Development, Analysis, and Demonstration of a Rudimentary fastText Classification Model for COVID-19 Misinformation

Alexandra Zharchuk

2024-05-17

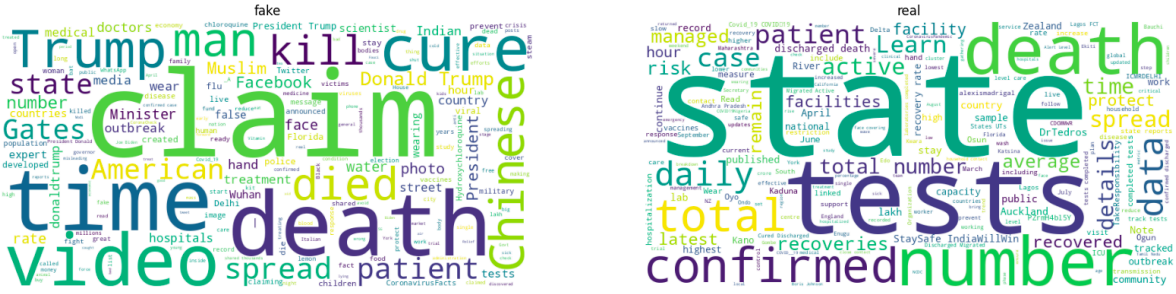


Figure 1: Significant keywords in fake vs. real labeled tweets generated with WordCloud

Introduction

Since its initial appearance in March 2020, the novel coronavirus has undeniably altered history. Leading to over 7 million global casualties from 2020 to 2024, the virus and its resulting disease have become widely recognized topics of conversation.

The urgency surrounding the virus has generated countless narratives including but not limited to its symptomology, treatment, outcomes, and speculative origins. Many of these narratives are validated by scientific investigation and propagated by authorized information channels (government, local, scientific, academic) on popular social media sites.

On the contrary, there exists a subsection of narrative that presents unsubstantiated information - often propagated through the same social networking outlets used by authorized entities. These narratives are often based on unverified claims and false information about various aspects of COVID-19 and its causative agent.

The distribution and origins of COVID-19 misinformation are broad and thus difficult to fully classify. However, American government intelligence suggests that parties involved in COVID-19 misinformation propagation are malicious non-state actors, governments, and organizations[1]. Misinformation from these sources is often generated by automated processes (bots) and disseminated on Twitter, Facebook, Instagram, and other social media platforms. Such misinformation content is often generated to instill chaos, mistrust, and confusion. Other sources of misinformation are attributed to the internet presence of high-level politicians, celebrities, and other public figures[2] .

There is general agreement that misinformation hinders public health progress and generates public confusion. Additionally, there is an inverse relationship between the exponential increase of generated internet content and the collective ability to process it. The loss of information literacy paired with colossal data generation brings a specific challenge to misinformation mitigation efforts.

An appropriate solution to mitigate misinformation spread is the application of machine learning methods (MLM) for online misinformation filtering. Given the utility of MLM in handling large, live data - these methods can be employed to detect and filter out false information in real-time.

As machine learning becomes more advanced and accessible, it is of interest to explore text identification methods using MLM techniques such as natural language processing (NLP). A particular MLM, fastText, is notable for its efficiency in this domain.

---

[1] Kavanagh, J., & Rich, M. D. (2021). **Truth Decay in the COVID-19 Pandemic: Exploring Sources and Impacts of Misinformation**. In S. J. Gentry & K. E. Boyce (Eds.), *Pandemics and the Crisis of Information* (pp. 115-136). Springer. https://doi.org/10.1007/978-3-030-73955-3_7

[2] Newman, N., Fletcher, R., Schulz, A., Andı, S., Robertson, C. T., & Nielsen, R. K. (2020). **Types, Sources, and Claims of COVID-19 Misinformation**. Reuters Institute for the Study of Journalism. Retrieved from https://reutersinstitute.politics.ox.ac.uk/types-sources-and-claims-covid-19-misinformation

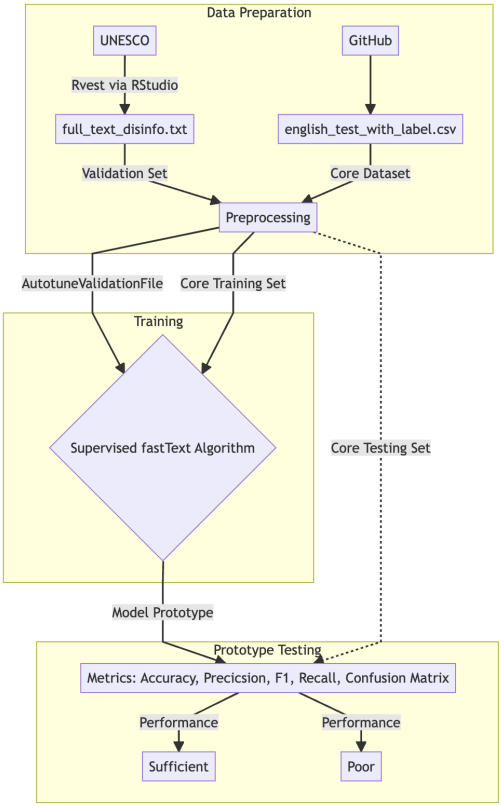
fastText is a popular NLP algorithm developed by formerly Facebook's AI Research Lab. As well as showing success in text classification, fastText is notable for its ability to quickly process large datasets. Its efficiency, scalability, and ease of use make it an ideal candidate for developing misinformation sentiment analysis tools.

In this analysis, fastText's capabilities are leveraged to develop a rudimentary text classification model trained on online COVID-19 Twitter content. The ideal goal of the model development process is an effective demonstration of fastText's efficacy in labeling text as COVID-19 misinformation.

Abstract

A fastText model is developed to label COVID-19 content as 'fake' or 'real' information. The model is trained, tested, evaluated, and deployed on 4 pieces of selected COVID-19 related text content. Customized preprocessing methods for misinformation-specific data are explored.

Data Collection and Summary



Core Dataset:

The core dataset serves as the main dataset for training and testing the fastText model. The original source can be found in the GitHub repository for ArXiv article '*Combating COVID-19 Misinformation on Social Media: A Scalable, Semisupervised Learning Approach*'.[3]

The original dataset contains 3 columns with 2140 observations in total:

`id tweet label`

The training dataset consists of English-language tweets related to COVID-19. The data was sourced from social media platforms and labeled with sentiment (positive, negative, neutral) and misinformation tags (real, fake). The training data comprised tweets with corresponding labels, while a separate validation set was used for model tuning. More information on the core dataset is available on its parent GitHub repository page.

Validation Dataset:

The validation set, used for autotuning fastText hyperparameters, is derived from a large UNESCO dataset on COVID-19 misinformation.[4]

The source dataset contains 34 columns with 5645 observations in total.

`s_no`	`Secondary_Country`
`Source`	`Primary_Country`
`Recoded_Main_Narrative`	`Entry_Date`
`Motive`	`Publication_Date`
`Motive_Description`	`Title`
`Reported_On`	`Direct_Post_4`
`Distrib_Channel`	`Direct_Post_3`
`Misinfo_Type`	`Direct_Post_2`
`Key_Words`	`Direct_Post_1`
`Summary Coder`	`Twitter_Reference`
`Notes`	`Retrieve_from_3`
`Secondary_Language`	`Retrieve_from_2`
`Primary_Language`	`Retrieve_from_1`
	`Region`

The misinformation itself was derived from the URLs listed in the `Reported_On` column. Webpage content was scraped with Rvest in RStudio and compiled into `full_text_disinfo.txt`.

[3]Github Repository: Sharma, K., Qian, F., Jiang, H., Ruchansky, N., Zhang, M., & Liu, Y. (2021). Combating COVID-19 Misinformation on Social Media: A Scalable, Semisupervised Learning Approach [Code repository]. Retrieved from https://github.com/diptamath/covid_fake_news

[4]UNESCO. (n.d.). ESOC COVID-19 Misinformation Dataset. Retrieved from https://www.unesco.org/en/world-media-trends/esoc-covid-19-misinformation-dataset

The derived dataset sourced text from 5645 URLs and yielded 1815376 characters and 268636 words total.

Because the fastText model weights are not affected by this data, labels were not required when setting the fastText `autotuneValidation` parameter.

Pre-Processing

Stopword removal, label formatting, and character removal methods were embedded into a custom "clutter removal" function and applied to the core dataset. Because fastText automatically tokenizes and vectorizes text, the techniques have been omitted from the preprocessing workflow.

Custom Stopword Removal	**Label Formatting**	**Custom Character Removal**
A customized list of stopwords was created and added to default *sklearn* stopwords. Custom stopwords are categorized as those present in all COVID-19 related content. The full list of topic-exclusive keywords can be found in this document's code index.	Each text entry was formatted to include an appropriate `__label__`prefix required by fastText.	Special characters such as "#", "@", strings such as "https", "t.co", emojis, punctuation, and whitespace were removed to aid data readability.

As a final preprocessing step, the core dataset was split into training and testing sets. The testing set was created by methodically removing labels to hide true labels from the model. Labels were retained in the training set.

Model Training

The labeled and pre-processed core data were used to train the fastText model, sourced directly from the *fasttext* Python library.

Hyperparameter Selection

The 'autotune' feature of fastText was included to optimize hyperparameters. This feature employs an external validation file for hyperparameter tuning. The tuning duration parameter was manually set to 20 seconds. The label parameter was manually set for the model to better identify text entries with the `__label__` prefix.

Evaluation

Evaluation was performed by applying testing data to the model prototype shortly after training. The models' performance was evaluated with the following classification metrics: precision, accuracy, recall, F1-score. A confusion matrix was generated to visualize the distribution of true positive, true negative, false positive, and false negative classifications.

	precision	recall	f1-score	support
fake	0.87	0.92	0.89	1020
real	0.92	0.87	0.90	1120
accuracy			0.89	2140
macro avg	0.89	0.90	0.89	2140
weighted avg	0.90	0.89	0.89	2140

Figure 2: Classification Metrics

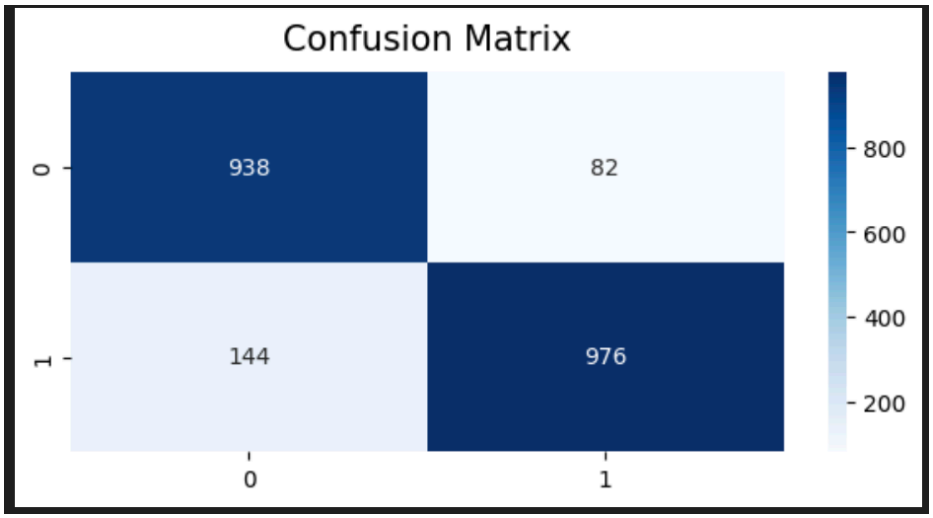


Figure 3: Confusion Matrix

Results

The model classification metrics show an accuracy of ~0.89-0.90. Precision scores for labels "fake" and "real" are 0.87, 0.92 respectively. The model indicates strong performance in classifying the sentiment of tweets related to COVID-19 but does show a tendency in mislabeling "fake" text as "real".

Label Prediction Demonstration Results

The model correctly labeled 4 of 4 text inputs. See Code Index for label prediction details.

Conclusion

This analysis demonstrates the potential of fastText in the classification of COVID-19 misinformation. By accurately classifying sentiment, this tool can help identify, label, and mitigate the spread of false information on social media platforms. The simplicity of this tool also provides a useful introduction to machine learning application in misinformation classification.

Discussion & Limitations

The results indicate that the fastText model is moderately effective in classifying misinformation sentiment in text related to COVID-19. The preprocessing steps, particularly the custom stopword removal, were significant to the model's accuracy score. This approach can be extended to other misinformation topics where sentiment is dependent on topic-specific keywords.

Limitations of the Analysis:

1. **Validation Data Source**: It is not known to what extent the validation dataset influences model optimization. The webpage content in question was derived from URLs that link to sources that reported on the source of misinformation. The content was not directly derived from misinformation content and contained text written in languages other than English. It is possible that model performance can be improved by sourcing from direct misinformation content links in the validation set. Additionally, content should contain information written in the English language.
2. **Low Observation Size of Testing and Training Data**: The size of the training and testing data is relatively small in comparison to the validation set. The model's ability to classify text labels could be improved with the inclusion of larger training and testing datasets.
3. **Low Number of Testing Iterations**: The number of testing iterations was limited to one for the brevity of this analysis. Model performance could be improved and further examined by increasing the number of testing iterations and accompanied testing datasets.

4. **Uniform Training Data Source**: Because the model was trained only on content derived from Twitter, there may be a knowledge gap in the model's corpus. When applied to classification on content outside of Twitter, the model may show inconsistent prediction power.
5. **Homogenous Testing and Training Data**: The model's performance may have been compromised by uniformity in the testing and training data. Both the testing and training data are identical except for the absence of labels in the training split.
6. **Low Number of Prediction Capacity Demonstrations**: This analysis included 4 demonstrations of the model's predictive capacity. The 4 demonstrations are indicative of performance, but not conclusive. Performing more predictions can finalize the model's true prediction power.

Future Direction

As stated in the limitations, the model can be further evaluated with the addition of larger and heterogeneous training and testing data, a more specific validation dataset, and increased testing iterations for the model. Including misinformation data from sources other than Twitter in training data can aid in the model's prediction capacity. Furthermore, the true classification power of this model can be assessed with further label prediction trials.

Code Index

Environment Setup

```
import string
import numpy as np
import pandas as pd
import seaborn as sns
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import fasttext
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
import matplotlib as plt
from matplotlib import pyplot as plt
import sklearn.metrics as skm
```

```
from sklearn.metrics import classification_report, accuracy_score, precision_score,
 ↪  recall_score, f1_score, precision_recall_fscore_support as score
from sklearn.metrics import confusion_matrix
```

```
path = ('/Users/alexz/Programming/Python Projects/Sentiment Analysis of COVID-19
 ↪  Misinformation with
 ↪  fastText/Sentiment-Analysis-of-COVID-19-Misinformation-with-fastText')
```

```
train_data_english = pd.read_csv('/Users/alexz/Programming/Python Projects/Sentiment
 ↪  Analysis of COVID-19 Misinformation with fastText/Sentiment-Analysis-of-COVID-19-
 ↪  Misinformation-with-fastText/english_test_with_labels.csv')

peripheral_text =('/Users/alexz/Programming/Python Projects/Sentiment Analysis of
 ↪  COVID-19 Misinformation with fastText/Sentiment-Analysis-of-COVID-19-
 ↪  Misinformation-with-fastText/full_text_cleaned.txt')

validation_set = []
with open(peripheral_text, 'r') as f:
    # Comment: Validation set
    for line in f:
        line = line.replace("    ", "\t").strip()
        list = validation_set.append(line)
        list = validation_set.append('\n')
validation_set = pd.Series(validation_set)
```

Creating Custom Stopword List and Preprocessing Functions

```
#create custom stopword list
custom_stopwords = set([
'covid', 'covid-19', 'coronavirus', 'pandemic', 'virus',
'vaccine', 'vaccination', 'vaccinated', 'vaccinate',
'symptoms', 'cases', 'infection', 'infected', 'infections',
'health', 'healthcare', 'hospital', 'doctor', 'nurse',
'mask', 'masks', 'quarantine', 'lockdown', 'isolation',
'positive', 'negative', 'test', 'testing', 'tested',
'who', 'cdc', 'government', 'authorities', 'official', 'officials',
'social', 'distancing', 'guidelines', 'rules', 'regulations',
'information', 'misinformation', 'news', 'article', 'post', 'comment', 'share',
```

```
'people', 'person', 'individual', 'individuals', 'group', 'groups',
'update', 'report', 'reported', 'reporting', 'today', 'yesterday', 'tomorrow',
'day', 'days', 'week', 'weeks', 'month', 'months', 'year', 'years'
'https', 'http', 'www', 'com', 'org', 'net', 'gov', 'edu', 'html', 'php', 'asp',
 ↪  'aspx',
'jsp', 'php', 'cfm','#', 't',
 ↪  'co','http','https','s','COVID','COVID19','Corona','amp','a','b','c',
'd','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
"IndiaFightsCorona", "CoronaVirusUpdates","MoHFW_India","RT", "NPR", "CDCgov", "WHO",
 ↪  "HHSGov",
"CNBC", "CNN", "BBCWorld", "nytimes", "CNN" , "BBCWorld", "nytimes", "washingtonpost",
 ↪  "guardian", "Reuters",
"CDC", "0o", "0s", "3a", "3b", "3d", "6b", "6o", "a", "a1", "a2", "a3", "a4", "ab",
 ↪  "able", "about",
"above", "abst", "ac", "accordance", "according", "accordingly", "across", "act",
 ↪  "actually",
"ad", "added", "adj", "ae", "af", "affected", "affecting", "affects", "after",
 ↪  "afterwards",
"ag", "again", "against", "ah", "ain", "ain't", "aj", "al", "all", "allow", "allows",
"almost", "alone", "along", "already", "also", "although", "always", "am", "among",
"amongst", "amoungst", "amount", "an", "and", "announce", "another", "any", "anybody",
 ↪
"anyhow", "anymore", "anyone", "anything",
"anyway", "anyways", "anywhere", "ao", "ap", "apart", "apparently", "appear",
 ↪  "appreciate",
"appropriate", "approximately", "ar", "are", "aren", "arent", "aren't", "arise",
 ↪  "around",
"as", "a's", "aside", "ask", "asking", "associated", "at", "au", "auth", "av",
 ↪  "available",
"aw", "away", "awfully", "ax", "ay", "az", "b", "b1", "b2", "b3", "ba", "back", "bc",
 ↪  "bd", "be",
"became", "because", "become", "becomes", "becoming", "been", "before", "beforehand",
 ↪  "begin",
"beginning", "beginnings", "begins", "behind", "being", "believe", "below", "beside",
 ↪  "besides",
"best", "better", "between", "beyond", "bi", "bill", "biol", "bj", "bk", "bl", "bn",
 ↪  "both",
"bottom", "bp", "br", "brief", "briefly", "bs", "bt", "bu", "but", "bx", "by", "c",
 ↪  "c1", "c2",
"c3", "ca", "call", "came", "can", "cannot", "cant", "can't", "cause", "causes", "cc",
 ↪  "cd",
"ce", "certain", "certainly", "cf", "cg", "ch", "changes", "ci", "cit", "cj", "cl",
 ↪  "clearly",
```

```
"cm", "c'mon", "cn", "co", "com", "come", "comes", "con", "concerning",
 ↪  "consequently",
"consider", "considering", "contain", "containing", "contains", "corresponding",
 ↪  "could",
"couldn", "couldnt", "couldn't", "course", "cp", "cq", "cr", "cry", "cs", "c's", "ct",
 ↪  "cu",
"currently", "cv", "cx", "cy", "cz", "d", "d2", "da", "date", "dc", "dd", "de",
 ↪  "definitely",
"describe", "described", "despite", "detail", "df", "di", "did", "didn", "didn't",
 ↪  "different",
"dj", "dk", "dl", "do", "does", "doesn", "doesn't", "doing", "don", "done", "don't",
 ↪  "down",
"downwards", "dp", "dr", "ds", "dt", "du", "due", "during", "dx", "dy", "e", "e2",
 ↪  "e3",
"ea", "each", "ec", "ed", "edu", "ee", "ef", "effect", "eg", "ei", "eight", "eighty",
"either", "ej", "el", "eleven", "else", "elsewhere", "em", "empty", "en", "end",
 ↪  "ending",
"enough", "entirely", "eo", "ep", "eq", "er", "es", "especially", "est", "et",
 ↪  "et-al", "etc",
"eu", "ev", "even", "ever", "every", "everybody", "everyone", "everything",
 ↪  "everywhere",
"ex", "exactly", "example", "except", "ey", "f", "f2", "fa", "far", "fc", "few", "ff",
 ↪  "fi",
"fifteen", "fifth", "fify", "fill", "find", "fire", "first", "five", "fix", "fj",
 ↪  "fl", "fn",
"fo", "followed", "following", "follows", "for", "former", "formerly", "forth",
 ↪  "forty",
"found", "four", "fr", "from", "front", "fs", "ft", "fu", "full", "further",
"furthermore", "fy", "g", "ga", "gave", "ge", "get", "gets", "getting", "gi", "give",
"given", "gives", "giving", "gj", "gl", "go", "goes", "going", "gone", "got",
 ↪  "gotten",
"gr", "greetings", "gs", "gy", "h", "h2", "h3", "had", "hadn", "hadn't", "happens",
"hardly", "has", "hasn", "hasnt", "hasn't", "have", "haven", "haven't", "having",
"he", "hed", "he'd", "he'll", "hello", "help", "hence", "her", "here", "hereafter",
"hereby", "herein", "heres", "here's", "hereupon", "hers", "herself", "hes", "he's",
"hh", "hi", "hid", "him", "himself", "his", "hither", "hj", "ho", "home", "hopefully",
 ↪
"how", "howbeit", "however", "how's", "hr", "hs", "http", "hu", "hundred", "hy", "i",
"i2", "i3", "i4", "i6", "i7", "i8", "ia", "ib", "ibid", "ic", "id", "i'd", "ie", "if",
"ig", "ignored", "ih", "ii", "ij", "il", "i'll", "im", "i'm", "immediate",
 ↪  "immediately",
"importance", "important", "in", "inasmuch", "inc", "indeed", "index", "indicate",
```

```
"indicated", "indicates", "information", "inner", "insofar", "instead", "interest",
"into", "invention", "inward", "io", "ip", "iq", "ir", "is", "isn", "isn't", "it",
 ↪  "itd",
"it'd", "it'll", "its", "it's", "itself", "iv", "i've", "ix", "iy", "iz", "j", "jj",
 ↪  "jr",
"js", "jt", "ju", "just", "k", "ke", "keep", "keeps", "kept", "kg", "kj", "km", "know",
"known", "knows", "ko", "l", "l2", "la", "largely", "last", "lately", "later",
 ↪  "latter",
"latterly", "lb", "lc", "le", "least", "les", "less", "lest", "let", "lets", "let's",
"lf", "like", "liked", "likely", "line", "little", "lj", "ll", "ll", "ln", "lo",
 ↪  "look",
"looking", "looks", "los", "lr", "ls", "lt", "ltd", "m", "m2", "ma", "made", "mainly",
 ↪  "make",
"makes", "many", "may", "maybe", "me", "mean", "means", "meantime", "meanwhile",
 ↪  "merely", "mg",
"might", "mightn", "mightn't", "mill", "million", "mine", "miss", "ml", "mn", "mo",
"more", "moreover", "most", "mostly", "move", "mr", "mrs", "ms", "mt", "mu", "much",
"mug", "must", "mustn", "mustn't", "my", "myself", "n", "n2", "na", "name",
"namely", "nay", "nc", "nd", "ne", "near", "nearly", "necessarily", "necessary",
"need", "needn", "needn't", "needs", "neither", "never", "nevertheless", "new",
"next", "ng", "ni", "nine", "ninety", "nj", "nl", "nn", "no", "nobody", "non",
"none", "nonetheless", "noone", "nor", "normally", "nos", "not", "noted", "nothing",
"novel", "now", "nowhere", "nr", "ns", "nt", "ny", "o", "oa", "ob", "obtain",
"obtained", "obviously", "oc", "od", "of", "off", "often", "og", "oh", "oi",
"oj", "ok", "okay", "ol", "old", "om", "omitted", "on", "once", "one", "ones",
"only", "onto", "oo", "op", "oq", "or", "ord", "os", "ot", "other", "others",
"otherwise", "ou", "ought", "our", "ours", "ourselves", "out", "outside", "over",
"overall", "ow", "owing", "own", "ox", "oz", "p", "p1", "p2", "p3", "page",
"pagecount", "pages", "par", "part", "particular", "particularly", "pas", "past",
"pc", "pd", "pe", "per", "perhaps", "pf", "ph", "pi", "pj", "pk", "pl", "placed",
"please", "plus", "pm", "pn", "po", "poorly", "possible", "possibly", "potentially",
"pp", "pq", "pr", "predominantly", "present", "presumably", "previously", "primarily",
 ↪
"probably", "promptly", "proud", "provides", "ps", "pt", "pu", "put", "py", "q", "qj",
"qu", "que", "quickly", "quite", "qv", "r", "r2", "ra", "ran", "rather", "rc",
"rd", "re", "readily", "really", "reasonably", "recent", "recently", "ref",
"refs", "regarding", "regardless", "regards", "related", "relatively", "research",
"research-articl", "respectively", "resulted", "resulting", "results", "rf",
"rh", "ri", "right", "rj", "rl", "rm", "rn", "ro", "rq", "rr", "rs", "rt",
"ru", "run", "rv", "ry", "s", "s2", "sa", "said", "same", "saw", "say",
"saying", "says", "sc", "sd", "se", "sec", "second", "secondly", "section",
"see", "seeing", "seem", "seemed", "seeming", "seems", "seen", "self",
```

```
"selves", "sensible", "sent", "serious", "seriously", "seven", "several",
"sf", "shall", "shan", "shan't", "she", "shed", "she'd", "she'll", "shes", "she's",
"should", "shouldn", "shouldn't", "should've", "show", "showed", "shown",
"showns", "shows", "si", "side", "significant", "significantly", "similar",
"similarly", "since", "sincere", "six", "sixty", "sj", "sl", "slightly",
"sm", "sn", "so", "some", "somebody", "somehow", "someone", "somethan",
"something", "sometime", "sometimes", "somewhat", "somewhere", "soon", "sorry",
"sp", "specifically", "specified", "specify", "specifying", "sq", "sr",
"ss", "st", "still", "stop", "strongly", "sub", "substantially",
"successfully", "such", "sufficiently", "suggest", "sup", "sure", "sy",
"system", "sz", "t", "t1", "t2", "t3", "take", "taken", "taking", "tb",
"tc", "td", "te", "tell", "ten", "tends", "tf", "th", "than", "thank",
"thanks", "thanx", "that", "that'll", "thats", "that's", "that've", "the",
"their", "theirs", "them", "themselves", "then", "thence", "there",
"thereafter", "thereby", "thered", "therefore", "therein", "there'll",
"thereof", "therere", "theres", "there's", "thereto", "thereupon",
"there've", "these", "they", "theyd", "they'd", "they'll", "theyre",
"they're", "they've", "thickv", "thin", "think", "third", "this", "thorough",
"thoroughly", "those", "thou", "though", "thoughh", "thousand", "three",
"throug", "through", "throughout", "thru", "thus", "ti", "til", "tip",
"tj", "tl", "tm", "tn", "to", "together", "too", "took", "top", "toward",
"towards", "tp", "tq", "tr", "tried", "tries", "truly", "try", "trying",
"ts", "t's", "tt", "tv", "twelve", "twenty", "twice", "two", "tx", "u",
"u201d", "ue", "ui", "uj", "uk", "um", "un", "under", "unfortunately",
"unless", "unlike", "unlikely", "until", "unto", "uo", "up", "upon",
"ups", "ur", "us", "use", "used", "useful", "usefully", "usefulness",
"uses", "using", "usually", "ut", "v", "va", "value", "various", "vd",
"ve", "ve", "very", "via", "viz", "vj", "vo", "vol", "vols", "volumtype",
"vq", "vs", "vt", "vu", "w", "wa", "want", "wants", "was", "wasn", "wasnt",
"wasn't", "way", "we", "wed", "we'd", "welcome", "well", "we'll",
"well-b", "went", "were", "we're", "weren", "werent", "weren't",
"we've", "what", "whatever", "what'll", "whats", "what's", "when",
"whence", "whenever", "when's", "where", "whereafter", "whereas",
"whereby", "wherein", "wheres", "where's", "whereupon", "wherever",
"whether", "which", "while", "whim", "whither", "who", "whod", "whoever",
"whole", "who'll", "whom", "whomever", "whos", "who's", "whose", "why", "why's",
"wi", "widely", "will", "willing", "wish", "with", "within", "without", "wo",
"won", "wonder", "wont", "won't", "words", "world", "would", "wouldn", "wouldnt",
"wouldn't", "www", "x", "x1", "x2", "x3", "xf", "xi", "xj", "xk",
"xl", "xn", "xo", "xs", "xt", "xv", "xx", "y", "y2", "yes", "yet",
"yj", "yl", "you", "youd", "you'd", "you'll", "your", "youre",
"you're", "yours", "yourself", "yourselves", "you've", "yr", "ys",
```

```
"yt", "z", "zero", "zi", "zz", "COVID19", "Afghanistan", "Albania",
"Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina", "Armenia",
"Australia", "Austria", "Azerbaijan", "Bahamas", "Bahrain", "Bangladesh",
"Barbados", "Belarus", "Belgium", "Belize", "Benin", "Bhutan", "Bolivia",
"Bosnia and Herzegovina", "Botswana", "Brazil", "Brunei", "Bulgaria", "Burkina Faso",
"Burundi", "Cabo Verde", "Cambodia", "Cameroon", "Canada", "Central African Republic",
 ↪
"Chad", "Chile", "China", "Colombia", "Comoros", "Congo (Congo-Brazzaville)",
"Costa Rica", "Croatia", "Cuba", "Cyprus", "Czech Republic (Czechia)",
"Democratic Republic of the Congo", "Denmark", "Djibouti", "Dominica",
"Dominican Republic", "Ecuador", "Egypt", "El Salvador", "Equatorial Guinea",
"Eritrea", "Estonia", "Eswatini", "Ethiopia", "Fiji", "Finland",
"France", "Gabon", "Gambia", "Georgia", "Germany", "Ghana", "Greece",
"Grenada", "Guatemala", "Guinea", "Guinea-Bissau", "Guyana", "Haiti",
"Honduras", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Iraq",
"Ireland", "Israel", "Italy", "Jamaica", "Japan", "Jordan", "Kazakhstan",
"Kenya", "Kiribati", "Kuwait", "Kyrgyzstan", "Laos", "Latvia",
"Lebanon", "Lesotho", "Liberia", "Libya", "Liechtenstein", "Lithuania",
"Luxembourg", "Madagascar", "Malawi", "Malaysia", "Maldives", "Mali",
"Malta", "Marshall Islands", "Mauritania", "Mauritius", "Mexico", "Micronesia",
"Moldova", "Monaco", "Mongolia", "Montenegro", "Morocco", "Mozambique",
"Myanmar (formerly Burma)", "Namibia", "Nauru", "Nepal", "Netherlands",
"New Zealand", "Nicaragua", "Niger", "Nigeria", "North Korea",
"North Macedonia", "Norway", "Oman", "Pakistan", "Palau", "Palestine State",
"Panama", "Papua New Guinea", "Paraguay", "Peru", "Philippines", "Poland",
"Portugal", "Qatar", "Romania", "Russia", "Rwanda", "Saint Kitts and Nevis",
"Saint Lucia", "Saint Vincent and the Grenadines", "Samoa", "San Marino",
"Sao Tome and Principe", "Saudi Arabia", "Senegal", "Serbia", "Seychelles",
"Sierra Leone", "Singapore", "Slovakia", "Slovenia", "Solomon Islands",
"Somalia", "South Africa", "South Korea", "South Sudan", "Spain",
"Sri Lanka", "Sudan", "Suriname", "Sweden", "Switzerland", "Syria",
"Taiwan", "Tajikistan", "Tanzania", "Thailand", "Timor-Leste", "Togo",
"Tonga", "Trinidad and Tobago", "Tunisia", "Turkey", "Turkmenistan",
"Tuvalu", "Uganda", "Ukraine", "United Arab Emirates", "United Kingdom",
"United States of America", "Uruguay", "Uzbekistan", "Vanuatu",
"Vatican City (Holy See)", "Venezuela", "Vietnam",
 "Yemen", "Zambia", "Zimbabwe"])
```

```
all_stopwords = ENGLISH_STOP_WORDS.union(custom_stopwords)
default_stopwords=set(STOPWORDS)
```

```
def remove_stopwords_custom(text):
    if isinstance(text,str):
        return ' '.join([word for word in text.split() if word not in all_stopwords])
    else:
        return ' '.join([word for word in str(text).split() if word not in
        ↪ all_stopwords])
        return text

def remove_stopwords_default(text):
    if isinstance(text,str):
        return ' '.join([word for word in text.split() if word not in
        ↪ default_stopwords])
    else: #make into string if not already
        return ' '.join([word for word in str(text).split() if word not in
        ↪ default_stopwords])
        return text
```

```
def remove_clutter_custom(text): #dependent on stopword removal fx
    text = text.apply(remove_stopwords_custom)
    text = text.str.replace(r'#', '') #Remove instances of '#'
    text = text.str.replace(r'@', '') #Remove instances of '@'
    text = text.str.replace(r'-', '') #Remove instances of '-'
    text = text.str.replace(r':', '') #Remove instances of ':'
    text = text.str.replace(r';', '') #Remove instances of ';'
    text = text.str.replace(r'!', '') #Remove instances of '!'
    text = text.str.replace(r'.', '') #Remove instances of '.'
    text = text.str.replace(r'?', '') #Remove instances of '?'
    text = text.str.replace(r',', '') #Remove instances of ','
    text = text.str.replace(r'/', '') #Remove instances of '/'
    text = text.str.replace(r' \ ', '') #Remove instances of '\'
    text = text.str.replace(r'>', '') #Remove instances of '>'
    text = text.str.replace(r'<', '') #Remove instances of '<'
    text = text.str.replace(r'^', '') #Remove instances of '^'
    text = text.str.replace(r'*', '') #Remove instances of '*'
    text = text.str.replace(r'_', '') #Remove instances of '_'
    text = text.str.replace(r' " ', '') #Remove instances of ' " '
    text = text.str.replace(r'~', '') #Remove instances of '~'
    text = text.str.replace(r'[^\x00-\x7F]+', '', regex=True) #rm emojis
    text = text.str.replace(r'\s+', ' ').str.strip() #Further clean up
    text = text.apply(remove_stopwords_custom)
    return text
```

```

def remove_clutter_default(text): #dependent on stopword removal fx
    text = text.apply(remove_stopwords_default)
    text = text.str.replace(r'#', '') #Remove instances of '#'
    text = text.str.replace(r'@', '') #Remove instances of '@'
    text = text.str.replace(r'-', '') #Remove instances of '-'
    text = text.str.replace(r':', '') #Remove instances of ':'
    text = text.str.replace(r';', '') #Remove instances of ';'
    text = text.str.replace(r'!', '') #Remove instances of '!'
    text = text.str.replace(r'.', '') #Remove instances of '.'
    text = text.str.replace(r'?', '') #Remove instances of '?'
    text = text.str.replace(r',', '') #Remove instances of ','
    text = text.str.replace(r'/', '') #Remove instances of '/'
    text = text.str.replace(r' \ ', '') #Remove instances of '\'
    text = text.str.replace(r'>', '') #Remove instances of '>'
    text = text.str.replace(r'<', '') #Remove instances of '<'
    text = text.str.replace(r'^', '') #Remove instances of '^'
    text = text.str.replace(r'*', '') #Remove instances of '*'
    text = text.str.replace(r'_', '') #Remove instances of '_'
    text = text.str.replace(r' " ', '') #Remove instances of ' " '
    text = text.str.replace(r'~', '') #Remove instances of '~'
    text = text.str.replace(r'[^\x00-\x7F]+', '', regex=True) #rm emojis
    text = text.str.replace(r'\s+', ' ').str.strip() #Further clean up
    text = text.apply(remove_stopwords_default)
    return text
```

Applying the remove_clutter_custom() function

```
train_data_english['tweet'] = remove_clutter_custom(train_data_english['tweet'])
validation_set = remove_clutter_custom(validation_set)
validation_set.to_csv(path + '/validation_set_cleaned_May1624', index=False,
 ↪  header=False)
```

Function for formatting text for fastText model. Adds '**label**' to each line of text in text containing column.

```
def add_label(row):
    return f"__label__{row['label']} {row['tweet']}"
```

Adding labels to each row in training set, then sending the labeled data off to a separate csv.

```
train_data_english['labeled_text'] = train_data_english.apply(add_label, axis=1)
```

```
train_data_english['labeled_text'].to_csv('/Users/alexz/Programming/Python
 ↪  Projects/Sentiment Analysis of COVID-19 Misinformation with fastText/Sentiment-
 ↪  Analysis-of-COVID-19-Misinformation-with-fastText/train_data_labeled_May1624',
 ↪  index=False, header=False)

labeled_text = ('/Users/alexz/Programming/Python Projects/Sentiment Analysis of
 ↪  COVID-19 Misinformation with fastText/Sentiment-Analysis-of-COVID-19-
 ↪  Misinformation-with-fastText/train_data_labeled_May1624')
```

Model Training

```
# Train the fastText model
model = fasttext.train_supervised(input='/Users/alexz/Programming/Python
 ↪  Projects/Sentiment Analysis of COVID-19 Misinformation with fastText/Sentiment-
 ↪  Analysis-of-COVID-19-Misinformation-with-fastText/train_data_labeled_May1624',
 ↪  autotuneValidationFile=(path + '/validation_set_cleaned_May1624'),
 ↪  autotuneDuration=20, label= '__label__')
```

```
Progress:  2.5% Trials:   3 Best score:     nan ETA:  0h 0m19sProgress:  5.1% Trials:   3 Best score:
Training again with best arguments
Read 0M words
Number of words:  11199
Number of labels: 2
Progress: 100.2% words/sec/thread:  235325 lr: -0.000161 avg.loss:  0.581356 ETA:  0h 0m 0sProgress: 100.0
```

Model Evaluation

```
train_data_english['tweet'].to_csv('/Users/alexz/Programming/Python
 ↪  Projects/Sentiment Analysis of COVID-19 Misinformation with fastText/Sentiment-
 ↪  Analysis-of-COVID-19-Misinformation-with-fastText/tweet_contents.txt',
 ↪  index=False, header=False)
tweet_contents = ('/Users/alexz/Programming/Python Projects/Sentiment Analysis of
 ↪  COVID-19 Misinformation with fastText/Sentiment-Analysis-of-COVID-19-
 ↪  Misinformation-with-fastText/tweet_contents.txt')
```

```
tweet_contents=pd.read_csv(tweet_contents, header=None)
```

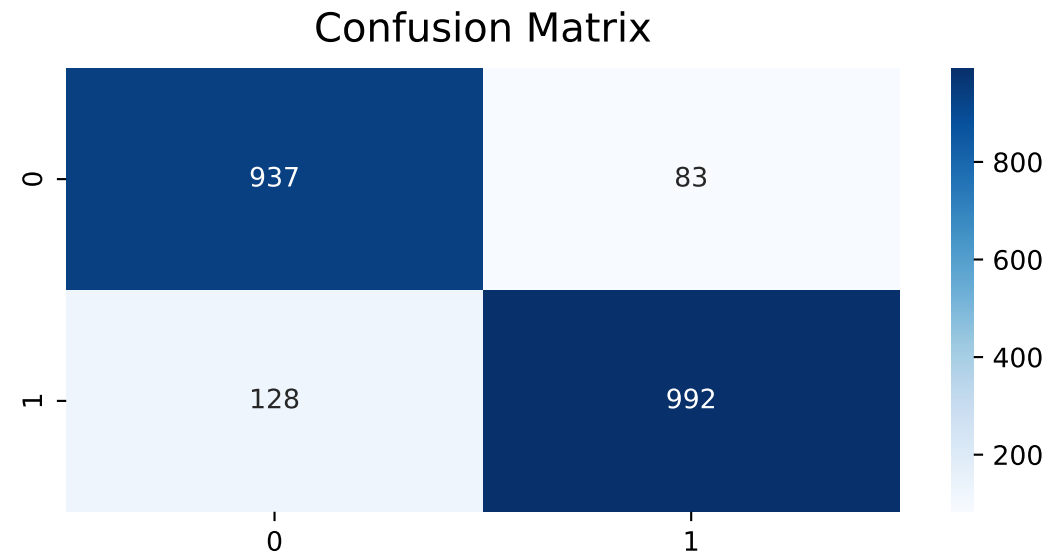
For loops for the creation of predicted and actual label lists to be used in metrics assessment.

```
predicted_labels = []
for row in tweet_contents[0]:
    prediction = model.predict(row)
    label = prediction[0][0]
    label = label.replace('__label__', '')
    predicted_labels.append(label)

actual_labels = []
for label in train_data_english['label']:
    label = label.replace('__label__', ' ')
    actual_labels.append(label)
```

Plotting a confusion matrix, and routing test/train labels into *sklearn* classification report.

```
conf_matrix = confusion_matrix(actual_labels, predicted_labels)
# Create confusion matrix
plt.figure(figsize=(7,3))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix', fontsize=15, pad=10, loc='center')
plt.show()
print(classification_report(actual_labels, predicted_labels))
```



	precision	recall	f1-score	support
fake	0.88	0.92	0.90	1020
real	0.92	0.89	0.90	1120
accuracy			0.90	2140
macro avg	0.90	0.90	0.90	2140
weighted avg	0.90	0.90	0.90	2140

Label Prediction Demonstration

```
def prediction_wizard(input_text=None):
    if input_text is not None:
        prediction = model.predict(input_text)
        label = prediction[0][0]
        label = label.replace('__label__', '')
        print(label)
    else:
        input_text = input('Enter the text you would like to predict: ')
        prediction = model.predict(input_text)
        label = prediction[0][0]
        label = label.replace('__label__', '')
        print(label)
```

```
#From random tweet on Twitter:
 ↪  https://twitter.com/PoisonDeathShot/status/1789495792039870702
prediction_wizard('2021 - Rosa Koire, Who Warned About the Global “Agenda 21” Back in
 ↪  2012, Describes the Plan & Where We’re Headed.  It Should Scare the Shit Out of
 ↪  You. (RIP)')
```

fake

```
#From https://www.cdc.gov/vaccines/covid-19/clinical-considerations/interim-
 ↪  considerations-us.html
prediction_wizard('Healthcare providers who administer the Moderna COVID-19 Vaccine
 ↪  (2023-2024 Formula) to individuals ages 6 months through 11 years should ensure
 ↪  the correct volume of the vaccine (0.25 mL) is withdrawn from the vial and
 ↪  administered to the recipient. Discard vial and excess volume after extracting a
 ↪  single dose.')
```

real

```
#Taken from https://www.cdc.gov/vaccines/covid-19/info-by-product/index.html
prediction_wizard('Janssen COVID-19 Vaccine is no longer available in the U.S. All
 ↪  remaining U.S. government stock of Janssen COVID-19 Vaccine expired May 7, 2023.
 ↪  Dispose of any remaining Janssen COVID-19 Vaccine in accordance with local, state,
 ↪  and federal regulations.People ages 18 years and older who received 1 dose of
 ↪  Janssen COVID-19 Vaccine should be considered to have received a single-dose
 ↪  Janssen primary series.People ages 18 years and older who received 1 or 2 Janssen
 ↪  COVID-19 Vaccine dose are recommended to receive 1 bivalent mRNA dose (Moderna or
 ↪  Pfizer-BioNTech) at least 2 months after completion of the previous dose.')
```

real

```
#Taken from video summary: https://www.bitchute.com/video/SyfZYzVU1U7s/ ; 'COVID
 ↪  VACCINES HAVE HIGHEST ‘KILL RATE’ IN MEDICAL HISTORY – MEDIA BLACKOUT'
prediction_wizard('Covid mRNA vaccines are now officially the deadliest drugs in the
 ↪  history of Western medicine, killing and injuring hundreds of millions of people
 ↪  around the world as the fallout from the mass roll out continues to snowball. Big
 ↪  Pharma and the global elite have blood on their hands and they are using mainstream
 ↪  media to whitewash and cover up the greatest crime in history.World-renowned OBGYN
 ↪  physician Dr. James Thorp has blown the whistle on the massive cover-up, warning
 ↪  the public about the disturbing numbers that governments, Big Pharma and the
 ↪  mainstream media are working overtime to keep hidden from the public.')
```

```
fake
```