Indian Statistical Institute
Kolkata

M.Tech. (Computer Science) Dissertation

Clustering of Web Search Queries to Identify Users' Intent

A dissertation submitted in partial fulfillment of the requirements for the award of M.Tech.(Computer Science) degree of ISI

Author:
Sayantan Sarkar
Roll No:MTC1215

Supervisor:
Dr. S.K Parui
CVPRU

CERTIFICATE

This is to certify that the Dissertation entitled **Clustering of Web Search Queries to identify User's Intent** is a bona fide record of independent research work done by **Sayantan Sarkar (Roll No. MTC-1215)** under my supervision and submitted to **Indian Statistical Institute, Kolkata** in partial fulfillment for the award of the Degree of **Master of Technology in Computer Science**.

Supervisor:

Dr. Swapan Kumar Parui
CVPR Unit
ISI, Kolkata

Acknowledgements

Coming towards the end of my journey of M.Tech(CS) Course at **Indian Statistical Institute**, it is my pleasure to acknowledge the contribution of everyone associated with my Dissertation for this last leg of my course.

First of all, I want to express my heartfelt gratitude to **Dr.Swapan Kumar Parui** for being my advisor, guide, mentor and anchor for my first interaction with the world of research.It will be forever memorable how he helped me shape the Problem Statement, and was there for my guidance for all the roadblocks with those rapid fire meetings I imposed on him.

I would like to thank **Dr.Mandar Mitra** for being ever encouraging in sharing the primary literature required for this project. It has been a sheer pleasure to get my doubts clarified by him, where instead of giving a stock solution, he nudged to towards a direction of thought that I could develop upon.

I would also like to thank **Dr. Subhas Chandra Nandy**, who has been the single most important guiding force throughout the duration of this course. He helped grapple with my most bizarre doubts which he not only gave a proper shape but also actively encouraged me to debate.

A special thanks to all my professors in Indian Statistical Institute, who made this journey not at all easy, but exciting none the less.

The most heartfelt of acknowledgment goes to the whole team of **Microsoft Bing STCI, Hyderabad**, who not only accepted me as their intern, but made me feel like an integral part of this organization. Thank you to **Mr. Deepinder Gill** for chalking out the small project that went on to become this minuscule dissertation. Each an every of those 30 minutes long conversations with you shaped months of my work.

Thank you to **Mr. Kapil Malhotra** for being the manager you were to me. For the fact that you did not take anything for granted, I was saved from being complacent not only during the internship but till date.

Special Thanks to **Mr. Vikas Mittal** always talking time out either to point the basic flaws in my work, that I was tempted to overlook or to provide me with this dataset which is today the foundation stone of this project.Special Thanks to **Mr. Akshad Vishwanathan** because I never expected to that someone else can make more sense out of my work than I can.

Finally, I cannot end without a vote of thanks to my friends who mostly mentally and sometimes physically carried me across this rush of two years.Thanks to **Sachu, Jayadev, Amit, Badri, Raghu, Saurabh, Tamal, and many more** that this page fails to accommodate.

Abstract

Over the past two decades, the expectation of people from a Search Engine has changed drastically. Today, people use Search Engine to accomplish task like *finding a restaurant, searching for review, finding forum solution, booking a travel plan* instead of looking up website lists. But Search Engine still serves to provide a list of web-sites on the input query. In this dissertation work, we propose a method to identify User Intention behind a search query generated by the User. This work paves the first step towards identifying and hence serving the User Task.

First we try to find a keyword representation of the queries which is used to group the keywords into keyword groups that convey the same intent. Hence, the grouping proposed not only uses the Query string but as well the User ID, Location of User, Clicked Results, etc as meta data to the query to generate a set of robust keyword group invariant to spelling and intention. This process is aided by a Wikipedia Database for reference. We then propose few Unsupervised Clustering Models to cluster the Web Search Queries into User Intent clusters. All these set of clusters are evaluated manually to assess the pros and cons of the proposed clustering models. Hence, we conclude by generating User's Intent Clustering technique and also identifying properties of an ideal clustering technique for Web Search Queries.

Contents

Chapter 1

Introduction

Web-Search Engines over a period of two decades has been one of the most fundamental portals of Internet that governs people and their choices[1]. Though Internet in theory is open for direct access, Web Search engines have stood out to be the entry portal for Internet as a whole. The reason behind this dynamics is, that like encyclopedia, but better than Encyclopedia, Web Search engine goes a step further to be the link between our thoughts and necessities from the Internet and the relevant outcomes Internet provides to us in return[20].

The key reason for popularity of certain Search Engines over the others is because of the user-friendly interface that they provide. It allows the User to present his/her needs by a list of keywords in form of query, and the underlying ranking system, spits out a list of websites that it feels will best satisfy the interests of the User. Hence, this model is used by the User to give a series of queries to the Search engine that ultimately leads to the successful completion of the Web-mediated tasks such as booking ticket, finding reviews, accessing news, etc.

Although today Search Engines have very complicated underlying machinery that provides assistance in the form of auto completion of queries, auto correction, diversification of results, the essential system is an Information Retrieval system that generates a set of web page links for the input query. Hence, we shall proceed to look at this problem from the perspective of Information Retrieval. As with IR, if the results are not satisfactory then the User refines the search keywords, and get a new set of results, and keeps on repeating until the requirement is partially fulfilled.

Therefore, we cannot look at User Search Process as a single search but it is a transaction[30] that takes place with the Search Engine where the results generated create a feedback within the User that direct the modification of the Search Query.But this search-look-refine model is not an essentially

effective model, and certainly not an efficient model. As a search engine, it fails to assist the user for his quick satisfaction, but keeps the onus on the user to identify the right set of keywords that will trigger his true intention.

As it is already evident, the reason behind this issue is that commanding such a diverse User Base of billions, search engines cannot depend on the User to find proper keywords to represent their intention. A same user intention can be represented by thousands of starkly different query strings that can generate varied results. But only a small subset of those results will be relevant for the User. Hence, the task of a search engine should include identifying and understanding the intent expressed by the user, even if the query submitted is not the best representation of that interest.

This research attempts to bridge that gap between User Intent and Search Query string with the help of Information Retrieval. The idea behind this approach is that search behaviors and patterns can be revealed by analyzing the Query Logs, that record the search queries and corresponding feedbacks of the Users[6, 25, 18, 29]. Hence, if we can identify the group of queries that represents a common behavior of search, then it completes our necessity of mapping a big set of search queries to a particular User Intention[19]. Therefore this will assist the search engine to identify the intention of the User simply by looking up the input query in the table. What remains is simply generating the best rest of results to suit that intention instead of suiting just the Query String.

To identify the real intent behind a query, we need to understand that queries submitted to Search Engine need not always be informational as opposed to general assumptions in IR problems.The general web searches can be classified into three distinct categories[6, 25].

Navigational	Queries whose intent is to reach a specific website
Informational	Queries whose intent is to acquire some information from one or more websites.The exact website does not matter till it is a valid source.
Transactional	Queries whose intent is to assist in performing some web felicitated task.

Therefore this provides us with a further necessity for Web Search Query Clustering because we can label this cluster into one of these three categories. This labeling is necessary as the service expected from the Search Engine is starkly different for each of the three categories. The primary step to provide such content specific assistance by the search engine is to identify the category of intention which can be achieved with Query Clustering.

There has been significant if not extensive previous work on this subject. Though our work intends to look at it from a different perspective where we utilize not only the User input query but as well the further feedback responses provided by the user to create a logical clustering of the data. To understand the current status of the literature on this topic, the past works can be classified into three broad techniques discussed below[20].

Time-based Time-based techniques were part of earliest work on Query Clustering that worked to identify meaningful search sessions. The basic idea behind these methods was the fact the time period between two successive queries is prime indicator if the queries belong to same set of transactions. There is assumed to be a burst of queries by an user during a session, with a time gap of no activity before a new session[28] starts.

But this technique had two major drawbacks, primarily being the fact that it can only identify sessions of an individual user. It provides no approach to connecting these sessions among different users. Hence, the keywords that are generated to identify the User Intention expressed in these query sessions are very limited to be considered a general representation of that topic[13]. Also, over the time as browsers progress, multi-tab browsing became the De-facto method of search. Hence, this leads to the situation where the same User performs two different sets of searches for two different intentions from the same browser. Hence, the search queries of two sessions are interleaved. They do not hold onto the assumption that query sessions are supposed to be separated by a gap of low activity. Hence, we cannot anymore claim to separate Query sessions just on the basis of their temporal order.

Content-based Another major tract of work approaches to exploit the lexical content of queries to identify similar queries by the same user and by different users. There are many methods proposed to address this issue to finding lexical similarity between a set of keywords associated with different keywords.

But the primary problem that this method suffers from is vocabulary-mismatch problem[27, 20] which is basically two different queries that share the same topic but do not share any common keywords between them(eg: 'Sachin Tendulkar', 'Cricket batsman'). In our research we propose a method of Wikification to circumvent that problem to some extent.

ad-hoc This is the set of works the primarily uses Statistical similarity between set of queries by the same user and by different users to find

if these queries belong to the same cluster. There are varied statistical measures[14] used for this process that includes combining time and content of query to create Query-flow graph. Also, our work uses this method where we use the information about the web links clicked by the User from the results generated by his search, and the amount of time spent by the user on those web pages as a measure of their satisfaction with the search.

This set of techniques is best suited for the analysis of this kind of problem of query clustering as it drags all the good factor of different techniques and combines them in a statistically efficient manner.

This gives us an idea of the basic problem that we intend to address in this research project. It also identifies the outline of the results that we expect out of this process. In addition, we highlight the current state of work in this field on which we will pile on our contribution. In the next section will shall proceed to give a brief overview of our process and methods that we will follow thereafter.

Chapter 2

Overview

In our research, the standard input to be considered throughout is a Search Engine Query Log of Bing Search Engine. This dataset is provided to us by Microsoft Search Technology Center, India which lists all the search queries submitted to Bing Search Engine all over the world and some associated attributes which are discussed in detail later.

The goal of this research is to identify a method that is logically and experimentally valid in identifying Search Query Clusters from the dataset. That is given the dataset, it will partition all the search queries in that dataset into certain logical clusters. The validity of those clusters is represented by the quality of User Intent they are able to identify. Also we shall continue and test the compactness of the different set of clusters generated to reach a conclusion about the validity and usefulness of our proposed process.

The primary focus of our method will be proper Preprocessing of data that will follow an Ad-hoc technique[20]. It will involve the preliminary cleaning of queries into set of valid keywords which are cross referenced with some dictionary and are useful in portraying the intent of the User behind those queries. This will be followed by lexical-based analysis of Wikification[15]. This process will intend to expand the vocabulary of this keyword groups such that we can avoid the vocabulary-mismatch problem. The large number of wikified words will help to find a common link between to search queries that may not be connected by a common keyword (eg. 'Sachin Tendulkar','cricket batsman' share many common wikified keyword including 'cricket')

The pre-processing will continue to use the temporal proximity of queries submitted by same Users to generate a more compact set of groups for keywords. This will be achieved by identifying the query session associated with same User ID. The proximity of these queries with respect to small time gaps between them will be the factor considered while grouping them as queries with similar User Intent.

As the last step of pre-processing we shall continue with an Ad-Hoc technique where we will identify the feedback of the user to the results displayed for the submitted Search Query. This feedback is essentially the URL[12] clicked by the user out the displayed results and the amount of time spend on them. We will assume the more time an User spends on an URL, more is the satisfaction of the user with that URL with respect to the search query submitted by the user that led to that URL. This, will help us to find a measure to pair groups of keywords that share same kind of satisfaction from the user for similar kind of Website Domains visited. This will be the last statistical measure used to identify groups of keywords that share a common link of intention.

Once, the pre-processing is completed, we will built a Vector Space Model that will associate each search query with a Query Vector that is representative of that query string. Henceforth, this vector will be used in our Clustering models to generate requisite query clusters out of the dataset.

As last leg of our process, we will discuss four such Clustering Models that will take the Query Vector set generated by our Vector Space Model and will generate clusters of these vectors. These clusters will in turn correspond to cluster of queries, which we intend to identify. We will point out the pros and cons of these four models and thereby the validity of the results generated by them.

This will end our process and we shall proceed to analyses the quality of the result produced in terms of Query Clusters and conclude the goodness of the solution to the problem that we intend to solve.

Chapter 3

Method Proposal

3.1 Dataset

The standard dataset used for our research is a Bing Search Engine Query Log. This is a temporally ordered set of all queries that are submitted to a Bing Search Engine anywhere across the world. This data is generally a stream of queries, along with certain captured Meta data about the user who initiated this search query and the corresponding results.

For every item in this dataset following attributes are associated:

1. **Query:** This is the actual query string given as input by the user.
2. **Time stamp:** This is the registered GMT time of the submission of the query to the search engine. It is of the format DD/MM/YYYY HH:MM:SS
3. **Temporary User ID:** To protect the identity of the user, the search engine assigns a random User ID to this User. All search queries originating from this same User is stamped with this common User ID. But this ID is temporally variable. If the user continues the transaction with the search engine for a long period of time, after a certain time period, he is assigned a new User ID to prevent User Tracking. Also, every time the User stops the transactions, the User ID is reset for him.
4. **Country:** This identifies the current country from where the User is initiating the search. This helps the Search Engine to give country specific results to the query.
5. **Language:** This records the language in which the Search Engine is being accessed. This helps the search engine to report linguistically similar results at a higher rank in the results.

6. **Location:** This identifies the Location within the country where the User is supposed to be located while initiating the queries.

7. **Clicked URL:** This is an array of URL s that are accessed by the User in response to the results displayed by the search engine in reply to his query.

8. **Clicked URL Access Time:** This is an Array of same size as the earlier attribute. This records the time spend by the User(in seconds) on that respective URL . This time period is recorded until he closes the link or follows some other trail of link from that web page to leave the URL.

The dataset provided to us covers has 100,000 similar items sorted temporally according to the Time stamp of the initiation of the search. The Query Log starts recording from 25th December 2013, 12:00:00AM and stops recording the 100,000 items at 25th December 2013, 12:00:45AM. Hence, the whole query log contains the search submitted to Bing Search Engine over the period of 45 seconds across the world.

3.2 Pre-processing

3.2.1 Bag of Words

As the queries submitted are given as keywords by the User and generally do not form a logical sentence, the relative ordering of the keywords of the query is not considered. It is assumed that the query is not a structured query, but a set of keywords upon which the User expects the results[23].
Hence, each query is converted into a set of keywords that constitute them. While converting the query into set of keywords, any kind of punctuation or unrecognized symbol is ignored and considered irrelevant.
Also, as our Language Processing of the queries is done assuming queries in English, any non-English queries are removed from the Dataset and henceforth not considered in our analysis.

3.2.2 Stop Word Removal

Once the query string is converted into a set of keywords, the task of Stop Word removal is to consider the valid keywords that are responsible to capture the intention of the User.
The Stop Words are therefore irrelevant as they are the words that are used

to bind a sentence or a phrase, such that the keywords are connected in an order.
As for our analysis we are assuming that the internal order of the keywords are irrelevant to the query intention, therefore the stop words do not convey any more information to the intention of the User.
To remove the stop words, a standard NLP Stop word list is used as a reference. All the keywords are cross referred to this Stop Word lists, and accordingly the stop words are removed from the keyword set associated with each query[23].
Some common Stop Words are: because, an, about, etc

3.2.3 Keyword List

After stop word removal, it is assumed that all the remaining keywords are essential keywords, i.e they convey certain information about the intention of the user behind the query.
Now, for further processing, a new list of keywords is created where each identified keyword after last processing is a new item of the list. Hence, this is a list of the unique keywords present in the Query Log. With each such keyword, a set of queries are identified where this keyword is used.

This list of keyword is adapted to account for small spelling variations in the keywords. Unlike, other text processing application, in our case, a vast majority of Users generate the Query List. Hence, we cannot control the quality of Users. This introduces large variations of spellings in the keywords that mean the same. This is further aggravated by high proportion of Proper Nouns in the keyword list whose spelling is not controlled.

Thus, we have a list of 10-12 different spellings for simple and oft repeated words which obviously have the same intention.Hence, the first step is to make the keyword list robust to such spelling variations. Essentially, we need to merge keywords with similar spelling into a single keygroup. Even if all weird spelling variations is not detected, the obvious misspells should be captured in this keygrouping on spelling.

We achieve the same with a metric that is defined to capture the spelling difference between two words. This is essentially used to find set of keywords that are close in spelling to one another and therefore convey the same word.

Definition 1 (Modified Edit Distance) It is way of quantifying the dissimilarity between two strings by counting the minimum number of operations(Insertion, Deletion, Modification) of characters required to convert one string into other string. The modification essentially accounts for the modified penalization to for the operations as per the following rules:

1. Penalty of 1 is imposed on Modification of Consonant
2. Penalty of 0.5 is imposed on Modification of Vowel
3. Penalty of 2 is imposed on Modification of First character if Consonant
4. No penalty is imposed for insertion/deletion of repeated consecutive character

As per this definition, any two keywords that have a Modified Edit Distance of ≤ 3 between them is considered essentially an misspell of other.

But this still does not accounts for the fact that keyword can occur as variation of the same word as per its tenses and its position in Part of Speech(eg.eat, eating, ate; ball, balls, bowling). They essentially are variations of the same word with same intent. But they cannot be grouped together by simple Modified Edit Distance as they can have large edit distances between them as they are not variations in spelling.

Hence, we borrow the concept of Porter's Stemmer[24],that given an word of English, identifies the stem of the word. Hence, a second round of matching is performed where the stems of pairs of keyword are compared with their Modified Edit Distance and thereby combined if they are under the threshold of 2.5. This generates our required keyword list where each keyword is associated with its spelling and grammatical variations.

Definition 2 (Porter's Stemmer) It is an algorithmic process to reduce inflected/derived words to their stems/roots. The stems need not be the linguistic stem, but it is required that related words map to the same stem even if its not the original root.

This keyword list is used in our further processing of Wikification to identify associate words for a keyword.

3.2.4 Wikification

This process is intended to identify a set of words that convey similar intention as to the keyword that is part of the query. The assumption behind the process of Wikification is that there are multiple keywords that express the same intention of the User. Hence, the same intention can be represented by different set of keywords for different Users.
This leads to the necessity to identify a set of words that can convey the same intention as that of the keyword. Though it is not possible to identify exhaustive set of such words, Wikification is a process that intends to identify the most common words that can be associated with the keyword.

In general text processing applications, the set of synonyms of the keyword were considered the set of words that are similar to the keyword. Hence, WordNet or similar dictionaries were used to identify the synonyms.
But this assumption does not hold true for search queries, because of two primary reasons:

- An extremely varied set of users submit search queries to the Search Engine. Hence, unlike text processing, where standard set of texts are considered as inputs, search queries consist of words that are part of urban linguistic. These words do not have any reference in the standard available dictionaries
- Search Queries, unlike text documents, consists of disproportionate amount of Proper Nouns. But no standard dictionary can provide any reference to Proper Nouns.

Hence, using synonyms to identify similar words to a keyword of the search query is not plausible. This is addressed by the process of Wikification that uses the Standard English Wikipedia Log as the reference to find similar words instead of Dictionary[20, 15]. This Wikipedia log is a database of all the English web pages of Wikipedia sorted according to their Page Title. Each web page is an article on certain topic or item.
Thus, Wikipedia logs circumvent the issues put forward by the search queries. Firstly, Wikipedia is the most comprehensive collection of Proper Nouns, any kind of proper noun referring to any person/location/object has a reference in Wikipedia. As it is peer maintained, it can also be assumed that this web article corresponding to the proper noun is most up to date. Thus, it for any proper noun keyword in the search query, Wikipedia can provide the most up to date reference of associated words that can be used to expand the search query and to identify the intention of the user.
Secondly, also as Wikipedia is updated by general public instead of an institution, new Urban Vocabulary finds quick reference in Wikipedia unlike traditional dictionaries. Hence pages defining the meaning of new word connotation is created as soon as the word starts to trend, and deleted once the words become obsolete. This helps in providing a reference to this new urban vocabulary to connect them to Standard English Words that provide closes definition of them.

WikiLinks List

The Wikipedia logs used for our analysis are in form of a WikiLinks list. All the Wikipedia pages that are relevant to our text processing requirements

are stored in form of an alphabetically sorted list.
Hence, this list has around 57,16,808 items, each corresponding to a Wikipedia Page sorted according to the title of the page. Now, each Wikipedia Article in turn refers a set of other Wikipedia Articles in its description. These articles referred are relevant articles that are necessary for complete information of the parent article. Hence, we can consider these articles as the references of our Parent Article.
This information is captured in our WikiLinks list, where for each Wikipedia web page listed is associated with a set of Wikipedia web pages, that have an outgoing web link from this Wikipedia page. Hence, each Wikipedia page has outgoing links to a small subset of 57,16,808 other Wikipedia pages, which is captured in this list.

Wikification of Keywords

Given a keyword List and a WikiLinks List, the Wikification of keywords is achieved by the following algorithm:

Algorithm 1 An algorithm to identify associated Wikipedia topics to a keyword

for all item in keyword List **do**
  Search matching WikiLinks item with same title as keyword, using Dictionary Search
  if Matching Wikilinks Item Not Found **then**
    Identify a WikiLinks web page, which has a smallest Modified Edit Distance* from the Porter's Stem of the keyword
    if $SmallestEditDistanc \leq 1.1$ **then**
      Associated WikiLinks web page is considered a match
    else
      keyword is flagged as 'no match found' and Continue to next keyword
    end if
  end if
  For the selected WikiLinks page, outgoing links are identified
  if $|OutgoingLinks| = 1$ **then**
    along with this outgoing link, Web page titles of the 2nd level outgoing links is also associated with the keywords
  else
    all the outgoing link Web page titles are associated with the keywords
  end if
end for

Thus, each keyword of the keyword list belongs to either of two categories

- A matching Wikipedia page has been found, and this keyword is associated with a set of Wikipedia Page Titles that are considered to be the associated topics to that keyword.
- No matching Wikipedia page is found, and hence this keyword cannot be associated with any topic. This occurs, mainly due to these following reasons
  - High degree of spelling mistake by the User in the Keyword
  - Multiple merged keyword, forming a nonsensical word
  - An acronym that has no direct reference in Wikipedia

Let the set of keywords that has been identified be known as $keyword_{found}$. The remaining set of keywords are denoted as $keyword_{notfound}$

Keyword Similarity Graph

As our intention is to identify set of keywords that convey the same user intention, we need to group the keywords according to the intention conveyed. The assumption behind the process of Wikification is that as the Wikipedia topics associated with each keywords identifies all the possible contexts that keyword can be used.
Hence, if two keywords have a considerable overlap of similar contexts in form of common Wikipedia topics associated with it, we can claim that these two keywords share similar intention. But the challenge is to identify a measure of this overlap.
For our case, we can represent this keywords as a Complete Graph[4], where the edge weight between any two keywords(represented as nodes) is a measure of the extent of overlap of their Wikipedia Topics.This graph can be represented as follows:
$Graph : KeywordSimilartityGraph : G_{WikiSimilar}$
$Nodes(N) : keyword_{found}$
$Edges(E) : (u, v) : \forall u, v \in keyword_{found}$
$EdgeWeight : W_{u,v} = \sum_{i \in \{u_{WikiLinks} \cap v_{Wikilinks}\}} \frac{1}{WikiLinkCount(i)}$
$WikiLinkCount(i) = Numberofincomingreferencesto Wikipediaarticlei$

Thus, each edge has a weight corresponding to all the common Wikipedia Web pages shared by the pair of keywords. Also, the edge weight depends of the factor $WikiLinkCount(i)$ such that has following properties:

- if $WikiLinkCount(i)$ is high for a Wikipedia Article i, that signifies that many Wikipedia articles refers to i. Hence, i is a generic topic. Eg: Food, Medicine, Country are generic topics
- if $WikiLinkCount(i)$ is low for a Wikipedia Article i, that signifies i corresponds to a niche topic, that is not referred by many articles, hence contains more information. Eg: Oscar 2012, Indian Cricket Team, etc

Thus, for a pair of keywords, if they share Wikipedia Topics that are niche, they are more closely related with each other and hence get a higher edge weight that keywords sharing generic Wikipedia topic. Therefore the **Keyword Similarity Graph** generated captures the relative similarity between every pair of keywords from the set $keyword_{found}$

Keyword Grouping using Markov Cluster Algorithm

Definition 3 (Markov Cluster Algorithm) is an unsupervised cluster algorithm for graphs based on the simulation of stochastic flow in graphs. This is based on the postulate that natural groups in graphs have property such that if a Random Walk in the graph visits a dense cluster will likely not leave the cluster until many of its vertices have been visited.[9]

Given our weighted graph **Keyword Similarity Graph**, we intend to use MCL Algorithm to identify such dense cluster of keywords in the graph. This keyword groups identified will have a proximity due to their high edge weights. Hence, as the edge weights are due to the Wikification of the keywords, in turn we will get a partition of the set $keyword_{found}$ into keywords group that share the same intent on virtue of the meaning of the keyword that constitute them.

Algorithm 2 MCL Algorithm[9]

Input: Weighted undirected Graph Matrix, power parameter e, inflation parameter r
Add self loops of Unit value at each node
Normalize the matrix across each column, to generate a Transition Matrix. Edge Weight is equivalent to probability of transition from one node to another.
Expand by taking e^{th} power of the matrix
Inflate my taking the inflation of resultant matrix with parameter r
Repeat last two steps until steady state is reached
Interpret resulting Matrix to discover clusters

Definition 4 (Inflation) Given a matrix $M \in \Re^{nXn}$, $M \geq 0$ and a real non-negative number r, the matrix resulting from rescaling each of the columns of M with power coefficient r is called $I_r M$.

$$(I_r M)_{pq} = \frac{(M_{pq})^r}{\sum_{t=1}^{k}(M_{tq})^r}$$

This inflation operator is responsible for both strengthening the strong currents and weakening the weak currents in the transition matrix. The extent of the effect is controlled by the parameter r.

Thus, the algorithm stops when a steady state is reached due to the following operations[9]:

Expansion Responsible for allowing flow to connect to different regions of the graph

Inflation Responsible for strengthening and weakening of currents

To interpret the clusters out of the steady state transition probability matrix, the vertices that have same columns are grouped together.
Thus, the number of clusters is equal to the number of unique columns in the steady state transition matrix, and each set of vertices with similar columns are elements of the same group.

For example:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the above matrix, partition of four vertices will be as $\{1\}, \{2, 3\}, \{4\}$.

Accordingly, we can find a $keygroupWiki_{found}$ which is a partition of $keyword_{found}$ such that keywords that are similar with respect to the Wikipedia topics are grouped together to signify common User Intention.

Wikipedia Based Keyword Groups

After the last section, we get a partition of the set $keyword_{found}$ that forms group $keygroupWiki_{found}$. Now all the remaining keywords, for whom no similar words could be identified from the Wikipedia Log, are added as separate entries to the current keygroup.

This retains all the information of the User Queries, in hope that in further steps of pre-processing, some common criteria can be identified to merge these keywords in logical groups. Thus, the final list generated is denoted as $keygroupWiki$.

3.2.5 User ID based Keyword Grouping

One of the primary meta data captured in the User Query log, is the temporary User ID. As mentioned in the dataset, the User ID is randomly assigned to an User and is temporally variable. Thus, there is two following factors that holds true for the User ID.

- A new User ID is assigned to the same User every time the User starts a new session with the Search Engine.
- If the user continues the same session with the Search Engine for a long period of time. The user ID is reset after a certain buffer period

Considering the aforementioned two conditions, we can assume that the sequence of search queries recorded under a common User ID as session separated. Thus, they have been requested by the User in a single session, over a stipulated period of time.

Also, the fact that our whole dataset spans a period of 45 seconds. The time gap between the first and last query by the same User ID is less than 45 seconds.

Thus we can safely assume, that all the queries submitted by the same User ID, are generated out of a single Intention by the User[28, 20]. Hence, they share the common intention of the same User. Hence, any this is a strong similarity criteria between the keywords that constitute the query.

User ID List

Going back to our initial dataset of Search Query Log, each item i.e a query in that list, was associated with an attribute for the Temporary User ID, the significance of which is discussed earlier.

Given this Search Query List, we create a User ID List, that is sorted according to the unique User ID present in the Query List. Hence, each item in the list is an User ID, and associated with that User ID is the set of all queries generated from that User ID.

In out sample dataset, the following statistics were observed regarding the query generation of the User ID:

1. **Average Number of queries** generated per User ID is around 1.7 Queries
2. **Maximum number of queries** generated by a User ID is 6
3. Around 55% of the User ID generates only 1 query during the session

Thus, we now have a User ID List generated from our Search Query Log.

Modification of Wikipedia based Keygroups upon User ID

In the last section, we have identified a keygroup List $keygroupWiki$, where each item was a group of keywords that convey the same intent due to the common Wikipedia Topics they share.

As we now have the added information of the User ID List, this information can be considered to modify $keygroupWiki$. The keyword groups can now be further merged to generate modified keyword groups.

The assumption behind this merging is that as all the keywords generated from a single User ID convey the same intent. Hence, if two keygroups have considerable number of keywords generated by a common User ID, both the keygroups convey same intent according to that user.

Therefore, we merge these two keygroups to generate a large keygroup set that satisfies the need to identify a common User Intent.

Merging Criteria

Given a set of keygroup $keygroupWiki$ where every item has a set of keywords, and associated set of User ID that generated those keywords, if I is an item of the set, let us define $UserIDNeighbour(I)$ as follows:

$\forall I \in keygroupWiki, UserIDNeighbour(I) = k$

$k \in keygroupWiki \; \& k \neq I \; \& \; |I_{UserID} \cap k_{UserID}| \; is \; maximum \; \forall k \in keygroupWiki$

It is evident from the given criteria that,

$UserIDNeighbour(I) = J \Longrightarrow UserIDNeighbour(J) = I$

Hence, merging of keyword groups is reduced to identifying such pair of keygroups that has maximum overlap between them. Each keygroup is merged with its identified pair, which satisfies the Merging Criteria.

If there is no identified pair for a keyword Group, no merging is done for the same.

User ID based Modified keyword Groups

Thus, after the merging is performed for all such pairs identified in the keygroup set $keygroupWiki$, we get a new keygroup set that has at most as many items as the aforementioned keygroup List. Hence, this keygroup list, $keygroupWikiUserID$ is more compact version of the keyword groups that share common User Intent.

3.2.6 Visited URL based Keyword Grouping

Up to this point, the process of identification of User Intent was solely based on the queries submitted by the User to the search engine. We either analyzed the individual queries submitted, or the set of queries submitted by a common user to generate groups of keywords that convey the same User Intention across demography.

But the last crucial meta data captured in our dataset are Clicked URL and URL Access Time[12]. Both of these attributes are a feedback to the results generated by the search engine upon the submitted search query. Hence, any positive response reflected by this feedback gives a sure mark of identification that the User Intention is satisfied.

Therefore, we can assume that our last level of merging of keyword groups need to depend on these two attributes. As these attributes quantifies the degree of satisfaction of the User upon the access time, they provide us a direct way to identify a Merging Criteria for the same.

Identifying Web Site Domain

In our original dataset of Search Query Log, each item is a search query that has an associated attribute of Set of URL visited by the user is response to the results of his search query. In general, the number of items in this list of URL s varies from 0 to 5.

For the given dataset, we parse this individual URL associated with each query, to identify the Web Site Domain of the URL. Hence, we generate a Web Site Domain list, that has all the unique Web Site Domains visited by the Users as part of the dataset.

Website Domain List has two attributes, the name of the Domain and number of queries in whose response this website was visited. Some examples of domains are Google,Wikipedia,facebook,imdb,rediff,etc

Access Time based Domain Pruning

In the Search Query Log, every identified website Domain, is associated with a access time. Now, this set of website Domain associated with each query are pruned in two steps:

- **Generic Domain pruning:** With the help of our Website Domain List, we can rank the domains according the number of queries in whose response those domains were visited, in descending order. Hence, the top domains are generic domains like, Google, Wikipedia, yahoo which fails to inform anything about the intention behind the query that gets

satisfied here. Therefore, such generic domains (manually verified to be the top 50 identified domains) are blacklisted and removed from the Domain list for every query. The leftover domains like imdb(film), soundcloud(music), npr(music), thetelegraph(news) are information specific domains, hereby retained.

- **Access Time based pruning:** In the next level, of the remaining set of website domains associated with each queries, we remove the domains whose corresponding access time is less than the median access time. This is based on the assumption that as the median access time is around 3 sec, any domain that has a lower access time failed to hold the attention of the User. Therefore that domain was not satisfying the User Intent, and needs to be pruned. If all the domains for a certain query gets pruned under this criteria, we can classify such query as an unsatisfied query.

Thus, after the aforementioned steps, we will get a pruned Search Query list, where each query is associated with a selected few domains and their corresponding Access Time. We can hereby assume that these Website Domains were satisfying the User Intention as it held their attention for a considerable period of time. Also, they are niche domains that support a narrow range of topics, so that we can use them to further modify our keyword groups into stronger partitions.

Modification of Keygroups upon Website Domains

We will use a method similar to that of previous section to merge keygroups upon a common User Intent highlighted by the Website Domains they share among themselves.

The keygroup List $keygroupWikiUserID$ generated in the last section consists of a set of keywords of common User Intention along with the associated queries that generate those keywords. Hence, every item of $keygroupWikiUserID$ can be associated with a set of Website Domains from the pruned Search Query List depending upon the queries that constitute them.

Thus, we can identify a measure of the degree of similarity upon the common domains shared by any two items of the keygroup List. This measure is representative as follows:

$\forall i, j \in keygroupWikiUserID$:

$DomainSimilarity(i,j) = \sum_{k \in Domain_i \bigcap Domain_j} (1 - \frac{|AccTime_i(k) - AccTime_j(k)|}{max(AccTime(k))})$

The validity of the above measure for our analysis is highlighted as follows:

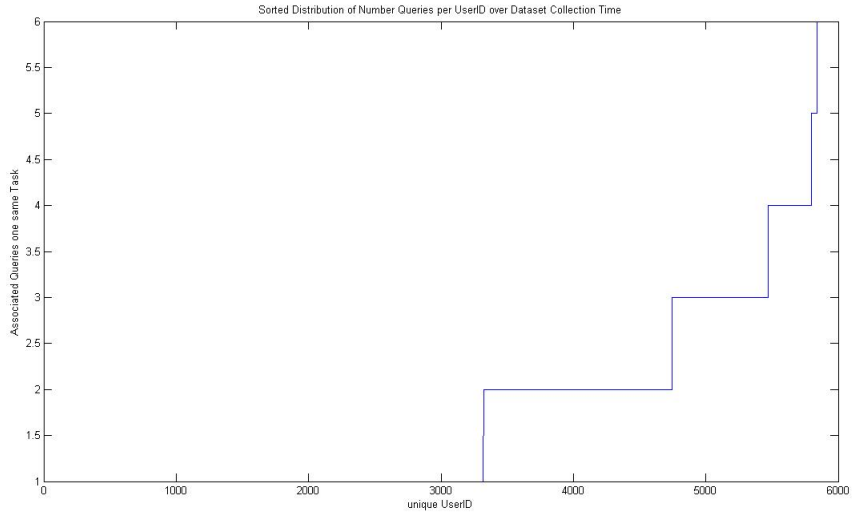

Figure 3.1: Distribution of Number of Task per User ID

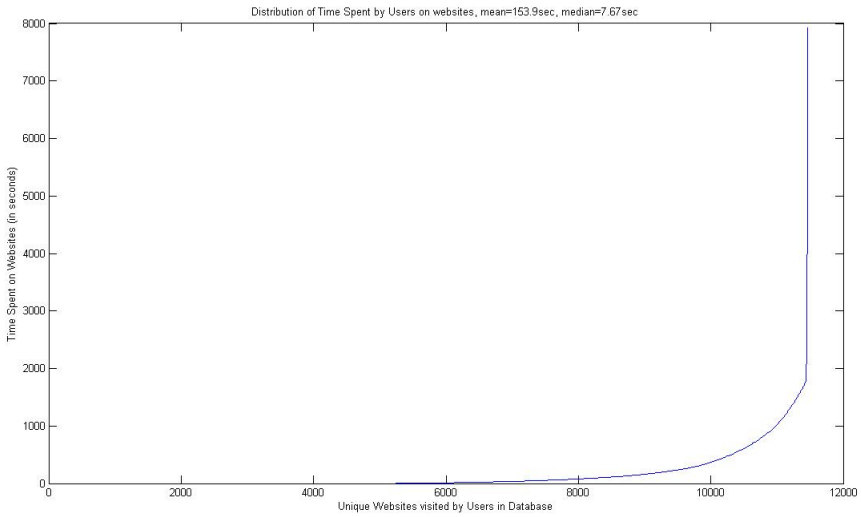

Figure 3.2: Distribution of time spent(in seconds) per Website

- The *DomainSimilarity* measure is dependent on all the individual domains that are common to both the keygroups
- The measure is low if the difference in Access Time for a particular domain is starkly high for two keygroups, as it indicates that this domain is not satisfying one keygroup users as much as it is satisfying the other.
- The measure is maximized if the Access Time is similar for both the keygroups. As we have already pruned out Domains with low Access time, this indicates that both the keygroups provide similar degree of satisfactory results to the User for both keygroups

Thus, we have a measure between any two items of $keygroupWikiUserID$ which can be used to merge appropriate pairs of keygroups to identify a final partition of the keywords upon User Intention.

Merging Criteria

The merging criteria is exactly similar to the process undertaken in last section to merge upon User ID. Given a set of keygroup $keygroupWikiUserID$ where every item has a set of keywords, and associated set of Domains accessed by those keywords, if I is an item of the set, let us define $DomainNeighbour(I)$ as follows:

$\forall I \in keygroupWikiUserID, DomainNeighbour(I) = k$

$k \in keygroupWikiUserID \; \& k \neq I \; \& \; DomainSimilarity(I, k)$ *is maximum* $\forall k \in keygroupWikiUserID$

Thus, upon the similarity measure *DomainSimilarity* identified earlier, we propose this merging criteria, which identifies pairs of keygroups that are best suited to merge.

It has to be noted that only a few of the keygroups qualify the Merging Criteria, only if the share a large degree of varied common domains. Thus, a merging upon this criteria is justified as it only ends up merging those pairs of keywords that are logically of same User Intention.

We hereby get a keygroup List which we will denote as $keygroupFinal$. This keygroup is at most as large as $keygroupWikiUserID$. Therefore, we can claim that it is a much more compact grouping of the keywords into logical keygroups identified by a common User Intention.

This keyword groups of $keygroupFinal$ will henceforth be used as the primary dimensions for the Vector Space Modeling of Search Queries for our Unsupervised Learning framework.

3.3 Vector Space Model of Search Queries

Vector Space Model(VSM) is one of the most fundamental techniques in Information Retrieval systems since inception[3, 2, 26]. One of the fundamental advantage of this method is to generate a relevance vector for any kind of heterogeneous document with respect to the features of the classes it is supposed to be classified.

In our case, we do not deal with the standard model of classifying documents into fixed number of classes, but the analogy prevails. We will in turn define a Query Vector that irrespective to the type and the length of the input query string from the User,will generate a fixed dimension vector that is representative of the query.

The basic idea behind the process depends on the factor that during the course of the Learning, the input Dataset remains constant. Hence with respect to the input dataset, and all the pre-processing performed to get structured data, we can represent any User Search Query relevant to that Search Query Log with a Query Vector, which is the Vector Space Model of that set of queries.

There are two primary types of Vector Model considered in out analysis and essential features of both are discussed here onwards.

3.3.1 Boolean Vector Model

In Boolean Vector Model, the vector space is dependent on the $keygroupFinal$ list generated in the last section. Thus, each item, which is a set of keywords, in that list forms an independent dimension in the Boolean Vector Space.

Here, the number of dimensions is equal to number of unique keyword groups in $keygroupFinal$. Hence for every input Search Query, the following algorithm is followed to generate the Boolean Query Vector

Algorithm 3 Algorithm to generate Boolean Query Vector

input: search query string
Initialize a Zero Vector $\overrightarrow{BoolQ}$ of size $|keygroupFinal|$
for all words in the Search Query String **do**
  Find X : $word \in X \ \forall X \in keygroupFinal$
  if X is found **then**
    Assign $\overrightarrow{BoolQ(Location(X))} = 1$
  end if
end for

Thus,for all valid keywords we the vector $\overrightarrow{BoolQ}$ has unary marked at

appropriate location. The vector can only take values of 0/1. Hence, it is a Boolean Vector Model.

Though the model is representative of all the keywords that constitute the Search Query, there is no captured information about the relative importance of the keyword. All the keywords in the case are assigned equal importance and their presence is denoted by 1. This, drawback is addressed in the next Vector Model

3.3.2 Term Weighted Vector Model

Term Weighting is a modification of Boolean Model that address the drawback mentioned above by accounting for the relative frequency of the appearance of keywords and keygroups in the dataset. Hence, it allows to associate a real value with each keyword that appears in the Input Search Query that signifies this relative priority of the keyword.

The Term Weighted Vector Model that we consider here is based on **keygroup frequency** which is nothing but the number of queries in the Search Query Dataset, any of keyword from the keygroup appears. Hence, each keygroup item of $keygroupFinal$ is associated with an unique value for keygroup frequency.

With respect to this above variation of the concept, the algorithm to generate a Term Weighted Vector for an input Search Query is as follows:

Algorithm 4 Algorithm to generate Term Weighted Query Vector

input: search query string
Initialize a Zero Vector $\overrightarrow{TermQ}$ of size $|keygroupFinal|$
for all words in the Search Query String **do**
  Find X : $word \in X \ \forall X \in keygroupFinal$
  if X is found **then**
    Assign $\overrightarrow{TermQ(Location(X))} = Weight(X)$
  end if
end for

Here, $Weight(X)$ is the function that decides the weight associated with the keygroup X as per the keygroup frequency as follows[22]:

$Weight(X) = \log_2(n/keywordFreq(X))$

$n = Total\ Number\ of\ queries\ in\ the\ Database$

$keywordFreq(X) = No.\ of\ queries(out\ of\ n)\ where\ keyword\ from\ X\ appears$

Thus, the vector generated for any input Search Query has real non-zero values for the dimensions corresponding to the keygroups which contribute keywords for the Query.

Also as per the definition of the weight vector, the more frequent the keygroup occurs for the Query dataset, the smaller is the weight.Hence, the relative importance of the keygroup contributes to this vector space model.

For our analysis here onwards, we consider predominately the Term Weighted Vector Model as it gives marginally better outcomes for certain methods of Unsupervised Clustering discussed later.

3.4 Unsupervised Clustering Models

As we now have a Vector Space Model, that successfully generates Query Vectors for input Search Queries, on the basis of last two sections we can claim that we currently have access to a Dataset of Query Vectors, where each item is a m-dimensional vector, where m is number of keygroups identified earlier by us.

Thus, now we have the standard input required for our Unsupervised Clustering Model, that tries to identify valid cluster partitioning of the Query Vector Dataset on the relative cohesiveness of the Query Vectors. All these models that will be discussed, consider the Query Vector Dataset as the standard input, and provides us an partition of this Dataset as output.

Once, the partition is available for this dataset, we can trivially identify the partition of the Search Query Log where each query has an one-to-one relationship with their corresponding Query Vector. Therefore the output of this Clustering Models are our required Search Query Clusters.

For our analysis, we consider 4 different Unsupervised Clustering Models, and run our dataset independently on all of them to get 4 set of results for comparison. The Clustering Models hereby selected are as follows:

1. **DBScan:** Density based Spatial Clustering is an hierarchical clustering model, which incrementally builds the clusters without supervision[11]

2. **KMeans:** k-Means Clustering is a Partition based clustering Model, that reassigning the data points to different partitions until it identifies a stable partition[21, 17, 10]

3. **Principal Direction Divisive Partitioning:** It is divisive clustering method that keeps partitioning the dataset by using its Principal Directions until steady state is achieved[5]

4. **Spherical Principal Direction Divisive Partitioning:** It is modification of the last method, which is same apart from the fact that we consider the top two orthogonal Principal Directions to find partition of the cluster[8, 7]

We will from here onwards discuss the basic structure of these Clustering Models and their corresponding algorithms.

3.4.1 DBScan Model

DBScan Model of clustering is based on density-reachability of a cluster. It tries to identify certain core data points in the Vector Space such that it has a high density of points surrounding them. These core points are connected with the density-reachable points around it to found a cluster[11].

Likewise the points that are not accessible from this core points by successive density reachable points are the Noise points for this cluster. These noise points in turn can belong to some other cluster or can form a cluster of their own with single element at worst case.

A cluster, which is a subset of the points of the database satisfies two conditions:

- All points within that cluster are mutually density-connected
- If a point is density reachable from a cluster point, it is marked for that same cluster.

Hence, by the above description it is evident that DBScan Model depends on the definition of two primary concepts, that is, density and neighborhood. Hence, DBScan required two parameters to be specified to the algorithm.

The first parameter is ϵ which is the measure of the radius of the neighborhood of a point to be considered to measure its neighborhood-density.

The second parameter is minPoints which is the minimum number of points required to be present in the neighborhood so that it can be considered as a dense region.Otherwise the region shall be labeled as noise.

The algorithm of DBScan works on the same method discussed above where it tried to find a Core point to a cluster. Once the core point is identified, the cluster grows by finding density-connected points in succession in the neighborhood. This process stops when we reach the boundary points of that clusters and the neighbors of the boundary point turns out to be less dense than our required criterion. This algorithm is expressed as follows

As mentioned earlier, the algorithm is straightforward iterative labeling of points into certain cluster depending on the cluster label of its neighborhood points and the density of its neighborhood. Thus, ϵ and minPoints plays an important role for the shape of the clusters formed.

Though, there is no standard method to identify these parameters, it is essentially database dependent. For our analysis, as supported by other

Algorithm 5 DBScan Algorithm[11]

DBSCAN(Database,ϵ,minPoints)
Initialize C=0
for all Unvisited Points P in Database **do**
  NeighbourPts=regionScan(P,ϵ)
  if $sizeof(NeighbourPts) < minPoints$ **then**
    mark P as Noise
  else
    C=C+1 (New Cluster Number)
    expandCluster(P,NeighbourPts,C,ϵ,minPoints)
  end if
end for

expandCluster(P,NeighbourPts,C,ϵ,minPoints)
add P t cluster C
for all points P' in NeighbourPts **do**
  if P' is not visited **then**
    mark P' as visited
    NeighbourPts'=regionScan(P',ϵ)
    if $sizeof(NeighbourPts') \geq minPoints$ **then**
      $NeighbourPts = NeighbourPts \cup NeighbourPts'$
    end if
  end if
  if P' is not member of any cluster **then**
    add P' to cluster C
  end if
end for

regionScan(P,ϵ)
return all points in P's ϵ-neighbourhood (including P)

similar Text Processing applications that implement DBScan, the ϵ value is selected by following criteria.

Suppose, for every point P in the dataset, Border Neighbor of P is the i^{th} Neighbour such that:

$BorderNeighbour(P) = i^{th}\ closest\ neighbour\ of\ P\ where$

$dis(P, Neig_{i+1}) - dis(P, Neig_i) \geq dis(P, Neig_{k+1}) - dis(P, Neig_k) \forall k \neq i$

Thus, Border neighbor is that neighborhood point beyond which the next neighbor point is farthest away. Hence, if there is a proper cluster present in the dataset, the Border Neighbor point should be have a distance of at most the diameter of the cluster from the point P. The distances considered in all these cases are the Euclidean Distances between the vectors identified.

Therefore on an average $dis(P, BorderNeighbour(P))$ is between the diameter/2 to diameter of the cluster where P belongs. The minimum occurs when P is the core point of that Cluster, and the maximum occurs when P is the outer edge point.

On the basis of this assumption, we set ϵ as

This is a proper neighbor distance to be scanned to identify all dense cluster points for the point P under consideration.Similarly, once this is identified, we can find out the number of points in the ϵ Neighborhood for all points P of Dataset.

Experimentally, we have found that a good value to be set of minPoints is as follows

$minPoints = median(regionScan(P, \epsilon)), \forall P \in Dataset$

This is the median number of points that exist in the $\epsilon neighbourhood$ of all the points in the dataset. Any neighbourhood containing more than this value is considered to be a dense cluster by our DBScan algorithm.

This completes all the criteria required by the DBScan Model to run the algorithm for the Query Vector List provided to identify the Search Query Clusters.

3.4.2 KMeans Model

K-Means has been the standard Clustering Model for a long time. The traditional version of KMeans performs the clustering my reassigning the Points from one cluster to another depending on its distance from the mean of those clusters. Thus, it is not a incremental method, but the number of cluster is known from the beginning[21].

The stopping criteria of KMeans is when the movement of the means of the k clusters are minimized, that is the reallocation of points from one cluster to another is reduced. KMeans is a self correcting Model for clustering as it starts with a random guess about the k clusters and its constituent points.

As the iteration progresses, the algorithm refines its guess to reach the stable partition.

Given a set of vectors $X = x_1, x_2 \cdots, x_d$ in an Euclidean Space $\Re^m$ we will henceforth denote the centroid of the set as follows[17]:

$m(X) = (1/d) \sum_{i=1}^{d} x_i$

Now suppose $\{\pi_j\}_{j=1}^{k}$ is a valid partition of the set of vectors X, then the corresponding centroids for each of those partitions will be denoted as $m_1 = m(\pi_1), m_2 = m(\pi_2), ..., m_k = m(\pi_k)$.

The last factor that is important t decide on the stopping criteria of the KMeans algorithm is the Quality of a given Partition. It is defined as the sum over all the point's euclidean distance to their respective centroids upon the partition they belong[10].

$Q_2(\{\pi_j\}_{j=1}^{k}) = \sum_{j=1}^{k} \sum_{X \in \pi_j} (||x - m_j||^2)$

Though this absolute value makes no sense, the idea is reduce the this value over successive iterations of the KMeans to make the clusters compact and logically valid. Thus, the difference between the Q_2 value of successive iterations gives us idea about the effectiveness of this iteration for the overall clustering.

In our traditional KMeans clustering, at every iteration the set of k centroids are calculated and accordingly, the points are reclassified into partitions whose centroid are closest to them. This naturally changes the centroid of that partition to a new position, which is re-calibrated in the next iteration and there on it continues.

This process is represented such that for $x \in \pi_i \subseteq X$ the centroid nearest to x is denoted as $m_{min(x)}$. Therefore[10]

$||x - m_{min(x)}|| \leq ||x = m_l||, \forall l$

Thus, the new partition that is generated from the old partition of $\{\pi_j\}_{j=1}^{k}$ is,

$nextKM(\{\pi_j\}_{j=1}^{k}) = \{\pi_j^{'}\}_{j=1}^{k}$

$\pi_i^{'} = \{x : \ min(x) = i\}$

Though this traditional KMeans[21, 10] fulfills the criteria of lowering the Quality function in each successive iteration, to reach a minima, it suffers from a major drawback, where it gets stuck at a local minima for some clinical cases. Thus, the final clustering generated is not the natural clustering expected from the algorithm.

Therefore, to avoid such scenarios, after the batch KMeans reaches its minima, a further step of incremental KMeans is applied to verify it is its local minima. If so, the incremental KMeans is used to move the partition from the local minima towards the natural partitioning expected from the Clustering Model.

Incremental KMeans is a special case of KMeans, where in every iteration, instead of reassigning all the points to new clusters according to their distance from centroids, a single vector is identified whose movement to new partition will cause maximum reduction to the Quality function. Hence, only this vector is reassigned to the new partition and the means the recalculated. This is known as First Variation.

Therefore $nextFV(\{\pi_j\}_{j=1}^{k}) = \{\pi_j^{'}\}_{j=1}^{k}$ is the movement of one vector x from partition π_m to partition π_n as per the above mentioned criteria[17].

As it is evident, the process is very computationally intensive and therefore is only performed after the traditional KMeans reach a steady state minima to avoid clinical cases.

The algorithm for KMeans accounts for these two steps mentioned earlier. It takes as input tol1 and tol2 that are the tolerance values of difference of Quality function between successive iterations to decide upon the termination point of the algorithm

Thus, the algorithm starts with a random partition and using the traditional KMeans algorithm, reaches a steady state where successive iterations does not improve the quality of the clusters beyond tol1.

After that the First Variation step, tries to refine the partitions by finding individual vectors that are wrongly classified and cannot be identified earlier. This continues until the tol2 level is reached terminating the algorithm.

Hence, this algorithm generates k partitions of the Search Query Log as per the Query Vector set given as input for the Dataset.

3.4.3 Principal Direction Divisive Partitioning Model

The third Model that is considered in our analysis is a useful for clustering in Text Processing as it takes advantage of the sparsity of the Query Vector matrix. It refrains from using any Distance measure on the dataset for the clustering and in higher dimension, the distance measure creates non cohesive clusters.Instead, this method tries to project the data into a lower dimensional plane to proceed to partition the data in that plane.

The model proceeds by dividing the entire dataset into two cluster using the Principal Direction. Each of the division is further subdivided into clusters using the same process recursively. This continues until we get the required number of clusters or some stopping criteria is met to halt the process.

The basic idea behind this process is the fact that though partitioning a set of vectors in $\Re^n$ is very tough, but if $n = 1$ then the problem is in 1-dimension. Hence, it is as simple as finding the biggest gap between to

Algorithm 6 An Algorithm for KMeans Clustering[17]

Input: Dataset, tol1,tol2

Start with an arbitrary partition $\{\pi_j^{(0)}\}_{j=1}^k$. Set the index of iteration at t=0.

repeat

Generate the partition $nextKM(\{\pi_j^{(t)}\}_{j=1}^k)$

if $Q_2(nextKM(\{\pi_j^{(t)}\}_{j=1}^k)) - Q_2(\{\pi_j^{(t)}\}_{j=1}^k) \leq tol1$ **then**

set $\{\pi_j^{(t+1)}\}_{j=1}^k = nextKM(\{\pi_j^{(t)}\}_{j=1}^k)$

$t = t + 1$

$exit = False$

else

$exit = True$

end if

until $exit = True$

repeat

Generate the partition $nextFV(\{\pi_j^{(t)}\}_{j=1}^k)$

if $Q_2(nextFV(\{\pi_j^{(t)}\}_{j=1}^k)) - Q_2(\{\pi_j^{(t)}\}_{j=1}^k) \leq tol2$ **then**

set $\{\pi_j^{(t+1)}\}_{j=1}^k = nextFV(\{\pi_j^{(t)}\}_{j=1}^k)$

$t = t + 1$

$exit = False$

else

$exit = True$

end if

until $exit = True$

successive points along the dimension and partition the data across that gap.

Therefore this Model tries to identify a line, such that all the vectors can be projected on that line. Thus, the vectors are reduced from n-dimensions, to their respective projections in 1-dimension. Henceforth, the partitioning is a simple task.

But the problem with this approach is to find the best line, so that the projection of the vectors on the line is representative of the true distribution of the vectors in n-dimensional space. Though there is no single line that can truly preserve all the information captured by the vector space, that the line that maximizes the variance of the projections is the best one-dimensional approximation of an n-dimensional set[5].

This line is defined by the eigenvector of the covariance matrix C corresponding to the largest eigenvalue. Since, C is symmetric and positive semidefinite, all the eigenvalues $\lambda_i, i = 1, 2, ..., n$ of the matrix are real and non negative., such that $\lambda_1 \geq \lambda_2 \geq .... \geq \lambda_n \geq 0$.

Also since we are using only λ_1 corresponding eigenvector to find the line of projection, the fraction of information preserved under this case is $\frac{\lambda_1}{\lambda_1+\lambda_2+...+\lambda n}$. Though the amount of information preserved is a fraction, it is high preservation in case of this sparse matrix.

Algorithm 7 Algorithm of PDDP[5]

Input: Dataset, k(number of clusters required)
Intialize $c = 1, cluster(c) = Dataset$
while $c < k$ current number of clusters is less than required **do**
  for $i = 1..c$ **do**
    $cov(i) = covarianceMatrix(cluster(i))$
    $peg(i) = PrincipalEigenvector(cov(i))$
    project $cluster(i)$ on $peg(i)$ denoted by $proj(i)$
  end for
  identify $i^{'} \leq c$ such that $proj(i^{'})$ is best for Maximum Gap Partition
  Partition $proj(i^{'})$ and correspondingly $cluster(i^{'})$ into $\{cluster(i^{'})_1, cluster(i^{'})_2\}$
  $cluster(i^{'}) = cluster(i^{'})_1$
  $cluster(c+1) = cluster(i^{'})_2$
  $c = c + 1$
end while
Output: $Cluster(1), Cluster(2), ..., Cluster(k) : \bigcup_{i=1}^{k} Cluster(i) = Dataset$

Thus, this algorithm generated k Clusters which is a k-way partition

of the Search Query Log successively using the Principal Direction of the chunks created in the process. Certain studies claims that for Text Processing applications this methods of clustering marginally improves upon the result given by KMeans Algorithm.

3.4.4 Spherical Principal Direction Divisive Partitioning Model

This model is a modification of the earlier PDDP model to account for two orthogonal Principal Directions to project the vectors and thereby partition them. Instead of a line, the projections are now onto a Unit Circle that needs to partitioned appropriately[8].

Similar the earlier approach, we want to find the best two-dimensional approximation of the Query Vector set, which is the plane that maximized the variance of the projections. This place is defined by the two eigenvectors of the covariance matrix C corresponding to the largest eigenvalues λ_1 and λ_2.

The preserved information under this projection is $\frac{\lambda_1+\lambda_2}{\lambda_1+\lambda_2+...+\lambda n}$. It is to be observed that this quantity is almost twice as much as the information preserved under PDDP and hence assures us of a better performance in clustering.

Similar to the last algorithm, this also successively partitions the Dataset is smaller and smaller chunks until it identifies k appropriate clusters as outcome. Thus, given the Search Query Dataset, this algorithm gives k query clusters as our output.

But unlike Maximum Gap Partition, Unit Circle Partition is not straightforward. In this all the points are plotted on the circumference of an unit circle on the plane defined by the Principal Directions. Hence, partition of the points indicated to find a chord, that cuts the unit circle into two parts and along with partitions the points on the either side of the partition as separate sets.

We shall now define a measure for the quality of the partition that will be used by our algorithm of Unit Circle Partition to identify the the best partition. This measure is a spherical objective function that needs to be maximized for an optimal partition[7]

$Q_s(\{\pi_1, \pi_2\}) = ||\sum_{z\in\pi_1} z|| + ||\sum_{z\in\pi_2} z||$

The following algorithm will try to find partition of the set of unit circle projections $projUnit(i)$. Let $Z = projUnit(i)$

The partition given as output to this algorithm is the Unit Circle partition of the projections of Query Vectors into the unit circle on the plane defined

Algorithm 8 Algorithm of sPDDP[8]

Input: Dataset, k(number of clusters required)
Initialize $c = 1, cluster(c) = Dataset$
while $c < k$ current number of clusters is less than required **do**
  for $i = 1..c$ **do**
    $cov(i) = covarianceMatrix(cluster(i))$
    $peg_1(i) = PrincipalEigenvector(cov(i)), peg_2(i) = PrincipalEigenvector(cov(i))$
    project $cluster(i)$ on the plane defined by $peg_1(i)Xpeg_2(i)$ denoted by $proj(i)$ such that $y \in proj(i)$, then y is 2-dimensional vector, $y = (y_1, y_2)$
    For $y \in Proj(i)$, if $y \neq 0$ then push y to the unit circle and denote by $z = \frac{y}{||y||}$. These are collected in the set $projUnit(i)$
  end for
  identify $i^{'} \leq c$ such that $projUnit(i^{'})$ is best for Unit Circle Partition
  Partition $proj(i^{'})$ and correspondingly $cluster(i^{'})$ into $\{cluster(i^{'})_1, cluster(i^{'})_2\}$
  $cluster(i^{'}) = cluster(i^{'})_1$
  $cluster(c+1) = cluster(i^{'})_2$
  $c = c + 1$
end while
Output: $Cluster(1), Cluster(2), ..., Cluster(k) : \bigcup_{i=1}^{k} Cluster(i) = Dataset$

Algorithm 9 Algorithm for Unit Circle Partitioning[8]

Input: $Z = \{z_1, z_2, ..., z_m\}$
Let $W = \{w_1, w_2, ...w_m, ..., w_{2m}\}$ be a set of two-dimensional vectors such that: $w_i = w_{m+i} = z_i, i = 1, .., m$
Reassign indices such that $w_j = e^{i\theta_j}, 0 \leq \theta_1 \leq \theta_2 \leq ... \leq \theta_{2m} < 2\pi$
for all $j = 1..2m$ **do**
  Set $x = \frac{(w_j + w_{j+1})}{2}$
  Set $x^{\perp} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x$
  Set $\pi_1^j = \{z : z \in Z,\ z^T x^{\perp} \geq 0\}$
  Set $\pi_2^j = \{z : z \in Z,\ z^T x^{\perp} < 0\}$
  Set $Q_s^j = Q(\{\pi_1^j, \pi_2^j\})$
end for
Let $Q_s^k = max_{j=1...m}\ Q_s^j$
Output Partition $\{\pi_1^o, \pi_2^o\}$ **:** $\{\pi_1^k, \pi_2^k\}$

by the Principal Directions. Thus, correspondingly we find the partition of $projUnit(i)$ into $\{projUnit(i)_1, projUnit(i)_2\}$. Also, as each element of the set $projUnit(i)$ has one-to-one relationship with a vector in $cluster(i)$, we shall also get the required partitions, $\{cluster(i)_1, cluster(i)_2\}$.

Chapter 4

Experimental Results

In this section we discuss the results obtained by the implementation of the aforementioned proposed method. As the previous section of Method Proposal discuss, we assume throughout the implementation that the we are provided with only the discussed Dataset as input.

The process, does not assumes any other external information about the dataset apart from those discussed in detail earlier. On this data set, we implement the step-by-step Pre-processing again as indicated earlier. The preprocessing generates Keyword group List, which continues as input to the last sections of this proposed method.

To conclude the method, we generate a vectorization of the dataset with the help of our Vector Space Model. This vectorization is the only carry forward to our Unsupervised Clustering Models. Hence, now we use the Query Vector list independently on the four proposed models of Unsupervised Clustering to generate four separate independent clustering of the same dataset.

This section first discuss all these four clustering results individually. It points out the characteristics highlighted by the Clustering Models in these results, and henceforth points out the pros and cons of that Unsupervised Clustering Model. Thereafter, it attempts to compare the clustering results of the four methods empirically with each other. This will help us to get a measure on the cohesiveness of the results and hence its dependence on the Clustering Model.

This will complete this section by highlighting all the facts about the results, so that we can proceed to the last sections to conclude the experiments and share our views about the process on a holistic level.

spongebob games ghetto spongebob right move spongebob moves in	vodafone rewardz vodafone myvodafone.vodafone.in Vodafone Bill Download
sync contacts from facebook how to sync contacts from facebook to windows phone how to sync contacts from facebook to windows phone	red potatoes mashed baked sweet potatoes with brown sugar twice baked potatoes baked mashed red potatoes
bing google images bing images google images bing Www.bing.com bing crosby	

Table 4.1: Some Good Clusters of DbScan Model

4.1 Analysis of Unsupervised Clustering Models

4.1.1 DBScan Model

As the Model of DBScan indicates, the algorithm does not includes any criteria for the homogeneity of the cluster sizes for the clusters hereby generated. Hence, the clusters vary in size from 2 queries to around 500 queries per cluster.

Also, the clusters primarily tries to group the keywords that share maximum number of common keygroups in their constituent queries. Hence, though in most cases they are grouped as per single common keygroup they share, sometimes they grouping is also upon two common keygroups a set of queries have in common. This gives this model an advantage to distinguish between sets of queries that looks similar but can be further sub-clustered.

The total number of clusters generated is around 1100 for the 9999 queries considered by the model. It includes the generic clusters of facebook, youtube, Google, gmail and yahoo that groups all the queries of these types as separate clusters accounting for the minute variations of spelling and some added keywords that does not change te basic intent of the requirement. In addition few of the good clusters generated are mentioned in Table 4.1.

The table shows cluster where multiple keywords like 'facebook' and 'win-

dows' were used to create it. Also, the baked potatoes cluster highlights the elasticity of the cluster where both queries containing 'baked' and 'potatoes' and also queries containing only 'potatoes' were joined together as a tangible grouping. This helps to identify flexible cluster that are not rigidly grouped but the grouping is expanded without compromising on the intention.

Also, the cluster of 'bing' and 'google' highlights the help of Wikification for joining these two starkly different keywords together as search engines and hence create a logically common cluster for them.

But, we also need highlight the shortcomings of DBScan Mode by Table 4.2 which also touches upon the reason behind the failure of the model in such cases.

In one of the cluster 'Family Barn Game' was classified into 'Barnes and Noble' cluster while both has different intention.This is due to the fact that Barn as a word has different interpretation for proper nouns, and our method provides no guidance to distinguish between different interpretation of same word in context of proper nouns.

Also, another problem persistent of DBScan Model is over clustering where certain clusters are over dissected in certain cases to dilute the intention behind them. The big email and social network cluster of many email service providers has a common intention, but the DBScan algorithm goes ahead to identify a different cluster for 'google search' and 'facebook login' which has same intent as the parent cluster.

The further subdivision is done due to the extra number of keywords shared by these children clusters, but the algorithm fails to detect the fact that is added keyword does not alters the intention of the clusters to be further partitioned.

On overall consideration, out of the four models proposed, DBScan Model is the second best model in terms of the cluster quality, inspected manually in comparison to clusters generated by human decisions.

4.1.2 Principal Direction Divisive Partitioning Model

This model of clustering, generates a much homogeneous set of clusters by size, with certain outliers. As the process assumes universal cluster at the beginning and proceeds to break chunks out of it in steps, the chunks broken are of uniform size. But as some final exit criteria is satisfied, more often than not, a large chunk is left back in the end that is remains unclustered and hence without any good intention.

This method generally keeps on breaking the datasets into chunks until a stopping count of clusters are given. To keep up the comparison between models, we stop the clustering after 1100 clusters are generated. This is

barnes and noble barnes noble barnes and noble barnes noble Family barn Game barnes noble barne an nobles	gmail.com chase.com google www.msn.com food network www. google.co.uk yahoo www.facebook.com hotmail.co.uk sign in facebook youtube google google google facebook prime rib. ca lottery www.google.co.uk youtube yahoo mail free aol email hale village online hotmail google mail online facebook Facebook facebook
google search.com google.com search google search google search google search google search.com google search google search.com	facebook homepage login facebook homepage login Facebook homepage login facebuk homepage login

Table 4.2: Some Bad Clusters of DBScan Model

harris county toll road authority indiana hunting seasons	how do you delete songs from your iphone Use iPod camera
youtube to mp3 youtube to mp3 youtube to mp4	gmail login inbox Gmail Sign In Gmail Sign In Gmail Sign in Gmail Account Gmail Sign In
www.ebay.co.uk www.ebay.com ebay uk ebay.co.uk ebay uk ebay.co.uk ebay uk www.ebay.com www.ebay.com	yahoo sign in yahoo sign in mail Sign into Yahoo! Mail hotmail.com login email yahoo mail sign in yahoo mail sign in yahoo sign in hotmail.com login email

Table 4.3: Some Good Clusters of PDDP Model

because our last model generates similar amount of clusters before satisfying its own exit criteria.

In this case though, unlike the previous model, there is no guarantee about the homogeneity of the queries that constitute a cluster. As unlike DBScan, the clusters are not broken by simple distance measure, but by the projection of this sparse vectors on some single dimension, the chunks are sometimes mixed in intent.

Hence, it creates quite a few dubious clusters that club together multiple intention queries. But to some extent it removes the issue of over clustering highlighted in the last section as the clustering is no more just simply on distance measure.

Few good and innovative clusters generated by this model are mentioned in the Table 4.3 and discussed hereafter.

As usual, this clustering model is successful in identifying the big clusters of 'ebay' 'yahoo' 'gmail' 'google' and famous 'facebook' that has all the combinations of their web page name with some added keywords. But the primary intention behind all such queries being able to access these Service Provider websites. This model,like earlier, identifies such big clusters.

Also this model goes ahead to identify certain logical clusters by deduction from 'Wikification' and 'User ID Grouping'. Hence, it successfully groups 'iphone' and 'ipod' queries into a single cluster as they are flagship

product of same company.It also merges a query about a state 'Indiana' and its constituent county 'harris county' into same cluster which would have been missed by a human eye. Hence,first time this clustering model provides us new information about the clusters.

But as expected by the fact that the clusters generated by this model are heterogeneous, lot of clusters are therefore of mixed intentions, which reduces the impact of this model. Few such bad clusters are hereby highlighted in Table 4.4

The clusters shows that the model generate two separate clusters, where service providers like 'skype' and 'gmail' and put into one cluster while 'yahoo' and 'hotmail' are in a separate cluster. This partition is homogeneous in the cluster size, but heterogeneous in its content because if the intention upon which the clustering was supposed to be performed was 'email service provider', then 'gmail' stand misclassified with 'skype'.

Also, although this method generates clusters that are not that obvious but shares a common intent, this as expected includes a decent amount of misclassification, for example 'ea sports fifa' query is clustered in a recipe cluster where it clearly is a misfit.

The same example is repeated where a 'lottery' cluster contains a query on recipe, 'slow roasted prime rib recipe'. The last issue is created due to the Wikification that causes over-condensation of keywords, such that 'bristol city council' is clustered with 'illinois lottery' as there is a bristol city in illinois. But the fact is, there are multiple bristol cities around the world, including in UK which is query mostly refers. Thus, Wikification independently draws excess conclusions from the data which needs to be filtered by the location of other meta data associated with the query to sort such conflations.

On a overall analysis of the type of clusters generated as compared to clustering by human intelligence, this clustering model is the third best model out of the four proposed in this experiment.

4.1.3 Spherical Principal Divisive Partitioning Model

This model of clustering is an extension of the earlier model where we increase the retained information in the projections that are used to generate the clusters. Hence, the basic idea behind the clustering remains same.

In this case also the clusters are homogeneous in size, apart from few outliers. But the real boon or bane occurs due to the heterogeneity of the queries that constitute the cluster. The queries and not simply logically group able.Thus, though it ends up generating interesting clusters, there are

skype sign in gmail login inbox Gmail Sign In Gmail Sign In jb hi fi Gmail Sign in Gmail Account wells fargo online sign in netflix sign in help password Gmail Sign In	yahoo sign in yahoo sign in mail Sign into Yahoo! Mail hotmail.com login email yahoo mail sign in yahoo mail sign in yahoo sign in hotmail.com login email
recipe for potato bake ea sports fifa rib roast recipes egg bake recipes Christmas red cabbage	ct lottery ct lottery slow roasted prime rib recipe
bristol city council illinois lottery	

Table 4.4: Some Bad Clusters of PDDP Model

also clusters that combine queries of multiple intentions, making the cluster ambiguous.

Unlike the previous method, this clustering gives large clusters of size around 60-80 queries. Hence, it helps to group successfully the service provider queries, but causes over grouping is more niche intent groups.

Few good clusters generated by this model are discussed here along with Table 4.5. One of the clusters generated is the 'lottery cluster' which collects the queries to different kind of lottery service providers. The good thing is that it even identifies queries for lottery service providers that do not have the word 'lottery' in the query. This is due to the grouping upon the 'User ID' where it is assumed that a user is making subsequent queries with the same intention of Lottery Results.

Also, this clustering identifies a recipe cluster, which successfully identifies a food outlet 'Starbucks' in this cluster due to the Wikification of 'cappucino' which includes Starbucks as a outlet. This clustering model also identifies 'basketball' cluster that lists queries for two basketball clubs 'foothills club' and 'iowa state cyclones'. Also, looking into the Wikification of the club information, it identifies acbl as a web service to provide information about results of basket ball.

But as pointed out earlier, the large chunks of clusters generated by this model, does not circumvents the issue of ambiguous clusters created but

daily millions results nc lottery nc lottery nyc lottery daily millions results wa lotto winning numbers nc lottery hp govt.nic.in www.y8games.com	starbucks northwest indiana restaurants open christmas day godiva cappuccino liqueur recipes Kale Recipes Rachael Ray
iowa state cyclones basketball acbl.org club results acbl.org club results foothills club west parcel 24	

Table 4.5: Some Good Clusters of sPDDP Model

instead aggravates it. Hence, certain clusters created makes no sense, and is therefore highlighted in the Table 4.6.

A large cluster generated likewise is shown here, where there is no common intention highlighted by the queries of the cluster. Looking at the queries it is evident that it has queries with intention 'lottery' 'christmas and boxing day' 'bank' and 'game'. But there are separate standalone clusters for these intentions, where these queries are absent. They end up being partitioned from the projections into this ambiguous cluster.

Also, there is a 'blackberry' cluster that contains a query 'janak positioning & surveying system pvt ltd' which got included due to the over condensation of keywords due to 'UserID' as the user who searched 'blackberry' followed by searching 'janak surveying' and hence the algorithm assumed they share a common intent which is a case where our assumption breaks down.

Overall, though this method gives to us more bold clusters than earlier method that identifies insightful intentions that cannot be identified by Human verification. But this boon is combined with the bane of rampant under clustering creating ambiguous clusters that makes this method the worst of the models under consideration.

4.1.4 KMeans Model

This is the final Model under consideration. This model is different from all the models discussed above as this is the only model under consideration

kim k. christmas eve pics lloyds tsb internet banking is Myer Miranda open Boxing Day UK Lottery Results lloyds tsb internet banking medinah country club Christmas Giftcards itunes Top 10 Family Reunion Resorts jerry lee lewis www.lottery.co.uk is Myer Miranda open Boxing Day www.mysmartbox.com john lewis john austin shepard john lewis taney county missouri maps driving directions www emailyahoo the uk lottery Fase book john lewis 8 ball pool today show recipes how do I add a new game I got to steam Bulk Handgun Ammo what time does sainsburys open on boxing day in lord street southport is Myer Miranda open Boxing Day . az lottery oakland county jail john lewis Redlands Christmas Lights Tour Map g50 christmas lights lloyds tsb internet banking mcdonald's vernon ct kim k. christmas eve pics st george bank how do I add a new game I got to steam ps4 sign in problems christmas what fast food is open on christmas day . az lottery Redlands Christmas Lights Tour Map www.bankofamwerica.com	download instagram on blackberry janak positioning and surveying system pvt ltd blackberry bold 9790

Table 4.6: Some Bad Clusters of sPDDP Model

that assumes initially no universal clustering and builds cluster by joining queries one by one. Thus, unlike divisive clustering propagated by the earlier methods, this is essentially an agglomerative method of clustering and therefore gives unique set of results.

This model generates clusters that does not adheres to any homogeneity of size as the queries keep on piling on a cluster until the distance measure satisfies an exit criteria. This makes it suitable for our problem as even our problem demands heterogeneous clusters to be considered as part of the solution. Also, as the queries are collected by a distance measure, the queries grouped makes logical sense and generally do not try to make ambiguous clusters as it breaks the cluster before ambiguity can propagate in the cluster.

But as expected, the clustering model thereby suffers from 'curse of dimensionality' where each query vector being so sparse suffer due to grouping by distance measure. Thus, the over clustering remains where the distance between similar clusters differing by a non-consequential keyword is so great that it ends up generating two clusters of same intention.

We will hereby highlight some good clusters generated from this model in Table 4.7 . One cluster identified is a 'tube videos' clusters that identifies 'youtube' and 'redtube' into a same cluster and accounts for all the spelling variations that occurs for them under the same intent.

Similarly, multiple keywords are connected to generate a cluster for 'Merry Christmas' which include logical addtion of 'vanessa white' who announced her binge eating on christmas challenge just a day before this dataset is collected. Hence the fact was reflected on her Wikipedia page, finding its way down here. This highlights the one major help of Wikification where using Wikipedia as the reference database helps to keep a track of all the recent major updates that finds its way to the clustering. Hence, this keeps the grouping of keywords elastic and up to date.

As expected this model is successful identifying Service Provider group clusters not only for the big ones likes 'facebook' or 'google' but also for 'skype' and 'att'.A major cluster identified is the 'Samsung Galaxy' cluster that not only collects queries containing both 'Samsung' and 'galaxy' keyword under its wing, but also brings together queries for 'samsung' service queries containing just 'galaxy' in the keyword but intends same as 'samsung galaxy'. This is due to the agglomerative model adopted here that gives a elastic freehand for the grouping of these queries.

Similarly, we end up with an 'Airline' Cluster that collects various airline service provided under one window making a sensible cluster. There is a 'food cluster' that collects 'foodnetwork' along with 'carrots' and 'roast' keywords due to the Wikification, and the common Website feedback shared by the Users generating these queries. This highlights the importance of grouping

on basis of Websites and their relevant feedback from user.

Also another 'food' cluster is generated from the Wikification, where 'yam' 'marshmellow' 'macaroni' and 'cheese' are successfully detected as food products that belong together. Hence, it gives us insightful clustering from this Model.

But as expected, certain degree of conflict in interest and over clustering is retained in this clustering model. This is highlighted in Table 4.8 and discussed briefly.

One important bad cluster highlighted is the 'India Indiana cluster' where it is assumed that both are misspells of each other in reality they mean completely different locations. Though this is not the fault of the model but the assumptions made along the way.

Also, the issue highlighted is the importance of keywords, where the model ends up deciding 'password' as a more important keyword than 'facebook' and ends up creating a cluster on password while it is a non-consequential keyword and does not adds up to the primary intention of 'facebook'. Same issue is again highlighted over 'home pages' as being classified as an important keyword.

The last issue is that of over-condensation due to Wikification, where queries containing 'St.Andrews University' is naturally grouped in the 'Australia' Cluster as Australia has a University of that name and hence it is a logical grouping. But in actuality, there are multiple such universities of name 'St. Andrews University', the more famous one being that of Scotland that is mostly intended by this query.

It is evident, that with its minor defects, the 1000 clusters generated by this model are far better than the last two models and comparatively better than the DBScan Model. Hence, by the quality of clusters generated in comparison to clusters generated by Human Intelligence this model fares best out of the four. Out of the four models discussed and used in out Proposal, KMeans Model is the best model for clustering relevant to this model.

4.2 Comparison between Unsupervised Clustering Models

Now, as we have created four sets of partitions of the given Dataset. For all these four partitions, we have individually discussed the merits and demerits associated with the Clustering Model.

Apart from that, we also find that there is certain assumptions made

Tube 8 tube 8 you tube Redtube Porn Tube youi tube Hot Tube red tube YOU TUBE you tub ape tube you tube music learn piano songs you tube U TUBE red tube	merry christmas sms merry christmas sayings with dazzle vanessa white merry christmas webmd merry christmas greetings merry christmas in russian webmd.com vanessa rigaud
how to program att uverse remote att.net matt goss	skype how do iset up skype on tv skype sign in skype download for windows 7 64-bit
samsung galaxy s4 samsung galaxy case how do i reset my samsung account samsung how to put music onto samsung galaxy s4 samsung.com pureology 21 benefits samsung uk samsung galaxy young pureology 21 benefits samsung tv samsung syncmaster drivers https//account.samsung.com/mobile samsung microwave manuals act samslog in can i use my blackberry sim in galaxy finding you mac address samsung galaxy 4	delta airlines united airlines southwest airlines azal.airlines spirit airlines delta airlines southwest airlines southwest airlines american airlines flight schedule now american airlines www.southwest.com southwestairlines.com delta airlines southwest airlines tiger airlines
currys pc world currys currys electrical currys.co.uk www.currys.co.uk 0currys opening time on boxing day currys electrical	foodnetwork.com how to roast carrots odin how do you roast carrots fry's marketplace odin 64 bit harkins tempe marketplace
yam and marshmallow casserole 50 fresh yam and marshmallow casserole Paula Deen Macaroni and Cheese	

Table 4.7: Some Good Clusters by KMeans Model

ameristar casino indiana indian consulate houston perfect north slopes indiana indiana judgments indian in kuwait indian consulate houston Indian Consulate Atlanta indiana hunting seasons indian railways Indian Consulate Atlanta south indian bank xxx indian videos indiana government center northwest indiana restaurants open christmas day	facebook login and password account.live/password/reset facebook log in and password facebook login and password i forgot my password on my iphone ios7 windows live change password netflix sign in help password facebook log in and password apple password reset facebook login and password
facebook home page new leaf homes gmail login page facebook home page gmail login page facebook home page Facebook Home	st andrews university national australia bank internet banking st andrews university st andrews university rolex watches australia Google Maps Australia how many asians live in australia google maps australia pictures of australia mimco australia

Table 4.8: Some Bad Clusters of KMeans Model

	DBScan	**PDDP**	***sPDDP***	**KMeans**
DBScan	X	51.40%	47.20%	54.67%
PDDP	X	X	51.70%	38.71%
sPDDP	X	X	X	43.37%
KMeans	X	X	X	X

Table 4.9: Comparison between the four Unsupervised Clustering Models

during the Pre-processing stage that does not holds true always. Hence, this is also the cause of introduction for certain unwanted errors in the partitions. But it is clearly visible that the insights gained by this clustering, clearly outnumbers the few boundary cases which challenges the proposed method.

For the clustering methods discussed, there are two primary observations that we make about the required properties of an ideal clustering model for our case,

- Agglomerative Clustering is more useful for our Dataset that Divisive clustering as it assumes no homogeneity about the shape and size of the clusters and builds the convex cluster as per distance criteria.
- Clustering methods that depends on simple distance measure in high dimension are susceptible to the curse of dimensionality to such extent that it leads to over clustering in these cases. This over clustering is removed in projection based methods, but they have their own share of problems.

Hence, the ideal model required for our problem should be an agglomerative model, that of all things should not depend of simple distance measures to generate the clusters. This will incorporate all the major learnings from the different models.

This requirement is further highlighted by the comparison between Dataset partitions generated by these four clustering models. For the comparison proposed here, for every pair of Unsupervised Clustering Models, we verify the relative cohesiveness of the clusters. Thus, for every pair of Clustering Models, we compare if pair of queries that belong to same cluster in Model 1 , also do so in Model 2.

So for all possible pairs of query, this compares the membership of the pair of queries in same cluster for different Models. The result for the same is highlighted in the Table 4.9 below.

Thus, from the table we can see the membership generated by the Clustering Models are starkly different from each other. This confirms the fact

that the quality of clusters generated is highly dependent on the type of clustering used create that set of partitions.

The pros and cons of the different Unsupervised Clustering Models discussed in the last section is validated by the comparative cohesiveness between the pairs of clusters. As expected, as all the individual models brings a new method of grouping the query, no pair of clusters have a Similarity Score of more that 55% which is just about more than half of all the combinations of the queries.

Also, the pair of distance-based clustering models DBScan-KMeans and the pair of projection based clustering models PDDP-sPDDP has the highest Similarity score at 54.67% and 51.70% respectively. It is also highlighted that the best Model out of the four i.e KMeans Model is starkly different from both PDDP and sPDDP with Similarity Score 38.71% and 43.37% respectively.

Hence, we can conclude the comparison saying that PDDP and sPDDP though similar to each other are starkly different from DBScan Model and the KMeans Model. Moreover, the DBScan Model in the scale of similarity lies somewhere between the Projection based Models and the KMeans Model. Therefore the ideal clustering model needs to be similar to the KMeans Model, only burrowing few concepts of the Projection Model that circumvents the Curse of Dimensionality.

Chapter 5

Conclusion

This work is a step forward to address the issue of next-generation Search Engines that is getting more relevant day by day. Internet has become an integral part of our daily activities as virtual world and real world integrate on a daily basis. This puts web search on the forefront of this merger where it is the arterial connection between our requirements of the real world that translates to virtual world.

Our research is based on the fact that Web Searches are no more individual queries made by user to a look up table. They are now more than ever a means to accomplish tasks. So any development on search engines that are task based in analysis, and address our queries on the basis of task they portray rather than the raw query has to be depend on the primary problem of Task Discovery.

This research therefore was an attempt to simplify that process of task discovery. Where our approach starts with not considering Search Queries as strings but a collection of meta data that gives us an insight about the requirement of the User.

Once, we have the captured data our method goes forward to built a condensed set of keyword groups that not only considers the keywords on their own, but also accounts for simple spelling variations and expands the keyword into a keygroup as per the Wikipedia article associated with the keyword.

Now, over the process of Pre-processing these keygroups are condensed on two primary criteria. Firstly, they are condensed upon the User who generates multiple queries for same task and hence points to us that the different keywords he has used are indicative of the same requirement. Also, there is a information generated from the natural feedback of the User who visits certain set of web pages that are generated from his query. The amount of time the person visits the webpages generated from his query captures

his degree of satisfaction and therefore adds another criteria for keygroup condensation.

Our process then goes on to use for different Clustering Models to cluster the vectorized query set without supervision. This clustering models are different to one another in their clustering techniques. Hence, we generate starkly different sets of clusters, each with their own pros and cons.

Thus, this helps us to identify a good clustering method out of the four and as well guides us to the properties of an ideal clustering technique that will fulfill the generic requirements of this problem.

The roadblocks faced by us were challenging as we identified certain boundary case queries that were not compatible with the assumptions we made along the way. Hence, our method proposed fails if these two cases happen,

- If the keywords are Proper Nouns with multiple references, our method does not considers all the references by mostly the clusters are generated upon one reference which might not be the most prevalent reference in this scenario.

- The basic assumption that all the continuous stream of queries generated by same User over a short span of time are of same Task is mostly true, but fails for few cases. This cases leads to ambiguous grouping of keywords.

These are the issues faced by the theory of our proposed method. But another major issue is the real time implementation of this method. The method has a Time Complexity of $O(n^3)$ where n is the number of queries in the dataset. Thus for search engines where around 10^8 are generated under a minute, this complex algorithm is not at all realizable in real-time for analysis. Hence, it is required to greatly develop the underlying engineering associated with the proposed method to generate a distributed framework to deal with problems of comparative scale. A background Map-Reduce framework is at least necessary for the generating the keygroups that will be used by the Clustering Models.

As future work, we propose to consider primarily a relative valuation of the keygroups which can decide importance of that keygroup for dictating the intention represented by the cluster. In our proposal, all keygroups are of equi-importance, which leads to the scenario where non-consequential keygroups cause over-segmentation of the dataset which is of same underlying intention.

Also, though we have discussed four Clustering Models, none of the models are perfect for the problem requirement as each carries the baggage of their

own pros and cons. Hence, we have highlighted the properties that should be part of an ideal Clustering Model. Future work should also include identifying such Clustering Model that substantially improves the quality of the clusters generated. This will be a step towards the industrial implementation of 'Task Discovery' to usher a new era of Search Engines.

Bibliography

[1] BAEZA-YATES, R. AND RIBEIRO-NETO, B. 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[2] M. BERRY, S. DUMAIS, and G. O'BRIEN.Using linear algebra for intelligent information retrieval.57AM Review, 37(4):573-595, 1995.

[3] M. BERRY, Z. DRMAC, and E. JESSUP.Matrices, vector spaces, and information retrieval.5/AAf Review, 41(2):335-362, 1999.

[4] BOLDI, P.,BONCHI, F., CASTILLO, C., DONATO, D., GIONIS, A., AND VIGNA, S. 2008. The query-flow graph: Model and applications. In Proceedings of the 17th ACM International Conference on Information and Knowledge Management (CIKM08). ACM, New York, NY, 609618.

[5] D.L. BOLEY.Principal direction divisive partitioning.Data Mining and Knowledge Discovery, 2(4):325-344, 1998.

[6] BRODER, A. 2002. A taxonomy of Web search. SIGIR Forum 36, 2, 2, 310.

[7] I.S. DHILLON and D.S. MODHA.Concept decompositions for large sparse text data using clustering.Machine Learning, 42(1): 143175, Jan 2001.Also appears as IBM Research Report RJ 10147, Jul 1999.

[8] I.DHILLON,J.KOGAN,C.NICHOLAS, Feature Selectiong and Document Clustering,Survey of Text Mining II, 2008, pp 73-100

[9] STIJN VAN DONGEN , A New Cluster Algorithm for Graphs, National Research Institute for Mathematics and Computer Science, 1998.

[10] R.O. DUDA, RE. HART, and D.G. STORK.Pattern Classification, second edition.Wiley, New York, 2001.

[11] ESTER, M., KRIEGEL, H. P., SANDER, J., AND XU, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD96). ACM, New York, NY, 226231.

[12] GLANCE, N. S. 2001. Community search assistant. In Proceedings of the 6th ACM International Conference on Intelligent User Interfaces (IUI01). ACM, New York, NY, 9196.

[13] HE, D. AND GOKER, A. 2000. Detecting session boundaries from Web user logs.In Proceedings of the 22nd Annual Colloquium on Information Retrieval Research (BCS-IRSG). 5766.

[14] HE, D.,GOKER, A., AND HARPER, D. J. 2002. Combining evidence for automatic web session identification. Info. Process. Manage. 38, 5, 727742.

[15] JONES, R. AND KLINKNER, K. L. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In Proceedings of the 17th ACM International Conference on Information and Knowledge Management (CIKM08). ACM, New York, NY, 699708

[16] MEI KOBAYASHI,MASAKI AONO , Vector Space Models for Search and Cluster Mining,Survey of Text Mining II, 2008, pp 109-127

[17] J. KOGAN.Means clustering for text data.In Proceedings of the Workshop on Text Mining at the First SIAM International Conference on Data Mining, M.W. Berry, ed., pages 47-57, 2001.

[18] LEE, U., LIU, Z., AND CHO, J. 2005. Automatic identification of user goals in Web search. In Proceedings of the 14th International World Wide Web Conference (WWW05). ACM, New York, NY, 391400.

[19] LUCCHESE, C., ORLANDO, S., PEREGO, R., SILVESTRI, F., AND TOLOMEI, G. 2011. Identifying task-based sessions in search engine query logs. In Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM11). ACM, New York, NY, 277286.

[20] LUCCHESE, C., ORLANDO, S., PEREGO, R., SILVESTRI, F., and TOLOMEI, G. 2013. Discovering tasks from search engine query logs. ACM Trans. Inf. Syst. 31, 3, Article 14 (July 2013), 43 pages. DOI: http://dx.doi.org/10.1145/2493175.2493179

[21] MACQUEEN, J. B. 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, L. M. Le Cam and J. Neyman Eds., Vol. 1. University of California Press, Berkeley, CA, 281297.

[22] C. MANNING and H. SCHIITZZ.Foundations of Statistical Natural Language Processing.MYT Press, Cambridge, MA, 2000.

[23] CHRISTOPHER D. MANNING, PRABHAKAR RAGHAVAN, and HINRICH SCHÜTZE. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] PORTER, M. F. 1980. An Algorithm for Suffix Stripping Vol. 14. Morgan Kaufmann Publishers, San Francisco, CA, 130137.

[25] ROSE, D. E. AND LEVINSON, D. 2004. Understanding user goals in web search. In Proceedings of the 13th International World Wide Web Conference (WWW04). ACM, New York, NY, 1319.

[26] G. SALTON.7V SMART Retrieval Sy.vtem.Prentiee-Hall, Englewood Cliffs, NJ,1971

[27] SHEN, X., TAN, B., AND ZHAI, C. 2005. Implicit user modeling for personalized search. In Proceeding of the 14th Conference on Information and Knowledge Management (CIKM05). ACM, New York, NY, 824831.

[28] SILVERSTEIN, C.,MARAIS, H., HENZINGER, M., AND MORICZ, M. 1999. Analysis of a very large Web search engine query log. SIGIR Forum 33, 1, 612.

[29] SILVESTRI, F., BARAGLIA, R., LUCCHESE, C., ORLANDO, S., AND PEREGO, R. 2008. (Query) history teaches everything,including the future. In Proceedings of the 6th Latin American Web Congress (LA-WEB08). IEEE Computer Society, Washington, DC, 1222.

[30] SPINK, A., PARK, M., JANSEN, B. J., AND PEDERSEN, J. 2006. Multitasking during Web search sessions. Info. Process. Manage. 42, 1, 264275

[31] WEN, J. R., NIE, J. Y., AND ZHANG, H. 2002. Query clustering using user logs. ACM Trans. Info. Syst. 20, 1, 5981.