UWE Bristol | University of the West of England

Presentation by

Dr. Phil Legg

Associate Professor in Cyber Security

Security Data Analytics and Visualisation

9: Text Analytics

Date: Autumn 2019

Recap

- We have looked at analytics techniques, and visualisation techniques.
- We have surveyed the literature to see the challenges in this area.
- Previously we may have considered data to be confined to numerical tables…

- Let's now consider different forms of data that may be useful for understanding security challenges:
  - This week we will consider **Text** as a form of data.

Understanding Text

- As a primary form of communication, the written (or spoken) word provides a massive wealth of data for many applications.

- Natural Language Processing (NLP) is a subject area in it's own right!

- What kind of "processing" may we want?
  - Natural Language Generation – e.g., chat bots?
  - Topic Modelling – e.g., classification of documents?
  - Sentiment Analysis – e.g., understanding human emotion?
  - Text clustering – e.g., relationships between words?
  - Named Entity Recognition – e.g., the blue **car** drove down the **road**
  - Term Frequency Inverse Document Frequency – identifying key words of interest

Security Scenarios

- How does Text Analytics relate to security needs?
  - Understanding user emotion through use of language?
  - Classification of conversations?
  - Authentication of user based on use of language?
  - Understanding changes in use of language?

- What kind of data may we consider trying to analyse?
  - E-mail text data / Messaging data
  - Web page content
  - Social Media content
  - Transcribed audio
    - *... Probably others of interest too!*

Text Analytics

- Text clearly offers a wealth of information – *it is how humans communicate.*

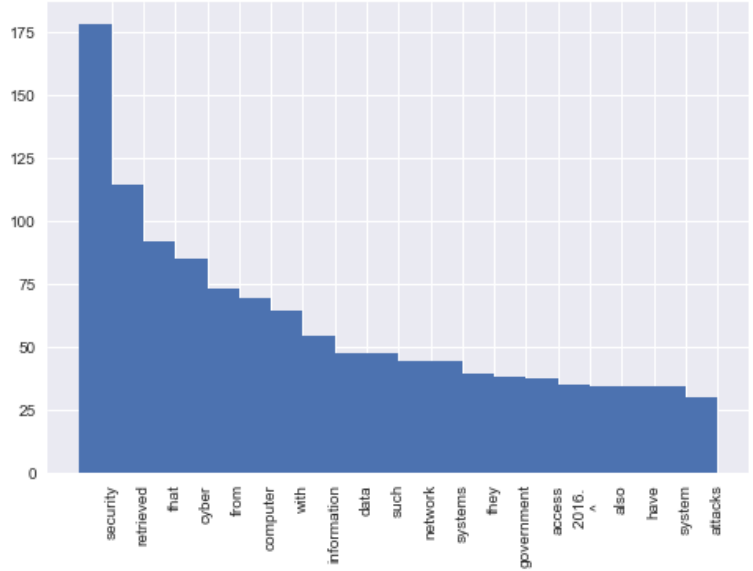
- Humans can not read/ process all this information effectively
  - We need automated approaches to better support users.

- How can we develop analytical tools that can handle text input data, and utilise this effectively?

Dictionaries and Counting

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from bs4 import BeautifulSoup
import urllib.request

### Open the web page and save this to a text file
file_name = 'wikipedia.txt'
url = "https://en.wikipedia.org/wiki/Computer_security"
with urllib.request.urlopen(url) as response, open(file_name, 'wb') as out_file:
    data = response.read()
    out_file.write(data)

### Open the text file using Beautiful Soup to parse the HTML
with open(file_name) as fp:
    soup = BeautifulSoup(fp, 'html.parser')

### Extract the text from the HTML and split based on spaces
text = soup.text
text = text.split(" ")

### For each word, if length is greater than 3
### then set to lowercase and append to our final text list
text_final = []
for t in text:
    if len(t) > 3:
        t = t.lower()
        text_final.append(t)

### Count the number of occurrences for each word
from collections import Counter
count = Counter(text_final)

### Split into labels and values for our chart
labels, values = zip(*count.most_common(20))
indexes = np.arange(len(labels))
width = 1

### Plot chart
plt.bar(indexes, values, width)
plt.xticks(indexes + width * 0.5, labels, rotation=90)
plt.show()
```

Counting the occurrence of words in a document *(or a probability if we divide by total number of words)*



Dictionaries and Counting

Counting the occurrence of words that appear in a given dictionary

```
dictionary_list = ['security', 'attack', 'data', 'encryption', 'cyber', 'crime']
dictionary_counts = {}
for t in text_final:
    if t in dictionary_list:
        if t not in dictionary_counts:
            dictionary_counts[t] = 0
        dictionary_counts[t] += 1
```

- Simple yet effective approaches for characterising text
- Suppose we had a more sophisticated dictionary, then we can do much more analysis using the same approach
- E.g., Linguistic Inquiry Word Count (LIWC) dictionaries
  - Classes include positive and negative emotions, verbs, etc.

- As a count…?

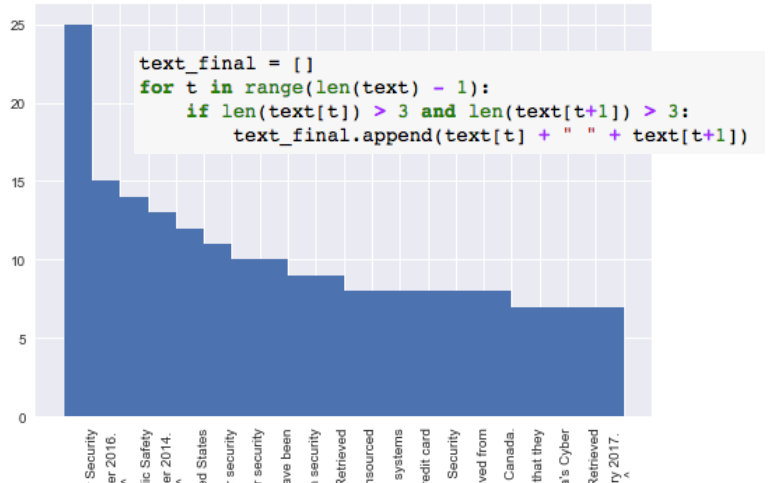
```
{'attack': 20,
 'crime': 12,
 'cyber': 85,
 'data': 47,
 'encryption': 5,
 'security': 178}
```

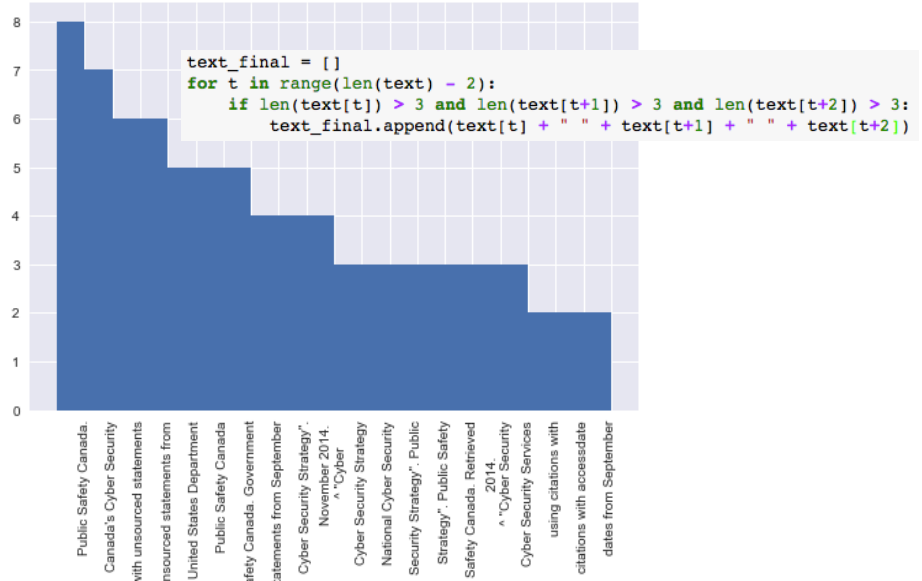
- Or as a probability…?

```
{'attack': 0.002148227712137
 'crime': 0.0012889366272824
 'cyber': 0.0091299677765843
 'data': 0.005048335123523 09
 'encryption': 0.00053705692
 'security': 0.019226638023 6
```

N-grams

- Previously, words were independent – however words have contextual meaning – need to know what occurs before / after!
- N-grams essentially groups (N number of words) together





TF-IDF

- Term Frequency – Inverse Document Frequency
- How can we avoid common (but not necessarily important) words?
  - Currently, we simply filter short words but this is not necessarily ideal.
  - Better way is to filter by importance, based on occurrence of a word across whole set of documents
  - If word occurs often in one document, but not often in other documents, it is likely to be important

- TF(t) = number times t appears in document / total number of terms in document
- IDF(t) = log (total number of documents / number of documents with t in)

- https://en.wikipedia.org/wiki/Tf–idf
- http://www.tfidf.com

TF-IDF

- Term Frequency – Inverse Document Frequency

- Consider a document with 100 words, and **cat** appears 3 times.
  - TF = 3 / 100 = 0.03
- Suppose we have 10 million documents, and cat appears in 1000 of these:
  - IDF = log(10,000,000 / 1,000) = 4

- TF-IDF = TF * IDF = 0.03 * 4 = 0.12

- **Cat** is weighted higher using TF-IDF than only using TF, to signify that it is more important in relation to the overall document set.

Recommender Systems

- How can a system ”learn” to make recommendations?
  - Used heavily in shopping basket analysis
  - Same principle can be applied in other applications
- Can be used to find relations between items (e.g., words)

Recommender Systems

- How can a system ”learn” to make recommendations?
  - Used heavily in shopping basket analysis
  - Same principle can be applied in other applications

	User 1	User 2	User 3	User 4	User 5
Bread	1	1	0	0	1
Milk	0	0	0	1	0
Cheese	1	1	0	1	0
Pizza	0	1	0	0	1
Bananas	1	0	0	0	1
Apples	1	1	1	1	0
Oranges	0	0	0	1	1
Chicken	0	1	1	0	0
Beef	0	0	0	1	0

Recommender Systems

- How can we predict what a New User may buy? (NEW1)

	User 1	User 2	User 3	User 4	User 5	NEW1	NEW2
Bread	1	1	0	0	1	?	?
Milk	0	0	0	1	0	?	?
Cheese	1	1	0	1	0	?	1
Pizza	0	1	0	0	1	?	?
Bananas	1	0	0	0	1	?	?
Apples	1	1	1	1	0	?	?
Oranges	0	0	0	1	1	?	?
Chicken	0	1	1	0	0	?	?
Beef	0	0	0	1	0	?	?

Recommender Systems

- How can we predict what a New User may buy? (NEW1)
- Can calculate the probability of purchase based on all previous purchases
- Apples are most likely purchase (0.8), followed by Bread and Cheese (0.6).

	User 1	User 2	User 3	User 4	User 5	NEW1	NEW2
Bread	1	1	0	0	1	0.6	?
Milk	0	0	0	1	0	0.2	?
Cheese	1	1	0	1	0	0.6	1
Pizza	0	1	0	0	1	0.4	?
Bananas	1	0	0	0	1	0.4	?
Apples	1	1	1	1	0	0.8	?
Oranges	0	0	0	1	1	0.4	?
Chicken	0	1	1	0	0	0.4	?
Beef	0	0	0	1	0	0.2	?

Recommender Systems

- What if we already have some knowledge about their purchase? (NEW2)
- If they buy Cheese, what else will they likely buy?

	User 1	User 2	User 3	User 4	User 5	NEW1	NEW2
Bread	1	1	0	0	1	0.6	?
Milk	0	0	0	1	0	0.2	?
Cheese	1	1	0	1	0	0.6	1
Pizza	0	1	0	0	1	0.4	?
Bananas	1	0	0	0	1	0.4	?
Apples	1	1	1	1	0	0.8	?
Oranges	0	0	0	1	1	0.4	?
Chicken	0	1	1	0	0	0.4	?
Beef	0	0	0	1	0	0.2	?

Recommender Systems

- We can omit users (columns) that do not buy Cheese, then calculate probabilities as before.
- Here, the data suggests that New User will certainly buy Apples (1), and possibly buy Bread (0.66).

	User 1	User 2	User 3	User 4	User 5	NEW1	NEW2
Bread	1	1	0	0	1	0.6	0.66
Milk	0	0	0	1	0	0.2	0.33
Cheese	1	1	0	1	0	0.6	1
Pizza	0	1	0	0	1	0.4	0.33
Bananas	1	0	0	0	1	0.4	0.33
Apples	1	1	1	1	0	0.8	1
Oranges	0	0	0	1	1	0.4	0.33
Chicken	0	1	1	0	0	0.4	0.33
Beef	0	0	0	1	0	0.2	0.33

Recommender Systems

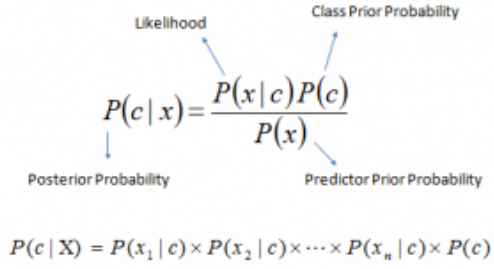
- With many columns (users), and many rows (items), we can obtain a much more confident prediction about user preference and behaviour.

- Suppose we want to study the use of language by users
  - Given a corpus of text, we could have a row for each possible word, and a column for each possible user.
  - This would provide a "model" of what words each user says
  - When we observe a *new* subset of words (e.g., part of a conversation), we can predict which user is likely to have said this.

- *See Jupyter Notebook – just finish the recommender bit for identifying user then we are done with this slide deck*

Spam Detection

- Developing a classifier to recognize patterns of spam e-mail, or phishing characteristics
  - We can use Naïve Bayes approach – based on Bayes theorem.
  - **"What's the probability of this e-mail being spam, given that it contains the word X ?**
  - We use spam and non-spam to learn the probabilities for new e-mails.
  - More detail at these links:
    - https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
    - https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
    - https://hackernoon.com/how-to-build-a-simple-spam-detecting-machine-learning-classifier-4471fe6b816e



More on Text Analysis

- **Recurrent Neural Networks**
  - Currently regarded as state-of-the-art (e.g., Long Short-Term Memory nets)
  - Suitable for learning sequential patterns in data (e.g., "the cat sat on the …")
  - Can be used for generation (e.g., chat bots)
  - Could also be used to recognise when unexpected sequence occurs? (e.g., is this genuine if not as expected?

- **Word2Vec Word Embeddings**
  - Similar to PCA – how can we represent relationship of words in a vector space?
  - Continuous Bag of Words (CBOW)
    - Given a set of words, what one word would fit with these?
  - Skip-grams
    - Given one word, what set of words would fit with this?