**Yelp Intelligent Reviews**

**Problem:**

**Find and screen the most important section of the review based on a key word.**

**e.g: Search query “deep dish pizza”**

**The output section of the review should be relevant to deep dish pizza.**

**The size of output and content is based on the programmer.**

*Algorithm*

1. Clean the Data using data cleaning strategies
2. Pre-compute the indices of starting of each word and sentence
3. Find where the query word(s) occur in the main document by storing the word starting indices with the priority( more consecutive the no. of query words are, higher the priority)
4. Apply KMeans clustering on the starting indices to cluster the indices which are closer.
5. Prioritize the clusters, based on weight, computed by using their priorities.
6. Compute the Starting sentence closer to the first member of the Cluster Group, and the ending sentence closer to the last member of the Cluster group, to make the sentence more readable
7. Return the short review by highlighting the keywords.

*How I approached the problem*

I put myself into the user’s shoe, and asked myself the following questions before I started implementing the algorithm

Data Cleaning and PreProcessing

* Remove multiple spaces with just one space

User Friendly features

* I would want the same results if I query “ pizza deep dish”, what I meant really was “ deep dish pizza” , “spicy indian chicken” or “ indian chicken spicy”
* If I query ‘pizza’, I would still want to view results with the word ‘pizzas’, but I don’t want a result like ‘ I am *deep*ly concerned…’ when I query ‘*deep* dish pizza’
* I would be more interested in a line if it matches my query directly, e.g: ‘this place is famous for *spicy indian chicken* for “spicy indian chicken” query, as it is a direct match.
* I would not be interested in spam, even though it occurs many times in one sentence.

e.g: “bacon bacon bacon bacon bacon bacon” for “bacon sandwich” query string.

The algorithm should intelligently mine which is relevant for me.

*Algorithm*

* How to best identify which part of the document might be most relevant to the user
* How long/short should the result be
* Other features such as size of the data, format, and the data type, data structures that should be used to speed up the processing.

*What my algorithm can do very well*

* My algorithm can return a highly relevant shortened review, if there are a lot of query keywords in the document to be parsed.
* My algorithm is good for multinational people, whose English is not that great, so they mean something else and they type something else by jumbling up the keyword query.
* My Algorithm can also recognize certain words that means/represents the same as was queried by the user, but wasn’t exactly the same.(e.g: singular, plural)
* My Algorithm can take care of spamming keywords, multiple keywords in 1 line.
* My Algorithm knows which set of keywords are important than others, and yields best result by prioritizing set of keywords.

*What my algorithm cannot do very well*

* It would not yield best results, when they keywords are sparse and distant from each other.

That is mainly, due lack of k- identifying strategies in using K-means.

I wanted to implement my own K-Means function ( with hierarchical clustering) which would work best if the data is sparse and distant, but could not due to lack of time.

*Testing strategies*

* I tested the data by computing my own query search on yelp website
* Used PyUnit as the testing tool
* Cases include

1. When query doesn’t match the document
2. When query keywords are very few
3. When query keywords are few and distant
4. When query keywords are more
5. When query keywords are jumbled, spelled wrong, case sensitive, slightly incomplete