Published in District Insights

District Data Labs Follow

Dec 29, 2017 · 10 min read · Listen

Beyond the Word Cloud

Visualizing Text with Python

By Lisa Combs and Prema Roman

In this article, we explore two extremely powerful ways to visualize text: word bubbles and word networks. These two visualizations are replacing word clouds as the defacto text visualization of choice because they are simple to create, understandable, and provide deep and valuable at-a-glance insights. In this post, we will examine how to construct these visualizations from a non-trivial corpus of news and blog RSS feeds. We begin by investigating the importance of text visualization. Next, we discuss our corpus and how to wrangle it. Finally, we will present word bubbles and word networks.

Why Do Text Visualization?

Edward Tufte examines "the unification of text and image" within *Centaur*, a ninth century manuscript depicting the constellation Centaurus filled with Latin text that elaborates on 24 of the stars depicted. The words within the image not only add information about the constellation, but also allow the reader to clearly see the centaur's image. This example shows the long history of text visualization and the importance of "continuity of showing and seeing." In this case, text is used as a powerful tool to inform the viewer of the "narrative metaphor grouping the stars into a constellation." Today, we see similar images created using large corpuses to outline images. (*Beautiful Evidence* p. 85)

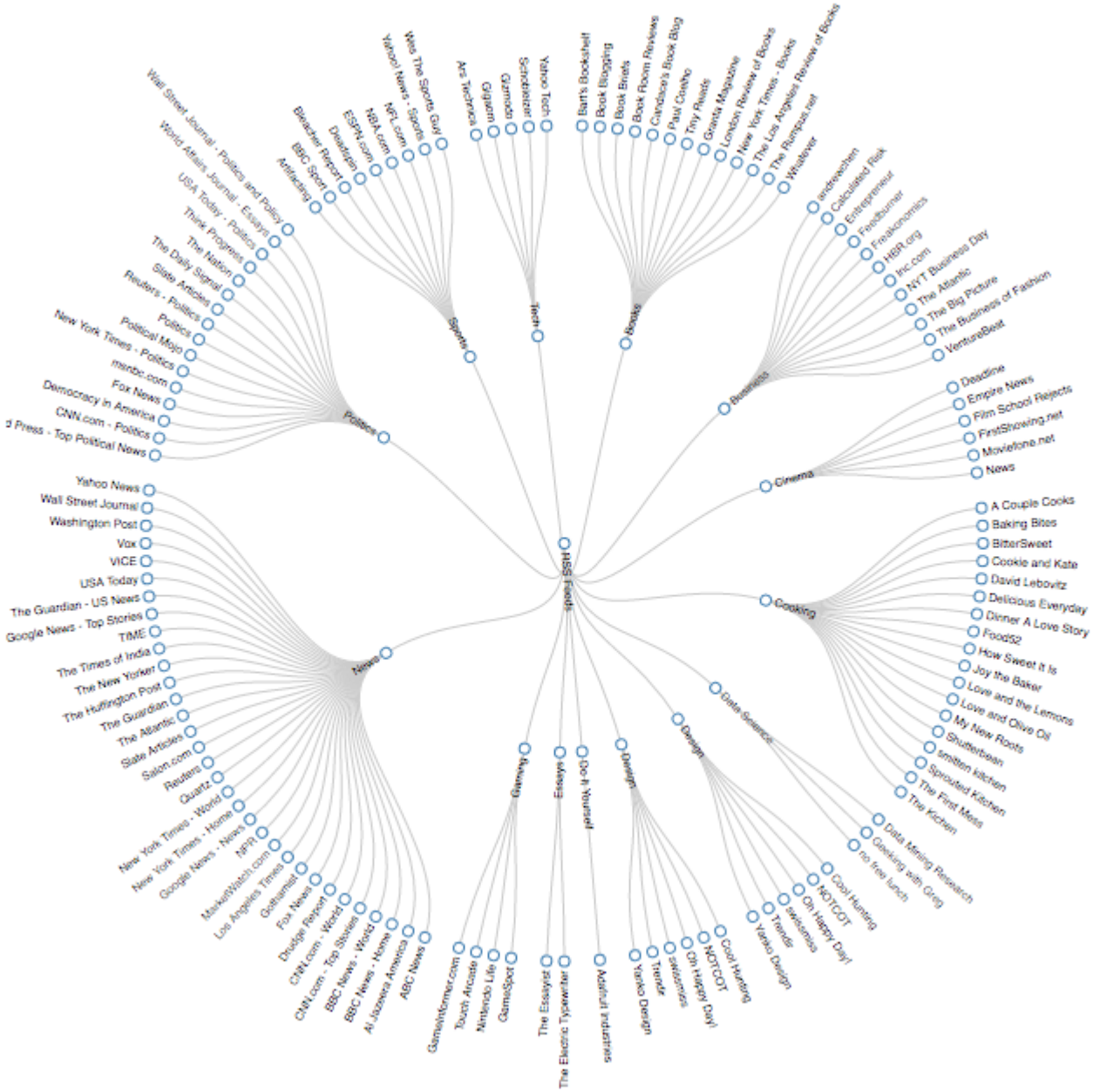
becomes more challenging. While there is a lot of valuable information to be extracted from such data, it is not natively in a computer understandable format, which makes it difficult to present insights while keeping the natural narrative intact.

Modern text visualization provides a solution to this challenge. Text visualization takes a media that we're very familiar with — language, and transforms the flood of information into a snapshot of the data that is more easily interpreted than painstakingly reading through hundreds of documents. David McCandless, a London based data journalist, gave a TED Talk entitled "The beauty of data visualization." In the talk, he noted that our sense of sight is the fastest of the five senses and that the "eye is exquisitely sensitive to patterns in variations in color, shape and pattern." He goes on to say that combining "the language of the eye" and the "language of the mind" provide a powerful combination to gather insights from data.

How to acheive a similar high quality "of display with text is an open question," Hearst states in *Search User Interfaces*. She continues, "Placing nominal data like words or names along an axis, unfortunately, is much like scrambling the years in a plot of change over time….Unfortunately, because text is nominal, many attempts to visualize text result in nonsensical graphs…" Through our two visualization suggestions in this post, we hope to present instructions for clear alternatives to summarize corpus content.

The Baleen Corpus

In order to demonstrate the effectiveness of text visualization, we have provided some examples of visualizations of an analysis on a non-trivial corpus. The source of these analyses is the Baleen corpus, a service that ingests news and blog posts from RSS feeds on a daily basis. The service has been collecting feeds since March 3, 2016 and has more than 52,000 articles in HTML format. The feeds span a variety of categories including news, politics, business, tech, and cooking. The following dendrogram chart shows the categories and feeds that are ingested into Baleen.



The articles are ingested into a hefty MongoDB database that is over 10GB. The database contains three primary collections:

- Feeds — contains metadata information about each feed, such as name, category, and the creation date
- Posts — contains details about the articles that were ingested, such as the feed that

For the purposes of this post, we will focus on the feeds and posts collections. If you would like a snapshot of the Baleen corpus, please request it via direct message on Twitter!

Extracting Data to Visualize

In order to extract the source data for analysis, the `feeds` and `posts` collections were joined to create a new collection called `feeds_posts`. Because the data set was so large, we used MongoDB's built in map/reduce functionality to create the new collection as an aggregation. The MongoDB code to join the posts is below:

```
mapFeeds = function() {
    var values = {
        feedsId: this._id,
        category: this.category,
  title: this.title,
    link: this.link
    };
    emit(this._id, values);
};

mapPosts = function() {
    var values = {
        postsId: this._id,
        postsTitle: this.title,
    feed: this.feed,
        url: this.url,
    pubdate: this.pubdate
    };
    emit(this.feed, values);
};

reduce = function(k, values) {
    var result = {}, postsFields = {
        "postsId": '',
        "postsTitle": '',
        "feed": '',
    "url": '',
    "pubdate": ''
    };
    values.forEach(function(value) {
        var field;
```


```
        } else if ("feeds" in value) {
            if (!("feeds" in result)) {
                result.feeds = [];
            }
            result.feeds.push.apply(result.feeds, value.feeds);
        }
        for (field in value) {
            if (value.hasOwnProperty(field) && !(field in
postsFields)) {
                result[field] = value[field];
            }
        }
    });
    return result;
};

db.feeds.mapReduce(mapFeeds, reduce, {"out": {"reduce":
"feeds_posts"}});
db.posts.mapReduce(mapPosts, reduce, {"out": {"reduce":
"feeds_posts"}});
```

The fields of interest in the joined collection are:

- Category (News, Politics, etc)
- Feed Name (CNN, FOX News, MSNBC, etc)
- Article title
- Article URL

Now that the data set was aggregated in the `feeds_posts` collection, we extracted the data category as JSON files using `mongoexport`. For example, below is a `mongoexport` query that pulls out all the article information from the "business" category:

```
$ mongoexport --db baleen --collection feeds_posts --query \
'{"value.feeds":{"$exists":true}, "value.feeds.pubdate" : { $type :
"date"}, "value.category" : "business"}' \
--out feeds_posts_business.json
```


content in the posts collection of the Baleen database using a package such as Beautiful Soup. However, we chose not to do so because we found Newspaper to be simpler and more effective in parsing HTML content. The parsing step in Newspaper is just one line of code and it worked for the majority of the articles that we parsed. The input for Newspaper is a URL (in this case, the article URL was used). The URL is used to go the specific web page, download the HTML document, and parse the document into a readable text file. The code can be modified based on how the text files need to be organized.

The code below organizes the files in the following manner in the file system:

Category > Feed > ArticleTitle_Date.txt

```
#!/usr/bin/python
import sys
import os
import json
import newspaper

input_file      = sys.argv[1]
output_root_dir = sys.argv[2]

from newspaper import Article

json_data = []
file = open(input_file, 'r')
try:
    for line in file:
        json_data.append(json.loads(line))
except:
    pass
file.close()

for i, json_val in enumerate(json_data):
        category = json_val['value'].get('category')
        feed = json_val['value'].get('title')
        output_dir = output_root_dir + '/' + category + '/' + feed
        try:
            os.makedirs(output_dir)
```


```
#note that parse below is a separate function
#not shown here for brevity
parse(feeds)
```

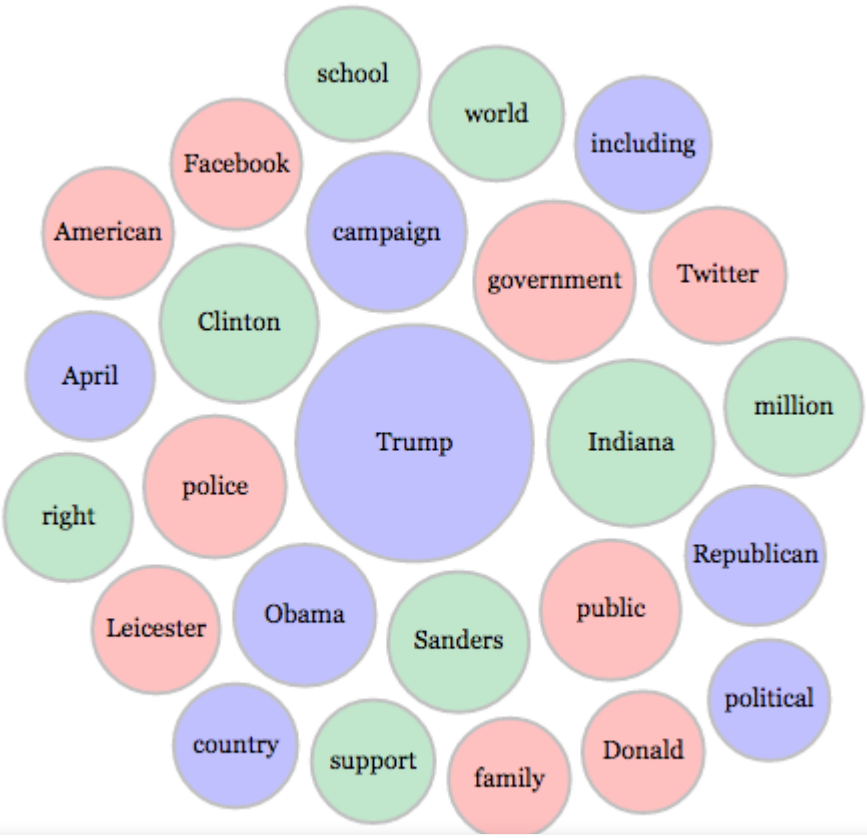
Beyond Word Clouds (Word Bubbles)

Since the early days of text visualization, word clouds have been used exhaustively as a means to represent text data. The idea behind word clouds is to use font size to denote frequency of usage of a given word. However, word clouds are becoming increasingly unpopular among data analysts as interest in Natural Language Processing grows. Jacob Harris, former NYTimes senior software architect, called word clouds "a shoddy visualization" that "support only the crudest sorts of text analysis." Word clouds are often confusing, difficult to read, and do not help convey any information about the text. The word cloud below haphazardly places text; some words are placed horizontally, while others are placed vertically. The reader has to spend effort to make out the words, which deters from understanding what the visual is about. Some word clouds add color to the words, which make the visual look pretty, but do little else.

Utilizing the `tm` and `wordcloud` R libraries, the Baleen corpus was stripped of whitespace, custom stopwords, and stemmed to create the default word cloud shown below:

Rather than using word clouds, we prefer to use word bubbles. The visual below shows the top 25 words used in Baleen's news feeds in May 2016. The words are enclosed in bubbles, which vary in size based on the word's frequency. The words are all presented horizontally, allowing the reader to focus on the bubble size to make comparisons. In addition, the visualization uses tooltips. A tooltip is a great tool to provide additional information without adding clutter to a data visualization. The reader simply needs to hover over a bubble of interest and the tooltip provides more detail about the data the bubble represents.

In this visualization, the word and its frequency are displayed. The bubbles are randomly filled with colors to beautify the image. A potential enhancement to the visual is to use shades of a single color to differentiate between the bubbles. For example, the smallest bubble can have the lightest shade and the largest bubble can have the darkest shade. With this change, both bubble size and color can be used to make comparisons between different words and their usage.



The Javascript code leverages d3.js to generate the word bubbles. The top 25 words were generated by month from February 2016 to May 2016 and stored as csv files and placed in directories by month. The script takes these files as input and dynamically generates the visualization based on the month chosen by the user.

The word bubbles are a well-organized alternative to show frequently appearing content within the corpus, but still miss connections between words; therefore, we will explore word networks next.

Beyond Word Clouds (Word Network)

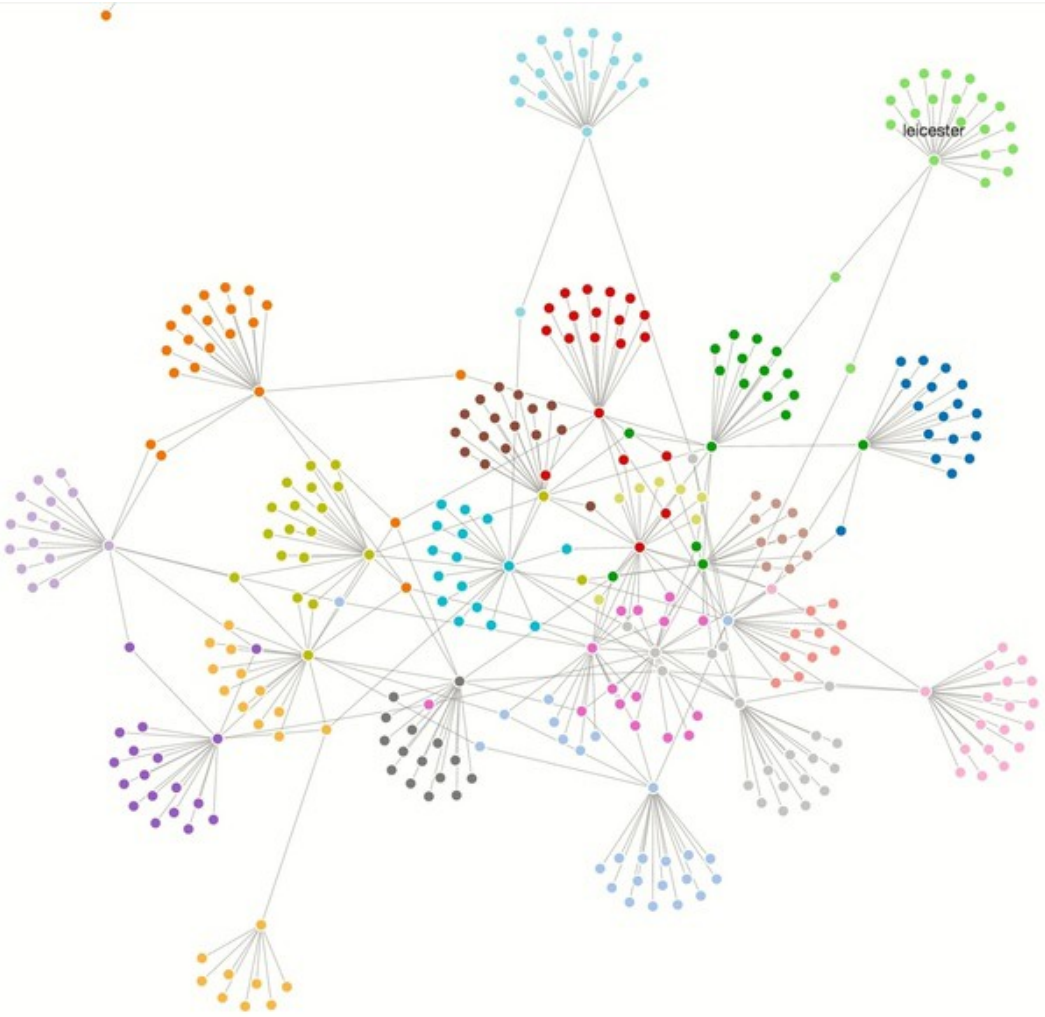
In order to create the network visualization, we use Python's NLTK library to find the top 30 one-word, non-plural people/organizations using named-entity recognition. Using these entities, we identify similar words from immediate context counts. In particular, if a word appears directly before or after the person/organization of interest multiple times, this is considered a similar word. The created Python dictionary is transformed into a JSON file where each node is a word, which links to a target, colored by group according to the base named-entity.

Example code for utilizing NLTK for this purpose is shown below. The corpus was also processed (tolower, custom stopwords removed, etc.) after it was read into Python. The dictionary `all_words` was later used to create the JSON for d3, which was modelled using this JSON.

```
import nltk
from collections import Counter

# The txt file is opened and tokenized
text_tokens = nltk.word_tokenize(file_text)

# Named-entity recognition
corp_tag = nltk.pos_tag(text_tokens)

corp_chunk = nltk.ne_chunk(corp_tag)
```


```
# Find the most common proper nouns
# (in our case, only the first word was kept to eliminate duplicates
# like first-name last-name, and last-name)
top_30 = Counter(proper).most_common(30)

# Find similar words
text = nltk.text.ContextIndex([word for word in text_tokens])

all_words = {}
for word, similar_words in words.items():
    all_words[word] = word
    for similar_word in similar_words:
        all_words[similar_word] = word
```

The network graph should give a more comprehensive depiction of the conversation happening within the text. Using context to link important people and organizations creates an understanding of the content, relationships, and the message within the text. For example, in early May, Leicester City Football Club won the Premier League title. Our network includes “Leicester,” which links to words like “BBC,” “CNN,” “self motivate,” “work,” “Manchester United,” and “streets.” Thus, the network graph tells part of the story of Leicester City, a self-motivated, hardworking team whose title-decider against Manchester United helped them claim victory. The Cinderella story reported both in the UK and in the US sparked celebrations in the streets of Leicester.

This force-directed network graph depicts the Baleen corpus as a narrative. Some of the proper nouns relate directly to one another (i.e. “Trump” and “Cruz”). The chart also shows how people and organizations within the text relate to each other through their similar words. Words that are shared by two frequently used proper nouns connect the node clusters together. When the similar word links to two proper nouns, it can show conceptual similarities between two topics of interest. By examining nodes *within* each cluster, we can gain insight into understanding each word of interest. Through the connections, we can find similarities *between* each cluster. The code used to generate the network graph can be viewed here.

Conclusion

As the scope of what we consider to be data increases, the role of the data visualizer

relevant data from two of the collections from the MongoDB database, outputting the results into JSON files, and using Python's Newspaper package to convert the HTML content into text. After extracting the data, we created two visualizations. Our word bubble chart uses bubble size to compare frequency of word usage, but also uses a tooltip to display the exact frequency. The network visualization shows potential relationships between different frequently used p[illegible] data in an attempt to visualize the greater conversation within the corpus. These are just two ways to improve text visualization, to move beyond the word cloud.

District Data Labs provides data science consulting and corporate training services. We work with companies and teams of all sizes, helping them make their operations more data-driven and enhancing the analytical abilities of their employees. Interested in working with us? Let us know!