Generating a Cybersecurity Thesaurus Based On Tweets

Vincent Fiore, Sukhjinder Nahal, Dmitry Matyunin, Emma Padilla, Jaikishin Satpal, and Andreea Cotoranu
Seidenberg School of CSIS. Pace University, Pleasantville, New York

***Abstract* -- With millions of live thoughts being tweeted on a daily basis, how could such a large amount of text data be analyzed and researched? By creating a thesaurus which contains a set of collected list of words that can be used to index and search such large data sets. One important step for analyzing such data is to perform pre-processing, which is used to clean up the noise, or irrelevant data, from large data sets. This is accomplished by []. For this study, thousands of cyber security related tweets were pulled using the Python programming language for the purpose of creating a cyber security thesaurus. The pulled data was analyzed, correlated, and processed for relevant words. As a result, a thesaurus of cyber security concepts from twitter data was created.**

***Index terms* -- Tokenization, Stop Words, Lemmatization, Cyber Security, Classification, Twitter**

I. INTRODUCTION

Analytics based on Twitter data has gained relevance over the past few years, largely due to the social media platform allowing a wide variety of users to quickly send out short messages, or tweets. This allows users to quickly jot down and share their thoughts, ideas, and even news stories they hear. This ease of communication may explain why there are millions of users, and amongst those users, are cybersecurity professionals and enthusiasts who are constantly posting cybersecurity related tweets, such as breaking news of the latest zero-day or malicious code found from an independent research. These users' tweets can be analyzed for research purposes, such as creating a cybersecurity based thesaurus.

As the field of cybersecurity offers a wide range of topics, specific Twitter accounts can be narrowed down based on their relevance to the subject. Due to the nature of our research resulting in a large variety of the words, such narrowing process will be needed in order to remove irrelevant words. This process will be later discussed.

Twitter benefits this particular research because many of the terms in the field of cybersecurity are fluid. Terms and definitions are frequently changing, making it difficult to narrow down exactly what we are looking for. One way to resolve this issue is to create algorithms that find related terms based on Twitter data. This allows us to create a data set that can be easily updated and will stay relevant even as the terms themselves change.

The aim of this study is to build a thesaurus of cybersecurity concepts based on Twitter data. This will be done by analyzing tweets to generate such a thesaurus, including both synonyms and related words. The final results will produce a searchable CSV file that will contain a number of relevant words and their compatible findings.

The results of this data will provide future researchers a comprehensive and reliable cybersecurity thesaurus. This will be significant in fields of text analytics that may focus on cybersecurity. For example, terms can be grouped together based on synonyms, which can then be used to track the prevalence of certain ideas or terms over time. Continuing research on Twitter will reap more benefits as this data will already be built on a dataset that they know is relevant.

II. LITERATURE REVIEW

Previous research has been conducted on the creation of a thesaurus based on text corpora through machine learning. One particular research comes from Ionian University, where semantic thesaurus was created. The first approach to building this thesaurus was preprocessing the data, which is an important step when a analyzing such data. Other important steps include tokenization, basic morphological tagging, removal of stop words, and removal of data that is exclusive to tweets but may not actually be relevant.

Data in the real world is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. There are several data preprocessing techniques such as data cleaning, data integration and data reduction. Data cleaning removes noise and correct inconsistencies in data. Data integration merges data from multiple

sources into a coherent data store such as a data warehouse. Data reduction reduces data size by, for instance, aggregating, eliminating redundant features, or clustering. [13].

Kermanidis, author of the research from Ionian University, goes on to describe the process of building a semantic thesaurus. Ontologies are defined as hierarchical structures of domain concepts that are enriched with semantic relations linking the concepts together.

In another study that consisted of sentiment analysis where different models were of data collection was used to compared on Twitter data. Such models included Unigram, and Feature Tree. While the study found that the Feature Tree models outperformed Unigram, it was the way the data was collected that was interesting. The process began with the collection of manually annotated Twitter data that was then used to experiment against a random sample of streaming data. The advantage of this approach was that tweets were collected in a streaming fashion that represented a true sample of actual tweets of language and content. Such twitter data included the use of emoticons that were collected for this study. A manual annotated dictionary of emoticons was then created, where the emoticons were then mapped to their polarity. An acronym dictionary that consisted of English translations of over 5,000 frequently used acronyms was downloaded and used. 11,875 manually annotated Twitter data (tweets) were collected from a commercial source that archived real-time streaming data. Though there was no restriction of language or location, Google translator was used to convert the tweets to English prior to the annotation. Each tweet was then labeled as positive, negative, neutral or junk, with junk meaning the tweet could not be understood by the annotator or was not properly translated. After eliminating the "junk" tweets, the data sample was reduced to 8,753 tweets [10].

An emoticon dictionary was prepared by labeling 170 emotions from Wikipedia with their emotional state. Then each emoticon was labeled with extremely positive, positive, neutral, negative and extremely negative. An acronym dictionary was also used, where acronyms such as "lol" were translated to "laughing out loud". Each tweet was then pre-processed by various rules such as replacing all the emoticons with their sentiment polarity via the dictionary, replacing all URLs and Twitter targets, such as @John, with tags, such as ||U||. Slang such as "cooool" was changed to "cool". The purpose of this slight alteration was so the researcher can determine the term is slang but still be able to analyze it and conduct Boolean searches. Statics of different subjects of the data such as number of stop words, English words, Twitter tags and so on were also taken [10].

In another study, two preprocessing methods were used to utilize formal concept analysis that were then presented. One method extended the set of attributes that described objects input data table by the new attributes. The second of replaces said attributes with new attributes. Both methods include the new attributes as being defined by certain formal concepts that were computed from an input data table. Selected formal concepts were obtained by boolean factor analysis described by Formal Concept Analysis (FCA), which is a method often used for data preprocessing before the data is processed by another data machine learning method. A decision tree was also used, which is the most common method in data mining and machine learning. It can take a finite number of values and assign a class label, often depicted by a table [13].

III. DEFINITION OF A THESAURUS

When most users think of the word thesaurus, a word list of synonyms is the first concept that comes in mind. But in fact, the words found in a thesaurus are not always synonyms of an original word. A thesaurus can be used to provide words that are connected with other words where the same idea might be most effectively expressed by a different word. This was the mindset when such statement was kept in mind when such that was based on Twitter data was created.

The most popular approach to creating a thesaurus is the "Top-Down" approach, where the actual phrases that appear in text are used as a key to organizing such material. The index and thesaurus are built out from the text, but are not added on. Due to this approach, not all the words are properly represented. For instance, concepts, which are dependent on description, are named and placed within the thesaurus as part of its maintenance [10].

The "Bottom-up" approach, on the other hand, allows one to build an ACP thesaurus by allowing every identifiable phrase that occurs, and is identified in the text, to be tracked. The historical information about each phrase allows automated methods to reduce the amount of human effort involved in such indexing efforts. This approach demonstrates a semi-automatic method of building a thesaurus from phrases occurring in text [10].

The American College of Physicians attempted to produce an index of their medical publications from a "Bottom-Up" approach, where nominal phrases were extracted from the text of a material with a long word list, using data from Unified Medical language Systems (UMLS) Metathesaurus along with current lexical and linguistic processing tools is feasible. Nominal phrases were then normalized to match the UMLS Metathesaurus and combined to create concept classifications. Nominal

phrases that did not match the Metathesaurus were treated as uncategorized terms, and were later reviewed and merged into existing or newly created concept classes. All of the concepts were then grouped together in larger descriptor classes, which provided the basis for the index. These descriptor classes and the hierarchical arrangements in which the concepts occurred in the UMLS served as the basis of the ACP thesaurus [10].

IV. DATA MINING TOOLS AND TECHNIQUES

Data mining involves utilizing different techniques to discover patterns from a large datasets. One of the related areas in data mining is text mining, which is the process of discovering high quality information from text documents. High quality is a term that refers to some combination of relevance, novelty and interestingness. Text documents contain data from both structured and unstructured data. Structured data is data that resides in a fixed field within a record or file. This data can be found in databases and spreadsheets, while unstructured data is the opposite of structure data. Semi-structured data is the data that is neither raw nor typed in a conventional database system.

Text mining tries to solve the issues that occur in the areas of data mining, machine learning processing, information retrieval, knowledge management, and classification. It is a technique that extracts information from both structured and unstructured data and finds patterns. Some applications of text mining include information retrieval, information extraction, categorization, and natural language processing [15].

Information extraction is a method that identifies keywords and relationship within text. This is useful when dealing with large volumes of data because predefined sequences are being searched. Relations between people, identified places and time are inferred to give the user meaningful information, as opposed to traditional data, where data mining assumes that the data that is being mined for is already in the form of a relational database [15].

Categorization identifies the main themes of a document by inserting the document into a predefined set of topics. The document is treated as a bag of words, where categorization counts words as they appear from the bag and identifies the main topic of the document, rather than processing the actual information, as compared to information extraction. In short, categorization relies on a glossary for predefined topics, and relationships are identified by looking for synonyms, narrower, related, and large terms [15].

The first step in text mining is data cleaning. Data cleaning is the process through which we remove unwanted words and characters from our text corpus. This step is important because the nature of tweets imbues much irrelevant data in each post. This process includes extraction, tokenization, stop-word removal and lemmatization [13][15].

VI. METHODOLOGY

1. Data Set

In order to begin work on the thesaurus, data from Twitter was needed to be collected into one easily accessible file. Although the Twitter Application Programming Interface (API) could have been used to automatically pull relevant tweets at runtime, it was determined that this process would have taken too long to be performed during each run. Additionally, as a major part of this research was processing the text, a stable corpus was required to ensure that the processing was correct. With a permanently downloaded file, this is much easier as the results of each attempt can be directly compared.

The first step in this process began with the identification of relevant Twitter users. The word ‘malware’ was entered into the Twitter search engine, which allowed us to view related tweets and Twitter accounts, to select, or “follow”, the desired Twitter accounts for this research. Accounts that were selected for this research had to be dedicated to cybersecurity and were up to date on the latest cybersecurity news and research. This ensured that the data collected were from those that were passionate and knowledgeable on the topic. A total of 20 relevant accounts were used for this research.

After finding the appropriate Twitter accounts, a python library called Tweepy was used to download user tweets into a CSV file. After this process was completed, the CSVs were then manually combined into one final file with over 42,000 tweets.

2. Data Pre-Processing

Once the corpus was assembled, the data preprocessing was ready to begin. As mentioned earlier, it is necessary to remove irrelevant for all text analytics research. When it comes to Twitter data, however, the need is even greater due to the various symbols, unicode, and other characters used. For example, the syntax used for Twitter replies, “@,” for example, must be automatically removed in addition to the user referenced.

Python was the preferred language for this research due to its multitude of data libraries available, such as Pandas and the Natural Language Toolkit (NLTK). Such libraries are ideal for this type of work, [14] as the act of removing stop words can be very

tedious. To remove the stopwords, NLTK contains a module that has a list of stopwords stored in 16 different languages; English was chosen for this scenario [5].

Stop words are words that include prepositions and pronouns that do not give meaning to a document, such as "the, in, a, an, with" and so on. Because stopwords can take up to 40-50% of the raw data, they are often removed from documents as they're not measured as keywords in text mining applications. This also reduces the dimensionality of term space. While there are different ways to measure stopwords, one common way the classical method, where the removal is based on a pre-compiled list [14],[16].

Once Pandas and NLTK libraries were utilized, links, numbers, Twitter usernames, "rt", "via" and "&" were removed. Web links that were automatically obfuscated by the Twitter API were also removed. Unicode characters also were stripped from the tweets in addition to numbers and single letters. Removing unicode in particular is an important step as Python does not correctly support emoji or special characters. This was done by the built-in method in NLTK that directly removes all non-HTML entities from the tweets.

The final step of data preprocessing was lemmatization. This process intelligently removed many common word suffixes and combined words within context. For instance, words with similar meanings, such as "well," "best," and "better," can all be lemmatized to simply "good." This kept meaning simple and allowed for our next step of calculating word frequencies. This process is crucial for our research as it helps to concatenate words based on their meaning and remove words that may not be exact duplicates, but have identical meaning in context [13].

3. Tokenization

Once the preprocessing stage was completed, the next step was to split the data into individual words so that we can perform operations on each words separately. This process is known as tokenization and it works by separating words using space and punctuation. The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging. A part-of-speech tagger processes a sequence of words and attaches a part of speech tag to each word in the form of a tuple.

Once the part of speech for the word was determined, it was then compared to another list of other recognized parts of speeches that were similar to the final data set. Nouns and adjectives were heavily focused on. As a result, of both data preprocessing and tokenization, the final data set was greatly narrowed down and ready to be analyzed.

4. Data Analysis

Once this final list was complete, the frequency of words that appeared in the final corpus was reviewed and stored with its corresponding word. This held every finalized word in the Twitter data along with its frequency within the the corpus. This list was then sorted by frequency before being printed to the researcher.

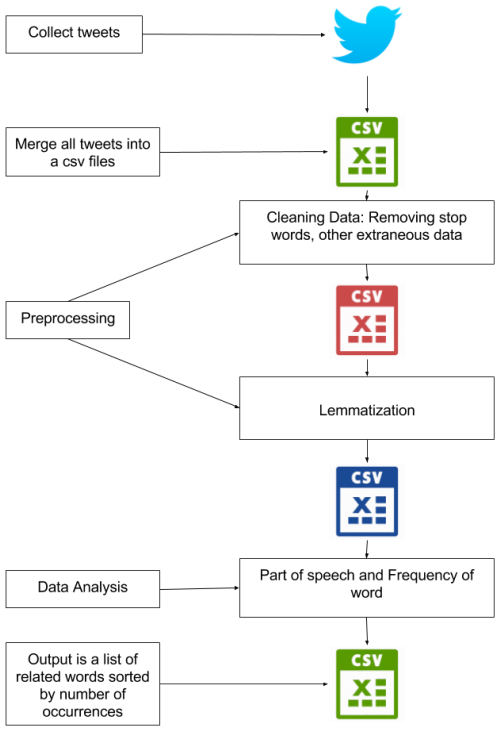

Figure 1: Workflow of the research

This step produced a list of a few hundred individual words, which could then be manually examined by a researcher. By further removing irrelevant words, the words most relevant to the original search term began to appear. This manual searching was necessary as there words that still appeared very frequently, but could not be picked up by the original preprocessing steps. For example, words such as "via" and "RT," which are both used in the Twitter lexicon to denote retweets, appeared frequently in any search performed. This was a common occurrence as many users retweet a large number of tweets and Twitter does not always enforce a pattern for this behavior. Upon finding these sorts of patterns about which words appeared most frequently without contributing to the research at hand, we were then able to

go back and remove them before they were ever attached to the final data.

Determining the number of recurrences of certain words will also be an important step in this process in order to identify synonyms. One method that has been particularly common in previous research is the creation of word clouds. This method creates "clouds" of words that are sorted by size to illustrate which occur the most frequently. In the case of searching for synonyms, this step may help when it comes to manually determining which words may be relevant for any given entry into the thesaurus. Furthermore, these word clouds can also sort words based on relevance to other words in the cloud. This can act as a logical sorting method for certain entries, which can help to narrow down which words may or may not end up being relevant for later use [16].

For an additional visualization of the final data, the word list was fed into a word cloud generator. Word clouds take words and their frequencies and plot them randomly in an image. The size of each word directly correlates to the frequency with which they appear, with more frequent words appearing largest and least frequent words smallest. This was done to provide the user with a visual representation of some of our findings and to make this process more understandable from the point of view of a lay person. Specifically, by showing a visual representation of the frequency, it's clear how we selected our final list of words for the thesaurus.

Figure 2: The word cloud produced for "malware"

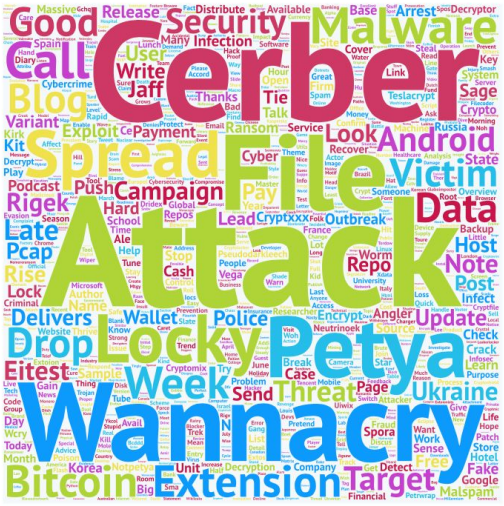

Figure 3: The word cloud produced for "ransomware"

V. HYPOTHESIS

The primary assumption that our research is built on is that this Twitter data will contain enough relevant information to build a cybersecurity thesaurus. With this considered, it is believed that we can find relevant information for almost any term that relates to the field. Based on the overwhelming size of the corpus that we are analyzing, trends should present themselves based on the analysis of word frequencies.

We believe that analyzing these frequencies is especially valuable due to the short form nature of Twitter. In 140 characters, it becomes unlikely that a user can write more than one or two sentences at a time. This means that any Tweet that contains the word we are searching for will most likely also contain words related to it. For example, if we are searching for the word "malware," it is unlikely that a user's tweets will deviate from that subject greatly.

This research will be particularly valuable for further research in the cybersecurity field. Different forms of text analytics and machine learning require primer words than can be searched for to discover trends. While this type of searching can be straightforward if the subject is relatively simple, cybersecurity poses a particularly interesting issue in this regard. Since the field is so frequently changing, searching for related terms for even the most basic topics can be wildly different depending on when the searches are carried out. [13] For example, when looking for data about recent malware campaigns, using the names of malware from even a few months prior can be detrimental. In these cases, using words that may be completely outdated can severely limit the research that is performed.

Twitter allows for data that is constantly updating and easily flows with the common discourse on the subject matter. For example, by searching through tweets over time, it can be possible to specifically identify which types of malware are most common during any given time period. Using real cybersecurity professionals as a basis for the data we collected also keeps the final product relevant. This allows for further researchers to be querying data that is always both relevant to their current work and is based upon real world usage. By selecting users that are known to be professionals in this subject, we can also ensure that the data is not tainted by those who are not adding to the common discussion on the subject.

VI. RESULTS

1. *Initial Results*

Our initial results were based on the word "malware." This word was proven to be an excellent starting point and allowed for very focused relevant words to be produced. Many words come into light that were specifically related to the subject in the exact way we had hoped. Specifically, we found a variety of words that were related to the subject of malware, such as types of malware, antiviruses, words related to hacking, and countries that have a reputation of producing some of the most popular types of malware.

Much of this data comes as predicted, and proves that this model works correctly in identifying relevant words. This also reassured that we selected the right group of Twitter accounts to follow for this research. In this data, one surprise that came to light was the inclusion of the names of countries that have been suspected of producing malware [15]. Russia, in particular, appeared more than 50 times. This is most likely because of the recent ransomware attacks that many believe have Russian origins. [11]

malware	***Android, PCAP, malspam, target, bank, hacker, rigged, spam, email, campaign, exploit, ransomware, elitist, infect, russian, angler, Thesas, CCleaner, Cisco, Wannacry, Dridex, Kaspersky, Gootkit, government, Kronos***

Table 1: Sample of the relevant thesaurus entry for the word "malware"

ransomware	***Attack, Cerber, Wannacry, Petya, Locky, spread, extension, malware, Bitcoin, Security, Android, campaign, victim, target, threat, data, Rigek, Update, Ukraine, Wallet, EITest, exploit, police, Cryptxxx, payment***

Table 2: Sample of the relevant thesaurus entry for the word "ransomware"

2. *Further Results*

We also tested our program on a number of other cybersecurity related terms such as: "ransomware," "encryption," "DDoS," "phishing," "vulnerability," "hacker," and "backdoor." Each of these words were processed the same way the original "malware," and our program worked without any additional changes needed. For these terms, we discovered results that were in line with the prior term. The words produced related words that could be traced back to specific incidents and phrases that made sense in their own context. Furthermore, these results also appear to be useful for the same research purposes that we are trying to achieve.

Specifically, when looking at some of the individual results, we are able to extract meaning from the related words. One of the most interesting results came from our analysis of the word "ransomware." This word was specifically chosen due to its relation to the original "malware" and because we knew that it would be straightforward to confirm the relationship of words in the results. We specifically were hoping to see the names of some major ransomware attacks in our findings, and were pleased to see that six major ransomware attacks were spotted in our list of 25 related words.

VII. CONCLUSION

Overall, we were very pleased with the results for all of the tested words. We believe that the trends shown from this data prove that it is useful for further research and provided accurate thesaurus creation. Specifically, the results that showed the names of malware attacks prove that this program is particularly accurate and

useful.

When looking at the results produced by "ransomware," the trends shown are assuring for further research. As mentioned earlier, the results showing the names of ransomware attacks are valuable for research in the cybersecurity field. Furthermore, this word also showed the results that did not specifically name ransomware attacks. The program was able to identify words like "Bitcoin," "wallet," and "payment." These are clear references to the actual details of the inner workings of ransomware attacks, which help to further flesh out the type of information a researcher might want to consider when looking for this specific term. These results are difficult to manually find and would require any researcher to spend a considerable amount of time learning before they could come to these same sort of results on their own.

In terms of the time relevance of the data, we also believe that the program was a clear success. Again, citing the ransomware results, we see obvious trends in the attacks that were found by the software. All of these attacks were performed relatively recently, and show that the data can continually be updated as the common lexicon of these terms expands. In fact, all a user would need to quickly update these terms for the newest results would be to download new tweets for Twitter's API and place it in a CSV file for the program. This allows for any user of the program to continually update their own data with minimal effort. For research that needs to stay up to date, we felt that making it as simple as possible to stay connected to the real-time conversation was important. In this respect, we feel that we were able to succeed.

Our results for the word "encryption" also proved to be interesting and differed greatly from some of our other search terms. For this word, many of the results did not have any sort of attack or specific point to name. Instead, we found that users were mostly concerned with the companies and software that they used and their implementations of encryption. Words like "iMessage" and "WhatsApp" came to the top of the frequency list, as these pieces of software are widely considered to be at the forefront of the encryption debate for many people. Similarly, words like "government" and mentions of Great Britain's version of the NSA were near the top of this same list. This helps to illustrate the concern users have in regards to encryption and the varying opinions on the matter.

These different type of words, that focus on less technical details and instead seemed to be mentioned in natural conversations, help to illustrate the flexibility of our program. It was clearly able to identify words regardless of their meaning and instead found words that were truly related to the original search term. This helps to show that we would have no issue in the future even if further researchers test words that we cannot currently predict and may not currently be apart of the cybersecurity lexicon.

REFERENCES

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Pasonneau, "Sentiment analysis of Twitter data," Columbia University, June. 2011.

[2] A. Ali, "Ransomware: A Research and a Personal Case Study of Dealing with this Nasty Malware," Issues in Informing Science and Information Technology Education, 14, 87-99, Mar. 2017.

[3] V. Balakrishnan, E.L. Yemoh, "Stemming and Lemmatization: A Comparison of Retrieval Performances," Lecture Notes on Software Engineering, Aug. 2017.

[4] V.P. Baradadi, A. Mugabuskaka "Corpus Specific Stop Words to Improve the Textual Analysis in Scientometrics," European Research Council Executive Agency, Jun. 2015.

[5] S. Bird and E. Loper, "NLTK: The natural language toolkit," Proc. of 42nd Annual Meeting of the Association for Computational Linguistics, 2004.

[6] K.L. Kermanidis, "Learning to Build a Semantic Thesaurus from Free Text Corpora without External Help," Intech, pp. 145-186, Jan 2009

[7] W. McKinney, 'Python for Data Analysis', Sebastopol: O'Reilly, 2014.

[8] A Mollett, D.M., Patrick Dunleavy, "Using Twitter in university research; Teaching and impact activities," pp. 1-11, 2011.

[9] S. Moon, H. Park, C. Lee, and H. Kwak, "What is Twitter, a Social Network or a News Media?", pp. 1-10, 2010.

[10] S.J Nelson, T.Khum, D. Radzinski "Creating a Thesaurus from Text: A Bottom Up Approach to Organizing Medical Knowledge," The American College of Physicians, Jun 1998

[11] D. O'Brien, "ISTR Ransomware 2017," Internet Security Threat Report, Jul. 2017.

[12] J. Outrata, "Preprocessing input data for machine learning by FCA," Palacky University, Oct. 2010

[13] J. Plisson, Nada Lavrac, and Dunja Mladenic. "A Rule based Approach to Word Lemmatization," Proceedings of the 7th International multi-conference Information Society IS-2004, Ljubljana: Institut "Jožef Stefan", pp. 83-86, 2004.

[14] A. Schofield, Mans Magnusson, David Mimno "Pulling Out the Stops: Rethinking Stopword Removal for Topic Models," Proceedings of EACL, 2017.

[15] S. Vijayarani, M.Nithya, J. Ilamath "Preprocessing Techniques for Text Mining - An Overview," International Journal of Computer Science & Communication Networks, Feb 2015

[16] Y. Wu, T. Provan, F. Wei, S. Liu and K. Ma, "Semantic-Preserving Word Clouds by Seam Carving," IEEE Symposium on Visualization vol. 3, June 2011.

APPENDIX

```
import pandas as pd
import operator
import re
from nltk.corpus import stopwords
from nltk import word_tokenize, pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize.casual import _replace_html_entities

import csv

df = pd.read_csv('merged.csv', encoding = "ISO-8859-1") #Reading the Twitter Corpus file

df.columns = ["A", "B", "C"]

freqDict = {}

#Start of helper functions.

#Part of speech finder
pos = lambda tokens: pos_tag(tokens)

#Lemmatizer
lemmatize = lambda posTokens: [processPosTagsAndLemmatize(*wordPos) for wordPos in posTokens]

#Returns lemmatization based on PoS
def processPosTagsAndLemmatize(word, pos):
    return lemma.lemmatize(word, treebankToWordnetPOS(pos))

#Replaces unicode
def unicodeReplacement(tweet):
    return _replace_html_entities(tweet)

#Helper function for PoS Tagging
def treebankToWordnetPOS(treebankPosTag):
    return {'J': wordnet.ADJ,
            'V': wordnet.VERB,
            'N': wordnet.NOUN,
            'R': wordnet.ADV}.get(treebankPosTag[0], wordnet.NOUN)

#Declares Lemmatizer
lemma = WordNetLemmatizer()

#End of helper functions

def dictionary(keyword):
    wordCount=0
    for each in df["C"]:
        if keyword in each.lower():
            wordCount = wordCount+1
            text = each.lower() #Makes each Tweet lowercase
            text = unicodeReplacement(text) #Removes unicode
            text = re.sub(r"http\S+", "", text) #Removes links
            text = re.sub(r'[0-9]+', '', text) #Removes numbers
            text = re.sub(r'@(\w)+', '', text) #Removes Twitter usernames
            text = re.sub(r'\W*\b\w{1,3}\b', '', text) #Removes single letters
            text = re.sub(r"rt", "", text) #Removes "rt"
            text = re.sub(r"via", "", text) #Removes "via"
            text = re.sub(r"&", "", text) #Removes "&"
            text = re.sub(r"icymi", "", text) #Removes "ICYMI"

            text = ' '.join([word for word in text.split() if word not in stopwords.words("english")]) #Removes stop words

            tokens = word_tokenize(text) #Tokenizes the tweets

            tagged = pos(tokens) #Grabs part of speech

            tagged = lemmatize(tagged) #Lemmatizes
            tagged = pos(tagged) #Grabs part of speech again because it is removed in lemmatization

            for word in tagged:
                if word[1] in ("NN","NNS","NNP","NNPS","JJ","JJR","JJS"): #Checks if the word is a noun or adjective
                    if word[0] not in freqDict: #If word is not already in the frequency list, add it
                        freqDict[word[0]] = 0
                    freqDict[word[0]] += 1 #Once word is in the frequency list, increase its frequency

    sorted_freqDict = sorted(freqDict.items(), key=operator.itemgetter(1)) #Sorts the dictionary by frequency
    sorted_freqDict.reverse() #Reverses the order

    print("\nWord Count = " + str(wordCount) + "\n") #Prints total frequency of search word
    #print(sorted_freqDict)

    for word in sorted_freqDict:
        print (word) #Prints each word and frequency


    #The following lines print the dictionary to a CSV file and are optional
    with open('%sWordCloud.csv' %keyword.lstrip(), 'w') as csv_file:
        writer = csv.writer(csv_file)
        for key, value in sorted_freqDict:
            writer.writerow([key, value])

#The lines to run the code
keyword = input("Enter keyword to be searched: \n")
dictionary(" " + keyword)
```