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 4, 2020

ABSTRACT

In this paper, we apply Natural Language Processing (NLP) and machine learning (ML) methods to the task of detecting misinformation. We use a dataset of 560 tweets to train both binary and one-class classification algorithms. NLP techniques allow us to transform the raw tweet text into a vector representation to which we apply latent variable methods in order to create word embeddings. We use the word embeddings to obtain tweet vector representations which are subsequently used as inputs for the classifiers. We also propose a novel explainability metric that allows us to capture an explainability score for a classifier, similar to accuracy or other evaluation metrics.

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

1 Introduction

Massive volumes of content are posted to social media platforms daily. Many users rely partially or entirely on social media as a source of news and information. This makes the detection of "unreliable" information, or misinformation, a pressing and challenging task. This is especially true of detecting misinformation during high-impact events when information changes rapidly. Clearly, manually labeling content as potentially unreliable is not feasible with the size of the data in question. In this paper, we propose to apply machine learning to the task of classifying misinformation in tweets.

In this study, we use "unreliable" to refer to tweets labeled as misinformation and "reliable" to refer to tweets which do not contain misinformation. The data as well as the labeling rules come from Boukouvalas et al. (2020); the authors manually labeled 560 tweets as unreliable or reliable. The tweets were collected from a sample of 282,201 Twitter users in Canada between the dates January 1, 2020 and March 13, 2020. The dataset is balanced for race, gender, and age via Conditional Independence Coupling (CIC) (Boukouvalas et al. 2020). From this large dataset, two experts reviewed a random sample of 1,600 tweets and created a perfectly balanced dataset of 560 labeled tweets (Boukouvalas et al. 2020).

We apply NLP techniques to the raw tweet text in order to obtain a vector representation of the text data. We then use latent variable methods, Principal Component Analysis (PCA) and Independent Component Analysis (ICA), in order to reduce the dimensionality of our data and to obtain statistically independent features. We subsequently implement both binary and one-class classification. Implementation is in Python.[1]

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.

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

[1]Please see the GitHub repo for the original code.

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 employ standard preprocessing, to include removing stop words and some punctuation as well as lemmatizing words. Specifically, we remove special characters which are not punctuation (e.g., parentheses, question marks, exclamation points, periods, commas, hyphens, colons, and semicolons) or alphanumeric characters (i.e., letters or numbers), convert all text to lowercase, remove stop words, and lemmatize words using NLTK's WordNetLemmatizer aided by NLTK's part-of-speech tagger. After preprocessing, we create word embeddings via the word-word co-occurrence matrix followed by latent variable methods.

> "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 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 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. We train the embeddings on the full set of 560 tweets.

2.1.1 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.

2.1.2 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.[2] 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.

2.1.3 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.

2.1.4 Shifted PMI (SPMI) and shifted PPMI (SPPMI)

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).

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

[2] 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.

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.

2.1.5 Start and 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.

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 the word-word 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 a unitary orthogonal matrix whose columns are the left-singular vectors, and $\mathbf{V}^T$ is a unitary orthogonal matrix whose columns are the right-singular vectors.

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).

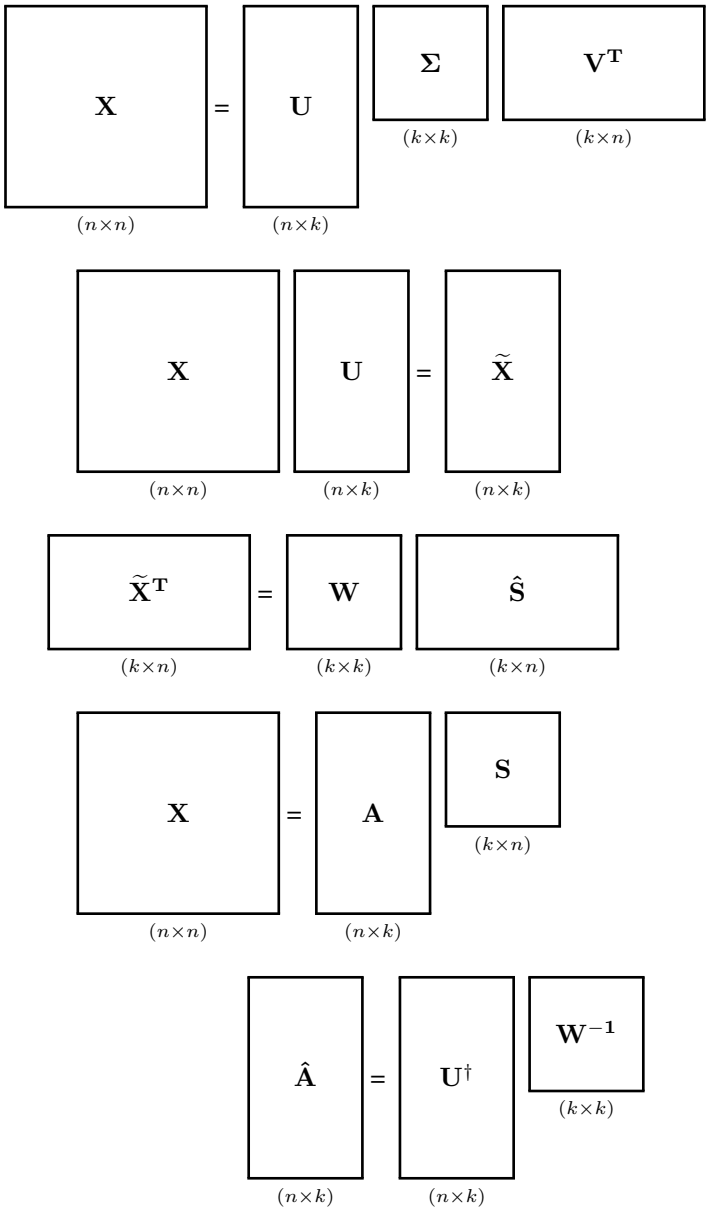

Figure 1: Truncated Singular Value Decomposition followed by Independent Component Analysis.

2.2.3 Latent variables pipeline

We use the scikit-learn implementation of FastICA to obtain the word embeddings from the word-context matrix. Because we also impose dimensionality reduction, the FastICA algorithm consists of PCA followed by ICA decomposition. This process is represented visually in Figure 1. We take the estimated $\hat{\mathbf{S}}$ matrix as our word embeddings which we subsequently use to obtain tweet embeddings as the inputs to our classification models.

Starting 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. $\mathbf{U}$ is of shape number of target words by number of SVD features, $\mathbf{\Sigma}$ is a square matrix with k (equals number of SVD features) singular values, and $\mathbf{V}^T$ is of shape number of SVD features by number of context words. In theory, we could take the $\mathbf{U}$ matrix as our word embeddings. However, we go a step further in order to obtain statistically independent latent variables by applying ICA to the SVD output.

Under the ICA model, we assume $\mathbf{X} = \mathbf{AS}$. To obtain estimates for $\mathbf{A}$ and $\mathbf{S}$, we use the SVD output:

$$\tilde{\mathbf{X}} = \mathbf{W}\hat{\mathbf{S}}$$

where $\tilde{\mathbf{X}} = \mathbf{XU}$, $\mathbf{W}$ is the mixing matrix, and $\hat{\mathbf{S}}$ is a matrix of ICA features by context words. This allows us to obtain ICA word embeddings from the $\hat{\mathbf{S}}$ matrix, whose rows are linear combinations of the SVD features. We use the context word embeddings as our ICA word embeddings. Another possibility is to obtain the target word embeddings from the $\hat{\mathbf{A}}$ matrix. Were we to use these embeddings instead, we would need to go a step further to obtain $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}} = \mathbf{U}^{\dagger}\mathbf{W}^{-1}$$

where $\mathbf{U}^{\dagger}$ is the pseudo-inverse of the $\mathbf{U}$ matrix.

2.3 Classification

In this study, we perform both one-class classification (also referred to as outlier detection) and binary classification in order to automate the detection of unreliable tweets. By using the Twitter dataset produced by Boukouvalas et al. (2020), we can appropriately apply a binary classification model because the dataset was structured as perfectly balanced with 280 reliable tweets and 280 unreliable tweets. However, we believe it is also valid to frame the research question as a one-class classification problem: in the real world, we might expect the number of reliable tweets to far outweigh the number of unreliable tweets. More generally, depending on how we define misinformation, we may see severe imbalance in the ratio of misinformation to reliable information. In such a situation, outlier detection models may be better suited to correctly identifying unreliable content than binary classification models. In terms of algorithms, for outlier detection, we employ one-class support vector machines (OCSVM), isolation forest, and local outlier factor models (LOF). For binary classification, we use binary SVM.

2.4 Evaluation

In order to evaluate our experiments, including featurization methods and classification algorithms, we have identified two areas for comparison: performance 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. Evaluating the explainability of one method versus another proves to be a more complicated task. For the ICA word embeddings, we propose a new framework to obtain global 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 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 6).

2.4.1 Explainability tools

In order to assess the explainability of our word embedding plus classifier pipeline, we first turn to LIME (Ribeiro, Singh, and Guestrin 2016), a popular explainability model which produces local explanations for classifier decisions. In the context of an NLP classification task, LIME attempts to provide an accurate explanation for why the classifier in question made a classification decision for a given document based on the featurization method used. For this study, we can apply LIME to obtain an explanation for why the SVM classifier classified a given tweet as unreliable based on the words which appear in that tweet; LIME can tell us which class the classifier associates each word with, as well as the relative importance of each word in the machine's decision-making process for that tweet.

While LIME is a brilliant tool for assessing explainability and classifier trustworthiness (Ribeiro, Singh, and Guestrin 2016), we believe ICA can provide interpretability on its own. Because ICA is not a black box, we can access information about how important certain words are from the estimated $\hat{\mathbf{A}}$ matrix. However, it is important to note that whereas LIME takes as inputs the pipeline from raw text to model features and the trained classifier so that it can produce a local explanation for a particular tweet prediction, the ICA explainability framework we propose is instead a global explainability approach which does not take class predictions (or even the classifier) into consideration.

Our proposed framework involves obtaining word importance weights by extracting the word weights from the $\hat{\mathbf{A}}$ matrix. Recall that the ICA decomposition begins with the $\hat{\mathbf{U}}$ matrix from the SVD factorization of the word-word co-occurrence matrix. Thus, the rows of $\hat{\mathbf{U}}$ are the target words, and its columns are topics in the sense that they are weighted linear combinations of words. Therefore, the rows of $\hat{\mathbf{A}}$ are also the target words, and the columns of $\hat{\mathbf{A}}$ are concepts, or weighted linear combinations of SVD topics.[3] Though simplistic, our method involves a logical aggregation of these weights such that we can obtain global word importance values. We define the importance of the i^{th} target word as follows:

$$g_i = \frac{1}{k} \sum_{j=1}^{k} |a_{ij}|$$

where k is the number of SVD features (and therefore the number of ICA features), and $|a_{ij|}$ is the magnitude of the i^{th} word's importance in topic j.

With these values, we can produce plots similar to those from the LIME output which show the relative importance of each word within a tweet. As mentioned above, an important difference is that SVD and ICA are unsupervised methods, so our word importance weights do not have any class association, whereas LIME provides both importance magnitude and sign (signifying class association).

2.4.2 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) 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). 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.

[3] We could also reimagine the concepts as topics themselves with adjusted word weights.

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

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.

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.[4] 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). OCSVM appears to outperform both isolation forest and local outlier factor across all

[4]For non-macro-averaged binary classification results, see Table 9

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

metrics. The three outlier detection models performed best on word embeddings with fewer dimensions compared to the binary classification model. OCSVM performed best on embeddings with a length of 100, whereas LOF and isolation forest had optimal results with embeddings of length 50.

3.1.2 Binary classification

For binary classification, we show results for our binary SVM experiment. The embeddings of length 250 appear to have resulted in the best classification performance with regard to every metric except AUC (though the AUC performance is comparable across all embeddings). Compared to the outlier detection models, binary SVM performed best on longer word embeddings of size 250. The accuracy for binary SVM, 0.807, improved upon the best accuracy performance from all of our one-class model experiments, which was 0.671, by over 13 percentage points.

3.2 Explainability

To assess explainability according to our novel metric, we conducted 4 experiments:

- Experiment 1: Only table 1 words
- Experiment 2: Table 1 words plus manual additions
- Experiment 3: Stemming on table 1 words
- Experiment 4: Stemming on table 1 words plus manual additions

Table 1 words refers to words that are listed as examples for linguistic features of misinformation in Table 1 from Boukouvalas et al. (2020). Manual additions are words we added after reviewing the corpus vocabulary and noting terms that fell within one of the 17 linguistic feature categories but were not explicitly provided as examples. Experiment 1 was a strict interpretation of explainability which relied solely on words provided in Table 1 from Boukouvalas et al. (2020); experiment 2 included these words plus our manual additions. Experiments 3 and 4 are different in that we used stemming in order to cast a wider net; for example, if "panic" is a Table 1 word, then stemming (via NLTK's snowball stemmer) allows us to catch any instances of "panicked", "panics", "panicky", etc. It is therefore not surprising that experiments 3 and 4 both show relatively higher explainability scores than the results for experiments 1 and 2. It is also unsurprising that the experiments which incorporated expansions of the table 1 words vocabulary (experiments 2 and 4) resulted in higher explainability scores than experiments 1 and 3.

We have also reported the results both in aggregate and disaggregated by correctness of prediction. We believe the disaggregated results are useful because we would expect the classifier to do a poorer job of associating table 1 words with the unreliable class when it incorrectly classifies an unreliable tweet as reliable. This assumption appears to hold based on our results; the "correctly predicted" tweets have higher explainability scores than the "wrongly predicted" ones.

4 Discussion

Future work on this subject can pursue several directions. An initial improvement concerns the ICA implementation: FastICA is only one of several ICA algorithms, and it may be worth investigating the use of other ICA algorithms to see whether a different algorithm (for the same model) produces different results. In a similar vein, we might consider conducting multiple ICA runs on the same word-context matrix and then taking the most representative vector representations from those runs. In our current study, we run ICA one time and use those vector representations for the word embeddings.

Two of the more straightforward additions would be to compare the performance of other binary classification models and to compare word embeddings created with different latent variable methods. Specifically, it would be interesting to

Table 5: Explainability Scores

Experiment	Penalty	No Penalty
1: Correctly predicted	0.331	0.534
1: Wrongly predicted	0.222	0.278
1: Aggregated	0.326	0.521
2: Correctly predicted	0.356	0.593
2: Wrongly predicted	0.074	0.315
2: Aggregated	0.342	0.579
3: Correctly predicted	0.396	0.593
3: Wrongly predicted	0.444	0.500
3: Aggregated	0.399	0.588
4: Correctly predicted	0.378	0.619
4: Wrongly predicted	0.148	0.407
4: Aggregated	0.367	0.608

compare binary SVM with a neural net model. For latent variable methods, instead of SVD followed by ICA we could perform non-negative matrix factorization (NMF), dictionary learning (DL), latent semantic analysis (SVD alone), or latent Dirichlet allocation (LDA).

Another extension of this work could incorporate other linguistic features from the dataset. Whereas we have focused on the vocabulary, it would be of interest to see whether part-of-speech tag counts, punctuation counts, the use of all-capitalized text, or sentiment analysis might be discriminative features in the task at hand.

More involved work could consist of improvements upon our ICA explainability framework or our explainability metric. A starting point for an extension of the explainability metric could be the incorporation of the feature (word) weight magnitudes instead of a binary weight based on sign (which signifies class association) alone.

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.

Appendix

Table 6: 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'

Table 7: Binary SVM 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 8: Binary SVM 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 9: Binary SVM results (not macro-averaged)

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