VerTex

Francisco Vargas

April 12 2017

Introduction

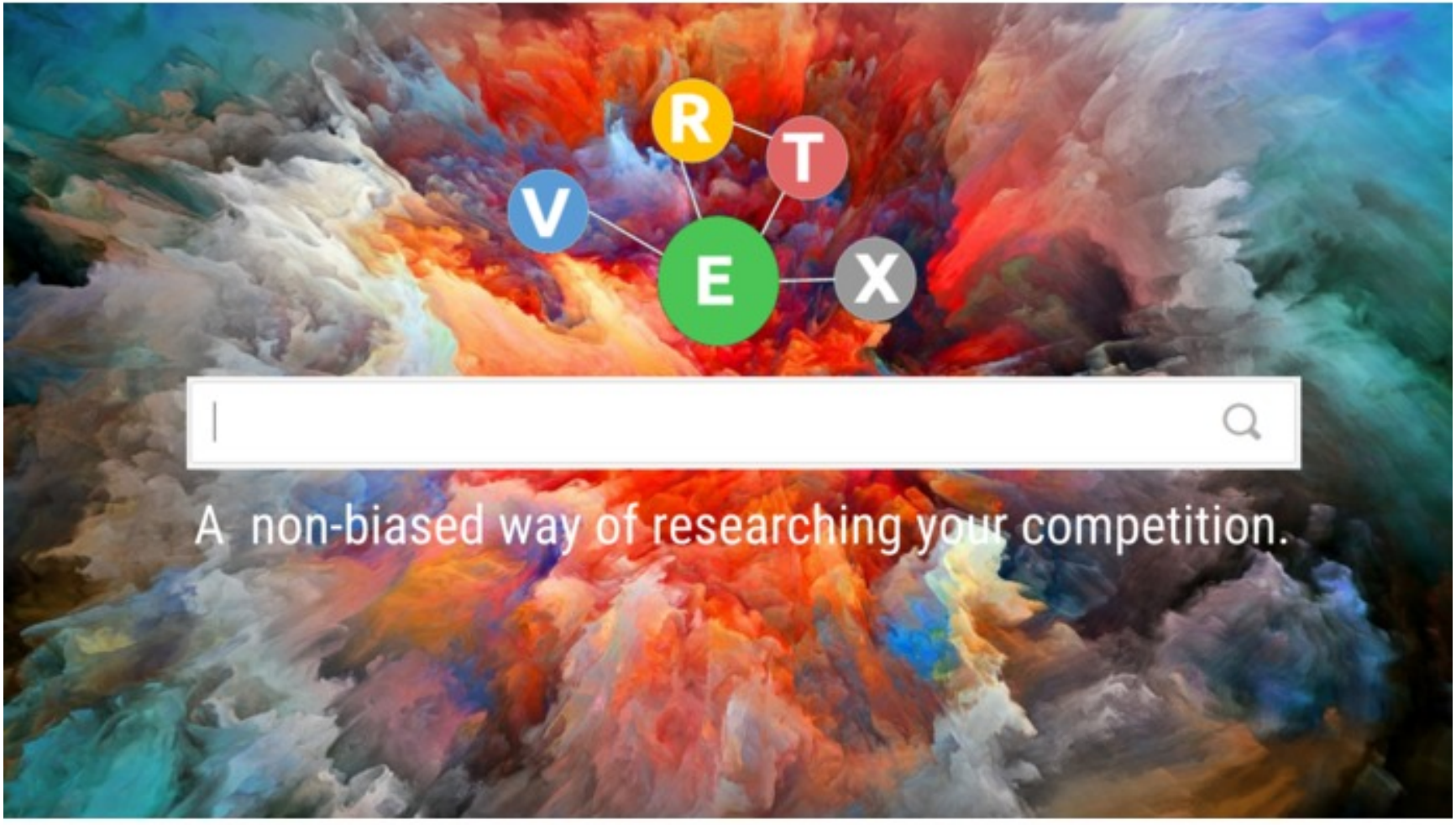
Presenting a "clever" Bing scraping agent.

- VerTex : An NLP/ML web based search engine

Demos

If time allows I can show some small clips (about a minute) on the facebook hackathon and a speech skype conversational agent.

VerTeX



VerTeX - Motivation

- Most search engines cache results and use it to bias our searches.
- HITS and PageRank are great algorithms however ranking results on number of incident and outgoing links ignores semantical and taxonomical content within the sites.
- Ever wondered what is beyond page 10 of your search results ?

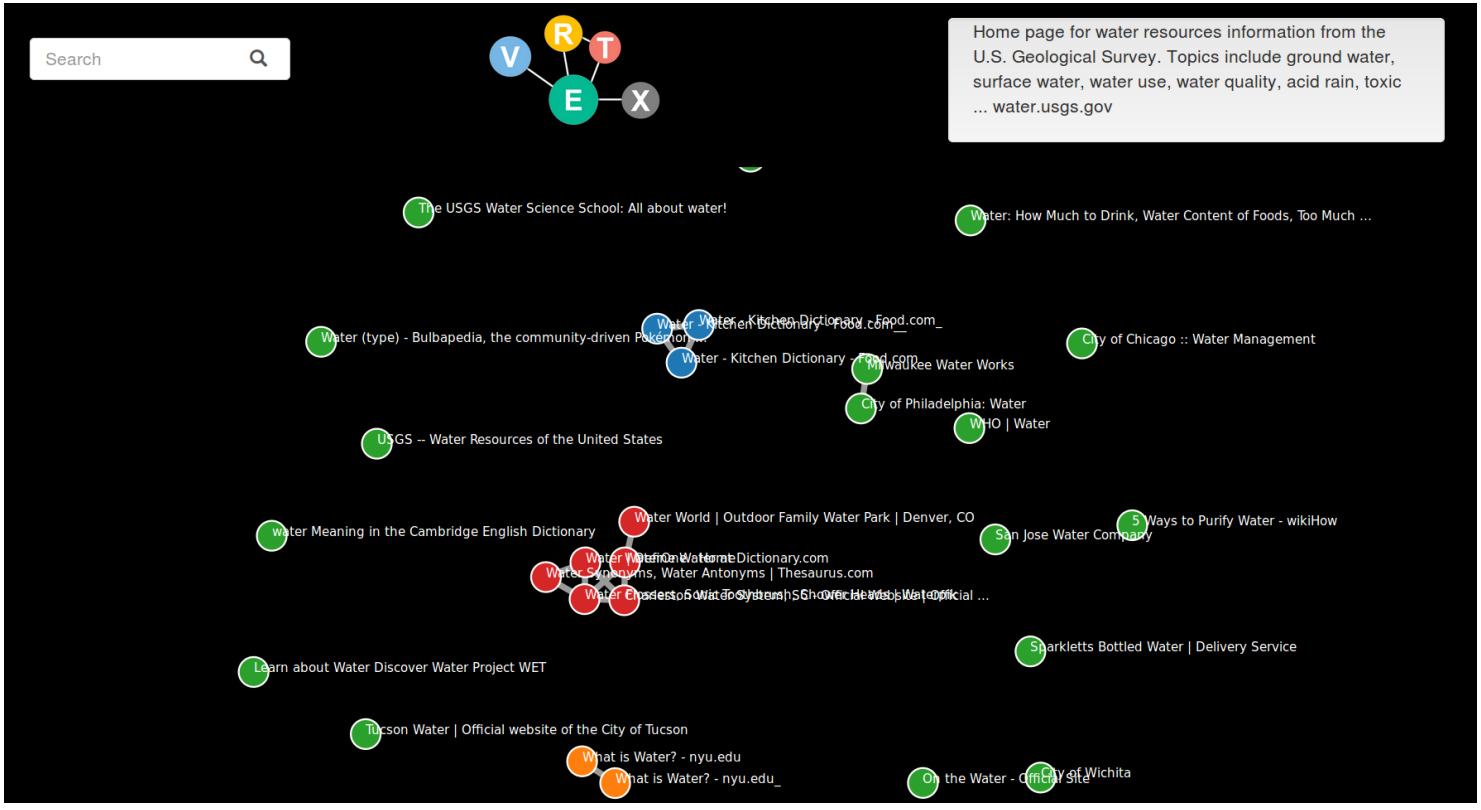
VerTeX - Solution ?

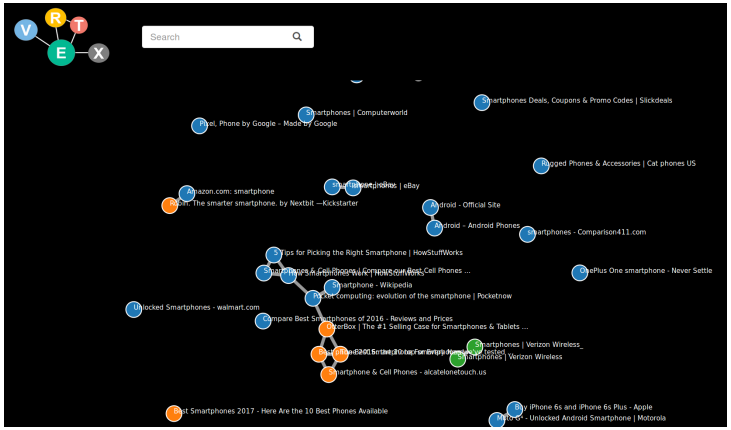
- Take the "entire" result set R and sample webpages $r \in R$
- Incorporate a bias in the sampling to acknowledge PageRank
- Construct a weighted graph $G(R, E)$ where the edges are estimated using NLP on the text for every $r \in R$

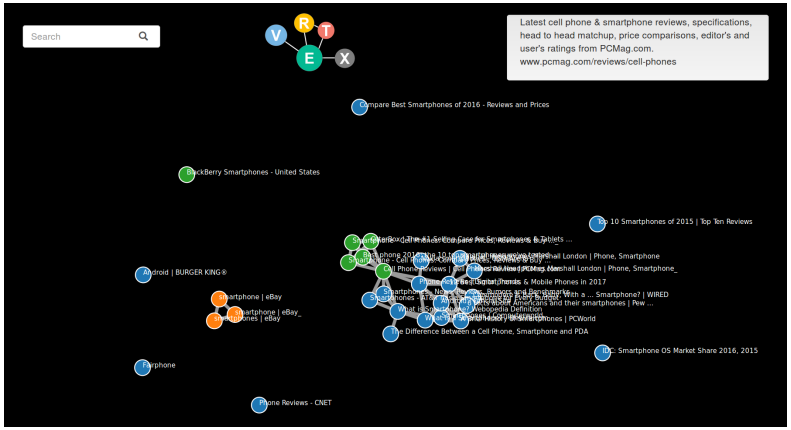
$$E = \left\{ (x, y) \quad | \quad (x, y) \in R \times R \bigwedge \mathit{NLPConnected}(x, y) > \alpha \right\}$$

$$w(x, y) = \mathit{NLPConnected}(x, y)$$

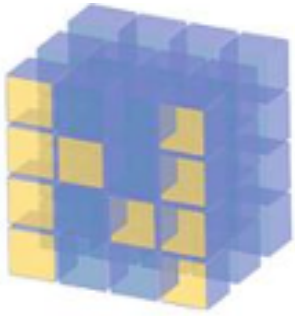
VerTex - Images







VerTex - Tech Stack



NumPy

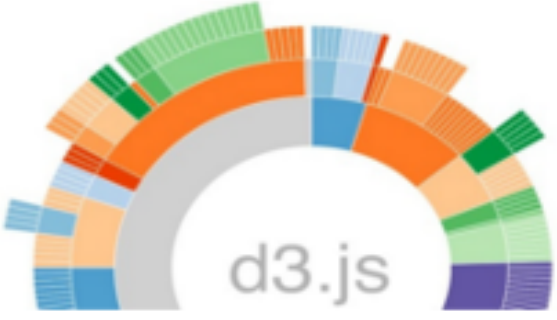
scikit learn

Flask

web development,
one drop at a time

bing

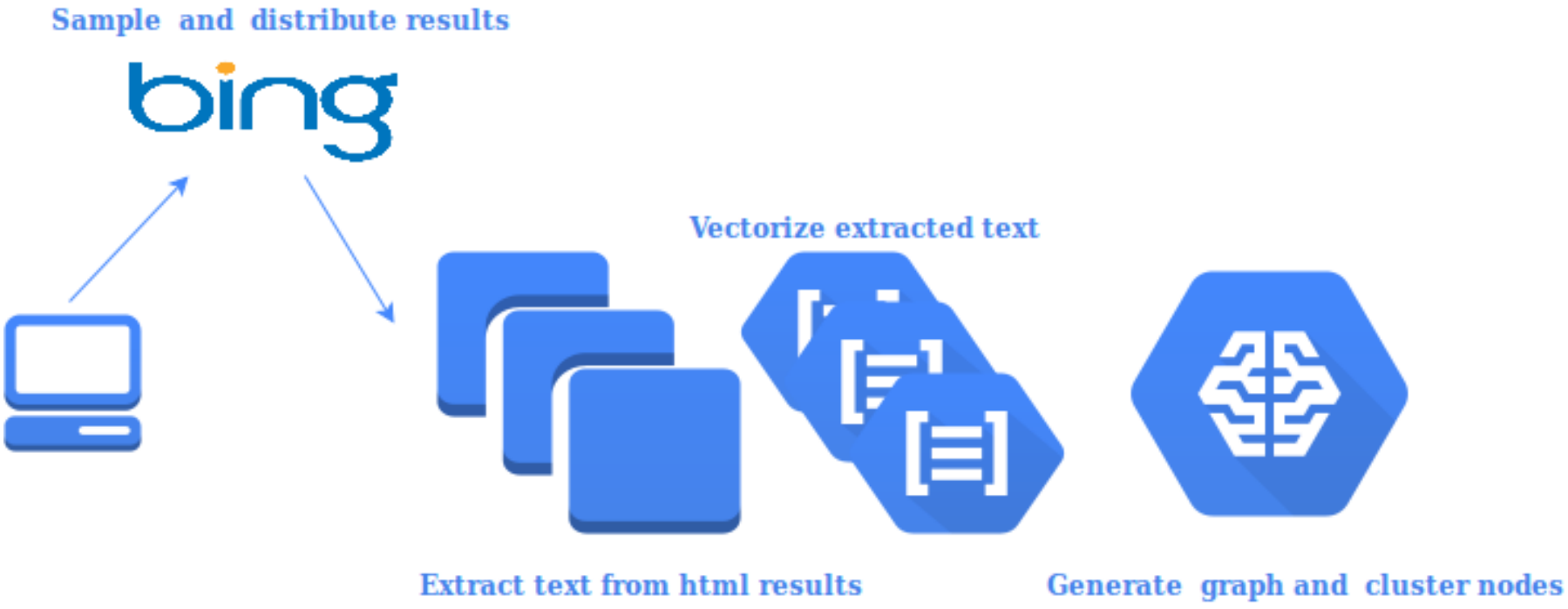
Natural Language Analysis
with Python NLTK



gensim

topic modelling for humans

VerTex - Pipeline



VerTex - Algorithms (Bing Sampling)

- A number N of desired results to be viewed is set (in VerTex it is set to 30 by default)
- Set $\beta \approx 1$.
- Randomly sample $N * \beta * 2^{-1}$ results from top 50 ranked bing results.
- Then in the range of results (50,100] sample $N * \beta * 2^{-2}$.
- Then sample $N * \beta * 2^{-3}$ in the range (100, 150] until $N * \beta 2^{-n}$ is bellow 1.

VerTex - Algorithms (HTML Parsing)

- Most python libraries failed at extracting the text from the HTML source.
- In order to extract the text, the english dictionary was loaded into a hashset data structure and the html string was split by space into a list of candidate words.
- For each candidate word all words which were not in the hashset were removed from the list. Each member check operation cost $O(1)$ due to hashing.

VerTex - Algorithms (HTML Parsing)

```
import requests
html_string = "<head> ... </body>" # from a url
html_list = html_string.split(" ") # O(n)
hash_set = set(requests.get("http://www-01.sil.org/
    linguistics/wordlists/english/wordlist/wordsEn.txt"
    ).content.split("\r\n"))
clean_html = []
for word in html_list: # O(n) amortised time
     if word in hash_set: # amortised O(1)
          clean_html.append(word)
```

VerTex - Algorithms (HTML2Vec)

- Each html document (Bing query result) is a node in the graph.
- Stemming algorithm used to pre-process words by mapping them to their root words.
- Html extracted text was mapped into a euclidean space using a tf-idf vectorizer ($\mathbf{doc} \in \mathbb{R}^{|corpus|}$)
- Word2Vec is used to normalize the tf-idf highlighting relevance to initial Bing search.

VerTex - Algorithms (Graph Creation)

- Cosine similarity measure used between document vectors:

$$\left(w(i,j) = cos(\theta_{ij}) = \frac{\mathbf{doc_i} \cdot \mathbf{doc_j}}{|\mathbf{doc_i}||\mathbf{doc_j}|} \right)$$

- A threshold α was determined statistically.
- If $cos(\theta_{ij}) > \alpha$ an edge is created between docs i and j in the graph.

VerTex - Algorithms (Taxonomy Clustering)

- Gaussian Mixture Model for document clustering
- A Gaussian Mixture Model iteratively fit k Gaussians on a set of vectors Φ using the Expectation Maximization algorithm.
- The number of clusters determined by silhouette analysis.
- An alternative - elbow method

Demo

- https://youtu.be/mVK-9fuMW7Q