Exemplar Queries
A New Way of Searching

Davine Mottin
Yannis Velegrakis

Matteo Lissandrini
Themis Palpanas
Topics

- What is an Exemplar Query?
- Motivating Example
- Features
- Exemplar Query Modeling
- Algorithms for Evaluating EQ
- Experimental Evaluation and Findings
- Conclusions
What is an Exemplar Query?

- **Definition**: A query paradigm that considers a user query as an example of the data in which the user is interested.

- The user knows one single element among those that are expected to be in the desired result set.

- It works as an example of what the elements of interest that are expected to be returned by the search engine are.
What is an Exemplar Query?

- It is **not** “query by example”, which is used simply to communicate to the query evaluation engine the conditions in a more user-friendly way.

- It is **not** “query relaxation”, which aims at producing more generic versions of a query.

- These queries have some of the properties of the original query at their core.
Motivating Example

- Consider a student who needs to perform a study on company acquisitions in the Bay area
- So he writes a query which contains the words “acquisitions” and “Bay Area”
Motivating Example

- An article on the takeover of del.icio.us by Yahoo! may not be returned if “acquisition” and “Bay area” are not explicitly mentioned in the text...
- He actually knows about the acquisition of YouTube by Google
- But searching for this acquisition will not result for others, such of del.icio.us by Yahoo!
Motivating Example

• Furthermore, he doesn’t look for acquisitions in the kind of Opel by General Motors (GM), namely because:
  – Google and Yahoo! Are both in California, while GM was founded in Michigan
  – IT companies against automotive industry

• Therefore, we need a method for inferring the set of elements that the user is interested in from his single sample
Features

- Suitable for various of users:
  - Student
  - Curious citizen
  - Investigator
  - Lawyer
  - Reporter

- Can form the basis of a new form of search engines that uses them as the main query evaluation mechanism

- Can be used to enhance the services that existing search engines are currently offering
Exemplar Query Modeling

- Evaluating EQ is divided into two steps:
  1. Evaluate the user query and identify its structure in the data repository
  2. Examine the data store to find similar structures

- We focus on the second step
Exemplar Query Modeling

- Data store is based on a graph data model with **labels** on the edges

![Diagram](image.png)

**Figure 1: Exemplar Query Evaluation**
Exemplar Query Modeling

- Using graph-isomorphism
- Doing it in a way that takes into consideration also labels on the edges
- Interested only on the $k$ most promising results
- Looking for significantly reducing search time and space

Note: this is just one possibility of modeling. As long as there is a standard query evaluation methodology and some similarity function that can be used that fits a specific use case, the exemplar queries can be answered.
3 algorithms will be discussed:

1. **XQ**: The brute-force solution - finding isomorphic graphs to the query graph - is exponential in nature.

2. **FastXQ**: An efficient iterative pruning schema that pre-computes a representation of the neighborhood of each node.

3. **ApFastXQ**: An approximate algorithm for pruning the search space, keeping only the subgraph portion that is closer to the user-query, i.e., contains the top-k answers.
Basic Definitions:

- $\mathcal{L}$ - infinite set of labels
- $\mathcal{T}$ - infinite set of atomic values
- $\mathcal{O}$ - infinite set of object identifiers
- $\mathcal{V} = \mathcal{T} \cup \mathcal{O}$
- An **object** is a representation of a real world entity or concept and is modeled through an object identifier and a set of attributes for that identifier
- An **attribute** of an object $o \in \mathcal{O}$ is a triple $\langle o, \ell, v \rangle$ where $\ell \in \mathcal{L}$ and $v \in \mathcal{V}$
Algorithms for Evaluating EQ

Basic Definitions:

• A database $D$ is a pair $\langle O, A \rangle$ where $O \subseteq O$ and $A \subseteq O \times L \times (O \cup T)$, both finite

• This way, $D$ can be represented as a graph $G(N, E)$ (or: $\langle N, E \rangle$), where:
  
  $N = \{ n \mid n \in O \lor \exists \langle n', l, n \rangle \in A \}$

  $E = \{ n \rightarrow n' \mid \langle n, l, n' \rangle \in A \}$

• In previous example:
  
  - “Google”, ”YouTube” and “Menlo Park” are nodes in $N$
  - “acquired” and “foundedIn” are edges in $E$
Algorithms for Evaluating EQ

Basic Definitions:

- A database $D$ is **edge-preserving isomorphic** to a database $D'$, denoted as $D \simeq D'$, if there is a bijective function $\mu$ from the nodes of $D$ to the nodes of $D'$ such that for every edge $n_1 \xrightarrow{\ell} n_2$ in $D$, the edge $\mu(n_1) \xrightarrow{\ell} \mu(n_2)$ is in $D'$. 
Algorithms for Evaluating EQ

Basic Definitions:

- A query $Q$ is a database whose graph representation is a connected-directed graph.
- An answer to a query $Q$: $\langle N_Q, E_Q \rangle$ on a database $D$ is any sub-graph $D'$: $\langle N_{D'}, E_{D'} \rangle$ of $D$ that is isomorphic to $Q$, i.e., $D' \simeq Q$, and there is a bijective function $\mu$ such that:
  \[ \forall n_Q \in N_Q, n_{D'} \in N_{D'}: \mu(n_Q) = n_{D'} \Rightarrow n_Q \equiv n_{D'} \]
- Considering edge label only isomorphism as the implementation of this similarity.
Algorithms for Evaluating EQ

Basic Definitions:

- The set of all such sub-graphs, denoted as $\text{eval}(Q)$, is referred to as the answer set of the query.
Algorithms for Evaluating EQ

The Basic XQ Algorithm

- Search for sub-graphs of the database $D$ that are isomorphic to the query. The query is denoted as the user sample $S$: $S = \langle N_S, E_S \rangle$, where $N_S \subseteq N$, $E_S \subseteq E$

1. A node $n_S$ from $S$ is randomly selected to serve as a seed, and is checked against all $n \in D$, which initialize sets with a single node, as potential answers

2. Then, iteratively tries to expand the sets by adding adjacent nodes from $S$ and $D$ and checks if they are still isomorphic

3. If at any point a set becomes isomorphic to $S$, it’s added to a result set
Algorithms for Evaluating EQ

The Basic XQ Algorithm

- Final set is denoted as $\mathcal{Q}$

### Algorithm 1 XQ

**Input:** Database $D$: $\langle N, E \rangle$

**Input:** User Query $Q$

**Output:** Set of relevant answers $\mathcal{Q}$

1. $\mathcal{Q} \leftarrow \emptyset$
2. $S \leftarrow \text{eval}(Q)$
3. $n_s \leftarrow \text{selectARandomNode}(S)$
4. **for each** $n \in N$ **do**
5. \hspace{1em} $A \leftarrow \text{FINDISOMORPHICSUBGRAPH}(S, n_s, D, n)$
6. \hspace{1em} **if** $A \neq \emptyset$ **then**
7. \hspace{2em} $\mathcal{Q} \leftarrow \mathcal{Q} \cup A$
8. \hspace{1em} $\text{Rank}(\mathcal{Q})$
9. **return** $\mathcal{Q}$
Algorithms for Evaluating EQ

FastXQ Algorithm

• **Problem**: Searching for possible matches of the user sample is an expensive operation, for sub-graph isomorphism is an NP-complete problem

• **Solution**: FastXQ uses a better technique for comparing nodes

• It relies on an **IterativePruning** algorithm, which effectively rejects nodes that can’t be in any isomorphic mapping

• It significantly reduces the search space
Algorithms for Evaluating EQ

FastXQ Algorithm

- The idea is to store in advance a compact representation of the neighborhood of each node, i.e., nodes and edges that are at a fixed distance $d$ from each node.
- This provides an effective way to compare nodes, allowing the pruning to remove the non-matching nodes without having to actually visit their neighborhood.
- May lead to false positives, but these can be removed by XQ, which now has less nodes to compare.
Algorithms for Evaluating EQ

FastXQ Algorithm

- Let \( n \in N \) be a node of a database \( D = \langle N, E \rangle \). The node \( n_i \in N \) is a \( d \)-neighbor of \( n \) if there exists a shortest path from \( n \) to \( n_i \) of length at most \( d \). The \( d \)-neighborhood of \( n \), denoted as \( N_d(n) \), is the set of \( d \)-neighbors of \( n \).

- For every node \( n \), label \( \ell \) and distance \( i \):
  \[
  W_{n,\ell,i} = \left\{ n_1 \mid n_1 \xrightarrow{\ell} n_2 \lor n_2 \xrightarrow{\ell} n_1, \ n_2 \in N_{i-1}(n) \right\}
  \]
  - These are the reachable nodes from \( n \) at distance \( i \) and with a path ending with label \( \ell \).

- \( W \) compactly represents the neighborhood of a node.
Algorithms for Evaluating EQ

FastXQ Algorithm

- So, if $W$ is computed for the nodes of $S$, these can now be compared with the nodes of $D$, in order to know in advance which nodes can be pruned.

- A node $n \in N$ of $D = (N, E)$ matches a node $n_s \in N_s$ in the user sample (and therefore is not pruned), if for each label $\ell$ and a distance $i \leq d$, $|W_{n,\ell,i}| \geq |W_{n_s,\ell,i}|$

- This is a way to prune irrelevant nodes from $D$
Algorithms for Evaluating EQ

FastXQ Algorithm

- First, the algorithm selects a node in the sample $S$ as a starting node

- **Starting node selection**: the node $n_s \in N_S$ which has the **minimum sum** of [incoming and outgoing edges (*)] + [connectivity to different labels (**)] i.e., selecting: $\min\{Sel(n)\}$, where:

$$Sel(n) = freq(n)^{(\ast)} + \sum_{i=1}^{d} \frac{1}{i} \sum_{W_{n,\ell,i}} |E^\ell|^{(\ast\ast)}$$

- Less probable the combination of labels for $n_S$ means higher pruning power.
Algorithms for Evaluating EQ

FastXQ Algorithm

- After selecting $n_{min}$ as starting node, the algorithm retrieves all the nodes in the database $D$ that match it.
- $\mu \subseteq N_S \times N$ is the mapping between the user sample and database nodes.
- First candidates are denoted as: $\mu(n_{min})$
- Then it iteratively checks, for each user sample node $n_S$ not yet visited, that each adjacent edge of $n_S$ matches the edges adjacent to the candidate edges of $n \in \mu(n_S)$. 
Algorithms for Evaluating EQ

FastXQ Algorithm

- If it doesn’t, $n$ is removed from the mapping $\mu(n_S)$

- Otherwise, every node $n_1$ adjacent to $n$ becomes a candidate for the user sample node $n_{S'}$, adjacent to $n_S$. So it is inserted into: $\mu(n_S)$

- Finally, the user sample node $n_S$ is marked as visited and removed from the candidate list
Algorithms for Evaluating EQ

**FastXQ Algorithm**

- The correctness of FastXQ is based on the following Theorem:

Let $D = \langle N, E \rangle$ be a database, $S$ a user sample, $\mathbb{N}_d$ and $\mathbb{N}_d^S$ be the $d$-neighborhood of $D$ and $S$ respectively. If there exists a subgraph-isomorphism $\mu: N_S \rightarrow N$, then:

$$\forall n_S \in N_S, \quad \mathbb{N}_d^S(n_S) \subseteq \mathbb{N}_d(n), \text{ where: } n \in N, n \in \mu(n_S)$$
Algorithms for Evaluating EQ

FastXQ Algorithm

- In the worst case, IterativePruning will have to traverse the entire database for each node.
- Thus, the complexity of the algorithm is: \( O(|N| \cdot (|N_S| + |E_S|)) \)
- The reduced-size database is now being fed to the XQ Algorithm.
**Algorithms for Evaluating EQ**

**ApFastXQ Algorithm**

- ApFastXQ is an approximate algorithm for finding solutions to an exemplar query.
- It prunes the search space even more than FastXQ.
- It is doing it by removing in advance nodes that are likely to not be relevant for the user.
- On the previous example, Yahoo!-del.icio.us is more relevant to the user than GM-Opel. ApFastXQ tries to result the first, and prune the second.
- However, some correct answers may be filtered out.
Algorithms for Evaluating EQ

ApFastXQ Algorithm

- ApFastXQ models a portion of the database, called: “Relevant Neighborhood”, which is the subset of nodes with higher proximity to the nodes of the user sample.

- The intuition behind this is that nodes in the graph that are located far from the user sample will be also semantically distant from the user’s intention as expressed in the exemplar query.

- To this end, it uses Personalized PageRank Vector (PPV) principles.
What is **PPV**?

- **PPV** is a personalized view of the importance of pages on the web.
- It relies on PageRank algorithm, redefining importance according to user preferences.
- Meaning, it works as PageRank algorithm, but takes into account the user preferences.
- Rankings of a user’s text-based query results can be biased according to a PPV instead of the global importance distribution.
Algorithms for Evaluating EQ

What is PPV?

- $N$ is the set of pages in the web
- $P$ is the user’s set of preferred pages
- $u$ is a $|N| \times 1$ preference vector, initialized with the user’s preferences: for each page $p \in P$, $u(p)$ denotes the user’s preference for $p$ (and 0 if $p \notin P$)
- Requirement: $\sum_{p \in P} u(p) = 1$ (or: $|v| = 1$)
- $A$ is a $|N| \times |N|$ matrix, where: $A_{i,j} = \begin{cases} \frac{1}{d_{out}(j)}, & j \rightarrow i \\ 0, & \text{else} \end{cases}$
- $A$ is called transition probability matrix
Algorithms for Evaluating EQ

What is PPV?

- Random surfer model simulates a surfer who is looking at page $p'$ at a certain time step, and at the next time step jumps to a random out-neighbor of $p'$

- $c$ is the teleportation probability: at each step, a surfer jumps to a random page $p \in P$ with probability $c$, and with $(1 - c)$ skips on $P$. Typically, $c \approx 0.15$

- For a given preference vector $u$, the PPV equation is written as the following Markov chain:

\[
(*) \quad v = (1 - c)Av + cu
\]
Algorithms for Evaluating EQ

What is PPV?

- A solution $v$ to equation (*) is a steady-state distribution of random surfers, such that a surfer teleports to page $p$ with probability $c \cdot u(p)$, or moves to a random out-neighbor otherwise.

- By Markov Theory, a solution $v$ with $|v| = 1$ always exists and is unique.

- The solution $v$ is the Personalized PageRank Vector (PPV) for preference vector $u$.

- If $u$ is distributed uniformly, i.e., $u = (\frac{1}{|N|}, \ldots, \frac{1}{|N|})$, then $v$ is the global PageRank vector, giving no preference to any pages.
Algorithms for Evaluating EQ

Back to ApFastXQ Algorithm

• RelevantNeighborhood Algorithm uses Adaptive Personalized PageRank Vector (APPV), an extension of PPV to find the Relevant Neighborhood

• It models the computation of PPV, which is used as an estimate of the distances of the nodes in the graph from the subset of nodes in the user sample

• “user preferences” are expressed through the query $Q$

• APPV differs from PPV due to the semantic of edges:
  − Some relationships between nodes are more informative than others
  − User query’s labels should be treated differently, because they represent the user preference
Algorithms for Evaluating EQ

ApFastXQ Algorithm

- $N$ is the number of nodes in database $D$

- $A^D_{i,j} = \begin{cases} I(e^\ell_{i,j}) , & i \xrightarrow{\ell} j \\ 0 , & \text{else} \end{cases}$

  where $I(e^\ell_{i,j})$ is the amount of information carried by $e^\ell_{i,j}$:

  $I(e^\ell_{i,j}) = I(\ell) = \log \left( \frac{1}{P(\ell)} \right) = -\log P(\ell)$,

  where: $P(\ell) = \frac{|E^\ell|}{|E|}$

- $A^S$ is the same, but where only entries for edges whose label appears also in $S$ are assigned a non-zero value
Algorithms for Evaluating EQ

ApFastXQ Algorithm

- $\tilde{A} = A^S + A^D$: $\tilde{A}$ normalized is the transition probability matrix: more relevance is given to edges carrying more information

- $p$ is an $|N| \times 1$ preference vector: $p[i] = \begin{cases} \frac{1}{|N_S|} & , n_i \in N_S \\ 0 & , \text{else} \end{cases}$

- $c$ is the teleportation probability, as at PPV

- The APPV $\nu$ is defined as the stationary distribution of the Markov chain with state transition given by the matrix:

\[ \nu = (1 - c)\tilde{A}\nu + cp \]
Algorithms for Evaluating EQ

ApFastXQ Algorithm

- A threshold $\tau$ determines which nodes will be taken into account.
- The APPV $v$ is returned containing the scores that have been accumulated through each iteration on every node.
- $D' = \langle N_{D'}, E_{D'} \rangle$ contains only those nodes with a score higher than $\tau$ and the edges connected to them.
Algorithms for Evaluating EQ

ApFastXQ Algorithm

Figure 2: A visualization of APPV
Algorithms for Evaluating EQ

ApFastXQ Algorithm

Algorithm 3 RELEVANTNEIGHBORHOOD

Input: User Sample $S : \langle N_{S}, E_{S} \rangle$
Input: Database $D : \langle N, E \rangle$
Input: Teleportation probability $c$
Input: Threshold $\tau$
Output: Subgraph $D' \subseteq D$

1: $\bar{A} \leftarrow$ ADJACENCYNORMALIZED$(D, S)$
2: $p \leftarrow [0] \times N$
3: for each $q_i \in N_{S}$ do
4: \hspace{1em} $p[q_i] \leftarrow 1/|N_{S}|$
5: $v \leftarrow$ COMPUTEAPPV($\bar{A}, p, c, \tau$)
6: $N_{D'} \leftarrow$ NEAREST$(N, v)$
7: $D' \leftarrow$ GETSUBGRAPH$(D, N_{D'})$
8: return $D'$
Algorithms for Evaluating EQ

Ranking Query Answers

• What left to be done is to rank the answers, sort them to the user, and/or select only the top-$k$ candidates

• The score for a node $n$ and label $\ell$ is:

$$\sigma(n, \ell) = \sum_{i=1}^{d} \frac{I(\ell) |W_{n,\ell,i}|}{i^2}$$

• the structural similarity between a node $n_s$ of the user sample $S$ and $n$ is:

$$S(n_s, n) = \frac{\sum_{\ell \in \mathcal{L}} \sigma(n_s, \ell) \sigma(n, \ell)}{\sqrt{\sum_{\ell \in \mathcal{L}} \sigma(n_s, \ell)^2} \sqrt{\sum_{\ell \in \mathcal{L}} \sigma(n, \ell)^2}}$$
Algorithms for Evaluating EQ

Ranking Query Answers

- The final similarity between the nodes is:
  \[ \rho(n_s, n) = \lambda S(n_s, n) + (1 - \lambda)p[n] \]

- \( \lambda \) close to 1 favors results that are mostly similar to the neighbors of the user sample nodes
- \( \lambda \) close to 0 favors results that are close to the original query
- Choosing \( \lambda \) is data dependent
- The final score of a query answer \( \mathcal{Q} \) is:
  \[ finalScore(\mathcal{Q}) = \sum_{n_s \in S} \sum_{n \in \mathcal{Q}} \rho(n_s, n) \]
Experimental Evaluation and Findings

- Full dump of the Freebase knowledge base in August 2012 (denoted “Real”) - fully connected graph of 53 million nodes and 213 million edges
- 90 queries from the AOL (America Online) query log to use as the query test set, and manually mapped them to the knowledge base
- $d = 3$ since it proved to be a reasonable choice
- $\lambda = 0.3$ retrieves most qualitative results
- $\tau = 0.003$
- Comparison to previous works:
  - QueryReformulation
  - EQ-Graph
  - NeMa
Experimental Evaluation and Findings

Usefulness

Figure 3: Percentage of relevant and irrelevant results per query
Experimental Evaluation and Findings

Usefulness

![Bar Chart]

**Figure 4:** Percentage of satisfied and dissatisfied users.
Experimental Evaluation and Findings

Comparisons to Previous Works

Figure 5: Comparison of methods applied to the Exemplar Query task
Experimental Evaluation and Findings

Comparisons to Previous Works

**Figure 6:** Time vs Size of graph with NeMa and our approach (Real dataset).
Experimental Evaluation and Findings

Comparisons to Previous Works

Table 1: Top-5 results with NeMa for “Google YouTube Menlo Park”

<table>
<thead>
<tr>
<th>Q1: Google - YouTube - Menlo Park</th>
<th>Google - YouTube - Menlo Park</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo! - LAUNCH Media - Stanford University</td>
<td>Yahoo! - Musicmatch - Stanford University</td>
</tr>
<tr>
<td>Yahoo! - Right Media - Stanford University</td>
<td>Yahoo! - Inktomi Corporation - Stanford University</td>
</tr>
</tbody>
</table>

Table 3: Results for exemplar query “Google YouTube Menlo Park”

<table>
<thead>
<tr>
<th>Q1: Google - YouTube - Menlo Park</th>
<th>Google - AdMob - Menlo Park</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google - DoubleClick - Menlo Park</td>
<td>Yahoo! - del.icio.us - Santa Clara</td>
</tr>
<tr>
<td>Microsoft - Powerset - Albuquerque</td>
<td>A&amp;E Television - Lifetime Ent. Services</td>
</tr>
</tbody>
</table>
Experimental Evaluation and Findings

Pruning Effectiveness

Figure 7: Pruning time gain distribution (Real dataset).
Experimental Evaluation and Findings

Calibrating Relevant Neighborhood

**Figure 12:** Count vs AP-FastXQ threshold (Real dataset).

**Figure 13:** Time vs AP-FastXQ threshold (Real dataset).
Experimental Evaluation and Findings

Calibrating Relevant Neighborhood

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>( P@1 )</th>
<th>( P@5 )</th>
<th>( P@10 )</th>
<th>( P@50 )</th>
<th>( P@100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.002</td>
<td>1</td>
<td>0.99</td>
<td>0.99</td>
<td>0.85</td>
<td>0.75</td>
</tr>
<tr>
<td>0.003</td>
<td>1</td>
<td>0.97</td>
<td>0.94</td>
<td>0.80</td>
<td>0.73</td>
</tr>
<tr>
<td>0.004</td>
<td>1</td>
<td>0.95</td>
<td>0.93</td>
<td>0.71</td>
<td>0.60</td>
</tr>
<tr>
<td>0.005</td>
<td>1</td>
<td>0.94</td>
<td>0.92</td>
<td>0.66</td>
<td>0.56</td>
</tr>
</tbody>
</table>
Experimental Evaluation and Findings

Scalability

Figure 8: Count vs number of answers (QSize-x dataset).

Figure 9: Time vs number of answers (QSize-x dataset).
Experimental Evaluation and Findings

Scalability

Figure 10: Count vs number of nodes (GSize-x dataset).

Figure 11: Time vs number of nodes (GSize-x dataset).
Conclusions

- Exemplar queries is applied on a graph-based data model
- Requires search for graph-isomorphism in order to evaluate a query
- Can be evaluated by XQ, FastXQ and ApFastXQ
- As it can be confirmed by studies, the proposed system for evaluating EQ is efficient, effective and useful