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

November 30, 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 punctuation as well as lemmatizing words. 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

To create word embeddings, we use the word-context co-occurrence matrix which, unlike the Bag-of-Words method, is able to incorporate information about context from the raw text data. 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 co-occurrence 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 the target word are scanned for context words. According to some sources, a window size of four or larger captures semantic representations of the words, whereas smaller windows capture more syntactic representations.

- Raw counts vs PMI (or PPMI)
- Shifted vs unshifted PMI (or PPMI)
- Laplace smoothing
- Use of start & end tokens

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 $\mathbf{e}_j$ (and, therefore, $\mathbf{v}_i$) is of size 250x1.

Overall, it seemed that larger context windows prevailed - a window size of 15 +/- performed the best (I tried window sizes of 1, 2, 4, 6, 10, 15, and 20). I find this interesting because tweets are such short posts (perhaps 30 words encompasses the entirety of the tweet for most tweets). Incorporating add-one Laplace smoothing and shifted [P]PMI decreased performance across all metrics (accuracy, precision, recall, ROC AUC, and F1 score). Interestingly, PMI often outperformed not only raw frequencies but also PPMI (positive PMI). However, with other optimal hyperparameters

held constant, there was virtually no difference in performance between PMI and PPMI. Optimal text cleaning included removing special characters which were not punctuation (parentheses, question marks, exclamation points, periods, commas, hyphens, colons, and semicolons) or alphanumeric characters (letters or numbers), converting all text to lowercase, removing stop words, and lemmatizing words (using NLTK's WordNetLemmatizer aided by NLTK's part-of-speech tagger).

The best results included ROC AUC of 0.94, accuracy of 0.89, F1 score of 0.90, precision of 0.86, and recall of 0.94. These scores are all higher than the best results from last week and were obtained using the same nested CV procedure (3 folds for inner loop; 5 folds for outer loop). NOTE: These should be replaced with the correct numbers.

- 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 $k = 250$.

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).

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 $\mathbf{U} = \mathbf{A}\mathbf{S}$. 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.

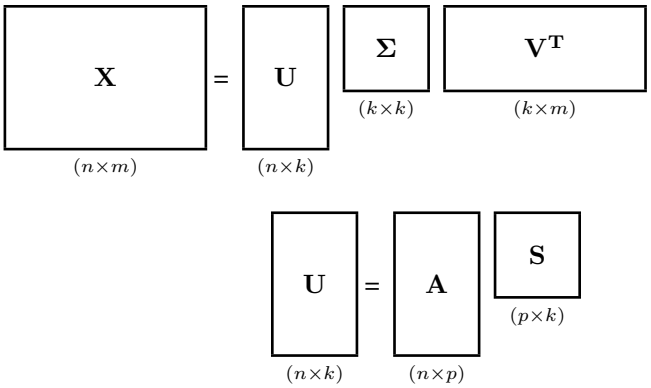

Figure 1: Truncated Singular Value Decomposition followed by Independent Componenet Analysis.

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 co-occurrence 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 1).

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).[1] 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.

3 Results

Report evaluation metrics for...

- One-class SVM
  - Performance
  - Explainability

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

- Binary SVM
  - Performance
  - Explainability

4 Discussion

- Interpret results
- Future work
  - 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.

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.

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

Table 1: 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 REEEEEEEEEEEEEEEEE'
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'