VeritasVigil: The truth Watchman

Saigourav Sahoo

May 17, 2025

1 Introduction

It begins on a storm-lashed night. A flickering lamp, the rustle of yellowing paper, and a spine-chilling dispatch slides onto your desk. Headline after headline screams from the shadows-some whispering truths soaked in danger, others spinning illusions designed to deceive and disturb. The city sleeps, but you, the detective of digital deceit, must stay awake.

To sift through this haunted stack of news-some real, some wickedly fake—and uncover the facts hidden beneath their frightful masks. These aren't just hoaxes; they are meticulously crafted phantoms of journalism. Some headlines are stretched with unnatural screams (sooooo scary!), others riddled with cryptic marks and unsettling patterns.

2 Objective

The goal of this task is to develop a report using Natural Language Processing tools and supervised learning algorithms to classify true and fake news. The purpose of the report is to explain the entire process that goes from the creation of the data set to the obtaining of the classification accuracy.

3 Understanding the Dataset

the dataset contains two csv files: "True.csv" and "Fake.csv" whose news are true and fake respectively. Each dataset contains the following columns:

1. title: the title of the news
2. text: some text regarding the news that gives some details about the news
3. subject: which type of news it is
4. date: on which date the news was published

For most of the processing purposes, I have used only the titles of the news reports. Since the accuracies are pretty high as is, there seemed to be no special point in including the text; further, the inclusion of the text column would have drastically increased training time.

Another thing to note is the shortness and crispness of the titles. Although the choice of using only the title column is largely based on trial and error, from a layman's perspective and from a human inference perspective, fake news is clear from the title itself.

4 Data Preprocessing for Custom Tokenization

To make a tokenizer that deals with contractions, emoticons and repeated characters, there is a need for some exploratory data analysis.

The idea is to surf through the text to list out contractions, emoticons and repeated characters, and to create a dictionary to replace them suitably. For this, I make use of string methods and the Regex (Regular Expression) library to enlist such expressions.

The use of a simple rule to find contractions - an apostrophe in the middle of a word - seems to return some contractions and all possessives (example: Trump's). Thus, we can manually go through this list and make the dictionary of necessary contractions. To make this manually feasible, the list is created from only the true news dataset following the argument that the true news dataset would generate a rather exhaustive list of contractions which should (for all practical purposes) generate a list agreeable to one generated by the fake news dataset.

The use of emoticons is rather rare in such datasets; thus, a dictionary created from the most common emoticons should suffice.

An exhaustive argument similar to contractions can be extended to repeated characters. Going through the dataset exhaustively and enlisting all repeated characters through a simple rule - any character of a word that is not a number repeated more than three times - generates the required dictionary to tokenize the same.

5 Mini Part-of-Speech Tagger and Rule-based Lemmatizer: Custom Design

The heart of the whole classification process is the implementation of a custom Rule-based Lemmatizer operating on a Part-of-Speech Tagger.

5.1 Inspiration

The uniqueness of a rule-based POS tagger and Lemmatizer is its simplicity. For a complex language like English which does not exactly obey lexicographic or grammatical rules for stemming words, we can still implement a stemmer that

”strictly” follows a given set of rules - in this case, some basic dictionary substitutions and suffix rules - to obtain root words. They are not perfect, agreed, but that is where we can use a major ”flaw” of classification algorithms - they simply do not care. They do not comprehend the true meaning or contextual meaning of a word as humans can. So, even if the root words are not exactly what they should be, the classification algorithm still works well enough.

5.2 Rules used

Basic rules are used to Lemmatize the text.

1. Articles: articles tagged as ART are to ensure that the rest of the rules do not tamper the articles since they would be sensitive to the same.

2. Verbs: a simple suffix rule is used to stem the root words from suffixes -ing (helping), -ed (helped), -pt (slept), -ung(swung), -en(enlighten), -ise(scrutinise), -ize(digitize), -ate(interrogate) will be tagged as VERB and the root word is extracted by removing the suffix from them.

   problems: words like estate, state, late etc. would be tagged and stemmed incorrectly. Words are like create, cremate, skate etc. would be stemmed incorrectly

3. Nouns: for proper nouns, words that are entirely in uppercase or have their first letter in uppercase are tagged NOUN. Along with this, noun suffix rules is used. suffixes used for the same are -ion (creation), -age (shortage), -al (retrieval), -dom (boredom), -ee (employee), -hood (brotherhood), -ism (sexism), -ist (racist), -ment (embarrassment), -ness (shyness), -ry (bakery), -ship (friendship), -vity (creativity)

   problems: words in which suffix is attached after dropping letters (like trivia -¿ trivial) would be stemmed incorrectly. words like creativity and creation that have the same root word but would be stemmed differently. Plural nouns cannot be handled.

4. Adverbs: since adverbs are of many different types, the goal was to stem adverbs that use -ly (shyly) or -ily (heartily) as suffixes.

   problems: words in which suffixes are attached after dropping last letter (true -¿ truly) are stemmed incorrectly. Words like rally, jolly etc. would be tagged as ADV instead of NOUN

5. Adjectives: simple suffix rules are used to stem adjectives to root words. the suffixes used are -er (louder), -est (loudest), -less (clueless), -ful (beautiful), -ous (nervous), -able (portable), -al (formal), -ish (childish), -ian (Canadian), -ic (poetic), -i (iraqi), -y (rainy)

   problems: words that are formed by dropping the last letter and adding suffix (beauty -¿ beautiful) are stemmed incorrectly. Some words are mistagged.

Words other than the ones stated above are tagged as UNK (unknown). These include untagged nouns, verbs, adjectives and adverbs as well as prepositions, conjunctions, determiners and modals. These unknown words are left untouched in the lemmatized title. It would have been possible to tag them as well but since the tagger was developed with the aim of being used in the Lemmatizer and since the classifiers give high enough accuracy without them being tagged, no need was felt to tag them.

6 Vectorization and Classification

Two vectorizers are implemented from the default classes of scikit-learn, namely the count-vectorizer and the TFID-vectorizer. These vectorizers are implemented on the lemmatized titles to create and extract features. These features are then fed to two different classification models: Naive Bayes Classifier and Support Vector Classifier.

While both models show almost similar results for both types of vectorizers, the support vector classifier works a little better than the Naive Bayes for both types of vectorizers.

6.1 Confusion Matrices

From the confusion matrices, it can be seen that the Naive Bayes Classifier has labelled more fake news as true news than the Support Vector Classifier. This may be attributed to its assumption of independent features, which may result in loss of contextual information, leading to this sort of misclassification. On the other hand, the Support Vector Classifier is more likely to label a true news as fake news.

6.2 ROC Curves

It is clear from the Receiver Operating Characteristic Curve that the classifiers used for both Bag-of-Words and TFIDF Vectorizers are near-perfect models, with Area Under Curve tending to 1 (rounded off to nearest 2 decimal places).

7 Custom vs Off-the-shelf methods

Off-the-shelf tools are statistical and rule-based and thus, handle textual information much better than the methods implemented in this task by the custom stemmer. However there are a few advantages that custom stemmer has on the default tools:

1. handling of informal texts: The custom tokenizer includes dictionaries to resolve issues with informal texts such as elongated words and emoticons. This feature is not available if a direct pipeline is implemented using a library, say Spacy.

2. interpretability: The statistical tools employed to lemmatize texts are complex and trained rigorously on a plethora of corpora. Therefore, to understand them requires heavy contextual knowledge. A rule-based stemmer is easy to comprehend and design.

3. scope of expansion: There is scope to expand the custom desgin to suit specific corpora. Here, the rules are desgined to rudimentarily identify fake news by the use of informal language. Similarly, this custom design can be expanded based on what is expected out of the specific corpora. This, in some cases, actually improve model performance.

8 Conclusion

The objective of the task has been achieved by two classifiers operating on two different types of vectorizer.

9 References

https://dictionary.cambridge.org/grammar/british-grammar/suffixes
https://www.geeksforgeeks.org/text-classification-using-scikit-learn-in-nlp/
https://www.geeksforgeeks.org/rule-based-stemming-in-natural-language-processing/
https://www.geeksforgeeks.org/generating-word-cloud-python/
https://www.geeksforgeeks.org/auc-roc-curve/
https://www.geeksforgeeks.org/naive-bayes-classifiers/
https://www.geeksforgeeks.org/support-vector-machine-algorithm/