Volume 4, No. 12, December 2013

ISSN-2229-371X

Journal of Global Research in Computer Science

RESEARCH PAPER

Available Online at www.jgrcs.info

AN APPROACH TO BUILD A WEB CRAWLER USING CLUSTERING BASED K-MEANS ALGORITHM

Nilesh Jain[1], Priyanka Mangal[2] and Dr. Ashok Bhansali[3]
Lecturer[1, 2] and Associate Professor[3]
Department of Computer Applications[1],
Department of Computer Science[2,3]
Mandsaur Institute of Technology, Mandsaur (M.P.), India[1, 2]
O.P. Jindal Institute of Technology,Raigarh[3]
Email: nileshjainmca@gmail.com[1], pmangal16@gmail.com[2]

Abstract: Central to any data-mining project is having sufficient amounts of data that can be processed to provide meaningful and statistically relevant information. But getting the unstructured data is only the initial stage and that data must be transformed into a structured format which is suitable for further processing.

In this paper we have proposed architecture for the web-crawling and arrange their unstructured data using cluster based algorithm. . The clustering process is based on the k-means algorithm. This paper is completely based on the focused crawler mechanism that only scans the pages by using general crawling policies.

Keywords: web-mining, clustering, data-mining-means, focused crawler

INTRODUCTION

WWW is the treasure of the information. It is a huge sea and in which finding the pearl is very tough process. Data mining using clustering based techniques can help us for such kind of work.

A crawler is a special program that visits Web sites, reads the page and after that collects raw contents in order to create entries for clustering. The major search engines on the Web all have such a program, which is also known as a "spider" or a "bot." Entire sites or specific pages can be selectively visited and indexed. Crawlers apparently gained the name because they crawl through a site a page at a time, following the links to other pages on the site until all pages have been read. [1]

Here is the process that a web crawler follows:

- Start from one preselected page. We call the starting page the "seed" page.
- Extract all the links on that page. (This is the part we will work on in this unit and Unit 2.)
- Follow each of those links to find new pages.
- Extract all the links from all of the new pages found.
- Follow each of those links to find new pages.
- Extract all the links from all of the new pages found.

Basically there are four types of crawlers: [2]

a) ***Traditional Crawler*** – visits entire Web (?) and replaces index
b) ***Periodic Crawler*** – visits portions of the Web and updates subset of index
c) ***Incremental Crawler*** – selectively searches the Web and incrementally modifies index
d) ***Focused Crawler*** – visits pages related to a particular subject

Focused Crawler:

- ➢ Only visit links from a page if that page is determined to be relevant.
- ➢ Classifier is static after learning phase.
- ➢ Components:
  - o Classifier which assigns relevance score to each page based on crawl topic.
  - o Distiller to identify *hub pages.*
  - o Crawler visits pages to base on crawler and distiller scores.
- ➢ Classifier to related documents to topics
- ➢ Classifier also determines how useful outgoing links are
- ➢ ***Hub Pages*** contain links to many relevant pages. Must be visited even if not high relevance score.

Focused Crawler:

A focused crawler based on a hypertext classifier was developed by Chakrabarti et al. [5]. The basic idea of the crawler was to classify crawled pages with categories in topic taxonomy. To begin, the crawler requires topic taxonomy such as Yahoo or the ODP.9 In addition, the user provides example URLs of interest (such as those in a bookmark able). The example URLs get automatically classified onto various categories of the taxonomy. Through an interactive process, the user can correct the automatic classification, add new categories to the taxonomy and mark some of the categories as “good" (i.e., of interest to the user). The crawler uses the example URLs to build a Bayesian classifier that can find the probability (Pr(cjp)) that a crawled page p belongs to a category c in the taxonomy. Note that by definition Pr(rjp) = 1 where r is the root category of the taxonomy. A relevance score is associated with each crawled page that is computed as: [5]

$$\sum_{c \in "good"}^{n} \Pr(c \mid p)$$

When the crawler is in a “soft" focused mode, it uses the relevance score of the crawled page to score the unvisited URLs extracted from it. The scored URLs are then added to the frontier. Then in a manner similar to the naïve best URL crawler, it picks the best URL to crawl next. In the “hard" focused mode, for a crawled page p, the classifier first finds the leaf node c* (in the taxonomy) with maximum probability of including p. If any of the parents (in the taxonomy) of c* are marked as “good" by the user, then the URLs from the crawled page p are extracted and added to the frontier.

Crawling Policy:

A Standard Crawlers having some well define policies to crawl the webpages. Below are the standard policies [2]

1) a *selection policy* that states which pages to download,
2) a *re-visit policy* that states when to check for changes to the pages,
3) a *politeness policy* that states how to avoid overloading Web sites, and
4) a *parallelization policy* that states how to coordinate distributed web crawlers.

Our Web crawler always respects the common crawling policies such as the following:

Our crawler accesses each site in a page-by-page manner with some intervals.

Though we interleave the crawling processes with the processes for detecting host aliases, chances are that an aliased server may be accessed simultaneously under different host names.

It always reads the robots.txt file and never crawls restricted pages.

You can specify directives to the crawler in robots.txt file at the top of your site (e.g., http://www.abc.com/robots.txt). For example, the following directive forbids our crawler to retrieve any content from your site.

```
User-agent: googlebot
Disallow: /
```

If you want to control the rate of access, specify Crawl-delay parameter in robots.txt file. For example, the following directs our crawler to access the site not more than once every 30 seconds.

```
User-agent: googlebot
Crawl-delay: 30.0
```

If the respective web page has the robots Meta tag included as follows, our crawler never crawls the page.

You can also protect the contents in a file-by-file manner with the robots Meta tags. If you put the following in the header of your HTML documents, our crawler will not follow the links found in the documents.

```
<META NAME="robots" CONTENT="nofollow,
noindex">
```

Architecture:

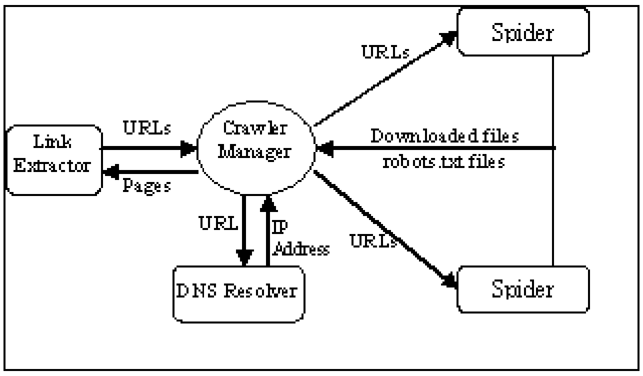

Figure 1 - Crawler Architecture

Volume 4, No. 12, December 2013

ISSN-2229-371X

Journal of Global Research in Computer Science

RESEARCH PAPER

Available Online at www.jgrcs.info

Figure 1 shows an easy architecture for web crawler:

Crawler Manager: takes a set of URLs from Link Extractor and sends the Next URL to the DNS resolver to obtain its IP address. This saves a lot of time because spiders do not have to send requests to DNS every time they want to download a page.

Robots.txt file: are the means by which web authors express their wish as to which pages they want the crawlers to avoid. Crawlers must respect authors' wishes as well.

Spider: downloads robots.txt file and other pages that are requested by the crawler manager and permitted by web authors. The robots.txt files are sent to crawler manager for processing and extracting? The URLs. The other downloaded files are sent to a central indexer.

Link Extractor: look through the pages downloaded by the spiders, extracts URLs from the links in those pages and sends the URLs to the crawler manager for downloading afterwards.

Any crawler must fulfill following two issues:

1) It must have a good crawling strategy

2) It has to have a highly optimized system architecture that can download a large number of pages per seconds.

Most of search engines use more than one crawler and manage them in a distributed method. This has following benefits:

- Increased resource utilization
- Effective distribution of crawling tasks with no bottle necks
- Configurability of the crawling tasks

Creating the URL Frontier

A URL frontier is the collection of URLs that the crawler intends to fetch and process in the future. URL frontiers generally work in one of two ways - a batch crawl or a continuous crawl. A batch crawl's frontier contains only new URLs, whereas a continuous crawl's can contain a seed set of URLs but new ones may be added (or existing ones removed) during the crawl.

Regardless of the type of crawl, the data store used to manage the frontier needs to be fast because it will slow down crawling otherwise. Below is a diagram that helps illustrate the URLs in a crawl frontier.

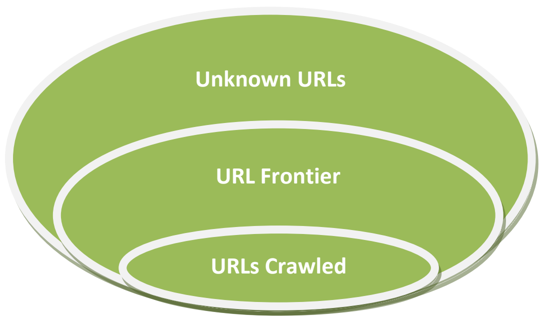

Figure2 – URL Frontier or Seed

Crawling is quite simple at its core:

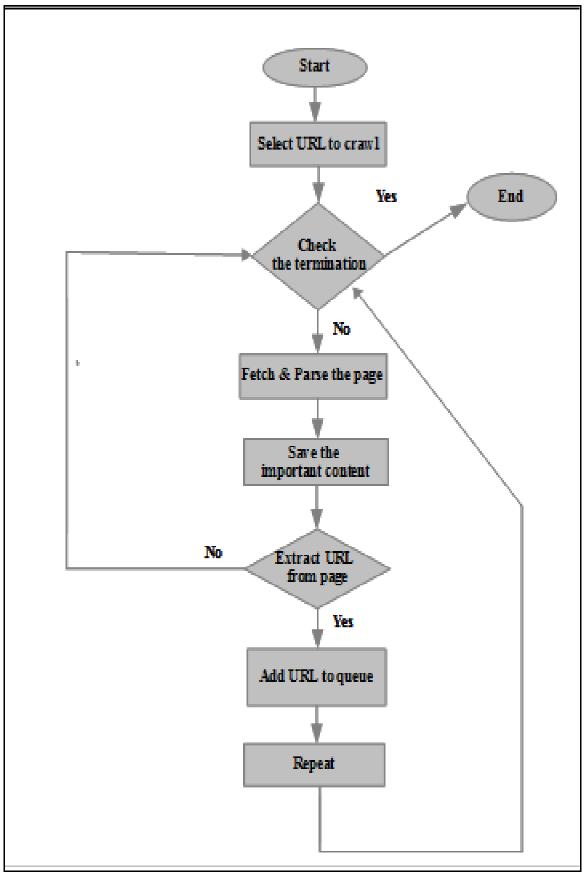

Figure 2 - Process for Web Crawling

Crawling and Scheduling

When it comes to crawling there are many details that allow one to do this quickly and at scale. If your crawl is less than 100 million pages, and you have time such that you don't need to distribute the crawling across servers, then crawling and scheduling is more manageable. The coordination of distributed crawling across several servers can get difficult quickly (and since it is much more complicated I will have to leave those details for a future post).

When it comes to fetching and parsing a page, there are several steps involved for each URL:

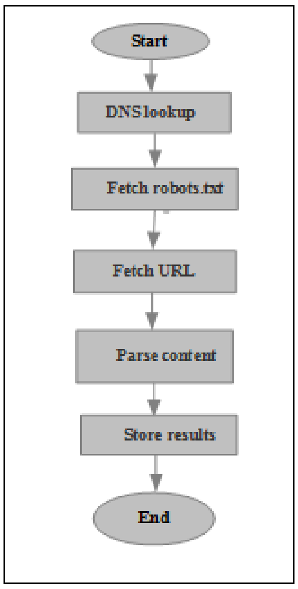

Figure 3 - Steps for parsing the URL

For the DNS lookup and robots.txt rules one can likely cache these such that any subsequent URLs from the same IP will be much faster and won't require a trip across the network.

Scheduling is not just about selecting which URLs to crawl, but is also about tracking the URLs that you have crawled so far. If you have a small crawl (<100 Million URLs), it probably makes sense to store these in memory otherwise your application performance can suffer reading from disk or across the network. It is possible to store URLs in plain text, although using a bloom filter (http://en.wikipedia.org/wiki/Bloom_filter) to hash URLs makes reading and writing much faster.

Extracting the information

Once the crawler is configured the last piece of the puzzle is extracting and structuring the data from different web pages.

The most important part of parsing web pages is that your parser is able to deal with messy markup and will be resilient to errors. Chances are you will want some smart logic that removes the page chrome (like navigation, headers/footers, search boxes, advertisements, etc.) so that only data is returned. It is also important to consider more than ASCII text, as it is common to run across Unicode URLs and content.

If you are trying to crawl the entire web there are more things to consider as the Internet is full of junk such as enormous or never-ending pages, with spider traps that look like dynamic content but are actually an infinite generation of links.

Once you have the html content it is necessary to navigate the page and DOM to find the specific parts of the page relevant or important to your cause. And then of course storing that information in a structured form, which may mean converting it to JSON, or transforming it to fit a schema and inserting it into the database. For extracting the information we are defining a analytical method.

Analytical Method: we have crawled the data and use an analytical methodology that spans domains, business/subject objectives, and information sources. Our method has three major phases:

1. Explore
2. Understand
3. Analyze

Each of these phases having the different capabilities that build on each other. However, it is not always necessary to use every phase or capability.

Explore: Internet is a sea of huge information and many times, we are in search of large repository of data. Depending on our information source and our business objective, not all of the information will be relevant. For example, if we are interested in analyzing the Web to understand details around a specific organization, we only need the part of the Web that pertains to that organization. Another example, in research, suppose we are analyzing references related to a domain area, we only need the references that are relevant to that paper. We use a combination of techniques to locate the relevant set of information from a larger set. With structured information, we can use various queries; for unstructured information, we can use search, and we can combine them in different ways using various set operations. We call this process *Explore*.

Queries: We use *seek* as the term for the query to describe how we build the structured definitions in a database to select the sub-area of records that is of relevant area. For example, we can select employees based upon their location, the product they have purchased, or the time frame that we wish to investigate. We can select papers based upon their authors, the references, or the category code. These are all typically structured informations that are stored in a database. These types of "seek "queries are quite simple to perform using the standard SQL query language to find the sub-collections of interest. This technique is very powerful and effective, given you have the appropriate attributes in the database and know which of their values will select the subset that is relevant to address the issue being analyzed.

Search is the process of finding those documents that contain specific words or statement in their unstructured text. We use search as the means to find collections of documents that have concepts of interest within them, rather than to find individual documents. Although it is a valuable tool, search is not the solution to all problems. The use of language does not always lend itself to easily disambiguate concepts. Some words have more than one meaning, known as homonyms. For example, using "shell" as a query will likely return information on sea shells, Shell Oil, Unix shell, egg shells, and many others. Disambiguation is one problem, but coverage is another. Some meanings can be described with more than one word, known as synonyms. For example, we have found that valium has more than 150 unique names—have fun typing that query.

Set operations: Because in many cases no single query or search is sufficient to get to the optimally desired collection for deeper analysis, we have found it necessary to be able to perform set operation on collections. The most commonly used operations are join and intersect. Joins are useful in combining multiple searches for synonyms. Intersection is useful when you are looking for the subset that has two attributes that could be from either the structured or unstructured fields. In some cases, when the result of a combination of queries and searches is still too large to effectively analyze in a reasonable time, sampling techniques may be used to select a statistically valid subset.

Recursion and expansion: Results of queries and searches can be used as input to subsequent Explore operations. This allows us to refine the subject of our mining study incrementally as we learn more about the data. Also, we can use query expansion to take the results of a query done on a subset of the data and apply it to the entire data collection.

Understand:

The result of the *Explore* phase is a collection of information that covers the topic of interest. The *Understand* phase is about discovering what the information contains. We have developed a unique method of creating structure from unstructured information through the process of taxonomy generation and refinement. We use a combination of practical steps, statistical techniques, algorithms, and a methodology for editing taxonomies that allows for the flexible capture of domain expertise and business objectives. We call this process *Understand*. The Understand process works in two directions: the analyst understands the underlying structure inherent in the unstructured information, and the models captured as a result of the analyst's edits represent an understanding of domain knowledge and business objectives.

Statistics are fundamental to our *Understand* process. We are all familiar with the idea of summarizing numerical data with statistical techniques. For example, a grade point average is a way to summarize your overall academic performance. It doesn't tell you everything, but most people have agreed that it is a pretty good indicator. What about something more complex like a sporting event? Pick your favorite sport—whether it is baseball, basketball, tennis, or football—and there are usually various ways to summarize the game or match that allow you to understand the essence of what transpired. Such summaries are no substitute for watching the game, but they can convey a lot of information about the game in a very small space.

Partitioning: If you have a large body of text, there is probably one or more natural ways to partition it into smaller sections. A book naturally falls into chapters, and each chapter into paragraphs, and each paragraph into sentences, just as a baseball game has innings and innings have outs. Breaking a large document into smaller entities makes it much easier to summarize the message of the text as a whole, because it makes statistics possible. If we try to summarize a baseball game without breaking it down by innings and outs, we are left with only the final score. But if we can break down the game into innings and measure what happened during each of these smaller units (e.g., hits, walks, outs), then we can create meaningful statistics such as *Earned Run Average or on Base Percentage*.

There may be many suitable ways to do partitioning, each with its own advantage. However, the best methods for partitioning are those that produce a section that talks about only one concept with respect to the questions we want answered. The level of granularity should roughly match that of the desired business result. The analogy in baseball is that we measure innings for pitchers and at-bats for batters. The different levels of granularity make sense for different kinds of outcomes that need to be measured. Similarly, if we want to understand the issues for which customers are calling into a call center, then individual problem records, which may span multiple calls, are the right partitioning. On the other hand, if we wish to understand better what affects customer satisfaction, we may decide to analyze each individual call record. A customer might be both satisfied and dissatisfied during the course of resolving an issue, and we want to isolate te interactions in order to analyze the underlying causes.

Feature selection: Once the partitioning granularity is properly adjusted, we need to decide what events we are going to measure and what statistics we will keep. In a baseball box score, we don't measure everything about the game. For example, we don't know the average number of swings each batter took, or the number of pitches each pitcher threw. We could measure these things if they were important to us, but that level of detail is not interesting to the average baseball fan. Similarly in statistical analysis of text, we could measure the average number of times each letter of the alphabet occurs. We could measure the average word length, or the number of words in sentences. In fact, such statistics are used as a means for roughly measuring how "readable" a section of text will be for readers of various grade levels. However, these kinds of statistics are not helpful to answer typical business questions, such as "What are my customers most unhappy about?" So what are the right things to measure about each text example? The answer is, it all depends. It depends on what we want to learn and what kind of text data we are dealing with. Word occurrence is a good place to start for most types of problems, especially those where you don't have much specific domain knowledge to draw upon and where the language of the documents is fairly general. When the text is more technically dense or focused on a very specialized area, then it may make sense to also measure sequences of words, also known as phrases, to get a more precise kind of statistic.

We use word and phrase occurrence as the features of a document. However, we don't use every word and phrase, because there can be a very large number of them and they are not all meaningful or useful. We use a combination of techniques to reduce the feature space to a more manageable size. We eliminate non-content–bearing words, called stopwords, such as "and" and "the." We also remove repetitive or structural phrases (we call them stock phrases). If every document contains "Copyright IBM" or "IBM Confidential," then it can safely be removed. We also combine features using a synonym list. This can be done manually where deemed appropriate or automatically through a technique called stemming. Stemming allows "jump," "jumping," and "jumped" to be treated as one. There are also various domain-specific synonym lists that can be used where stemming will fall short. Finally, we remove features that occur infrequently in the document collection because these tend to have little value in creating meaningful categories. Once we have reduced the features to a manageable size, we can use this to create summary statistics for each document. We call the collection of all such statistics for every document in a collection the *vector space model*.

Clustering: We use *clustering* to quickly and easily seed the process of taxonomy generation. Clustering is an algorithmic attempt to automatically group documents into thematic categories. These thematic categories, which together constitute taxonomy, give an overview of what information the document collection contains. There are many different clustering algorithms that could be used, and our approach could support them all. However, we have relied heavily on variations of the k-means algorithm, because it is fast and does a reasonable job. We have also developed our own algorithm, which we call *intuitive clustering*, that we also employ.

Taxonomy editing: Clustering is a wonderful tool, but we rarely find it to be sufficient. No matter how good the algorithmic approach to clustering becomes, it cannot embed the nuance of business objectives and the variations of language from different information sources within an algorithm. This is the critical missing element that our method incorporates. We have developed a unique set of capabilities that allow for an analyst or domain expert to quickly assess the strengths and weaknesses of taxonomy and easily make the changes necessary to align the taxonomy with business objectives.

Analyst knowledge about the purpose of the taxonomy trumps every other consideration. Thus, a category may be created by an analyst for reasons that have nothing to do with text features. An example would be a category of "recent" documents—those created most recently out of all the documents in the corpus. Depending on the business analysis goals, such a category may be very important in helping to understand emerging trends and issues.

Ideally, the name of a category should describe exactly what makes the category unique. An analyst may decide to change a system-generated name to one that is better aligned with the analyst's view of what the category contains. This category renaming process thus becomes an important way that domain expertise is captured.

In addition to the name, a category can also be described by choosing examples that best summarize the overall content. We describe these as "Typical Examples" because they are selected by virtue of having all or most of the features that typify the documents in the category as a whole. Using the vector space model, it is possible to automatically compare examples and select those that have the most typical content. By reading and understanding typical examples, it is possible for the analyst to make sense of a large collection of documents in a relatively short period of time.

It is also important to measure the variation within a category of documents. If there is a statistically large variation among the documents within a category, this may indicate that the category needs to be split up, or subcategorized. We call the metric that measures within category variation *cohesion*. Additionally, it is important to measure the similarity between categories. We call this *distinctness*. Categories with low distinctness scores indicate a potential overlap with another category. This overlap may indicate the need to merge two or more categories together.

The categories created using clustering and summarized with various statistics can also be edited based on this understanding. This is where analysts adds their domain knowledge and awareness of the business problem to be solved to the results—creating categories that are more meaningful.

There are many kinds of editing that are typically employed, at all levels of the text categorization. Categories can be merged or deleted. They can be created wholesale from documents matching individual words, phrases, or features. Categories can be edited—splitting off subsets of a category to create new categories. Documents can be selectively removed from one category and placed in another.

The taxonomy editing process can be thought of as the human expert training the computer to understand concepts that are important to the business. There may be many different types of categorizations that can be created on the same set of data, each representing a different important aspect of the information to the enterprise.

Visualizations: The visual cortex occupies about one third of the surface of the cerebral cortex in humans. It would be a shame to waste all of that immense processing power during the *Understand* process. We employ visualizations of taxonomies to create pictures of the information that the human brain can process in order to locate areas of special interest that contain patterns or relationships. There are many types of visualizations that can be used to show relationships in structured and unstructured information. Scatter plots, trees, bar graphs, and pie charts can all help in the process of understanding the information, and in modifying taxonomies to reflect business objectives.

The vector space model of feature occurrence in documents is the primary data source for automatically calculating visual representations of text. Using this representation, a document becomes not just words, but a position or point in high-dimensional space. Given this representation, a computer can "draw" a set of documents and allow a human analyst to explore the text space in much the same way an astronomer explores the galaxy of stars and planets.

Analyze:

At the end of the *Understand* phase, we have one or more taxonomies that represent characteristics of the unstructured information, along with a feature set that describes the individual documents that make up each taxonomy. But a taxonomy by itself rarely achieves the business objectives of mining unstructured information. The final step is to take combinations of structured and unstructured information and look for trends, patterns, and relationships inherent in the data and use that to make better business decisions. We call this process *Analyze*.

Trending: Timing is everything in comedy, in life, and in business. Knowing how categories occurred in the data stream over time will often reveal something interesting about why that category occurs in the first place. Trend analysis in data mining is also useful for detecting spikes in categories as well as in predicting how categories will evolve in the future. Trending can be interesting from a historical perspective, but it is usually most valuable when used to detect emerging events. If you can detect a problem in your business before it costs you a lot of money, which goes straight to the bottom line. If you can spot a trend before your competition, you have a leg up.

Correlations: Taxonomies capture the concepts embedded in unstructured information. Co-occurrence analysis reveals hidden relationships between these concepts and other attributes or between categories of different taxonomies. For example, we can look for a relationship between technology areas and companies to see where our competition is investing. Or we can find a correlation between a specific factory and a certain kind of product defect.

A correlation is based on the simple idea that two different phenomenon have occurred together more than expected. For example, if 100 customers who talked to a specific call center representative ended up dissatisfied with their overall customer experience, then depending on the total percentage of unsatisfied callers and the total percentage of calls that particular representative took, we could calculate whether there was a correlation between dissatisfaction and talking to this representative. Keep in mind that, even if there is such a correlation, it doesn't mean that this representative is actually responsible for the poor customer satisfaction. It could be that this person only works during weekends and that people who call on the weekends are generally more dissatisfied. This example serves to show that correlations are not causes. They are simply indicators of potential explanations that should be explored further. Think of them as "leading indicators" of business insights.

Classification : One you have a taxonomy that models an important aspect of the information, it is important to be able to apply this classification scheme to new unstructured data. Many classification algorithms exist, and we have incorporated a large variety of them into our approach, allowing us to select the best algorithm for a given taxonomy and information collection. The specific of how we do text classification is a more technical subject that is beyond the scope of this book. However, the general approach is to pick the algorithm that most accurately represents each category, based on a random sampling of the documents in that category being used to test the accuracy of each modeling approach.

K-Means Algorithm:

In our approach we have used K-Means Cluster based algorithm. K-Means is a widely use clustering algorithm. In that we use random seeds data and according to that arrange the clusters.

Given *k*, the *k-means* algorithm is implemented in 4 steps:

- Partition objects into *k* nonempty subsets
- Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
- Assign each object to the cluster with the nearest seed point.
- Go back to Step 2, stop when no more new assignment.

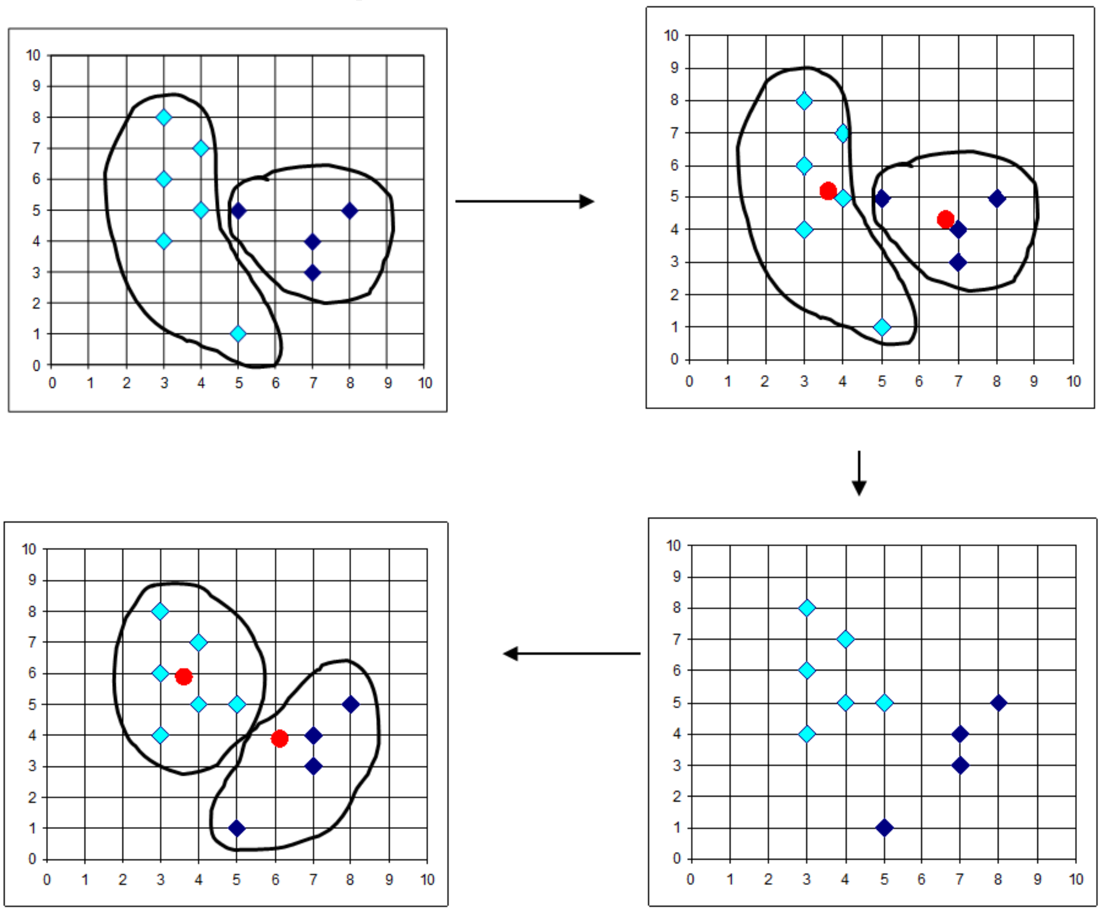

Figure -5 Clustering Process using K-Means Algorithm

CONCLUSION

In this paper we have proposed architecture for the web-crawling using focused crawler techquies. This paper also shows the method to convert unstructured data into structured data. K-Means algorithm is widely used algorithm which clusters the relevant data, and with this technique we can cluster the group of data in very effective manner.

REFERENCES

[1.] http://searchsoa.techtarget.com/definition/crawler
[2.] http://en.wikipedia.org/wiki/Web_crawler
[3.] http://cacm.acm.org/blogs/blog-cacm/153780-data-mining-the-web-via-crawling/fulltext
[4.] S. Chakrabarti. *Mining the Web*. Morgan Kaufmann, 2003.
[5.] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11{16):1623{1640, 1999.
[6.] W3C for Meta TAG Description [w3c.org]
[7.] http://en.wikipedia.org/wiki/Web_crawler
[8.] Dr. Vijay Singh Rathore and Nilesh Jain, “An Approach To Measure Research Related Domains Using Web-Mining Techniques” ,Journal of global research in computer science, 04-08