11/20/2023
CS 320.00
Prof: Anna Rafferty
Aidan Morris, A.J. Ristino
Final project

"Fake-News Classification using Machine Learning Models"

Abstract:

The phrase "fake news" has become synonymous with widespread mistrust of modern news coverage and journalistic efforts, oftentimes used by opposing political factions to discredit one another and attempt to seize control of the flow of information. Online media platforms are no exception to this unfortunate reality; site moderation on major news sites oftentimes find themselves overwhelmed with the sheer volume of content they are charged with policing. This results in a degree of uncertainty when interacting with online journalism. Fortunately, the problem of document-classification using machine learning models has been researched since the early twenty-first century. Our goals with this paper are twofold: first, explore the ML problem of text classification, and second, conduct our own experiments. The experimentation segment of this paper will consist of a comparative analysis of four models popular in text classification. We will examine the accuracy of each model in performing a binary classification using three different feature extraction methods, and two feature selection algorithms; comparing them to the results published in the literature.

I. Introduction:

The issue we are investigating is the classification of news articles as fake or real news. The current state of large media platforms has led to the rise of widespread misinformation that has fueled political and social turmoil, and obscured the truth of contemporary discourse. "A study conducted by Twitter reveals that fake news is 100 times quick[er to] spread and [share] than real news." (Fayaz, Khan, 2022) Due to the abundance of readily available information, and a lack of widespread regulation, it is oftentimes difficult for site moderation to effectively discern misinformation from quality journalism.

Our project, as stated above, consists of both a literary and experimental component. First, we will review a selection of associated literature, examine results, and attempt to follow the development of text-classification using ML over the past two decades. Second, using the information learned from our review, we will perform a comparative accuracy analysis on four types of ML models: Naïve Bayes (Gaussian and Multinomial, (GNB/MNB), k-Nearest Neighbors (kNN), and Support Vector Machines (SVMs).

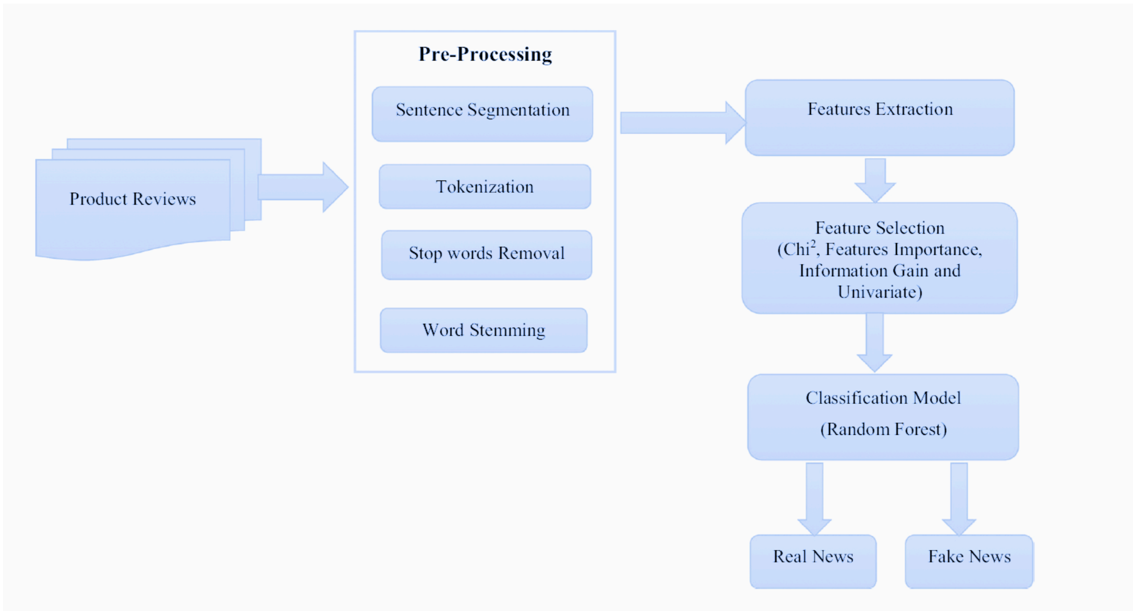
The intended input for all approaches are preprocessed documents separated into three constituent feature categories, as mentioned above. The intended output is either a 'Reliable' or 'Unreliable' label, based on the calculations performed by, first, the feature selection function and secondly, the classification performed by the ML model. With respect to our literary analysis, we will use our review to provide context and better explain how a complicated task such as classifying an entire text can be performed with impressive accuracy by an ML model.

The ultimate goal of our project is to use the sklib python library's built in ML infrastructure to perform a comprehensive analysis of best-approaches to solving the issues present within the problem of text classification; corpus processing, feature extraction, and classification. We will then compare our findings to those of our researched literature and speculate on results, using our findings as a reference.

II. Literature review

Our literary component consists of three different documents, published in 2005, 2019, and 2022. Each one examines the issue of using machine learning models to categorize texts, with varying results. These three texts covered a wide variety of approaches and informed our own experimental structure. Additionally, the timespan covered across all three documents paints a holistic picture of how continued research, advances in computation, and views on algorithms have changed the landscape of ML-model text classification.

Before diving into the literature, it's important to understand the components of the problem to better contextualize both ours and the literatures' experiments. At its core, ML text classification consists of three key components: feature extraction (pre-processing), feature selection, and training/classification. The below figure taken from Muhammad Fayaz et al.'s 2022 paper: "Machine learning for fake news classification with optimal feature selection" displays this process nicely:

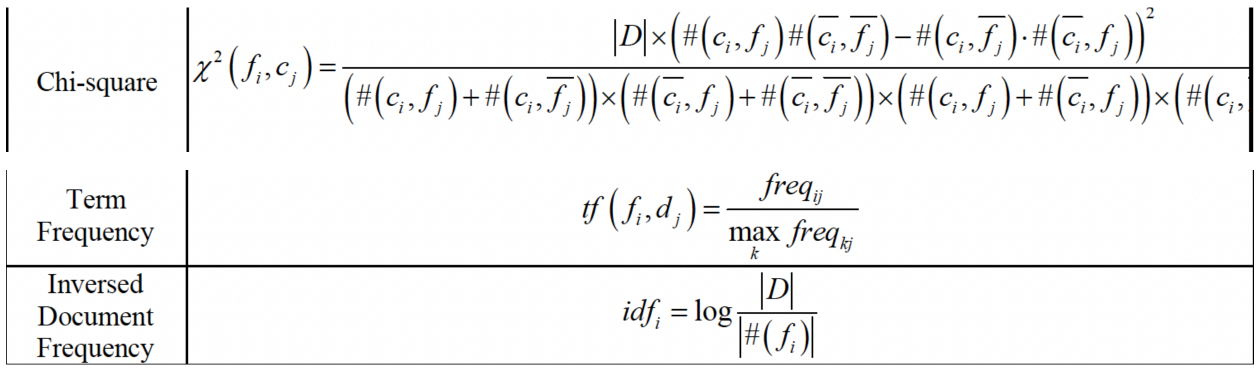


Proposed approach for fake news classification

The most notable element of this problem structure is that optimizations can be made in multiple places: Feature Extraction/Selection (FE/FS) and Classification Models. FE functions are responsible for vectorizing features taken from the pre-processed input, FS functions are responsible for picking the *n* most relevant features that can be fed into the ML model. With respect to the ML model chosen, not all models are built equally; some drastically outperform others when it comes to producing accurate predictions. It is through our lit review that we selected our desired models and FE functions for the experimentation portion of this project.

With respect to the literature itself, the first paper in our review is a 2005 survey paper entitled "Text Classification Using Machine Learning Techniques" Written by M. Ikonomakis et al. at the University of Patras in Greece. The paper itself did not conduct any original performance analysis, but rather served as an in-depth guide to the problem itself. Additionally, the paper is not directly linked to the classification of news as fake or real, but rather multiclass classification of documents. Despite this, much of the information present is still applicable to our own efforts, especially the breakdown of mathematical representations of feature extraction methods.

In section 3 of their paper, Ikonomakis et al. introduce popular choices for feature extraction functions, as well as popular feature selection techniques. Of concern to us were Chi-Squared in the former category, and TF-IDF in the latter. In addition to others, Ikonomakis provided the mathematical representation of the three terms:



Chi-square	$\chi^2(f_i,c_j)=\frac{\lvert D\rvert\times\left(\#(c_i,f_j)\#(\overline{c_i},\overline{f_j})-\#(c_i,\overline{f_j})\cdot\#(\overline{c_i},f_j)\right)^2}{\left(\#(c_i,f_j)+\#(c_i,\overline{f_j})\right)\times\left(\#(\overline{c_i},f_j)+\#(\overline{c_i},\overline{f_j})\right)\times\left(\#(c_i,f_j)+\#(\overline{c_i},f_j)\right)\times(\#(c_i,}$
Term Frequency	$tf(f_i,d_j)=\frac{freq_{ij}}{\max_k freq_{kj}}$
Inversed Document Frequency	$idf_i=\log\frac{\lvert D\rvert}{\lvert\#(f_i)\rvert}$

To explain what each technique's calculation accomplishes, let's first examine the feature selection method, Chi-squared (CS). CS examines dependencies between individual words (features) and class labels, where words that are more dependent on class labels are considered more informative to the model for classification. In other words, Chi-squared, when used as a feature selection function, identifies words that are statistically significant in differentiating between classes. In this manner, Chi-squared selects informative keywords that have a strong association with the target classes.

TF-IDF (a combination of the final two terms) is a measure of how important a term is within a document relative to its importance across a collection of documents. TF, or Term Frequency, represents the ratio of number of occurrences of a term to the total number of terms in a document. IDF or Inverse Document Frequency measures the importance of a term across a collection of documents. TF-IDF is calculated by multiplying the TF score and IDF score of a keyword together, becoming the vectorized input for a given keyword.

With respect to relevant performance metrics referenced in Ikomonaki's et al.'s paper: "In particular, they [Qiang et al.] evaluate the Vector and LSI methods, a classifier based on Support Vector Machines (SVM) and the k-Nearest Neighbor variations of the Vector and LSI models. Their results show that overall, SVMs and k-NN LSI perform better than the other methods, in a statistically significant way." (Ikonomakis et al., 2005) Noting that, in an external study, SVM and kNN performed exceptionally when coupled with LSI (applied PCA as feature processing). With respect to NB, in section five, Machine

Learning, Ikonomakis et al. observed: “However, its performance is often degraded because it does not model text well… Bayesian multinet classifier based on a new method of learning very large tree-like Bayesian networks… tree-like Bayesian networks are able to handle a text classification task in one hundred thousand variables with sufficient speed and accuracy.” In that, the base NB models encountered difficulty processing large text inputs.

Following suit, Dr. A. Kadhim of the University of Baghdad’s 2019 paper “Survey on supervised machine learning techniques for automatic text classification” is an excellent continuation of Ikonomakis et al’s paper. Much like the Greek research team, Dr. Kadhim investigated kNN, SVM and NB models. Using two different datasets of movie reviews (titled V1.0 and V2.0), Kadhim tracked the time it took for models to make predictions as well as resultant prediction accuracy. Kadhim found that “In terms of accuracy, SVM is the best algorithm for all tests (81.35%) using movie reviews in daset V2.0 and (76%) using movie reviews in dataset V1.0. SVM tends to be more accurate than other methods.” (Kadhim, 2019). Additionally, Khadim wrote in his conclusion that: “... it was observed that k-NN with TF-IDF term weighting representation scheme performs well in several text classification algorithms.” (Conclusion) which paints a favorable image for both SVM and kNN; though Kadhim neglected to mention what FE and FS functions he used in his experiment involving movie reviews.

The third and final paper in our literature review is M. Fayaz et al.’s report “Machine learning for fake news classification with optimal feature selection” published in January 2022. This report is an applied example of the problem of using ML to classify fake news, containing a comparative model and FS analysis of its own using the ISOT fake news dataset. An ISOT datapoint consists of 23 different features; Fayaz et al’s approach was to use four different FS functions: Chi-squared, Feature importance, Univariance, and Information Gain. The functions were used to extract the 14 most important features and feed them into 11 different ML models; with no reference to what FE function was used to build the keyword schedules. With respect to the results, we’ll mainly focus on Chi-squared and Information Gain. Among the models tested on the different classifiers were both kNN and three different variations of NB (Gaussian, Binomial and Monomial), but no SVM.

Fayaz et al’s analysis (Section five, analysis) aimed to demonstrate the superiority of the Random Forest (RF) model across all FE functions; which it achieved. The RF model’s best performance was 96.42% accuracy using the Information Gain FE function. With respect to IG, its mathematical notation, taken from Ikonomakis et al.’s paper, is written below:

Information Gain	$IG(t)=-\sum_{i=1}^{m}P(c_i)\log P(c_i)+P(t)\sum_{i=1}^{m}P(c_i\mid t)\log P(c_i\mid t)+P(\overline{t})\sum_{i=1}^{m}P(c_i\mid\overline{t})\log P(c_i\mid\overline{t})$

Information Gain, when used as a feature extraction method, uses a decision tree to split the data into subsets that are as homogenous as possible (i.e., similar value with respect to an examined variable). Applied to fake news classification, IG essentially examines keywords that are most effective at discerning truth or falsehood, ascribing them a value based on their relevance.

With respect to kNN and NB’s performance, the Gaussian version of NB demonstrated the worst performance overall, boasting the lowest accuracy across every FE function. The other two versions of NB cemented it as the worst performing model, achieving better performance than the Gaussian version, but falling below every other model. kNN, on the other hand, consistently landed itself in second place, right below RF as the second-most accurate model—with the highest recorded accuracy of 95.91%, using Information Gain as feature selection.

III. Methods

The dataset for our hands-on component was a free Kaggle CSV aptly named “Fake News”. The data is partitioned into three files: a training dataset, a testing dataset and an additional submission file that we will be ignoring (related to a competition ended in 2014). The content of the dataset is also from 2014, but for the sake of running this classification problem, this is not relevant. Each data point in the dataset is divided into three fields: Title, Author, and Text. To preprocess the data ourselves, we combined all three fields into one and then replaced all blank entries in the dataset (on the off chance there was one) with a blank string. Furthermore, each of the feature extractors performed their own preprocessing on the data, converting all words to lowercase and removing stop words (words that do not provide any meaning such as “an”). All extracted/selected features were words from the preprocessed data. As mentioned above, this is a binary classification problem with the labels: ‘Reliable’ or ‘Unreliable.’ as outputs. With respect to what data we actually used, we used SKLib methods to split the training dataset into a test partition and a training corpus.

We drew inspiration from our literature when choosing our models, settling on: k-Nearest-Neighbors, Naïve Bayes (Multinomial and Gaussian), and Support-Vector-Machines. As covered in class, kNN is a clustering algorithm that computes the classification of a datapoint by surveying its N nearest neighbors (using Euclidean distance) and inheriting the majority class shared amongst them. NB, on the other hand, is a probabilistic classification algorithm that calculates the posterior probabilities of classes based on observed features, making predictions by selecting the class with the highest probability. The Multinomial modifier, we applied to NB (following Fayaz et al.’s research), assumes a multinomial distribution of features in the dataset. The Gaussian variation applies the assumption of normalized data distribution. Lastly, SVMs aim is to find a hyperplane that maximizes the margin between different classes in feature space. This is accomplished by finding dimensional projections that allow for a hyperplane to be placed between two different classes to separate them.

We chose to test three different Feature Extraction methods for creating vectorized model inputs: TF-IDF, a Count Vectorizer (CV) and a Hash Vectorizer (HV). The mathematical representation and practical explanation for TF-IDF is written above in the literature review. The CV is fairly straightforward, essentially producing a vector of word counts in a given document, where the number of indices is equal to the size of the text’s vocabulary. The HV, on the other hand, uses a hash function to convert words into indices, making it far more memory efficient. Additionally, the HV automatically performs dimensionality reduction, which makes it less interpretable than the CV output. For each feature extraction method, we set the regularization metric to L2 (Euclidean) distance consistently across all tests.

For feature selection, we chose to pick the 30 best features from each keyword schedule created by the feature extraction methods. Originally, we planned to use Mutual Information as a stand-in for Information gain, since the library we used to build our models, SKLib, does not have an Information Gain function. Mutual Information, it turns out, is far too computationally intensive to be used efficiently on our dataset, so we decided to proceed without it.

Ultimately, we settled on two feature selection methods: Chi-Squared and an F statistic for classification (FC). For an explanation on how CS, see the literature review. With respect to FC, the selection process applies an F-statistic to evaluate the significance of differences in feature values, calculated by comparing variance within class means to the variance within classes. In context, this allows us to identify the most informative terms that help us distinguish between reliable and unreliable news articles using class means.

IV. Results

Our experiments examined the performance of each of the models we considered (i.e. GNB, MNB, kNN, and SVM), using testing accuracy as our performance metric while taking into account the runtimes of the models. For each model, we adjusted the feature extraction method and feature selection method to capture each possible combination of the two model attributes.

	Chi2	F-Test
TF-IDF Vectorizer	79%	**84.2%**
Count Vectorizer	79%	**84.2%**
Hash Vectorizer	79%	**84.2%**

fig. 1 - Gaussian Naive Bayes (GNB) Testing Accuracies

	Chi2	F-Test
TF-IDF Vectorizer	89.9%	**90.7%**
Count Vectorizer	89.9%	**90.7%**
Hash Vectorizer	89.9%	**90.7%**

fig. 2 - Multinomial Naive Bayes (MNB) Testing Accuracies

	Chi2	F-Test
TF-IDF Vectorizer	**91%**	**91%**
Count Vectorizer	**91%**	**91%**
Hash Vectorizer	**91%**	**91%**

fig. 3 - k-Nearest Neighbors (kNN) Testing Accuracies

	Chi2	F-Test
TF-IDF Vectorizer	**94.8%**	93.7%
Count Vectorizer	94.7%	94%
Hash Vectorizer	94.6%	93.8%

fig. 4 - Support Vector Machine (SVM) Testing Accuracies

For the GNB models, our results showed consistent testing accuracies across the feature extraction methods, with a constant 79% testing accuracy using the CS selection method and a constant 84.2% testing accuracy using the FC selection method (see fig. 1). Similar to the GNB models, MNB models also showed consistent testing accuracies across the feature extraction methods, with a constant 89.9% testing accuracy using the CS selection method and a constant 90.7% testing accuracy using the FC selection method (see fig. 2). We believe this is because Naive Bayes is a probabilistic model that only considers how the presence of a token affects the class probability. Thus, the weighting done by the IDF portion of the TF-IDF extraction method is not taken into account, resulting in no difference between it and the other two extraction methods.

The results for the kNN models demonstrate complete independence between the feature extraction method and testing accuracy. Our results recorded additional independencies between the feature selection method and testing accuracy, with the testing accuracy being a constant 91% with any combination of these parameters (see fig. 3). Since kNN classifies each data point based on the majority class of other nearby points, the way the most impactful features upon each data point's class are selected likely does not have a large impact upon the final classification. This is a reasonable assumption as, even if the two methods select some subset of different features, the selected features will still be some of the more impactful features towards determining the final classification.

The results for the SVM models show varied results depending upon the feature extraction and feature selection methods. The highest testing accuracy achieved was 94.8% coming from the combination of the TF-IDF extraction method and CS feature selection method (see fig. 4) used for input production.

When comparing the performance of the different models and extraction/selection method parameters, there is a clear hierarchy of performance between them. The best performing models were the SVMs, which generally showed higher testing accuracies compared to the other models, as well as the shortest runtimes, supporting our literature review findings. The SVMs were followed by the kNN models, with the average accuracy dropping from 94.2% to 91%. While the kNN models outperformed both of the Naive Bayes models, it is important to note that the model was more computationally taxing, taking significantly longer to train and predict than the others. Finally, the MNBs outperformed the GNBs, having average testing accuracies of 90.3% and 81.6% respectively.

V. Discussion

Using machine learning for text classification has been an ongoing area of research since the start of the 21st century. In this paper, we examined how the performance of this task has changed as new approaches have been developed, first by exploring the existing literature and then performing a comparative analysis on our findings. Our literature review proposed a multitude of possible models, a subset of which we settled on testing, namely GNBs, MNBs, kNNs, and SVMs. Furthermore, we tested varying the feature extraction and selection techniques based upon approaches in the literature. After testing these various combinations of models, extraction, and selection techniques, our results supported our literature. The GNBs and MNBs as examined by M. Ikonomakis et al. in 2005, while proving to be capable models, did not perform as well as the kNN and SVM models suggested by Dr. A. Kadhim in 2019. Our results also indicated that TF-IDF proved to be the best-performing feature extraction method across all models, providing equal or better accuracy. However, with respect to determining the most optimal feature selection method, our experiments returned mixed results; our most optimal model, SVM, was most accurate using CS, but the other three models tested performed best using the f-statistic.

Given more time and resources, we would have liked to examine additional models such as Random Forest and feature extraction methods such as Word2vec. As we read in both Muhammad Fayaz et al.'s analysis and other recent literature that we came across, Random Forest seems to have promise in the field of text classification, with potential to outperform SVMs given proper feature extraction and feature selection. Furthermore, we would have liked to perform hyperparameter testing across all models to find the best performing model on the dataset, but were unable to due to time constraints.

Our project raises some ethical concerns when considering widespread use of such models in real-world application. First, what accuracy threshold should be used when considering the viability of a given model? Furthermore, while the models that we tested averaged testing accuracies above 80% (above 90% excluding GNBs), none considered semantic meaning when predicting an article's classification. As such, these models are only considering the similarities and differences between a new article and past articles seen by the model. If society's definition of what makes an article reliable changes (such as a change in what vocabulary is considered "reliable"), or if the structure of reliable articles in the future differ significantly from reliable articles now, these models may not be able to account for these changes without retraining. Thus, they may come into conflict with societal change, classifying potentially reliable journalism as unreliable.

References:

Fayaz, M., Khan, A., Bilal, M., & Khan, S. U. (2022, January 29). *Machine learning for fake news classification with Optimal Feature Selection - Soft Computing*. SpringerLink. https://link.springer.com/article/10.1007/s00500-022-06773-x

IKONOMAKIS, M. (2005). WSEAS transactions on Computers. *WSEAS TRANSACTIONS on COMPUTERS*, *4*(8), 966–974. https://doi.org/10.37394/23205

Kadhim, A. I. (2019, January 19). *Survey on supervised machine learning techniques for automatic text classification - artificial intelligence review*. SpringerLink. https://link.springer.com/article/10.1007/s10462-018-09677-1