Enhanced Automatically Mining Facets for Queries and Clustering with Side Information Model

K. Vidhya and N. Saravanan

Abstract--- *In this paper describe a specific type of summaries that Query facet the main topic of given text. Existing summarization algorithms are classified into different categories in terms of their summary construction methods (abstractive or extractive), the number of sources for the summary (single document or multiple documents), types of information in the summary (indicative or informative), and the relationship between summary and query (generic or query-based). QD Miner aims to offer the possibility of finding the main points of multiple documents and thus save users' time on reading whole documents. The difference is that most existing summarization systems dedicate themselves to generating summaries using sentences extracted from documents. In addition, return multiple groups of semantically related items, while they return a flat list of sentences. In this paper, adding these lists may improve both accuracy and recall of query facets. Part-of-speech information can be used to check the homogeneity of lists and improve the quality of query facets. The side-information could not be incorporate into the mining process, because it can either improve the quality of the representation for the mining process, or can add noise to the process. Therefore, a principle way is required to perform the mining process, so as to maximize the advantages from using this side information. This dissertation proposes an algorithm which combines classical partitioning algorithms with probabilistic models in order to create an effective clustering approach.*

Keywords--- *Data Mining, Classification, TF-IDF, K-Mean Clustering, Statistical Mean Validation.*

I. INTRODUCTION

DATA Clustering is an automatic learning technique aimed at grouping a set of objects into subsets or clusters. The goal is to create clusters that are coherent internally, but substantially different from each other. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters. The aim of this thesis is to improve the efficiency and accuracy of document clustering. In this proposed system two clustering algorithms and the fields where these perform better than the known standard clustering algorithms.

K. Vidhya, M.Tech Student, Department of Information Technology, K.S.R College of Engineering, Tiruchengode, Tamilnadu, India. E-mail: vidhyait1994@gmail.com

N. Saravanan, Assistant Professor, Department of Information Technology, K.S.R College of Engineering, Tiruchengode, Tamilnadu, India.

DOI:10.9756/BIJSESC.8387

In other words, the goal of a good document clustering scheme is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). A distance measure (or, dually, similarity measure) thus lies at the heart of document clustering.

Clustering is the most common form of unsupervised learning and this is the major difference between clustering and classification. No super-vision means that there is no human expert who has assigned documents to classes. In clustering, it is the distribution and makeup of the data that will determine cluster membership. Clustering is sometimes erroneously referred to as automatic classification; however, this is inaccurate, since the clusters found are not known prior to processing whereas in case of classification the classes are pre-defined.

In clustering, it is the distribution and the nature of data that will determine cluster membership, in opposition to the classification where the classifier learns the association between objects and classes from a so called training set, i.e. a set of data correctly labeled by hand, and then replicates the learnt behavior on unlabeled data.

The goal of a document clustering scheme is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). A distance measure (or, dually, similarity measure) thus lies at the heart of document clustering. The large variety of documents makes it almost impossible to create a general algorithm which can work best in case of all kinds of datasets.

Challenges in Document Clustering

Document clustering is being studied from many decades but still it is far from a trivial and solved problem. The challenges are:

- Selecting appropriate features of the documents that should be used for clustering.
- Selecting an appropriate similarity measure between documents.
- Selecting an appropriate clustering method utilising the above similarity measure.
- Implementing the clustering algorithm in an efficient way that makes it feasible in terms of required memory and CPU resources.
- Finding ways of assessing the quality of the performed clustering.

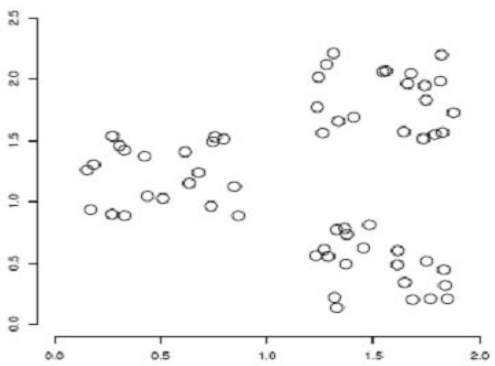

Figure 1.1: Cluster Structure

II. LITERATURE SURVEY

Weize Kong and James Allan [1] describe a Faceted search helps users by offering drill-down options as a complement to the keyword input box, and it has been used successfully for many vertical applications, including ecommerce and digital libraries. However, this idea is not well explored for general web search, even though it holds great potential for assisting multi-faceted queries and exploratory search. In this paper, explore this potential by extending faceted search into the open-domain web setting, which is call Faceted Web Search. To tackle the heterogeneous nature of the web, propose to use query-dependent automatic facet generation, which generates facets for a query instead of the entire corpus. To incorporate user feedback on these query facets into document ranking, we investigate both Boolean filtering and soft ranking models.

Krisztian Balog, Edgar Meij and Maarten de Rijke et al [2] explore the potential of combining IR with SW technologies to improve the end-to-end performance on a specific entity search task. We arrive at and motivate a proposal to combine text-based entity models with semantic information from the Linked Open Data cloud. The problem of entity search has been and is being looked at by both the Information Retrieval (IR) and Semantic Web (SW) communities and is, in fact, ranked high on the research agendas of the two communities. The entity search task comes in several flavors. One is known as entity ranking (given a query and target category, return a ranked list of relevant entities), another is list completion (given a query and example entities, return similar entities), and a third is related entity finding (given a source entity, a relation and a target type, identify target entities that enjoy the specified relation with the source entity and that satisfy the target type constraint.

Chengkai Li, Ning Yan et al. Et al [3] describes a faceted retrieval system for information discovery and exploration in Wikipedia. Given the set of Wikipedia articles resulting from a keyword query, Facetedpedia generates a faceted interface for navigating the result articles. Compared with other faceted retrieval systems, Facetedpedia is fully automatic and dynamic in both facet generation and hierarchy construction, and the facets are based on the rich semantic information from Wikipedia.

Wisam Dakka, Panagiotis G. Ipeirotis [4] describe a text-annotated data constitute a significant fraction of the information available in electronic form. Searching and browsing are the typical ways that users locate items of interest in such databases. Faceted interfaces represent a new powerful paradigm that proved to be a successful complement to keyword searching. Thus far, the identification of the facets was either a manual procedure, or relied on apriori knowledge of the facets that can potentially appear in the underlying collection. In this paper, we present an unsupervised technique for automatic extraction of facets useful for browsing text databases. In particular, observed through a pilot study, that facet terms rarely appear in text documents, showing that we need external resources to identify useful facet terms. For this, first identify important phrases in each document. Then, expand each phrase with "context" phrases using external resources, such as WordNet and Wikipedia, causing facet terms to appear in the expanded database.

Amaç Herdagdelen et al [5] describe a novel approach to query reformulation which combines syntactic and semantic information by means of generalized Levenshtein distance algorithms where the substitution operation costs are based on probabilistic term rewrite functions. We investigate unsupervised, compact and efficient models, and provide empirical evidence of their effectiveness. Further it explores a generative model of query reformulation and supervised combination methods providing improved performance at variable computational costs. Among other desirable properties, our similarity measures incorporate information-theoretic interpretations of taxonomic relations such as specification and generalization.

X. Xue and W. B. Croft et al [6] describe a Query reformulation modifies the original query with the aim of better matching the vocabulary of the relevant documents, and consequently improving ranking effectiveness. Previous models typically generate words and phrases related to the original query, but do not consider how these words and phrases would fit together in new queries. In this paper, a novel framework is proposed that models reformulation as a distribution of queries, where each query is a variation of the original query. This approach considers a query as a basic unit and can capture important dependencies between words and phrases in the query. Previous reformulation models are special cases of the proposed framework by making certain assumptions.

Idan Szpektor, Aristides Gionis, Yoelle Maarek et al [7] describe the ability to aggregate huge volumes of queries over a large population of users allows search engines to build precise models for a variety of query-assistance features such as query recommendation, correction, etc. Yet, no matter how much data is aggregated, the long-tail distribution implies that a large fraction of queries are rare. As a result, most query assistance services perform poorly or are not even triggered on long-tail queries. We propose a method to extend the reach of query assistance techniques (and in particular query recommendation) to long-tail queries by reasoning about rules between query templates rather than individual query transitions, as currently done in query-flow graph models.

III. PROPOSED METHODOLOGY

A. Query Facets Mining

A is query facet is a set of items which describe and summarize query one important aspect of a query. Here a facet item is typically a word or a phrase. A query may have multiple facets that summarize the information about the query from different perspectives. To automatically mining query facets from the top retrieved documents, QDMiner which discovers query facets by aggregating frequent lists within the top results.

Important information is usually organized in list formats by websites. They may repeatedly occur in a sentence that is separated by commas, or be placed side by side in a well-formatted structure (e.g., a table). This is caused by the conventions of webpage design. Listing is a graceful way to show parallel knowledge or items and is thus frequently used by webmasters. Important lists are commonly supported by relevant websites and they repeat in the top search results, whereas unimportant lists just infrequently appear in results. This makes it possible to distinguish good lists from bad ones and to further rank facets in terms of importance.

B. Overview of the System

In the proposed approach, given a query q, retrieve the top K results from a search engine and fetch all documents to form a set R as input. Then, query facets are mined by: List and context extraction Lists and their context are extracted from each document in R. “men’s watches, women’s watches, luxury watches ...” is an example list extracted.

List weighting All extracted lists are weighted, and thus some unimportant or noisy lists, such as the price list “299.99, 349.99, 423.99 ...” that occasionally occurs in a page, can be assigned by low weights. List clustering Similar lists are grouped together to compose a facet. For example, different lists about watch gender types are grouped because they share the same items “men’s” and “women’s”.

Facet and item ranking Facets and their items are evaluated and ranked. For example, the facet on brands is ranked higher than the facet on colors based on how frequent the facets occur and how relevant the supporting documents are. Within the query facet on gender categories, “men’s” and “women’s” are ranked higher than “unisex” and “kids” based on how frequent the items appear, and their order in the original lists.

C. Extracting List and Context

From each document d in the search result set R, extract a set of lists $L_d = \{l'\}$ from the HTML content of d based on three different types of patterns, namely free text patterns, HTML tag patterns, and repeat region patterns. For each extract list, we extract its container node together with the previous and next sibling of the container node as its context.

Free Text Patterns

In the free text patterns, extract all text within document d and split it into sentences. We then employ the pattern item{, item}* (and | or) {other} item, to extract matched items from each sentence. We name this sentence based pattern as TEXT. Further use the pattern {^item (:|-) .+$}+ to extract lists from some semi-structured paragraphs. It extracts lists from continuous lines that are comprised of two parts separated by a dash or a colon.

For a list extracted by the pattern $TEXT_S$, its container node is the sentence containing the extracted list.

HTML Tag Patterns

In the Html tag patterns, extract lists from several list-style HTML tags, including SELECT, UL, OL, and TABLE. It is named these simple HTML tag based patterns as HTMLTAG. SELECT For the SELECT tag, we simply extract all text from their child tags (OPTION) to create a list. Moreover, we remove the first item if it starts with some predefined text, such as “select” or “choose”. UL/OL For these two tags, we also simply extract text within their child tags (LI).

TABLE, it extracts one list from each column or each row. For a table containing m rows and n columns, extracts at most m þ n lists. For each column, the cells within THEAD or TFOOT tags are regarded as table headers and are dropped from the list. For a list extracted from a HTML element like SELECT, UL, OL, or TABLE by pattern $HTML_{TAG}$, its context is comprised of the current element and the previous and next element if any.

Repeat Region Patterns

The peer information is sometimes organized in well-structured visual blocks in web pages. Each block contains a restaurant record that includes four attributes: picture, restaurant name, location description and rating. In this method, extract three lists from this region: a list of restaurant names, a list of location descriptions, and a list of ratings.

To extract these lists, we first detect repeat regions in Web pages based on vision-based DOM trees. Here a repeat region is the region that includes at least two adjacent or nonadjacent blocks, e.g., M blocks, with similar DOM and visual structures. And then extract all leaf HTML nodes within each block, and group them by their tag names and display styles.

The names in the web page have the same tag name (<a>) and displaying style (in blue color), and they can be grouped together. Each group usually contains M nodes. Each two of them are from different blocks. At last, for each group, we extract all text from its nodes as a list.

For a list extracted from a repeat region, choose the lowest common ancestor element of all blocks of the repeat region as a container node (i.e., the smallest element containing the entire repeat region).

D. List Weighting

Some of the extracted lists are not informative or even useless. Some of them are extraction errors. The lists are navigational links which are designed to help users navigate between web pages. They are not informative to the query. The list is actually an extraction error: several types of information are mixed together. Then it is dispute that these types of lists are useless for finding facets. To should punish these lists, and rely more on better lists to generate good

facets. The system finds that a good list is usually supported by many websites and appears in many documents, partially or exactly. Finally, sort all lists by final weights for the given query.

E. List Clustering

In the system, do not use individual weighted lists as query facets because: (1) an individual list may inevitably include noise. For example, in the above Table 1, "watch brands" is noise. It is difficult to identify it without other information provided; (2) an individual list usually contains a small number of items of a facet and thus it is far from complete; (3) many lists contain duplicated information. They are not exactly same, but share overlapped items. To conquer the above issues, it group similar lists together to compose facets. Two lists can be grouped together if they share enough items. This means that two groups of lists can only be merged together when every two lists of them are similar enough.

The weight of a cluster is computed based on the number of websites from which its lists are extracted. More specifically, wc=|Sites (c)| where Sites(c) is the set of websites that contain lists in c. Note that, use websites instead of web pages because web pages from the same website usually share the same page templates and contribute duplicated lists. After that the clustering process, similar lists will be grouped into a candidate query facet.

F. Facet Ranking

After the candidate query facets are generated, it is evaluate the importance of facets and items, and rank them based on their importance. A facet c is more important if: the lists in c are extracted from more unique content of search results; and the lists in c are more important, i.e., they have higher weights. Here it is highlighted unique content, because sometimes there are duplicated content and lists among the top search results.

Unique Website Model

Because of the same website usually deliver similar information, multiple lists from a same website within a facet are usually duplicated. A simple method for dividing the lists into different groups is checking the websites they belong to. It is assume that different websites are independent, and each distinct website has one and only one separated vote for weighting the facet. i.e, let C (c) = Site(c) and recall that Sites (c) is the set of unique websites containing lists in c.

Context Similarity Model

In the Unique Website Model, the system used website as a simple signal for creating groups. Here it is assumed that lists from a same website might contain duplicated information, whereas different websites are independent and each can contribute a separated vote for weighting facets. Mirror websites are using different domain names but are usually publishing duplicated content. Different websites may publish content using the same software.

G. Item Ranking

In a facet, the importance of an item depends on how many lists contain the item and its ranks in the lists. As a better item is usually ranked higher by its creator than a worse item in the original list, calculate $S_{e|c}$, the weight of an item e within a facet c, by:

$$S_{e|c} = \sum_{G \in C(c)} \frac{1}{\sqrt{AvgRank\ c, e, g}}$$

where w(c,e,G) is the weight contributed by a group of lists G, and $AvgRank_{c,e,G}$ is the average rank of item e within all lists extracted from group G.

For each query, it first ask a subject to manually create facets and add items that are covered by the query, based on knowledge after a deep survey on any related resources (such as web sites related to the query).

H. Clustering with Side Information

The clustering text data with side information is a corpus S of text documents. The total number of documents is N, and they are denoted by T1 ... TN. It is assumed that the set of distinct words in the entire corpus S is denoted by W. Associated with each document Ti have a set of side attributes Xi. Each set of side attributes Xi has d dimensions, which are denoted by (xi1 ... xid). We refer to such attributes as auxiliary attributes. For ease in notation and analysis, we assume that each side-attribute xid is binary, though both numerical and categorical attributes can easily be converted to this format in a fairly straightforward way.

This is because the different values of the categorical attribute can be assumed to be separate binary attributes, whereas numerical data can be discretized to binary values with the use of attribute ranges. Some examples of such side-attributes are as follows:

- In a web log analysis application, we assume that xir corresponds to the 0-1 variable, which indicates whether or not the ith document has been accessed by the rth user.
- This information can be used in order to cluster the web pages in a site in a more informative way than a techniques which is based purely on the content of the documents. As in the previous case, the number of pages in a site may be large, but the number of documents accessed by a particular user may be relatively small.
- In a network application, corresponds to the 0-1 variable corresponding to whether or not the ith document Ti has a hyperlink to the rth page Tr.
- If desired, it can be implicitly assumed that each page links to itself in order to maximize linkage-based connectivity effects during the clustering process. Since hyperlink graphs are large and sparse, it follows that the number of such auxiliary variables are high, but only a small fraction of them take on the value of 1.
- In a document application with associated GPS or provenance information, the possible attributes may

be drawn on a large number of possibilities. Such attributes will naturally satisfy the sparsity property.

I. Content and Auxiliary Attribute [COATES Algorithm]

Content and auxiliary attribute-based text classification algorithm. The algorithm uses a supervised clustering approach in order to partition the data into k different clusters. This partitioning is then used for the purposes of classification. The steps used in the training algorithm are as follows:

- Feature Selection: In the first step, we use feature selection to remove those attributes, which are not related to the class label. This is performed both for the text attributes and the auxiliary attributes.
- Initialization: In this step, we use a supervised k-means approach in order to perform the initialization, with the use of purely text content. The main difference between a supervised k-means initialization, and an unsupervised initialization is that the class memberships of the records in each cluster are pure for the case of supervised initialization. Thus, the k-means clustering algorithm is modified, so that each cluster only contains records of a particular class.
- Cluster-Training Model Construction: In this phase, a combination of the text and side-information is used for the purposes of creating a cluster-based model. As in the case of initialization, the purity of the clusters in maintained during this phase.

Once the features have been selected, the initialization of the training procedure is performed only with the content attributes. This is achieved by applying a k-means type algorithm as discussed to the approach, except that class label constraints are used in the process of assigning data points to clusters. Each cluster is associated with a particular class and all the records in the cluster belong to that class. This goal is achieved by first creating unsupervised cluster centroid, and then adding supervision to the process. In order to achieve this goal, the first two iterations of the k-means type algorithm are run in exactly the clusters are allowed to have different class labels. After the second iteration, each cluster centroid is strictly associated with a class label, which is identified as the majority class in that cluster at that point. In subsequent iterations, the records are constrained to only be assigned to the cluster with the associated class label. Each iteration for a given document, its distance is computed only to clusters which have the same label as the document. The document is then assigned to that cluster. This approach is continued to convergence. The algorithm requires two phases:

- **Initialization:** We use a lightweight initialization phase in which a standard text clustering approach is used without any side-information. For this purpose the algorithm described. The centroids and the partitioning created by the clusters formed in the first phase provide an initial starting point for the second phase. We note that the first phase is based on text only, and does not use the auxiliary information.
- **Main Phase:** The main phase of the algorithm is executed after the first phase. This phase starts off with these initial groups, and iteratively reconstructs these clusters with the use of both the text content and the auxiliary information. This phase performs alternating iterations which use the text content and auxiliary attribute information in order to improve the quality of the clustering. We call these iterations as content iterations and auxiliary iterations respectively. The combination of the two iterations is referred to as a major iteration and each major iteration thus contains two minor iterations, corresponding to the auxiliary and text-based methods.

IV. CONCLUSION

The novel COATES Algorithm is used for high dimensional data. The algorithm involves removing irrelevant features based data preprocessing process and selecting representative features. In the proposed algorithm, a cluster consists of features. Each cluster is treated as a single feature and thus dimensionality is drastically reduced.

The significantly better results are found for the proposed method compared to existing methods, irrespective of the classifiers used. All the results reported in this paper demonstrate the feasibility and effectiveness of the proposed method. It is capable of identifying co-regulated clusters of genes whose average expression is strongly associated with the sample categories. The identified gene clusters may contribute to revealing underlying class structures, providing a useful tool for the exploratory analysis of biological data.

REFERENCES

[1] W. Kong and J. Allan, “Extending faceted search to the general web”, Proc. ACM Int. Conf. Inf. Knowl. Manage., Pp. 839–848, 2014.

[2] E.M. Balog and M. de Rijke, “Entity search: Building bridges between two worlds”, Proc. 3rd Int. Semantic Search Workshop, Pp. 9:1–9:5, 2010.

[3] C. Li, N. Yan, S.B. Roy, L. Lisham and G. Das, “Facetedpedia: Dynamic generation of query-dependent faceted interfaces for Wikipedia”, Proc. 19th Int. Conf. World Wide Web, Pp. 651–660, 2010.

[4] W. Dakka and P.G. Ipeirotis, “Automatic extraction of useful facet hierarchies from text databases”, Proc. IEEE 24th Int. Conf. Data Eng., Pp. 466–475, 2008.

[5] A. Herdagdelen, M. Ciaramita, D. Mahler, M. Holmqvist, K. Hall, S. Riezler and E. Alfonseca, “Generalized syntactic and semantic models of query reformulation”, Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. retrieval, Pp. 283–290, 2010.

[6] X. Xue and W.B. Croft, “Modeling reformulation using query distributions”, ACM Trans. Inf. Syst., Vol. 31, No. 2, Pp. 6:1–6:34, 2013.

[7] W.L. Bing, T.L. Wong and S. Jameel, “Web query reformulation via joint modeling of latent topic dependency and term context”, ACM Trans. Inf. Syst., Vol. 33, No. 2, Pp. 6:1–6:38, 2015.

[8] A.G. Szpektor and Y. Maarek, “Improving recommendation for long-tail queries via templates”, Proc. 20th Int. Conf. World Wide Web, Pp. 47–56, 2011.

[9] M. Damova and I. Koychev, “Query-based summarization: A survey”, Proc. S3T, Pp. 142–146, 2010.

[10] K. Latha, K.R. Veni and R. Rajaram, “Afgf: An automatic facet generation framework for document retrieval”, Proc.Int. Conf. Adv. Comput. Eng., Pp. 110–114, 2010.

[11] J. Pound, S. Paparizos and P. Tsaparas, “Facet discovery for structured web search: A query-log mining approach”, Proc. ACM SIGMOD Int. Conf. Manage. Data, Pp. 169–180, 2011.

[12] W. Kong and J. Allan, "Extracting query facets from search results", Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, Pp. 93–102, 2013.
[13] Y. Liu, R. Song, M. Zhang, Z. Dou, T. Yamamoto, M.P. Kato, H. Ohshima and K. Zhou, "Overview of the NTCIR-11 imine task", Proc. NTCIR-11, Pp. 8–23, 2014.