552	0.616	0.673	0.552
100	0.591	0.532	0.558	0.656	0.532
150	0.494	0.521	0.542	0.546	0.521
200	0.545	0.516	0.552	0.537	0.516
250	0.497	0.511	0.531	0.543	0.511
500	0.527	0.504	0.507	0.505	0.504

3 Results

Below we have included the results for both one-class classification and binary classification experiments. We report the standard performance evaluation metrics for three outlier detection models and one binary classification model. For our novel explainability metric, we report results for the binary classification set-up.

3.1 Performance

To assess performance, we report the macro-averaged ROC-AUC, accuracy, F_1-score, precision, and recall values. Overall, the binary classification experiment had much greater success than any of the three one-class classification models.

3.1.1 One-class classification

For one-class classification, we compared the performance of three models: one-class SVM (OCSVM), isolation forest, and local outlier factor (LOF).

3.1.2 Binary classification

For binary classification, we show results for our binary SVM experiment.

3.2 Explainability

4 Discussion

- Interpret results
- Future work

Table 3: LOF Results

Dimensions	AUC	Accuracy	F1	Precision	Recall
50	0.658	0.539	0.552	0.598	0.539
100	0.639	0.536	0.526	0.608	0.536
150	0.659	0.513	0.513	0.515	0.513
200	0.644	0.529	0.536	0.580	0.529
250	0.652	0.529	0.540	0.586	0.529
500	0.613	0.495	0.510	0.470	0.495

Table 4: Binary SVM Results

Dimensions	AUC	Accuracy	F1	Precision	Recall
50	0.903	0.804	0.801	0.818	0.804
100	0.911	0.796	0.793	0.817	0.796
150	0.906	0.795	0.792	0.810	0.795
200	0.901	0.800	0.798	0.815	0.800
250	0.904	0.807	0.804	0.827	0.807
500	0.908	0.789	0.785	0.814	0.789

- Different ICA algorithm (not FastICA)
- Multiple ICA runs + take most representative of those
- Better ICA explainability tool?
- Better explainability metric?
  - Weight by magnitude instead of binary yes/no weight?
- Other classifiers (e.g., neural nets)
- Incorporate other features:
  - Part-of-speech tag counts
  - Punctuation counts
  - Use of all-caps
  - Sentiment analysis

References

Boukouvalas, Zois, Christine Mallinson, Evan Crothers, Nathalie Japkowicz, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tülay Adalı. 2020. "Independent Component Analysis for Trustworthy Cyberspace During High Impact Events: An Application to Covid-19." *arXiv:2006.01284 [Cs, Stat]*, June. http://arxiv.org/abs/2006.01284.

Church, Kenneth Ward, and Patrick Hanks. 1989. "Word Association Norms, Mutual Information, and Lexicography." In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics* -, 76–83. Vancouver, British Columbia, Canada: Association for Computational Linguistics. https://doi.org/10.3115/981623.981633.

Firth, J. R. 1957. "A Synopsis of Linguistic Theory 1930-1955." *Studies in Linguistic Analysis*, 1–32.

Honkela, Timo, Aapo Hyvärinen, and Jaakko J. Väyrynen. 2010. "WordICA—Emergence of Linguistic Representations for Words by Independent Component Analysis." *Natural Language Engineering* 16 (3): 277–308. https://doi.org/10.1017/S1351324910000057.

Jurafsky, Dan. 2015. "Distributional (Vector) Semantics." Seminar lecture. Chicago, IL: Seminar lecture. https://web.stanford.edu/~jurafsky/li15/lec3.vector.pdf.

Levy, Omer, and Yoav Goldberg. 2014. "Neural Word Embedding as Implicit Matrix Factorization." In *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, 27:2177–85. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf.

Mikolov, Tomas, Kai Chen, G. S. Corrado, and J. Dean. 2013. "Efficient Estimation of Word Representations in Vector Space." *CoRR* abs/1301.3781.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." *arXiv:1602.04938 [Cs, Stat]*, August. http://arxiv.org/abs/1602.04938.

Salton, Gerard, and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill Computer Science Series. New York: McGraw-Hill.

Appendix

4.1 Table 1 words

Table 5: Misinformation rules from Boukouvalas et al. (2020)

Linguistic Feature	Example from Dataset
Hyperbolic, intensified, superlative, or emphatic language	e.g., 'blame', 'accuse', 'refuse', 'catastrophe', 'chaos', 'evil'
Greater use of punctuation and/or special characters	e.g., e.g., 'YA THINK!!?!!?!', 'Can we PLEASE stop spreading the lie that Coronavirus is super super super contagious? It's not. It has a contagious rating of TWO'
Strongly emotional or subjective language	e.g., 'fight', 'danger', 'hysteria', 'panic', 'paranoia', 'laugh', 'stupidity' or other words indicating fear, surprise, alarm, anger, and so forth
Greater use of verbs of perception and/or opinion	e.g., 'hear', 'see', 'feel', 'suppose', 'perceive', 'look', 'appear', 'suggest', 'believe', 'pretend'
Language related to death and/or war	e.g., 'martial law', 'kill', 'die', 'weapon', 'weaponizing'
Greater use of proper nouns	e.g., 'USSR lied about Chernobyl. Japan lied about Fukushima. China has lied about Coronavirus. Countries lie. Ego, global'
Shorter and/or simpler language	e.g., '#Iran just killed 57 of our citizens. The #coronavirus is spreading for Canadians Our economy needs support.'
Hate speech and/or use of racist or stereotypical language	e.g., 'foreigners', 'Wuhan virus', reference to Chinese people eating cats and dogs
First and second person pronouns	e.g., 'I', 'me', 'my', 'mine', 'you', 'your', 'we', 'our'
Direct falsity claim and/or a truth claim	e.g., 'propaganda', 'fake news', 'conspiracy', 'claim', 'misleading', 'hoax'
Direct health claim	e.g., 'cure', 'breakthrough', posting infection statistics
Repetitive words or phrases	e.g., 'Communist China is lying about true extent of Coronavirus outbreak - If Communist China doesn't come clean'
Mild or strong expletives, curses, slurs, or other offensive terms	e.g., 'bitch', 'WTF', 'dogbreath', 'Zombie homeless junkies', 'hell', 'screwed'
Language related to religion	e.g., 'secular', 'Bible'
Politically biased terms	e.g., 'MAGA', 'MAGAt', 'genetic engineer Hillary', 'Chinese regime', 'deep state', 'Free Market Fundamentalists', 'Communist China', 'Nazi'
Language related to financial or economic impact, money/costs, or the stock market	e.g., 'THE STOCK MARKET ISN'T REAL THE ECONOMY ISN'T REAL THE CORONAVIRUS ISN'T REAL FAKE NEWS REEEEEEEEEEEEEEEE'
Language related to the Trump presidential election, campaign, impeachment, base, and rallies	e.g., 'What you are watching with the CoronaVirus has been planned and orchestrated. We are 8 months from the next Presidential elections'

4.2 Word embedding results

4.2.1 150 dimensions

4.2.2 500 dimensions

4.2.3 Choosing dimension of word embeddings

All word-context matrices were formed using unshifted PMI_0 with no Laplace smoothing.

Table 6: Binary clf results using word embeddings dim 150

Method	AUC	Accuracy	F1	Precision	Recall
Raw counts	0.92	0.84	0.85	0.82	0.88
Raw counts, add-1 smoothing	0.92	0.85	0.85	0.82	0.89
Raw counts, add-2 smoothing	0.92	0.85	0.85	0.82	0.89
Raw counts, add-5 smoothing	0.92	0.84	0.85	0.82	0.88
PMI_0	0.93	0.86	0.86	0.82	0.92
PMI_0 add-1 smoothing	0.91	0.83	0.84	0.81	0.88
PMI_0 add-2 smoothing	0.92	0.85	0.85	0.82	0.89
PMI_0 add-5 smoothing	0.92	0.85	0.85	0.82	0.89
PPMI	0.93	0.85	0.86	0.82	0.91
PPMI add-1 smoothing	0.93	0.85	0.86	0.82	0.89
PPMI add-2 smoothing	0.93	0.84	0.85	0.81	0.90
PPMI add-5 smoothing	0.92	0.85	0.86	0.82	0.89
$SPMI_0$ ($k = 5$)	0.90	0.82	0.83	0.80	0.87
$SPMI_0$ add-1 smoothing	0.91	0.83	0.84	0.81	0.89
$SPMI_0$ add-2 smoothing	0.92	0.85	0.85	0.82	0.89
$SPMI_0$ add-5 smoothing	0.92	0.85	0.85	0.82	0.89
SPPMI	0.91	0.82	0.83	0.81	0.85
SPPMI add-1 smoothing	0.89	0.82	0.83	0.77	0.92
SPPMI add-2 smoothing	0.89	0.83	0.84	0.78	0.91
SPPMI add-5 smoothing	0.88	0.81	0.83	0.75	0.94

Table 7: Binary clf results using word embeddings dim 500

Method	AUC	Accuracy	F1	Precision	Recall
Raw counts	0.93	0.87	0.88	0.83	0.93
Raw counts, add-1 smoothing	0.93	0.87	0.88	0.83	0.93
Raw counts, add-2 smoothing	0.93	0.87	0.88	0.83	0.93
Raw counts, add-5 smoothing	0.93	0.87	0.88	0.83	0.93
PMI_0	0.93	0.87	0.88	0.83	0.94
PMI_0 add-1 smoothing	0.93	0.87	0.88	0.83	0.93
PMI_0 add-2 smoothing	0.93	0.87	0.88	0.84	0.93
PMI_0 add-5 smoothing	0.93	0.87	0.88	0.83	0.93
PPMI	0.93	0.86	0.87	0.83	0.92
PPMI add-1 smoothing	0.93	0.86	0.87	0.83	0.92
PPMI add-2 smoothing	0.93	0.87	0.87	0.83	0.92
PPMI add-5 smoothing	0.93	0.87	0.87	0.83	0.93
$SPMI_0$ ($k = 5$)	0.92	0.86	0.87	0.82	0.93
$SPMI_0$ add-1 smoothing	0.93	0.87	0.88	0.83	0.93
$SPMI_0$ add-2 smoothing	0.93	0.87	0.88	0.84	0.93
$SPMI_0$ add-5 smoothing	0.93	0.87	0.88	0.83	0.93
SPPMI	0.92	0.84	0.85	0.81	0.89
SPPMI add-1 smoothing	0.90	0.81	0.83	0.77	0.91
SPPMI add-2 smoothing	0.89	0.83	0.84	0.78	0.91
SPPMI add-5 smoothing	0.88	0.81	0.83	0.75	0.94

Table 8: Binary clf results using word embeddings with varying dimensions

Dimensions	AUC	Accuracy	F1	Precision	Recall
50	0.903	0.804	0.810	0.795	0.839
100	0.911	0.796	0.804	0.782	0.846
150	0.910	0.795	0.815	0.743	0.904
200	0.902	0.800	0.816	0.758	0.889
250	0.904	0.807	0.822	0.774	0.889
500	0.908	0.789	0.810	0.746	0.896