Diversifying top-k Answers in a Query by Example Setting

Grégory Smits, Marie-Jeanne Lesot, Olivier Pivert, Marek Reformat

Diversifying top-k Answers in a Query by Example Setting

Grégory Smits[1], Marie-Jeanne Lesot[2], Olivier Pivert[3], and Marek Z. Reformat[4]

[1]IMT Atlantique, Lab-STICC, 29280 Plouzané, France
gregory.smits@imt-atlantique.fr
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
marie-jeanne.lesot@lip6.fr
[3]University of Rennes – IRISA, UMR 6074, Lannion, France
olivier.pivert@irisa.fr
[4]University of Alberta, Edmonton, Canada reformat@ualberta.ca

Abstract. For a given data base T and a user query Q, the top-k answers are the k tuples from T that best match Q. The integration of a diversity constraint aims at avoiding returning redundant tuples, that are too similar one to another. This paper addresses the diversification question in the Query By Example setting, especially for approaches that can deal with possibly very different representative examples provided by the user. It proposes a new definition for diversity that depends on the query, in order to guarantee that the result set illustrates the diversity of the representative examples provided by the user, covering all components of the query. The paper proposes a numerical measure to assess diversity in that sense, an algorithm to identify such a diversified top-k set, optimising both the query satisfaction and the diversity measure, as well as its integration into a flexible querying approach.

Keywords: Querying by example, diversified top-k answers

1 Introduction

In order to exploit information stored in Data Bases (DB), users need to interact with the underlying Data Base Management System (DBMS) that relies on a query algebra and a, generally declarative, formal query language. Now most end users are not computer scientists and cannot express their information needs using formal languages. The Query By Example (QBE) paradigm, as introduced in [13], alleviates the query formulation step as the query is only expressed through a few examples of answers the user expects: from these representative examples, the QBE mechanism infers a formal query that can be submitted to the DBMS. This principle can be enriched to allow taking as input some counter-examples as well, i.e. an additional set of unwanted answers [2]. This expression of an information need through several representative examples means that different kinds of answers are acceptable. This implies that the query inferred from the user-provided examples should be of a disjunctive nature.

Providing users with a set of diversified answers generally means that the underlying querying system has to identify k tuples from the DB, k being a hyper-parameter, that best match the query and are not too similar one to another, see e.g. [12] for an overview of this research issue. The objective is to provide a complete view on the possible interesting answers the DB may contain by constraining them to differ one from another. A diversified top-k result set is classically defined as a set of answers that maximizes a pairwise dissimilarity among the returned answers.

This paper addresses this question of diversifying the result set of a query in the case where it has been inferred from representative examples of expected answers. It argues that a QBE setting requires a new definition of the notion of diversity, that has to take into consideration the query and not only the set of candidate answers. More precisely, a result set to a disjunctive query is said to be diversified if it covers all the user-provided representative examples from which the considered query has been inferred. The objective is so to provide users with answers covering all the different examples of answers they are willing to accept.

The contribution of this work are:

- an adaptation of the notion of diversity in a QBE setting,
- an algorithm to provide users with a diversified top-k result set, optimising both the query satisfaction and the diversity measure,
- the technical integration of this approach in a flexible QBE setting that explicitly models the disjunctive component of the user information need.

The paper is structured as follows. After positioning the research issues addressed in this paper wrt. existing works in Section 2, it details in Section 3 the proposed approach, discussing both the underlying notions and an algorithmic solution. Section 4 experimentally shows how this diversification strategy takes place in a QBE implementation, namely the DCQ strategy [7], additionally illustrating the relevance of the results it allows to obtain. Section 5 concludes and draws some perspectives for future works.

2 Related Works

This section positions the proposed approach wrt. existing QBE systems and result diversification strategies.

2.1 The Query-By-Example Paradigm

The QBE strategy, introduced by Zloof [13] in the 70's, aims at easing the interaction of a user with a DBMS [10]: it takes as input i) one or several example tuples provided by the user or ii) user-defined positive or negative evaluation of prototypical examples reflecting the content of the database. This paper focuses on the first case.

Formally, the QBE paradigm considers a queried table T, that may be the result of a join query, whose schema is $\{A_1, \dots A_p\}$. T stores a set of tuples $T =$

$\{t_1, \ldots, t_n\}$ where $t_{i=1..n} \in D_1 \times \ldots \times D_p$ and D_j is the domain of attribute A_j. This paper focuses on the case where the user provides a set $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ of examples to illustrate what he/she is looking for. The examples from $\mathcal{E}$ may be taken from T or be non-obesrved tuples. The QBE system then computes for each candidate tuple $t \in T$ a satisfaction score denoted $s_{\mathcal{E}}(t)$ that quantifies how much t matches the query Q implied by $\mathcal{E}$. As detailed below, existing approaches differ in the way the satisfaction degrees are computed. Without loss of generality, it can be considered to output scores in $[0, 1]$. Two hyper-parameters are used to control the result set: an integer k specifying the number of expected results and $\alpha \in]0, 1]$ a qualitative threshold regarding the satisfaction degree. For any query Q, the result set is defined as $\Sigma_Q^\alpha = \{t \in T / s_{\mathcal{E}}(t) \geq \alpha\}$. Finally, $\Sigma_Q^{k,\alpha}$ denotes the subset of Σ_Q^α containing at most k tuples that best match Q, i.e. with maximal satisfaction degrees. It may happen that $|\Sigma_Q^{k,\alpha}| < k$ when there are less than k answers that satisfy Q with a score of at least α. In the QBE case, the notation is slightly revised as Q is replaced by $\mathcal{E}$, leading to sets of results denoted $\Sigma_{\mathcal{E}}^\alpha$ and $\Sigma_{\mathcal{E}}^{k,\alpha}$ respectively.

Existing approaches to QBE may be categorized into three groups. The first one does not explicitly infer a formal query and considers that the user provided examples are independent one from another: it looks for the tuples in the database that are similar to at least one example (wrt. all the attributes) and, if provided, dissimilar to all counter-examples (wrt. at least one attribute). The approach by De Calmès et al. [2] relies on a case-based reasoning system to identify the candidate answers. It defines the satisfaction score $s_{\mathcal{E}}$ as a combination of similarity with positive examples and dissimilarity with counter-examples. In the case where only positive examples are available, Zadrozny et al. [11] propose a k-NN based QBE strategy to identify the tuples that are close to the provided expected answers.

A second, related, category does not infer a formal query either, but exploits dependencies between the provided examples, so as to extract from them an appropriate similarity measure that learns correlation from attributes: the Disjunctive Concept Querying (DCQ) strategy [7] relies on the Choquet integral to build a satisfaction score $s_{\mathcal{E}}$ that allows to interpret the provided examples as different types of expected results. More precisely, it allows to identify subsets of somewhat similar representative exemplars that emphasize the importance of shared combinations of values but without discarding more outlying examples that do not look like any other member of $\mathcal{E}$.

A third type of approach builds a formal query from the provided examples and counter-examples: in [4], the positive examples are analyzed as a whole to identify their most representative (i.e. most frequent) fuzzy predicates, seeing to it that these predicates do not also cover one of the unwanted answers. In [11], the inferred search condition is composed of fuzzy terms taken from a pre-defined vocabulary that discretizes each attribute domain in the DB. An interesting aspect of this approach is that it provides users with a linguistic description of the values shared by positive examples that are not shared by counter-examples.

2.2 Diversified Search

Combining the notion of satisfaction with that of diversity is a research question that has received a lot of attention, starting from the recommendation system framework [9]. It is now used in many application contexts, still including content recommendation [1], but also AI explanation [5] and DB queries [12] to name a few. Focusing on DB querying, it may be the case that many very similar tuples fully satisfy the submitted query, thus leading to a top-k result set containing one type of answer, hence the need for answer diversification strategies.

Diversity is defined and assessed in most existing works as a the result of a pairwise comparison of the candidate answers: denoting Σ a set of candidate answers and $dist$ an appropriate distance measure, it is basically defined as

$$div(\Sigma) = \sum_{t,t' \in \Sigma} dist(t, t') \tag{1}$$

Given a set of candidate answers Σ_Q^α, i.e. tuples associated with a sufficient satisfaction degree, a diversification mechanism aims at finding the subset denoted by $\tilde{\Sigma}_Q^{k,\alpha}$ that contains k answers as diverse as possible, i.e.:

$$\tilde{\Sigma}_Q^{k,\alpha} = \arg \max_{\substack{\Sigma \subseteq \Sigma_Q^\alpha, \\ \text{s.t.} |\Sigma| = k}} div(\Sigma) \tag{2}$$

Some approaches perform a post-processing clustering step on the set Σ_Q^α to determine its structure as groups of somewhat similar answers [8]. The diversified result set is then composed of the most representative tuples taken from each of these clusters. This clustering strategy obviously leads to an overall increase of complexity of the querying system and a significant computation time overhead, or a non-relevant partition if the clustered result set is too small.

To the best of our knowledge, the question of result diversification in the QBE setting has not been studied. The next section thus proposes a definition of a diversified result set dedicated to QBE systems where diversity is defined with respect to the provided set of examples and not only depending on the set of answers, so as to guarantee a complete coverage of the examples that have been used to infer the query.

3 Result Diversification in the QBE Paradigm

This section describes the proposed strategy for diversifying result sets in a QBE setting and an algorithm that allows to identify an optimal set of answers, where optimality depends both on the satisfaction score and the diversity measure.

3.1 Diversity wrt. a Set of Representative Examples

Given a set of representative examples $\mathcal{E}$, diversifying $\Sigma_{\mathcal{E}}^{k,\alpha}$ takes a definition that differs from existing approaches dealing with this issue, as reminded in Section 2.

Instead of maximizing the dissemblance between pairs of tuples in $\Sigma_{\mathcal{E}}^{k,\alpha}$, the presented approach aims at guaranteeing that the returned set of answers covers as much as possible the set of expected answers the user has specified.

Definition 1. *Given a similarity measure sim and η a similarity threshold, a set Σ of candidate answers is said to be a* **diversified result set** *with respect to the set of expected answers $\mathcal{E}$ if it covers each example in $\mathcal{E}$. The notion of coverage refers to a minimal similarity to at least one of the candidate answers.*

Formally, Σ is diversified with respect to $\mathcal{E}$ iff.:

$$\forall e \in \mathcal{E}, \exists t \in \Sigma \text{ st. } sim(e,t) \geq \eta,$$

where sim is an appropriate similarity measure (see e.g. [3]).

Given $\Sigma_{\mathcal{E}}^{\alpha}$ a set of candidate answers, the question is to find a subset of k candidates, subset denoted by $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$, that are diversified considering $\mathcal{E}$. Depending on $\mathcal{E}$, the queried table T and the parameter values (k, α, η), it may obviously be the case that such a subset does not exist.

The aim of a diversification approach is to find the optimal subset $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$ wrt. a numerical criterion of diversity. We propose to define diversity in a QBE setting as related to a fair coverage of the representative examples in $\mathcal{E}$. In other words, each user-provided example of expected answer e should be covered by the same number, $\lfloor \frac{k}{|\mathcal{E}|} \rfloor$, of tuples in $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$ that are sufficiently close to it. Denoting by $S_{\Sigma}^{e} = \{t \in \Sigma, \text{ st. } sim(t,e) \geq \eta\}$ the set of tuples in Σ that are sufficiently close to e, we propose the following diversity criterion:

$$div(\Sigma, \mathcal{E}) = \frac{1}{\lfloor \frac{k}{|\mathcal{E}|} \rfloor} \times \sqrt{\frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \left(|S_{\Sigma}^{e}| - \left\lfloor \frac{k}{|\mathcal{E}|} \right\rfloor \right)^2}. \tag{3}$$

Note that, according to that definition, a candidate answer t may cover several representative examples e simultaneously. Indeed, it may belong to several sets S_e when it is sufficiently similar to several e. The aim is then to find, from a set of candidates $\Sigma_{\mathcal{E}}^{\alpha}$, the subset with cardinal k that maximises diversity. This diversified result set is denoted by $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$ and its definition is identical to Equation 2 but instantiated with the proposed definition for diversity.

3.2 Diversification Strategy

This section introduces the algorithmic strategy we propose to compute the diversified result set $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$ that provides the best diversity wrt. $\mathcal{E}$. The first, preliminary, step consists in retrieving the set $\Sigma_{\mathcal{E}}^{\alpha}$ of the tuples from T that have a sufficient satisfaction degree with respect to $\mathcal{E}$. Algorithm 1 provides the pseudo-code of the proposed approach that is commented below.

To determine the set of tuples from $\Sigma_{\mathcal{E}}^{\alpha}$ that will belong to the diversified set of answers $\tilde{\Sigma}_{\mathcal{E}}^{k,\alpha}$, an empty list l_e is initiated for each element $e \in \mathcal{E}$. Then, $\Sigma_{\mathcal{E}}^{\alpha}$ is scanned in a decreasing order of the score $s_{\mathcal{E}}(t)$ and each candidate t is

Input: Candidate answers $\Sigma^{\alpha}_{\mathcal{E}}$; similarity measure sim; similarity threshold η; number of desired answers k
Output: Diversified answers $\tilde{\Sigma}^{\alpha}_{\mathcal{E}}$

```
1  Σ̃^α_E ← ∅
2  l_e ← [] for each e ∈ E
3  maxle ← 0
4  sort(Σ^α_E, s_E);            ▷ sort ts in Σ^α_E in a decreasing order of their score s_E(t)
5  foreach t ∈ Σ^α_E do
6  |  foreach e ∈ E do
7  |  |  if sim(t, e) ≥ η then
8  |  |  |  l_e.append(t)
9  |  |  |  maxle ← max(maxle, |l_e|)
10 |  |  end
11 |  end
12 end
13 i ← 0
14 while |Σ̃^α_E| < k and i < maxle do
15 |  if |Σ̃^α_E| + |E| ≤ k then
16 |  |  foreach e ∈ E do
17 |  |  |  Σ̃^α_E.add(l_e[i])
18 |  |  end
19 |  end
20 |  i ← i + 1
21 end
22 return Σ̃^α_E
```

Algorithm 1: Diversification of $\Sigma^{\alpha}_{\mathcal{E}}$.

appended to all lists l_e such that $sim(t, e) \geq \eta$. Finally, to build $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$, the first elements in each list are added to $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$, then the second ones and so forth, until $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$ contains k elements or $\Sigma^{\alpha}_{\mathcal{E}}$ has been fully scanned.

Note that a tuple $t = l_e[i]$ is added to the result list $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$ (l17 in Algorithm 1) only if t is not already present in $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$. It it already is, then the current representative example e is considered as already covered by the result list at the same level as the other representative examples.

As in the classical case, it may happen that the final diversified result set $\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}$ does not contain the desired number of answers k for two reasons. The first obvious one is due to a not sufficient number of candidate answers, i.e. if the preliminary step does not find at least k tuples in T that sufficiently satisfy Q. The second one comes from the constraint introduced line 15 in Algorithm 1. The meaning of this constraint is to ensure that the order in which the elements from $\mathcal{E}$ are processed (l16 in Algorithm 1) has no effect on the returned diversified result set. It indeed guarantees that, for a given round of the loop line 14, all the lists representing the different expected answers (the l_es) are processed or none, hence $|\tilde{\Sigma}^{k,\alpha}_{\mathcal{E}}| = |\mathcal{E}| \times \min(k, \min_{e \in \mathcal{E}} |l_e|)$.

The use of Algorithm 1 to diversify the result set of a query defined by examples of representative examples does not add a significant computation cost to the whole querying process. Sorting the tuples from $\Sigma^{\alpha}_{\mathcal{E}}$ in a decreasing order of their score $s_{\mathcal{E}}(t)$ is done in $\mathcal{O}(|\Sigma^{\alpha}_{\mathcal{E}}| \log_2(|\Sigma^{\alpha}_{\mathcal{E}}|))$. Then, the assignment of these candidate answers into the different lists is done in linear time and bounded by the number of tuples to diversify, in other words $|\Sigma^{\alpha}_{\mathcal{E}}|$. The soundness and correctness of this algorithm are stated in the following proposition:

Proposition 1. *Algorithm 1 returns the most diversified top-k result set of a representative examples $\mathcal{E}$ according to Definition 1 of the diversity criterion.*

The sketch of the proof is as follows: at each iteration of the loop starting at line 14, the number of answers covering each representative example $e \in \mathcal{E}$ is increased by 1 and line 15 guarantees that all the representative examples of answers are covered or none of them, at the given iteration. It thus gives the guarantee that $\tilde{\Sigma}^{\alpha}_{\mathcal{E}}$ covers each $e \in \mathcal{E}$ with a same ratio. The fact that the candidate answers are processed in a decreasing order of their respective query satisfaction degree ensures that $\tilde{\Sigma}^{\alpha}_{\mathcal{E}}$ contains the tuples that best satisfy the query.

4 Illustration

This section now illustrates an implementation of the proposed approach in a complete QBE process, named *Div*-DCQ. It also describes an illustration of its relevance through experiments: the latter confirm that the diversification step allows obtaining results of interest and that it does not induce a significant overhead in terms of computation time.

4.1 *Div*-DCQ

To show how query satisfaction can be combined with a diversity criteria among the returned answers, the proposed approach is implemented on top of the QBE strategy named DCQ introduced in [7]. This choice is first motivated by the availability of an implementation of this QBE strategy on top of a commercial RDBMS, namely PostgreSQL. Then, the underlying query inference strategy, from the user-provided examples of expected answers, relies on the CHOCOLATE approach introduced in [6] that is especially appropriate to infer a disjunctive concept underlying the user-provided representative examples. It is thus particularly relevant to build a QBE system on top of CHOCOLATE because, as compared to other QBE approaches (Sec. 2), it ensures that all the representative examples are taken into account during the computation of the satisfaction degree attached to each candidate answer.

The main principles of DCQ are briefly recalled hereafter, more details about the satisfaction degree computation may be found in [6]. They are illustrated with the 2D data shown in Figure 1 where the diamonds represent 5 representative examples building the considered $\mathcal{E}$ set.

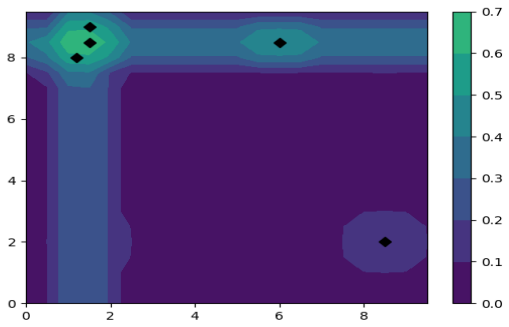

Fig. 1: Considered 2D illustration: the black diamonds show the 5 representative examples forming $\mathcal{E}$. The contour plot shows the values of the satisfaction degree computed by the CHOCOLATE method, as described in Example 1.

The first step consists in inferring a satisfaction function $s_{\mathcal{E}}$ from $\mathcal{E}$ that can be interpreted as a membership function to the fuzzy disjunctive concept exemplified by the examples in $\mathcal{E}$. Without entering into the details of the CHOCOLATE approach [6] used to build this membership function, let us underline two properties of interest it possesses. First, it captures possible situations of generalization among the user-provided expected answers, as illustrated by the contour plot in Figure 1 for the considered $\mathcal{E}$. The fact that three expected answers are in a same narrow subspace, around the point with coordinates $(1.5, 8.5)$, indeed gives more importance to its surrounding area. However, contrary to a mean aggregator for instance, the inferred membership function does not discard the two atypical expected answers, which constitutes the second property of interest. Still, it gives more weight to the point $(6, 8.5)$ as it shares a common y-value with other expected answers.

Technically, as shown below in Example 1, the stored procedure *infer_concept* is used to infer a satisfaction function, named here *myQBE*, that can be applied on the *testData* table, that can be a view as a result of a more complex join query.

Example 1. Use of the *infer_concept* procedure to infer a characteristic function, whose contour is depicted in Figure 1, from few examples of expected answers. The *testData* table contains, for the purpose of this illustration, 2,000 tuples generated using normal distributions around the five representative examples (i.e. the five black diamonds).

```
CALL infer_concept('testData', 'myQBE', { "x"=>1.5, "y"=>8.5, "x"=>1.2, "y"=>8,
    "x"=>1.5, "y"=>9, "x"=>6, "y"=>8.5, "x"=>8.5, "y"=>2 '});
```

Calling the procedure *infer_concept* leads to the creation of a user function named *'myQBE'* that can then be integrated in the selection clause of a query

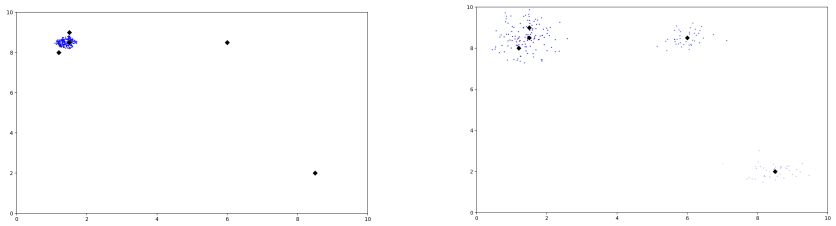

Fig. 2: Results for the query shown in Figure 1: (left) top-200 results, (right) diversified top-200 results.

as in Example 2: it retrieves 200 tuples ($k = 200$) from the *testData* table that satisfy $myQBE()$ with degree of at least 0.2 ($\alpha = 0.2$).

Example 2. Use of the *'myQBE'* user function in a selection clause of a query:

```
SELECT *, get_mu() as mu FROM testData WHERE myQBE() >= 0.2 LIMIT 200 ;
```

As shown in Figure 1, the area around coordinates $(1.5, 8.5)$ gets the highest satisfaction scores. As a result, the top-200 answers for the above query are all located in this area only as shown in the left graph of Figure 2. This illustrates that the result set composed of the tuples that best satisfy the selection condition may lack of diversity and representativity wrt. the different expected answers envisaged by the user.

To overcome this limitation, the proposed approach to diversify the result set may be activated by simply adding the *DIVERSIFY* keyword in the selection clause as shown in Example 3. The *DIVERSIFY* keyword indicates that an *a posteriori* diversification step has to be applied on the set of candidate answers.

Example 3. Call of the diversification process on top of the returned result set.

```
SELECT DIVERSIFY *, get_mu() as mu FROM testData WHERE myQBE() > 0.2 LIMIT 200 ;
```

The right part of Figure 2 displays the results of this modified query: it shows that it leads to a very different result set that now covers all the representative examples of expected answers.

4.2 Experimentations

First experimentations[1] have been conducted on artificial data so as to examine the cost overhead, showing it is negligible. Second, they emphasize the compromise achieved between the overall satisfaction of the returned result wrt. the query and the diversity of the answers. The experimentation context is the

[1] The experimentations are available for reproducibility as a Jupyter notebook at the following url http://people.irisa.fr/Gregory.Smits/fqas2023.tgz

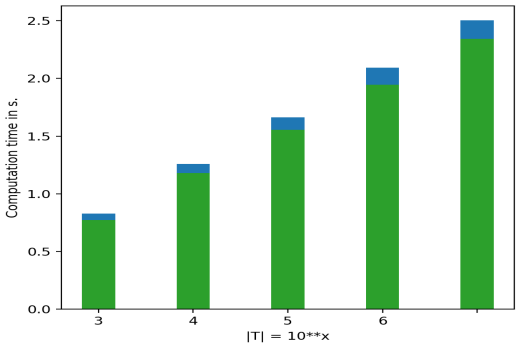

Fig. 3: Computation time wrt. the dataset size, the bottom part of each bar represents the candidate answers retrieval and the upper one the diversification

following. Considering 12 randomly generated reference points in a 4-dimension space, with a shared domain $[0, 10]$, tuples are generated as mixtures of Gaussian distributions around these points, thus forming 12 elliptic clusters.

Computation time Figure 3 shows the evolution of the computation time wrt. different sizes of datasets (from 10^3 to 10^7 with $k = 50$); for each size of dataset, 20 queries are executed and the average of the observed computation times is used. It confirms that most of the computation time is devoted to the retrieval of the candidate answers and that the diversification uses in average $\frac{1}{11}$ of the overall time. It is however worth mentioning that in these experimentations a sequential scan of T is performed and indexes may speed up this retrieval step, but such optimizations are entrusted to the DBMS.

Compromise satisfaction vs. diversity Figure 4 (left) depicts a comparison of the mean satisfaction, obtained on 20 queries, based on the $s_{\mathcal{E}}$ scores, of the top-k obtained without diversification and after diversification. In addition, Figure 4 (right) shows the gain obtained in terms of diversity of the returned answers when the proposed strategy is applied. These results show that, without paying the cost of a significant loss in terms of satisfaction, the proposed diversification strategy leads to a significant improvement of the result diversity, especially for low values of k. The definition of diversity considered in this work is related to an equal coverage of the different representative examples, quantified through a coefficient of variation around the expected number of answers for each representative example. So the closer to zero, the better the diversity degree is. One may also observe that, without diversification, a low value of k will often lead to situations as the one depicted in Figure 2 (left) where tuples around the most "important" representative example are returned only.

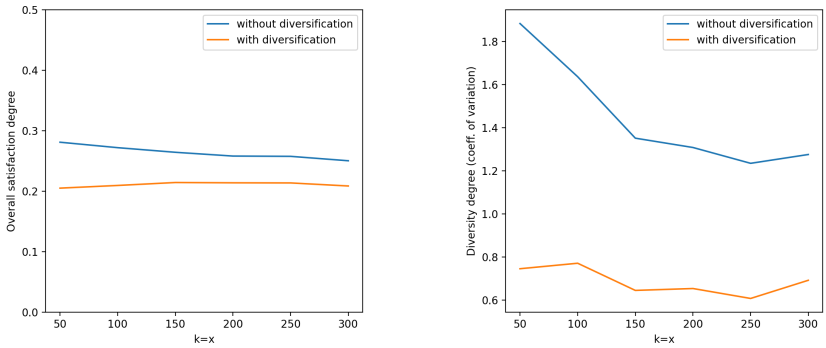

Fig. 4: Comparison of the overall satisfaction (left) and diversity (right) of the result set, for increasing k values

5 Conclusion and Perspectives

In the Query By Example context, this paper studies the issue of diversifying the set of tuples that constitute candidate answers to a query inferred from a set of representative examples. The notion of a diversified result set is redefined to fit the particularities of a QBE context. Diversity is related to a coverage of the different possible answers the user is expecting or willing to accept. An algorithm to diversify a set of candidate answers is proposed and it is shown that the cost overhead in terms of computation time is negligible compared to the execution time of the query itself. The first conducted experimentations illustrate that, without paying the cost of a significant decrease of the overall result satisfaction, the proposed diversification strategy provides a better overview of the different possible answers to a query inferred from user-provided representative examples.

Future works will perform a deeper study of the behavior of the proposed approach according to variations of the query parameters, e.g wrt. the number of provided representative examples and the minimal satisfaction degree. A longer term perspective is to find a strategy or at least a heuristic to avoid having to identify all the candidate answers and to rank order them before starting the diversification step.

References

1. Castells, P., Hurley, N., Vargas, S.: Novelty and diversity in recommender systems. In: Recommender systems handbook, pp. 603–646. Springer (2021)
2. De Calmès, M., Dubois, D., Hullermeier, E., Prade, H., Sedes, F.: Flexibility and fuzzy case-based evaluation in querying: An illustration in an experimental setting. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 11(01), 43–66 (sep 2003)

3. Lesot, M.J., Rifqi, M., Benhadda, H.: Similarity measures for binary and numerical data: a survey. International Journal of Knowledge Engineering and Soft Data Paradigms 1(1), 63–84 (2009)
4. Moreau, A., Pivert, O., Smits, G.: Fuzzy query by example. In: Proc. of ACM Symposium on Applied Computing, SAC'18. pp. 688–695 (2018)
5. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 conference on fairness, accountability, and transparency. pp. 607–617 (2020)
6. Smits, G., Yager, R., Lesot, M.J., Pivert, O.: Concept membership modeling using a choquet integral. In: Proc. of the Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'20). pp. 359–372 (2020)
7. Smits, G., Lesot, M.J., Pivert, O., Yager, R.R.: Flexible querying using disjunctive concepts. In: Flexible Query Answering Systems: 14th International Conference, FQAS 2021, Bratislava, Slovakia, September 19–24, 2021, Proceedings 14. pp. 29–40. Springer (2021)
8. Smits, G., Pivert, O.: Linguistic and graphical explanation of a cluster-based data structure. In: Scalable Uncertainty Management: 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings 9. pp. 186–200. Springer (2015)
9. Smyth, B., McClave, P.: Similarity vs. diversity. In: Case-Based Reasoning Research and Development: 4th International Conference on Case-Based Reasoning, ICCBR 2001 Vancouver, BC, Canada, July 30–August 2, 2001 Proceedings 4. pp. 347–361. Springer (2001)
10. Thomas, J.C., Gould, J.D.: A psychological study of query by example. In: Proceedings of the May 19-22, 1975, national computer conference and exposition. pp. 439–445 (1975)
11. Zadrozny, S., Kacprzyk, J., Wysocki, M.: On a novice-user-focused approach to flexible querying: The case of initially unavailable explicit user preferences. Proc. of the 10th Int. Conf. on Intelligent Systems Design and Applications, ISDA'10 pp. 696–701 (2010)
12. Zheng, K., Wang, H., Qi, Z., Li, J., Gao, H.: A survey of query result diversification. Knowledge and Information Systems 51, 1–36 (2017)
13. Zloof, M.M.: Query-by-example: A data base language. IBM Syst. J. 16(4), 324–343 (1977), https://doi.org/10.1147/sj.164.0324