**CSE5095 - Topics in Big Data Analytics**
Date: 04/08/2014
Note taker: Tham Hoang

**A STRAIGHT FORWARD Algorithm for records linkage take $O(n^2L^2)$ time**

$n \rightarrow$ The number of records.
$L \rightarrow$ The max length of any record.
Note: The distance between two records can be compute in $O(L^2)$ time or $O(L\tau)$ time where $\tau$ is a threshold on distance or $O(L+\tau^2)$ time.

**Blocking**

For every record R, generate $l-mers$ (for some $l < L$). In practice, $l = 3$ or $l = 4$.
Bucket the records based on $l-mers$.

**Note:**

The same record goes into $(L-l+1)$ buckets.
if $|\sum| = 36$, there are at most $36^l$ buckets. The total number of $l-mers$ is $(L-l+1)n$.
The expected size of each bucket $= \frac{n(L-l+1)}{36^l}$
Perform clustering in each bucket. We observe that if we have two strings that are very similar to each other, then they will share a small substring with a good probability.
The expected time for these clustering (using $O(L\tau)$ algorithm) is $O(\frac{n^2L^2}{36^2l}L\tau 36^l) = O(\frac{n^2L^2}{36^l}L\tau)$

**CLAIM:**

$\frac{L^2}{36^l} \ll 1$
Example: L = 36; l = 4; $\rightarrow \frac{L^2}{36^l} = \frac{30^2}{36^4} = \frac{900}{36^4} \ll 1$

**Generate a graph $G(V,E)$**

V $\rightarrow$ records
$(R_i, R_j) \in$ E if $R_i, R_j$ were in the same cluster in at least one bucket. Find the connected components in G. Each component is output as a cluster.

***K − means* clustering:**

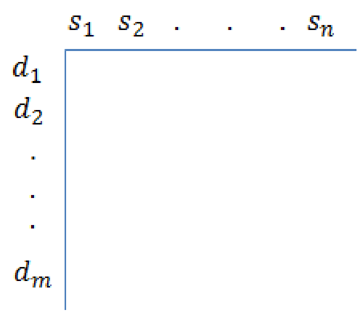
Input: $p_1, p_2, ..., p_n$
Output: $C_1, C_2, ..., C_k$
Let $q_1, q_2, ..., q_k$ be the centers of the clusters.
We want to minimize: $\sum\limits_{i=1}^{k} \sum\limits_{p_i^j \in C_i} (p_i^j - q_i)$

**Lloyd's Algorithm:**

Start with $k$ random points as centers. Associate each point with the center that is the closest to it. Now we have new clusters. Compute the center of each cluster. Associate each point with the center that is the closest to it. Repeat the above process until the clusters do not change any more.

**TEXT MINING**

Input: $D \rightarrow$ a list of drugs
$S \rightarrow$ a list of side effects
PUBMED $\rightarrow$ ABSTRACTS
Output: For every pair $(d,s) \in D \times S$, output the number of document in PUBMED in which $d$ and $s$ occur together.
Keep a matrix $M$



$M(d_i, s_j)$ = the number of co-ocurrences of $d_i$ and $s_j$.
We can solve problem in 1 pass through the data.

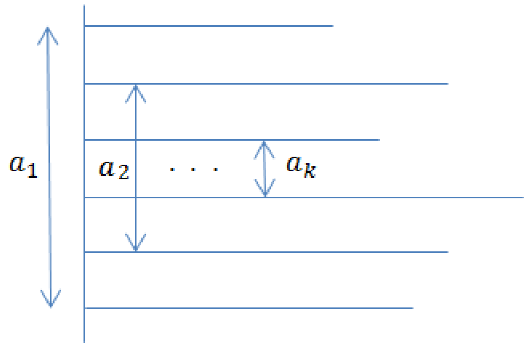
**<u>ONE IDEA:</u>** use a Generalized Suffix Tree
- Keep a GST $T_D$ for the drugs
- Keep a GST $T_S$ for the side effects.

When we bring in an abstract $A$

```
**For** each word $w \in A$ **do**
    Use $T_D$ and $T_S$ to check if $w \in T_D$ or $T_S$
    If $w \in T_D$ add $w$ to $L_1$
    If $w \in T_S$ add $w$ to $L_2$
**For** every $(w_1, w_2)$ with $w_1 \in L_1$ and $w_2 \in L_2$ **do**
    Increase the count of the relevant entry in $M$
```

**<u>Analysis:</u>**
Total run time is $O(\sum |A|)$ with $A \in PUBMED$.
The I/O complexity is one pass.

Another data structure can also be employed. The idea is to sort the drugs and side effects:



Given any word $w \in A$ with $w = a_1 a_2 ... a_k$ we can do a binary search in the sorted array (of drugs and side effects). The binary search is done in stages where in each stage the comparison is with respect to a single character. In the first stage the comparison is with respect to the first character, and so on. Each stage takes $O(\log n)$ time. Therefore, the time for searching $w$ is $O(k \log n)$ with $n = |D| + |S|$.

(To be continued)