Abstract—This paper presents IQP - a novel approach to bridge the gap between usability of keyword search and expressiveness of database queries. IQP enables a user to start with an arbitrary keyword query and incrementally refine it into a structured query through an interactive interface. The enabling techniques of IQP include: (1) a conceptual framework for incremental query construction; (2) a probabilistic model to assess the possible informational needs represented by a keyword query; (3) an algorithm to perform an optimal query construction.

I. INTRODUCTION

Keyword search requires neither a-priori knowledge of the database schema nor query construction skills and can be performed efficiently by novice users. However, this search paradigm lacks expressiveness to precisely describe a user’s informational need, and may return irrelevant or incomplete results. To cope with the limitations of keyword search, some recent approaches [4, 6, 7, 8] translate a keyword query into a ranked list of structured queries, such that the user can select the one that represents her informational need. However, the number of possible structural interpretations of a keyword query is typically large, such that ranking alone is not sufficient to efficiently identify the desired structured query. Moreover, even a theoretically optimal ranking algorithm can, at best, rank the most common informational needs highest, whereas the users with less frequent informational needs may not receive adequate results. For example, if the majority of users who issued the keyword query “London” in a book store were interested in a city guide of London, the results referring to Jack London as a book author will receive a low rank.

In this paper we present IQP, a novel system to bridge the gap between usability of keyword search and expressiveness of database queries. IQP allows a user to start with an arbitrary keyword query, and then incrementally refines the keywords into the desired structured query through an interactive interface. For instance, to find movies acted by Tom Hanks and produced in 2001, Alice issues a keyword query “Tom Hanks 2001”. IQP translates this query into a ranked list of structured queries, which give different interpretations to Alice’s keywords (Fig.1 (3)). Simultaneously, IQP generates a set of query construction options (Fig.1 (2)). If Alice cannot immediately identify the intended structured query in the query window, she can incrementally construct the query by selecting the proper query construction options. For instance, if Alice selects an option “Tom Hanks is an actor’s name”, the query window zooms into the subset of structured queries satisfying this option. At the same time, IQP presents another set of the query construction options to enable Alice to further clarify her informational need.

For instance, in [2, 9] we presented SUITS, a faceted interface enabling users to disambiguate the semantics of keyword queries. However SUITS lacks a theoretical foundation for verifying its effectiveness. With QUICK [10], we developed a formal framework for incremental query construction and applied it to Semantic Web data. Compared with the previous work, the contributions of IQP include: (1) a probabilistic framework that formally defines the process of incremental query construction; (2) a probabilistic model to estimate the probabilities of structural query interpretations; (3) an algorithm for generating an optimal query construction plan based on Information Gain, which enables a user to obtain the intended structured query with a minimal set of interactions.
II. QUERY CONSTRUCTION FRAMEWORK

IQP translates a keyword query to a set of structured queries and guides users through an interactive query construction process using a query construction plan.

A. From Keywords to Structured Queries

In the context of a relational database, a structured query is an expression of relational algebra. To translate a keyword query to a structured query, IQP first obtains a set of keyword interpretations, which map each keyword to an element of an algebraic expression. It then joins the keyword interpretations using a predefined query template [1, 3], which is a structural pattern that users usually employ to query the database.

For example, to translate a keyword query K="actor hanks 2001", IQP can create keyword interpretations \( \sigma_{\text{actor}} \), \( \sigma_{\text{hanks}} \), and \( \sigma_{2001} \) where 'actor' is interpreted as a table name, and 'hanks' and '2001' are interpreted to the values of two predicates. Finally, IQP can use the template \( T=\sigma_{\text{name}}(\text{Actor}) \bowtie \text{Acts} \bowtie \sigma_{\text{year}}(\text{Movie}) \) to join the keyword interpretations into the structured query \( \sigma_{\text{actor}} \bowtie \text{Acts} \bowtie \sigma_{2001} \).

We call the structured query resulted from the translation process described above a query interpretation. If a query interpretation Q contains keyword interpretations for all the keywords in a keyword query K, we call Q a complete interpretation of K. Otherwise we call Q a partial interpretation. Each complete interpretation of a keyword query is a possible structured query desired by the user. Each partial interpretation can be used as a query construction option. Given a keyword query K, the interpretation space of K is the entire set of the complete interpretations of K.

During the query construction process, IQP utilizes the subsumption relationships between the partial and complete query interpretations to reduce the interpretation space of a keyword query. Given a complete interpretation Q and a partial interpretation Q', we say that Q' is a sub-query of Q (or Q' subsumes Q), iff Q' is a sub-structure of Q. By selecting Q' as a correct partial interpretation, a user can eliminate all the complete interpretations that are not subsumed by Q' from the interpretation space.

B. Query Construction Plan

As shown in Fig.1, in each query construction step a user is presented with a list of query construction options, i.e. partial interpretations. The user selects the first option that subsumes the intended structured query. For simplicity we assume that the user decides on only one option at a time. If the option subsumes the intended query, the user accepts it, and the interpretation space of the keyword query can be reduced accordingly. The user evaluates the options one after another, until the intended structured query can be uniquely determined. This process can be modelled as a binary decision tree, called Query Construction Plan (QCP). In the plan, each node represents the remaining interpretation space of the query construction process, and the out-edges of each node represent the acceptance and the rejection of a partial interpretation.

III. PROBABILISTIC MODEL AND ALGORITHM

To obtain an optimal query construction plan, IQP needs: (i) a probabilistic model for estimating probabilities of query interpretations, and (ii) an algorithm to generate the MQCP.

A. Estimating Query Probability

Probability of Query Interpretation: Given a keyword query K, let the structured query Q be a complete interpretation of K. Then, \( P(Q|K) \) is the probability that the leaf is the user intended query interpretation. A QCP is a minimum query construction plan (MQCP) for the keyword query K, iff there is no other QCP of K which can offer a lower interaction cost.

Fig.2. Query Construction Plan as a Binary Tree

A fragment of a query construction plan for the keyword query K="hanks 2001" is shown in Fig.2. A query construction process is a traversal of the query construction plan. The user starts from the root of the plan. At each node, the user decides on the query construction option. After accepting or rejecting this option, the user moves to the node pointed by the corresponding edge. The process continues until the user reaches a leaf node, which is the desired structured query. It can be proved that such a binary tree can be uniquely transformed to an N-ary tree corresponding to the user interface in Fig.1, in which a user is supposed to select the first option matching her intent.

The key issue of query construction in IQP is to find an optimal plan, which enables the user to obtain the intended query with as less effort as possible. We measure the interaction cost of a query construction plan by the expected number of query construction options a user has to evaluate to construct Q is the depth of Q in the query construction plan. Therefore, the interaction cost of the plan can be calculated by:

\[
\text{Cost}(\text{QCP}) = \sum_{\text{leaf } \in \text{QCP}} \text{depth(leaf)} \times P(\text{leaf})
\]
If a keyword query $K$ has been used repeatedly in a database, we can directly estimate $P(Q|K)$ using the previous interpretations of $K$ in the query log of the database. However, in a large database, it is unlikely to find a sufficient number of records for a particular keyword query. Therefore, we decompose $P(Q|K)$ to a set of atomic probabilities which are much easier to obtain. As stated in Section II.A, a query is composed of a set of keyword interpretations $\{A_i|k_i\}$ and the query template $T$ of $Q$. Thus, the probability $P(Q|K)$ can be transformed to:

$$P(Q|K) = P(A_i|k_i), T|K$$  \hspace{1cm} (2)

To simplify the computation, we assume that (i) an interpretation of each keyword in the query keyword is independent from the other keywords; and (ii) the probability of a keyword interpretation is independent from the part of the query interpretation the keyword is not interpreted to. Based on these assumptions and Bayes’ rule, we can transform Formula 2 to:

$$P(Q|K) \propto \prod_{i,k} P(A_i|k_i)(T \cap A_i) \times P(T)$$  \hspace{1cm} (3)

where $P(T)$ is the prior probability that the template $T$ is used to form a query interpretation. $P(A_i|k_i)(T \cap A_i)$ represents the probability that, given that $T \cap A_i$ is a part of a query interpretation, $A_i|k_i$ is also a part of the query interpretation.

The probability of a partial interpretation $O$, i.e. $P(O|K)$, can be computed similarly as Formula 3.

### Estimating Probability of a Template

If the database possesses a query log that is statistically representative, $P(T)$ can be estimated directly using the log. Namely,

$$P(T) = \frac{\#\text{occurrences}(T)}{N}$$  \hspace{1cm} (4)

where $\#\text{occurrences}(T)$ is the number of queries using template $T$ and $N$ is the total number of queries. When the query log is not available, we assume all templates to be equally probable.

### Estimating Probability of a Keyword Interpretation

In this paper, we focus on two types of keyword interpretations. The first type interprets a keyword as a part of the query template, such as a table name, an attribute name or an operator name. The second type is in the form $\sigma_\in \in A(T):k_i$, where a keyword is interpreted as a value of an attribute. For the first type of interpretation, we can estimate the probability $P(A_i|k_i)(T \cap A_i)$ using query log too. Without a query log, our system can use some empirical values set by domain experts.

When a keyword is interpreted as an attribute value, we estimate probability of this interpretation using statistics obtained from the database instances. We model the formation of a query interpretation as a random process. For an attribute $AT_i$, this process randomly picks one of its instances $a_j$ and randomly picks a keyword $k_i$ from that instance to form the expression $\sigma_\in \in A(T)$. Then, the probability of $P(\sigma_\in \in A(T):k_i|\sigma_\in \in A(T))$ is formed through this random process. This probability can be estimated using Attribute Term Frequency (ATF), which is defined as:

$$\text{ATF}(k_i, AT_i) = P(\sigma_\in \in A(T)|\sigma_\in \in A(T)) = \frac{DF(k_i, AT_i) \times \text{avg}_{a_j \in A_T(k_i,a_j)} TF(k_i, a_j)}{DF(k_i, AT_i)}$$  \hspace{1cm} (5)

In the formula 5, $\text{avg}$ denotes average, $TF(k_i, a_j)$ is the frequency of keyword $k_i$ in an attribute instance $a_j$, and $DF(k_i, AT_i)$ is the frequency of the instances of $AT_i$ that contain $k_i$. The concepts of $TF$ and $DF$ closely correspond to the Term Frequency and Document Frequency used in Information Retrieval if we treat each attribute instance as a document [5].

### B. Query Construction Algorithm

Creation of the optimal query construction plan (MQCP) is a complex problem. Therefore, $IQ^p$ uses a greedy algorithm to construct a near-optimal query construction plan in a top-down fashion. The pseudo-code is presented in Algorithm 1.

```plaintext
Proc greedy_algo(QO, QS)
Input: QO := Query Construction Options; QS := Interpretation Space; Output: Query C := Final Structured Query;
Program:
while(true)
    if |QS| = 1, then //let QS=c return c;
    end if;
    partial_query best_r := null;
    float best_gain := \infty;
    for each R in QS, do
        if IG(QS|R) < best_gain, then
            best_gain = IG(QS|R);
            best_r = R;
        end if;
    end for;
    present R to user;
    if R is accepted, then
        QS := QS - all queries subsumed by R;
    else if R is rejected, then
        QS := QS - all queries subsumed by R;
    end if;
end while;
End Proc;
```

Algorithm 1. A Greedy Algorithm for MQCP

Instead of generating the entire plan, the algorithm generates query construction options one by one. Each time, it searches for the best query construction option, and presents the option to the user. If the user accepts the option, it keeps the query interpretations subsumed by this option and discards the rest. If the user rejects an option, it discards the interpretations subsumed by this option. Following that the algorithm proceeds to generate the next option. This process continues until only the intended structured query is left.

To find the best query construction option to be presented to the user, we select the option that can reveal as much information as possible about the intended structured query. We measure the amount of information by Information Gain (IG), which can be easily calculated using the probabilities of structured queries.
IV. EVALUATION

A. Datasets and Keyword Queries

We used a movie database with seven tables and more than 10M records and a lyrics database with five tables and around 400K records. Both databases were crawled from the Web. We extracted 108 IMDB and 76 lyrics multi-attribute keyword queries from the query log of a Web search engine. For each keyword query we manually constructed the corresponding structured query, serving as ground truth in our evaluation.

B. Effectiveness of the Probability Estimates

To assess how fast $IQ^p$ can enable a user to construct a structured query, we measured the number of query construction options a user needs to evaluate to obtain the intended structured query. Our experiments were performed in an automatic way. Based on the ground truth interpretations established a-priori, we let our system automatically accept the correct options and reject the incorrect options. The process of query construction stops when less than five complete query interpretations are left in the query window, in which the user is able to quickly identify the intended query.

To estimate the effectiveness of the proposed probabilistic model, we used three variations of probability estimates. The first variation, also called base line, assumes that all structured queries and query construction options are equally likely. The second variation, referred as (ATF, Tequal), applied the probabilistic model introduced in Section III-A. It uses the Attribute Term Frequency (ATF) to estimate $P(A_i|k_i\rightarrow A_j)$, but assumes equal probabilities of query templates. The third version, represented by (ATF, TLog), not only used ATF to estimate $P(A_i|k_i\rightarrow A_j)$, but also used the query log to estimate the probabilities of the query templates.

The experiment results for IMDB and Lyrics are shown in Fig.3a and Fig.3b respectively. In both figures, each data point on the X-axis represents a keyword query. Each Y-value is the number of query construction options a user needs to evaluate until she identifies the intended query. As shown in Fig.3a, for the IMDB dataset, using the base line estimate, a user needs to evaluate one to 20 query construction options. In more than 50% of the cases, the interaction cost is below ten. In more than 80% of the cases, the interaction cost is below 15. Occasionally, the cost can reach 20. By estimating the probabilities of structured queries, the interaction cost can be significantly reduced. As shown by the lines of (ATF, Tequal) and (ATF, TLog), in more than 70% of the cases, a user needs to evaluate at most five options to create a structured query. In most of the cases, the interaction cost falls below ten.

We observed a similar trend in the Lyrics results (Fig.3b). Using the baseline estimation, the interaction cost ranges between zero and 15. By applying our probabilistic model, especially by using the probability estimation (ATF, TLog), the interaction cost is reduced by around 50%. Both Fig.3a and Fig.3b show that Attribute Term Frequency (ATF) is a highly effective statistics. The usage-based probability estimation of the templates is also useful. However, its effectiveness differed in the two datasets.

ACKNOWLEDGMENT

This work has been partially supported by the FP7 EU Project OKKAM (contract no. ICT-215032).

REFERENCES