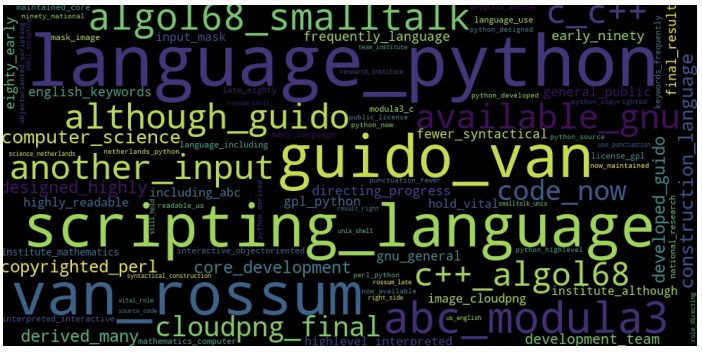
Bigram WordCloud for text visualization



WordCloud provides a means to perform high-level exploratory analysis of textual data. WordCloud is intended to highlight key terms and/or phrases in a visual representation. This is particularly useful for human viewers to filter and select interested documents to process. Pre-processing input text using stopwords, lemmatization followed by TF-IDF term frequency determination, length normalization leads to enhanced WordClouds. These techniques however might take single words(unigrams) out of context and lead to less effective WordClouds. One technique to overcome this is to use a combination of words - bigram and n-gram to derive WordClouds. This approach preserves the context of words and provides more meaningful WordClouds.

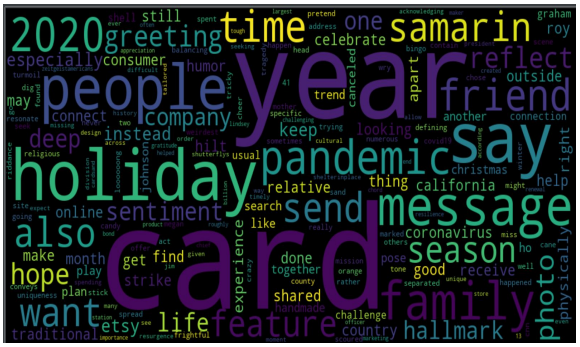
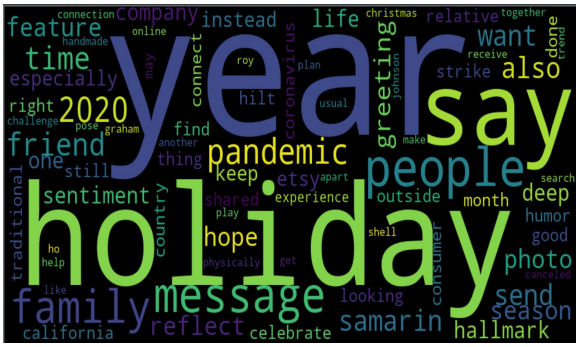
To analyze wordclouds in this effort we would be using news articles published by CNN at http://lite.cnn.com/en . Code for downloading and processing these articles would be available at https://github.com/rakesh-patnaik/tech_review. Instructions to execute would be available at https://github.com/rakesh-patnaik/tech_review/blob/master/README.md . This code downloads the latest news articles to generate WordCloud representations.

In order to compare efficiency of a bigram wordcloud we would need to perform a unigram wordcloud exercise to start with. Following are the steps the algorithm would execute to generate a unigram WordCloud.

- Lower case all words
- Tokenize the raw text string into a list of words where each entry contains a word
- Remove punctuation and other characters like @#$%^_&*, etc.
- Remove stop words
- Use lemmatization to remove closely redundant words
- Use the NLTK frequency distribution to determine the frequency of each unigram
- Fill a Python dictionary with each unigram and unigram frequency
- Render a word cloud

Of the two variations tried: most frequent words and least frequent words, in most cases the most frequent words seem to provide more accurate word cloud representation. For example if you consider the story “Your 'Seasons Greetings' cards this year won't be all holiday cheer” and the corresponding “Most Frequent” and “Least Frequent” representations. The word “Card” which the article is about is missing from the least frequent representation.

The one factor that could be leading to this is the lack of context around a word and hence the detachment to human perception.

Most Frequent Unigram WordCloud	Least Frequent Unigram WordCloud
	

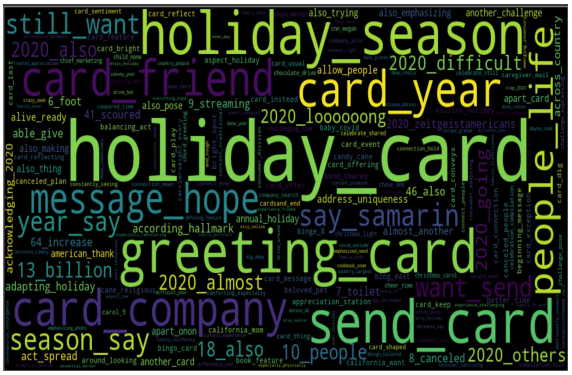
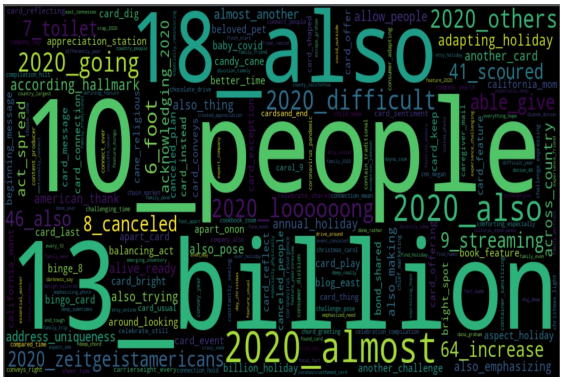
Moving on to bigram WordClouds, we would use two words to retain some perceived context around the words in the articles. The gist of operations we would perform are as follows:

- Lower case all words
- Tokenize the raw text string into a list of words where each entry is a word
- Call the NLTK collocations function to determine the most frequently occurring bigrams
- Print the top N frequently occurring bigrams to the screen
- Remove punctuation and other characters like @#$%^_&*, etc.
- Remove stop words
- Use lemmatization to consolidate closely redundant words
- Use the NLTK Bigram Collocation finder to determine the frequency of each bigram (explained below)
- Stuff a Python dictionary with the bigram and bigram measure raw frequency score
- Render a word cloud of bigrams

Collocations are words that occur together at some frequency. The wrodcloud rendering uses “Collocations = True” which determines if you’d like bigrams to appear in the word cloud along with unigrams. However, if the unigrams far outnumber the bigrams (which is the most likely case) when the word cloud reaches the “max_words” then you’ll only see the unigrams.

The word cloud uses a dictionary with key values that look like the following; A bigram from BigramCollocationFinder and a score from BigramAssocMeasures.rawfreq.
word_dict = {‘bigram1’: 0.000972355,
‘bigram 2’: 0.090127142}
Beyond simple term frequency you can insert any score you like for an n-gram into the dictionary and have this particular word cloud render n-grams per the score.
However, if the word cloud only receives words with the same frequency/score then it’ll vary the size of the words to make them fit rather than render them with the same size due to the same frequency/score.

A comparison of wordcloud representations again reveals that most frequent term provides more value than a least frequent bigram representation. This could be because of the size of the articles and smaller articles seem to provide less data to derive meaningful frequency distributions

Most Frequent Bigram WordCloud	Least Frequent Bigram WordCloud
	

Frequently occurring bigram WordCloud preserves context and for small document sizes provides the most valuable wordcloud. Least frequently occurring unigram or bigram wordcloud fail to represent the text as document sample sizes are smaller. One improvement is to look beyond simple commonly occurring bigrams and instead utilize 2 approaches to populate our word cloud: (a) a supervised approach to identify parts of speech such as an adjective, preposition and noun (b) an unsupervised approach that determines common, short, informative phrases.