DynaCet: Building Dynamic Faceted Search Systems over Databases

Senjuti Basu Roy[#], Haidong Wang[#], Ullas Nambiar[*], Gautam Das[#] and Mukesh Mohania[*]

[#]*Dept of Computer Sci. and Engg., University of Texas at Arlington. Arlington, TX, USA.*
{roy,haidong.wang,gdas}@cse.uta.edu

[*]*IBM India Research Lab. New Delhi, India.*
{ubnambiar,mkmukesh}@in.ibm.com

***Abstract*— Extracting information and insights from large databases is a time-consuming activity and has received considerable research attention recently. In this demo, we present DynaCet - a domain independent system that provides effective minimum-effort based dynamic faceted search solutions over enterprise databases. At every step, Dynacet suggests facets depending on the user response in the previous step. Facets are selected based on their ability to rapidly drill down to the most promising tuples, as well as on the ability of the user to provide desired values for them. The benefits provided include faster access to information stored in databases while taking into consideration the variance in user knowledge and preferences.**

I. Introduction

Facilitating effective search for data records within vast data warehouses is one of the primary challenges in recent years. For example, the warehouse of a large financial institution such as a bank contains information about its customers, their accounts and transactions and so on. Data is usually organized into multiple tables in such cases, each with numerous attributes (facets). These databases are used by enterprise users having diverse needs and expertise. For example, a data analyst might look for insights into customer behavior and hence needs to browse through a large subset of the database, while a customer service representative may only need to extract few tuples to answer a customer query. In either case, users have to formulate queries to get the required information. Of course if the relevant tuple is uniquely identifiable by an identifier, this problem is trivial. But in most real applications the user only has partial information about the tuple (e.g., perhaps the values of a few of its attributes) and thus it is necessary to enable an effective search procedure.

In recent years, faceted search [1] has received considerable research attention as an alternative to traditional querying mechanisms (SQL for database and keyword search for IR systems). One of primary reason is its ability to let a user iteratively define the intended query by adding or removing constraints (facet bindings) while browsing the data. This considerably reduces the burden on the user, as she needs not create the perfect query upfront - often a problem most users face when querying over third-party data sources whose underlying data distribution is hidden from the user. In particular, faceted interfaces are now available in many e-commerce web sites such as Amazon.com[1] and Ebay[2], over bibliographic databases like DBLP[3] and for browsing document collections[4]. However, current solutions for faceted search require predefined taxonomies over the facets. This makes the solutions too time consuming to deploy as one must first develop a universally acceptable taxonomy. In addition, they work primarily for small number of facets, typically in the range of 15-20[5]. This problem becomes particularly difficult in enterprise databases where each database can have hundreds of attributes (facets). Current faceted search solutions become ineffective in such cases as they need to have a built-in taxonomy and also do not easily support large number of facets. In fact, during evaluations, we found static faceted search interface to be inefficient and cumbersome for exploring databases with large facet sizes such as the Yahoo Auto database [6] which has large number of attributes.

In this demo, we present *DynaCet* [2]- a middleware system that sits between the user and the database and dynamically suggests facets for drilling down into the database. The facet suggestion model is driven by our intent to provide a *minimum-effort* database exploration solution for enterprise users. We focus on a simple but intuitive metric for measuring effort: *the expected number of queries that the user has to answer in order to reach the tuples of interest.*

The DynaCet Approach: The faceted search techniques developed by DynaCet will present the user with a set of queries after every refinement step - where each query consists of an attribute name and to which the user responds with a value from its domain. Our solution assumes a one-to-one mapping between attributes in the database and facets displayed to the user. Ideally, the refinement process terminates when a unique tuple has been isolated. We will elaborate our solution using the example below.

Example: Consider a user searching for a movie in a MovieDB(Name, Year, Genre, Director, Actor, Color, Language). The user may be interested in seeing a specific movie, but may only know a few of the attributes (such as an "action"

[1]http://www.amazon.com/
[2]http://www.ebay.com/
[3]http://www.l3s.de/growbag/demonstrators.php
[4]http://flamenco.berkeley.edu/demos.html
[5]http://www.miskatonic.org/library/facet-web-howto.html
[6]http://autos.yahoo.com/

movie by “Bruce Willis”); thus a search is necessary to narrow down the choices. □

A very simple faceted search interface is one where the user is prompted an attribute (e.g., Actor), to which she responds with a desired value (e.g., “Bruce Willis”), after which the next appropriate attribute (e.g., Genre) is suggested to which she responds with a desired value (e.g., “Action”), and so on. Thus the *first challenge* is to judiciously select the facets to be suggested, so that the user reaches the desired tuple(s) with *minimum effort*. The task then is to develop a faceted search solution that minimizes the average number of facets shown. Variants of this problem have been considered in the context of interactive question-answering systems [3]. We adopt a simple approximation algorithm from [3] based on constructing a decision tree with minimum average height (construction of an optimal decision tree is shown to be NP-complete). Each node of the tree represents an attribute, and each edge leading out of the tree is labeled with a value from the attribute’s domain.

However, current solutions (such as [3]) for building one such tree assumes that a user is able to answer questions on any attribute equally well. This is inapplicable in many situations; for example, a movie might be best disambiguated by specifying its Cinematographer, but in most instances the user is unlikely to know the right value to bind this attribute. Hence, the *second challenge* we have to deal with is the problem of uncertainty in user knowledge in the facet selection process. In particular, the uncertainty over an attribute refers to the probability of the user being able to provide a desired value for that attribute. We solve this challenge by extending the decision tree model [3] to account for such attribute uncertainties.

Our solution also works when the underlying data source provides a *ranked list* of tuples. This is a novel problem area in faceted search and makes DynaCet the first solution for this problem. We view the ranking as imposing a *skew* over the user preferences for the selected tuples, and thus DynaCet would select the facet that directs the user towards the most preferred tuples as efficiently as possible. This introduces further complications since these tuple preferences (or ranks) may change as the faceted search progresses.

An additional challenge is raised by the fact that the facet generation process must have real-time response. This happens when the user begins by asking an SQL or keyword query and then wishes to switch to a faceted interface for browsing the result. Dynacet provides scalable implementations of all the algorithms that uses principles of the Rainforest [4] framework to avoid multiple scans over the database for building scalable decision trees.

II. The DynaCet System

The architecture of DynaCet and the flow of information through the system is illustrated in Figure 1. The front-end of the system is a web-based user interface which enables user to build queries and provides navigational access into the database. The back-end consists of two components, the *Facet Component* and the *Ranking Component*. DynaCet is domain independent and requires read-only access to the underlying database, thus making it implementable over any database system.

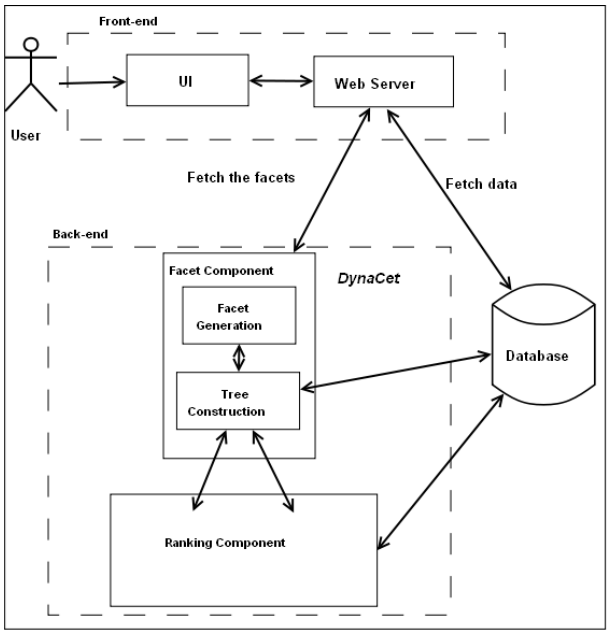


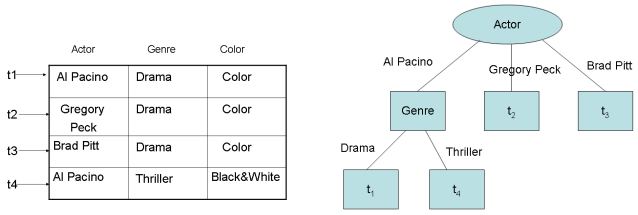
Fig. 1. Architecture of DynaCet

We have implemented our algorithms by leveraging the scalable decision tree framework Rainforest [4]. The *Facet Generation* module supports two modes of exploration over the facets - *Browse Only* and *Search and Browse*. In the *Browse Only mode*, a typical browsing session begins by showing suggested facets to the user. A user simply needs to select one of the facet values in order to move on to the next step in browsing. In this mode, the entire database is to be explored, hence the facet generation module uses pre-computed decision trees. However, for the *Search and Browse* mode, a more dynamic scenario is investigated. Here, a user can typically begin her search session by specifying one or more of her preferences in the form of a query. Next, the resultant tuple set is retrieved by DynaCet and faceted search is enabled on that set. Hence, in this case, decision trees are constructed online over search results. Essentially, we build a partial tree with a few “look ahead” nodes and then stay in sync with the user while she is exploring the partially constructed tree. Each of these two above mentioned mode can also work in conjunction with a Ranking component, where the Ranking module imposes a *skew* over the user preferences for the selected tuples. Different problem variants of DynaCet are discussed in more detail in [2].

A. Dynamic Single-Facet Generation

Essentially, a minimum-effort driven dynamic faceted search solution will involve building a decision tree that identifies each tuple unambiguously by testing the attribute values. Each node of the decision tree represents an attribute, and each edge leading out of the node is labeled with a value

from the attribute's domain as shown in Figure 2(b) (a toy movie database example).



(a) A small movies database (b) An optimal decision tree

Fig. 2. A Small Movie Database and an Optimal Decision Tree

The intuition behind constructing such a decision tree is to make the attribute that disambiguates the maximum number of pairs of tuples as the root of the tree, where an attribute A_l is said to disambiguate a pair of tuples t_i, t_j if $t_i[l] \neq t_j[l]$. Picking the attribute A_l as the root node partitions the database D into disjoint tuple sets $D_{x_1}, D_{x_2}, \ldots, D_{x_{|Dom_l|}}$, where each D_{x_q} is the set of tuples that share the same attribute value x_q of A_l and Dom_l is the domain[7] of A_l. Using this intuition, we seek to select as root attribute A_l that minimizes the function

$$Ambiguous(A_l, D) = \sum_{1 \leq q \leq |Dom_l|} |D_{x_q}|(|D_{x_q}|-1)/2 \quad (1)$$

This process is recursively repeated for all sets D_{x_q}, until each set reduces to a single tuple. We incorporate the uncertainty in user knowledge by assuming that users can respond to a question by either (a) providing the correct value for an attribute, or (b) with a "don't know". In either case, the system responds by suggesting a fresh attribute (facet). Detailed solution approaches of DynaCet can be found at [2].

B. Suggesting Multiple Facets

The solution above presents users with a single new facet at each step. To improve search effectiveness, it may be preferable to present the user with multiple facets at each step.

m-Facets Selection: DynaCet can suggest a set of m-Facets at every step where m is defined by the user. However, a user is restricted to select only one of the m facets for further refinement. The motivation behind such model is to reduce the probability that a user will follow "don't know" links. Specifically, given the set of m attributes A'' at the root, the probability that a user will be unable to answer any of the m questions is $\prod_{A_l \in A''}(1-p_l)$, where p_l is the probability that the user is able to provide the value for A_l. This cumulative value is smaller than the "don't know" likelihood for each attribute. Hence, we expect the navigation to be more efficient here.

Fixed m-Facets Interface: In certain applications, users would prefer to be presented with a single static interface, in which a reasonably large number of attributes are shown, and the user assigns values to as many of the presented attributes as she can. If the space available on the interface is restricted such that only m attributes can be shown (where m is less than the total number of attributes), the task is then to select the best set of m attributes such that the expected number of tuples that can be disambiguated via this interface can be maximized. Our Fixed m-Facets interface is designed towards achieving this goal.

C. Supporting Skew introduced by Ranked Answers

We also explore whether faceted search procedures can work *in conjunction with* ranking functions. This is a novel problem area, and to the best of our knowledge, has not been investigated before. Given a query Q, a ranking function typically assigns relevance scores $S(Q,t)$ to all selected tuples t, and a ranked-retrieval system will score and return only the top-k tuples.

In our approaches, we make one assumption: that the scores are normalized so that they are (a) positive, and (b) $\sum_{t \text{ selected by } Q} S(Q,t) = 1$. In other words, the ranking function can be imagined as inducing a non-uniform "probability distribution" over the selected tuples, such that $S(Q,t)$ represents the probability that tuple t is preferred by the user.

We propose solutions to this problem by developing a greedy heuristic that is motivated by our facet selection approaches presented earlier. Since the problem is NP-hard even without a ranking function, this problem too is intractable. Assume that we are at a particular node v of the decision tree. Let Q be the current query at that node. Thus Q is the initial query at the root, concatenated (i.e., AND'ed) with all conditions along the path from the root to v. Let D be the set of tuples of the database that satisfy Q. In this case, for any attribute A_l, function $Ambiguous(A_l, D)$ needs to be modified as follows:

$$Ambiguous(A_l, D) = \sum_{x_q \in Dom_l} \left(\sum_{t_i, t_j \in D_{x_q}, i<j} S(Q,t_i) \times S(Q,t_j) \right) \quad (2)$$

The quantity $\sum_{t \in D_{x_q}} S(Q,t)$ is the cumulative scores of all tuples in D_{x_q} and intuitively represents the probability that when the user is at the root, she will prefer any of the tuples in D_{x_q}.

Extensions to selecting multiple facets at each step are similar.

III. Demonstration

In this demonstration we will showcase DynaCet's domain independent approach for efficiently generating dynamic faceted search interfaces over databases. Figure 3 shows three views of a faceted search interface developed over the IMDB [8] database. We will give an end-to-end demonstration of the DynaCet system's ability to generate minimum-effort faceted search solutions that support *Single Facet, m-Facets and Fixed m-Facets* suggestion algorithms. We will use IMDB and Yahoo Auto dataset for this demonstration. In DynaCet, our

[7] We assume only categorical attributes; thus numeric attributes are required to be discretized in a pre-processing step.

[8] http://www.imdb.com

Fig. 3. Screen shot of DynaCet GUI

local copy of IMDB database (nearly 20 attributes) and Yahoo Auto dataset (more than 40 attributes) can be queried by the user. The upper right half of Figure 3 shows the parameter setting interface of DynaCet through which users will be able to provide their choices. In general, the demonstration will focus upon two broad aspects of search - *Browse Only* and *Search and Browse*.

Browse Only Mode: In this model, the user does not initiate the search with a query - rather DynaSet will recommend facet(s) for her. Consequently, a user is shown m different facets, to which she responds by selecting a value from one of the facet domain (or a "don't know"). Depending upon the user response, the next set of facets are dynamically suggested and the process repeats. The lower half of Figure 3 shows the interface from a typical browsing session over IMDB using DynaCet.

This model takes the advantage of a pre-computed decision tree and thus results in good response time. We will also provide a comparative evaluation of our proposed solutions with some existing attribute selection techniques in the demonstration. The user will be allowed to choose the m-Facets or the Fixed m-Facets algorithm in this mode.

Search and Browse Mode: In this mode the user will be able to start the exploration phase by providing a query through the form interface. The upper left part of Figure 3 shows the interface for querying. The query form shows only a few attributes from among the total set of attributes. A larger set can be seen by going to *Advanced Search*. In the figure, the user has asked for movies with Language="English" and Color="Color" which then results in only three dynamically generated facets being shown to the user.

Faceted Search in Conjunction with Ranking Function: The user will have an option of choosing an appropriate ranking function from the list of available ranking functions (provided as a drop-down list). A comparative cost evaluation between our proposed solution and a prior attribute ordering method [5] will also be shown.

IV. Summary

In this demonstration, we present DynaCet - a domain independent system that provides effective minimum-effort based dynamic faceted search solutions over enterprise databases. DynaCet's contributions include - (1) an efficient approach for generating facets for minimum-effort navigation over enterprise databases and (2) extensions that use uncertainty models over attributes as well as skew in tuple preference introduced by ranked-retrieval models.

References

[1] E.Stoica, M.Hearst, and M.Richardson, "Automating creation of hierarchical faceted metadata structures," in *NAACL-HLT*, 2007.

[2] S. Roy, H. Wang, G.Das, U.Nambiar, and M.K.Mohania, "Minimum-effort driven dynamic faceted search in structured databases," in *CIKM*, 2008.

[3] V.T.Chakravarthy, V.Pandit, S.Roy, P.Awasthi, and M.Mohania, "Decision trees for entity identification: Approximation algorithms and hardness results," in *PODS*, 2007.

[4] J.Gehrke, R.Ramakrishnan, and V.Ganti, "Rainforest - a framework for fast decision tree construction of large datasets," in *DMKD*, 2000.

[5] G.Das, V.Hristidis, N.Kapoor, and S.Sudarshan, "Ordering the atributes of query results," in *SIGMOD*, 2006.