Adaptation of language model of Information Retrieval for empty answers Problem in databases

Abdelhamid Chellal
IRIT
University of Toulouse III
Toulouse, France
Email: abdelhamid.chellal@irit.fr

Karima Amrouche
ESI
Alger, Algeria
Email:k_amrouche@esi.dz

***Abstract*—Information over the web is increasingly retrieved from relational databases in which the query language is based on exact matching, data fulfil completely the query or not. The results returned to the user contain only tuples that satisfy the conditions of the query. Thereby, the user can be confronted to the problem of empty answers in the case of too selective query. To overcome this problem, several approaches have been proposed in the literature in particular those based on query conditions relaxation. Others works suggest the use of fuzzy sets theory to introduce flexible queries. Another line of research proposes the adaptation of information retrieval (IR) approaches to get an approximate matching in databases.In this paper, we discuss an adaptation of language model of IR to deal with empty answers. The main idea behind our approach is that instead of returning an empty response to the user, a ranked list of tuples that have the most similar values to those specified in user's query is returned.**

Keywords*—*Information retrieval, empty answers, databases, language model, similarity.

I. Introduction

An information over the Web is increasingly retrieved from relational databases. Users will become benefit if they are allowed to search the underlying databases directly. In fact, a large number of databases such as digital libraries, scientific databases and travel reservation systems are available. Seeking for information on these databases is increasingly sought by ordinary users wishing to fulfil their informations needs.That new search applications are bringing the principles of information retrieval (IR) to database systems.

The notion of relevance isn't often present in relational database management systems (DBMS). Traditional interrogation approaches support only a boolean model query [1], [2]. The correspondence between tuple and query is exact, data fulfil completely a query or not. Thus, a select SQL query (Structured Query Language) returns only tuples that exactly meet the specified conditions. Therefore the user may be faced with empty answer problem or the over-abundant answer problem which aren't gracefully handled. The former occurs when none data satisfy the conditions whereas the later takes place where many data satisfy the conditions.

To handle the aforementioned problems, several solutions have been proposed in the literature many of them are based on automatic relaxation [3], [4], [5], [6]. These approaches tries to obtain automatically a less restrictive variant of the initial query by weakening the predicates in query's conditions. The challenge in this case is to find a form of relaxation that preserves, as much as possible, the semantics of the original query submitted by the user.

Other works propose the use of flexible queries [7], [8], [9], [10], that is queries with vague conditions whose semantics is based on fuzzy logic [11]. The result of a flexible query is a set of all answers satisfying, in some degree between 0 and 1, the query's conditions. The advantage of this approach is that the chance of obtaining an empty answer set is reduced. But, in the situation where no available data can satisfy a flexible query is not handling.

As mentioned, an empty answers or too little answers returned occurs when the user query is too selective. To avoid a frustrated situation for the user in the case of an empty answer, we think that it might be interesting to return a ranked list of data that approximately meet the user's need. The ranking function should capture user's preference by assessing a similarity between tuples in the databases and the query.

The Approximate matching, ranking and returning the most relevant results of a query are a popular paradigm in Information Retrieval (IR). A variety of IR models have been proposed, there are a lot of them in [12]. Some of IR models have been adapted to databases. In [1] an adaptation of the vector model was proposed to enable approximative matching in databases as solution for the empty answer problem. An other work [13] suggests the use of the basic probabilistic model to rank answers for handling over-abundant answer problem.

In this paper, a ranking function based on adaptation of language model of Hiemstra [14] is proposed to handle the empty answers problem. The purpose is that instead of returning an empty answer to the user, a ranked list of tuples with attributes values that are the most similar to those specified in the query is proposed. We use a uni-gramme language model to asses the similarity between value specified by the user in the query and those in the tuple.

In this article, we discuss only the case of queries on a single table with conjunctive conditions. The condition can have an atomic form ($a_i = q_i$) or a more general form with BEETWEN clause in the case of numeric attribute or IN clause for the categorical attribute.

The rest of this paper is organized as follows. We introduce

related work in section 2. Our approach for handling empty answer in databases is detailed in section 3 and section 4 presents a discussion. Section 5 details our experimental evaluation setup of proposed approach. We end with the conclusion in Section 6.

II. RELATED WORK

Several researches have been proposed to deal with empty answers problem. The first solutions that have focused on approximate matching in databases, are based on extending the relational model to fuzzy sets [11]. They propose methods express and evaluate flexible queries[3], [9], [10], [8], [15] through modelling of vagueness by using the theory of fuzzy sets. In these works a relational algebra and operators have been studied in detail to take into account degrees of preference and manipulate fuzzy relations [16]. A flexible query language, SQLf (Structured Query Language Fuzzy) [7], an extension of SQL the query language databases was specified.

An other category of researches suggests a relaxation of the query that fails to avoid empty answers. The principle of these approaches is to expand the scope of query conditions so that it can return responses that meet the original query, instead of presenting an empty result [3]. The original query is then rewritten as an approximate query by relaxing the query criteria range.

The data distribution in the database and knowledge discovery are used to choice the constraints, the relaxation order of specified attributes and the degree of relaxation for each attribute. [17], [4], [5], [18], [19], [20]. In [2] authors propose AQRR (Approximate query ranking & results), an automatic approach to relax query and rank results in which an evaluation of the distances between values for both kind of attributes (categorical and numeric) is proposed. AQRR assigns a weight for each attribute in the relaxation process. To rank tuples they generate user's contextual preferences from database and workload and use them to create a priori orders of tuples.

The closest work to our approach is [1] in which an adapting of the classic vector space model of IR is discussed to handle the problem of empty answers. The proposed relevance score function is an adaptation of the scalar product function of vector space model.

$$Score(T, Q) = \sum_{k=1}^{m} S(t_k, q_k)$$

With t_k is a value of attribute a_k in tuple T and q_k is the specified value of the same attribute in the query.

$S(t_k, q_k)$ is the degree of similarity between the values of attributes specified in the query and those existing in the tuple. It is estimated by:

$$S(t_k, q_k) = \begin{cases} QF(q_k) \times IDF(q_k) \; if \; t_k = q_k \\ 0 \; Otherwise \end{cases}$$

$QF(q_k) \times IDF(q_k)$ is an adaptation of the term frequency and inverse document frequency (tf-idf) common weighting technique used in IR. Where $IDF(q_k)$ represents the importance of value of the attribute a_k specified in query in the database. It is inversely proportional to its occurrence frequency in the database and it is evaluated according to the nature of the attribute (categorical or numeric).$QF(q_k)$ represents the importance of the value q_k in the workload.it is proportional with their frequency in the workload.

III. A LANGUAGE MODEL FOR THE EMPTY ANSWERS PROBLEM

To be able to use IR model, we consider each tuple as document, attributes values as terms and the table as the collection of documents. The main idea, behind our proposition, is to be able to have an approximate matching we assess the similarity between values. Therefore, we measure the probability of likelihood between query's values and those in the tuple by using a probalistic model (language model). This likelihood is measured according to the similarity between the attribute values. The tuples are then ranked according to their score and only the Top-k tuples are returned to the user.

A. Principle of the Proposed Approach

Language model approaches in information retrieval are based on the assumption: *when a user submits a query to an information retrieval system, he has already had in his mind one or many ideals documents that he is looking forward its* [21]. The query is then inferred by the user from these documents [22], [21]. The relevance of a document is considered as the probability that the document's language model would generate the terms of the query: $P(Q|M_d)$.

By analogy to IR, we consider that when a user makes a search in a database, he has already had in his mind the ideals tuples to his information need. He translates these ideals tuples to a query in which he specifies the values of attributes he wants. In general the query does not contain all the attributes of the tuple since the user is not supposed to know the schema of the database. Also only the most important attributes, according to the user are specified.

The idea behind our approach is as follows. We assume that each tuple in a table is generated by a language model M_T. The relevance score of each tuple to the query *Q* is estimated by computing the probability that the query is generated by the model of the tuple.

To estimate this probability, we suggest to use a unigram language model with smoothing. Using a language model with smoothing gives us an approximate matching. Furthermore, with a smoothing, tuples in which no values specified in the query occurs will not get a null score and enable to have an approximate matching. We chose smoothing of Jelineck Mercer, the same one used in the IR model of [14].

To provide an overview of our approach, we will follow the illustrative example below: Example: Consider the used car relation CarTable (Make, Model, Version, Year, Transmission, Color, Engine, Mileage). The attributes Make, Model, Color, Transmission and Engine are categorical attributes where the attributes Year and Mileage are numerical. Assume that we have the following empty answer query Q: Select * form CarTable where Model= Polo and Engine= Diesel and Year = 2005 and Mileage = 50000.

B. Relevance Score of Tuples

Let Q be a conjunctive query with atomic conditions of the form $(a_1 = q_1, a_2 = q_2, ..., a_n = q_n)$ on a table *D* which has the attributes $(a_1, a_2, ..., a_m)$ with $(m > n)$ and A_i is the domain of the attribute a_i. The relevance score of a tuple $T = (t_1, t_2, ..., t_m)$ from the table *D* for the query *Q* is evaluated as follows:

$$Score(Q, T) = P(q_1, q_2, ..., q_n | M_T)$$

In the databases there is a dependency between tuples, but takes it into account leads to the fact that all the conditions specified in the query have to be satisfied simultaneously. So, we'll not be able to have an approximate matching and the problem of empty answers will come back again. Since we are looking for handling this problem through the implementation of approximate matching between attribute values of tuples and those specified in the query, we assume that the values of attributes in the query are independent. Therefore we obtained:

$$Score(Q, T) = \prod_{q_i \in Q} P(q_i | T)$$

In order to estimate $P(q_i|T)$ we use a mixed model based on interpolation of Jelineck Mercer [21] that combines between the tuple language model and the table *D* language model as follow:

$$P(q_i|T) = \alpha P_{ML}(q_i|T) + (1 - \alpha) P_{ML}(q_i|D)$$

$P_{ML}(q_i|D)$ is the probability of producing the value q_i in the table *D*. It represents the importance of this value in the database through its frequency of occurrence in Table *D*. It is evaluated using the maximum likelihood estimation (ML) as follows:

$$P_{ML}(q_i|D) = \frac{f(q_i)}{\sum\limits_{v_i \in A_i} f(v_i)} \quad (1)$$

Where $f(q_i)$ is the number of tuples in table D having the value q_i in attribute a_i and $v_i \in A_i$ are the different values of the attribute a_i that exist in *D*. We note here that if the attribute a_i is not null, $\sum\limits_{v_i \in A_i} f(v_i)$ will be equal to the number of row in the table.

Considering $P_{ML}(q_i|T) = \frac{tf(q_i)}{|T|}$ is not interesting since the frequency $tf(q_i)$ of value q_i in any tuple is equal to either 0 or 1. In addition, in the case of empty answers, it's more likely that this frequency is equal to 0. Therefore, instead of using maximum likelihood estimation, to evaluate this probability we define $P_{sim}(q_i|T)$. It is the probability that the value q_i of the attribute a_i specified in the query Q is similar to the corresponded one in the tuple T. We attempt to measure the similarity between the value that the user is looking forward and the existing one in the tuple. More the two values are distinct more the probability is smaller. So $P_{sim}(q_i|T)$ is the probability that both values are similar and consequently the probability that the existing value in the tuple is relevant to the user.

Attribute values in a database are not similar. Its can be numerical or categorical. Therefore, the probability $P_{sim}(q_i|T)$ will be estimated in two various ways depending on the nature of each attribute. We take in consideration the features of each kind. To achieve this, we adopt the formulas that estimate similarity between values proposed by [2], [20]. Notice here that the main purpose of this two works is to assess the degree of relaxation of predicates according to the similarity with values specified by the user in his query.

1) Similarity between two numerics attribute values: A simple way to deal with numeric attributes is to discretize the domain of numerical attribute in several intervals and consider all values of attribute belonging to each rang as categorical. Thus, we can apply the same approach in assessing the degree of similarity between two attribute values regardless of their nature. This approach raises several constraints including the choice of intervals and their number. The major disadvantage of this technique is that tow values that belong to different intervals are treated as completely different independently of the real distance between them.

The aim is to assess the degree of similarity between two numeric values in an automatic way without the need of expertise and independently of the domain. For that, we adopt a function proposed by [2] that considers the continuous nature of numerical attributes. It evaluates the similarity between two values according to the distance between them with taking into account the distribution of different values in the database. The similarity between two attribute values v_1, v_2 is estimated by:

$$Sim(v_1, v_2) = \frac{1}{1 + (\frac{v_1 - v_2}{h})^2} \quad withh = 1.06\sigma \times n^{-\frac{1}{5}} \quad (2)$$

Where σ is standard deviation of the attribute values in *D* and n is a number of tuple in *D*. Thus, the probability $P_{sim}(q_i|T)$ is computed as follows:

$$P_{sim}(q_i|T) = \frac{Sim(q_i, t_i)}{\sum\limits_{t_k \in A_i} Sim(q_i, t_k)} \quad (3)$$

In this formula, we can see that when the query and the tuple contain the same value $(q_i = t_i)$, $Sim(q_i, t_i) = 1$ and $P_{sim}(q_i|T)$ will get the greatest value. In contrast, if t_i is very different from q_i, $Sim(q_i, t_i)$ tends to 0 and $P_{sim}(q_iT)$ tends to the minimum.

2) Similarity between two categorical attribute values : To assess the similarity between two categorical attributes values, we use an adaptation of the method proposed in [2], [20]. It is measured as the percentage of common AV-pairs (Attribute Value Pairs) that are associated to them. An AV-pair of a value v of attribute a_i represents all attribute values of the a_i and their frequency of occurrence in the database that belong to tuple having $a_i = v$. Precisely, it is the result of the following SQL query:

Select distinct a_j*, count(*)*
From D where $a_i = v$ *group by* a_j .

The answer set that contains each AV-pair as a structure is called the supertuple. The supertuple contains a set of

TABLE I. SUPERTUPLE FOR MODEL = POLO.

Attribute	Value
Make	Volkswagen:112
Mileage	10000-20000: 12, 20000-40000: 25,2
Color	Black: 46, Silver: 15,
year	2008: 17, 2007: 37,

keywords for each attribute in the relation not bound by the AV-pair. The table 1 shows the supertuple for value "Polo" of attribute "Model" in a used car database.

Given a categorical value, all the AV-pairs associated to the value can be seen as the features describing the value. The similarity between two values can be estimated by the commonality in the features (AV-pairs) describing them. The similarity between two AV-pairs can be measured as the similarity shown by their supertuples. The supertuples contains sets of keywords for each attribute in the relation and therefore Jaccard Coefficient which assesses the similarity between two sets is used to determine the similarity between two supertuples.

In this paper, the similarity coefficient between two categorical values is calculated as a sum of the set similarity on each attribute[11]. If we consider that C_{q_i} is a supertuple of value q_i specified in the query and C_{t_i} is supertuple for attribute value t_i in the tuple T, the similarity between these two values is estimated by:

$$VSim(C_{q_i}, C_{t_i}) = \sum_{j=1}^{m-2} J(C_{q_i}.A_j, C_{t_i}.A_j)$$
$$withJ(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (4)$$

Now, to estimate the probability $P_{sim}(q_i|T_j)$ in the case of categorical attribute we use this assessment but with normalization in order to have values between 0 and 1.

$$P_{sim}(q_i|T) = \frac{VSim(C_{q_i}, C_{t_i})}{\sum_{t_k \in A_i} VSim(C_{q_i}, C_{t_k})} \quad (5)$$

In our example, tuples that have $Model = Polo$ will get $VSim(C_{q_i}, C_{t_i}) = 4$ and those with for instance $Model = Clio$ get $VSim(C_{q_i}, C_{t_i}) = 1,8407$. This leads us to have an approximate matching between values of tuples and those specified in query. The first ten most relevant tuples returned for query Q in our example are presented in the table below:

C. Generalization for query with BETWEEN and IN Conditions

In Section III.1 and III.2, we assumed that a query is a conjunction of atomic conditions such as $a_k = q_k$. A useful generalization is the ability to specify a range/set of values for numerical/categorical attributes. Lets consider that the query Q has a condition $(C_1, and...and, C_m)$, where each C_k is generalized form as a_i IN $Q_i = (q_{i1}, q_{i2}, , q_{im})$ for categorical attributes or $a_i\ BETWEEN[v_{sup}, v_{min}]$ the range for numeric attributes. We generalize the estimation of the probability $P_{sim}(q_i|T)$ described in equation (2,3) and (4,5) and $P_{ML}(q_i|D)$ in equation (1) as follows:

- For numeric attributes values:

$$P_{sim}(q_i|T) = \begin{cases} \frac{1}{\sum_{t_k \in A_j} Sim(q_i, t_k)}\ if\ t_i \in [V_{sup}, V_{min}] \\ \frac{\max[Sim(V_{sup}, t_i), Sim(V_{min}, t_i)]}{\sum_{t_k \in A_j} Sim(q_i, t_k)} \end{cases} \quad (6)$$

And

$$P_{ML}(q_i|D) = \frac{\max[f(V_{sup}), f(V_{min})]}{\sum_{t_k \in A_j} F(t_k)} \quad (7)$$

- For categorical attributes values:

$$P_{sim}(q_i|T) = \begin{cases} \frac{m-2}{\sum_{t_k \in A_i} VSim(C_{q_i}, C_{t_k})}\ if\ t_i \in [q_{i1}, q_{im}] \\ \frac{\max_{q_{ik} \in Q_i} VSim(C_{q_i}, C_{t_i})}{\sum_{t_k \in A_i} VSim(C_{q_i}, C_{t_k}}\ Otherwise \end{cases} \quad (8)$$

Where m is the number of attribute and $m-2$ is the maximum value of the similarity between two categorical attribute values equation (4).

IV. DISCUSSION

With this proposed approach, answers returned are ranked according to their relevance score for the query. This score represents the degree of similarity between attribute values of the query and those of the tuple. Therefore, the set of returned tuples will be composed of three categories of tuple. The first subset is composed by all tuples in which all attribute values are equal to those specified in the query, if they exist, since we are in the case of empty answers. The second subset will gather tuples that have at least one attribute value specified in the query. The last group contains tuples that doesn't have any value specified in the query.

Notice here that in general the number of tuple that belong to the last category is very important. In order to avoid to overload the user with irrelevant tuple, we select only the first top-k tuples.

The closest work to ours is [1] where an adaptation of the vector space model is proposed to handle the problem of empty answers. The relevance score is assessed through two functions, $IDF(t)$ which represents the importance of the value in the database and $QF(t)$ which estimates the significance of the value t through its frequency of occurrence in database workload. The main differences between our approach and theirs beside the fact that ours is based on a probabilistic model (Language Model) are:

- The relevance score of a tuple depends both on the degree of similarity between attribute values specified in the query and those in tuple and the importance of values in the database which is determined by

their frequency of occurrence in the table. While the approach of [1] is based exclusively on the importance of values in the database and workload;

- In our approach, we do not need a database workload which is not always available;
- The results returned by [1] are grouped into several equivalence classes. Where all tuples in each class have the same score. To break ties among the tuples in each class, they compute the importance of not specified attribute values in the workload. While in our approach tuples have gotten a different score even if they belong to the same category.

V. Experimental evaluation

Evaluating and comparing the quality of different ranking database alternatives is challenging. Unlike Information Retrieval which relies on extensive user studies and available benchmarks, such infrastructure is not available for evaluating database ranking. Therefore we adopt the same test protocol that was followed by [1], [2] the two closest works to our.

To evaluate the capacity of our approach to return relevance tuple in approximate matching with query and the ranking quality, we implemented the described approach and conducted experiments to assess their effectiveness. For our evaluation, we set up a used car database CarTable(Make, Model, Version, Transmission, Year, Color, Engine, Mileage). It contains 16,842 tuples extracted from database of the website:www.ouedkniss.com. The attributes Make, Model, Version, Transmission, Color whereas Engine are categorical attributes and the attributes Year and Mileage are numerical attributes.

The proposed approach was implemented in two components: a Pre-processing component and Query processing component. The main task of the pre-processing component is to compute the similarity score between categorical values which are then stored in an auxiliary databases. It computes the parameters used by the similarity function for each attribute according to its nature. The Query processing component uses the data estimated in the Pre-processing component to compute a relevance score for each tuple in table.

In this experimental evaluation, we compare the performance of our adaptation of language model in databases with a baseline. We chose as baseline the list returned by the DBMS after submitting the same queries that we had already removed randomly one condition.

A. Test protocol

We requested 5 users to submit 3 too selective SQL queries to the databases CarTable according to their preference. Each query had on average of 2.8 of specified attributes. The majority of these queries returned an empty response (except for two queries where there ware only one tuple returned). Table 2 presents these queries.

Table 3 shows the results returned by our approach as response for the query number 4. In this example, we can see that since there are no tuples that meet the specified conditions in query number 4, our approach returns primarily the same cars that the user is sought after (vw polo diesel) with the closest year and mileage values to what were specified in the query (2005 and 50000 km). After that, our approach presents other kind of cars that belong to the same class and were made in the same year specified in the query (2004).

TABLE II. Queries of test protocol.

N	Query	Specified attributes	Response
1	Hyundai Accent gaz 2007	4	0
2	Polo gasoline 2007	3	0
3	Leon gasoline 2007	3	0
4	Polo diesel 2004 50000km	4	0
5	Logan gaz 2011	3	0
6	Accent diesel 2008	3	0
7	Nissan sunny gaz 2009 70000km	5	0
8	Renault Scenic gasoline 2004	4	0
9	Symbol gaz 2009 80000km	4	0
10	206 diesel 2006	3	0
11	Kia Picanto 2005 50000km	4	0
12	Yaris 2006 60000km	3	0
13	Yaris sedan 2008 60000km	4	0
14	Ibiza gasoline 2011	3	1
15,	Alto 2009 50000km	3	1

TABLE III. Tuples on approximate matching returned for query number 4.

Make	Model	Version	Engine	Year	Mileage
Volkswagen	Polo	1.9 sdi	Diesel	2005	43500
Volkswagen	Polo	1.9 sdi	Diesel	2005	57000
Volkswagen	Polo	1.9 sdi	Diesel	2005	90000
Volkswagen	Polo	1.9 sdi	Diesel	2005	99500
Volkswagen	Polo	1.9 sdi	Diesel	2005	102000
Volkswagen	Polo	1.9 sdi	Diesel	2005	117000
Volkswagen	Polo	1.9 sdi	Diesel	2005	130000
Volkswagen	Polo	1.9 sdi	Diesel	2005	130000
Renault	Clio	1.5 dci	Diesel	2004	42400
Peugeot	307	1.4 hdi	Diesel	2004	55000

Since, it is not practical to ask users to mark all tuples of the database according to their relevance (preferred or not) for each query and rank the whole query results by order of their preference. We used the strategy followed by [1], [2] which can be describe as follows: For each query Q_i, we generated a list of 60 tuples likely to contain a good mix of relevant and irrelevant tuples to the query. We did this by mixing the Top-30 results of our approach after removing ties with a few randomly selected tuples and adding tuples returned by traditional SQL queries which was constructed from the original user query by eliminating one of these conditions randomly chosen.

Finally, we presented the queries along with their corresponding list to each user in our study (our approach and a baseline). Each user's responsibility was to mark each tuple in the list relevant or irrelevant to the query Q_i and mark the Top 10 tuples that they preferred most. We then applied our ranking functions against the test queries. We compare the result obtained by the proposed approach against those obtained by the baseline.

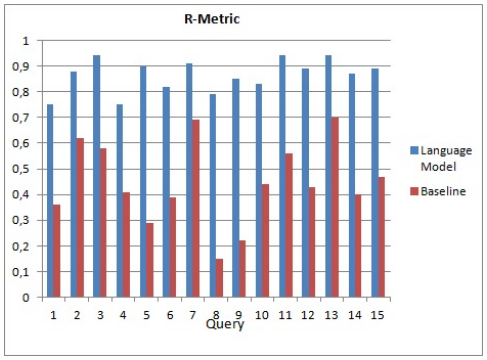

Fig. 1. R metric for the 15 queries obtained by the proposed approach and the baseline.

Fig. 2. Number or relevant tuples returned by proposed approach VS total number of relevant tuples by query.

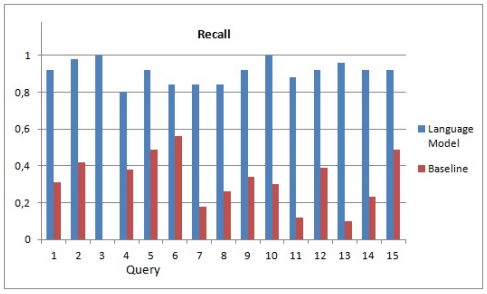

Fig. 3. Recall for each query obtained by the proposed approach and the baseline.

B. Experimental results

To evaluate the effectiveness of our approach, we adopt two types of measures: the standard collaborative filtering metric R-metric and recall. The R-metric is used to assess the ranking quality. The recall allows us to assess the ability of our approach to return relevant tuples to the query. In fact, unlike conventional IR the notion of relevance is different to the known RI, here it's more related to the user preference.

1) Quality sorting of returned tuples : To compare the ranking quality of results returned by the proposed function with the human responses, we used the standard collaborative filtering metric R. It is defined by the following equation: [1], [2]

$$R = \sum_i \frac{r_i}{2^{(\frac{i-1}{9})}} \tag{9}$$

In the equation, r_i is the user's preference for the i^{th} tuple in the ranked list returned by our approach (1 if it is marked relevant, and 0 otherwise). The intuition behind the R metric is that if relevant tuples are ranked low, they contribute less to the value of R with exponential decay.

Figure 1 shows the R metric values obtained for each query in the test protocol (R values are normalized by dividing by the maximum possible value for R). We observe that the ranking quality of our outlier approach outperforms the baseline.

The average of R-metric for the fifteen queries obtained by the proposed approach is 0,86 whereas it is 0,44 for the baseline. This result reveals that the relevant tuples are ranked high in a returned list which shows that the proposed approach is a good ranking function.

2) The quality of approximate answers : To assess the ability of our approach to return relevant tuples with approximate matching, we use the recall metric. Recall is the ratio of the number of relevant tuples retrieved to the total number of relevant tuples.

Figure 2 shows the number of relevant tuples returned by the proposed approach for each query of the test protocol whereas figure 3 presents a recall of the proposed approach compared with the recall of the baseline.

Where RTR is the number of relevant tuples for the query retrieved by our approach and RT is the total number of relevant tuples for the query.

Thus, we can see that our approach outperforms the baseline in all queries. the recall of the outlier approach is (0,90%) almost triples of the recall of the baseline (0.31%).

VI. Conclusion

We presented in this paper a new approach to address the problem of empty answers in the database. Through adapting the language model of IR, we propose a new approach to implement an approximate matching between attributes values in databases. Our approach is based in assessing the relevance of tuples according to the degree of similarity between attribute values of tuples and query.

We evaluated the proposed approach according to the test protocol followed by the two closest works. [1], [2]. The results obtained are motivating. Our outlier approach clearly outperforms the baseline, improving both ranking quality and recall which indicating that adapting the language model in database is beneficial for handling the empty answer problem. Our approach has shown promise, and is worthy of further investigation, especially more conclusive user studies and adaptation for the ranking results in databases to handle an over-abundant problem.

In perspective, it would be interesting to investigate generalization of the functions of relevance scores proposed in this article for queries on multiple tables. For join queries, the first solution that has come to our mind is to compute the cartesian product in an auxiliary table and applies the proposed approach on this table.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, "Automated ranking of database query results proceedings," in *In Proceedings of First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.

[2] X. Meng, Z. Ma, and L. Yan, "Answering approximate queries over autonomous web databases,," in *Track: XML and Web Data / Session: XML Querying*, 2009.

[3] S. Amer-Yahia and D. Srivastava, "Tree pattern relaxation," in *In Proceedings of EDBT02*, Prague, Czech republic, 2002, pp. 496–513.

[4] I. Muslea and T. J. Lee, "query relaxation via bayesian causal structures discovery," in *Proceedings of the national conference of artificial intelligence (AAAI-05)*, 2005, pp. 831–236.

[5] U. Nambiar and S. Kambhampati, "Answering imprecise database queries: A novel approach," in *ACM Workshop on Web Information and Data Management (WIDM)*, Nov. 2003.

[6] S. Moises and S. P. and, "Dealing with empty and overabundant answers to flexible queries," vol. 2, pp. 12–18, Feb. 2014. [Online]. Available: http://dx.doi.org/10.4236/jdaip.2014.21003

[7] P. Bosc and O. Pivert, "Sqlf : A relational database language for fuzzy querying," in *In Proceedings of IEEE Transaction on Fuzzy Systems, 3(1),*, 1995, pp. 1–17.

[8] P. Bosc, D. Dubois, O. Pivert, and H. Prade, "Flexible queries in relational databases, the example of the division operator," pp. 281–302, 1997.

[9] V. Tahani, "A conceptual framework for fuzzy query processing : A step toward very intelligent database systems," no. 13, pp. 289–303, 1977.

[10] O. Pivert, "Contribution linterrogation flexible des bases de donnes : expression et valuation de requłtes flous," Ph.D. dissertation, Universit de Rennes, Rennes, France, 1991.

[11] L. Zadeh, "Fuzzy sets," vol. 3, no. 3, pp. 338–353, 1965.

[12] A. Baeza-Yates and A. R.-N. Berthier, *Modern Information Retrieval: the concepts and Technology behind search (2nd edition) ACM Press*. Addison-Wesley, 2011.

[13] H. V. W. G. Chaudhuri S, Das G, "Probabilistic ranking of database query results," in *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004.

[14] D. Hiemstra, "A linguistically motivated probabilistic model of information retrievalg," in *Proc European Conference of Digital Library (ECDL98)*, Sep. 1998.

[15] P. Bosc, A. Hadjali, and O. Pivert, "Fuzzy sets and systems journal," no. 12, 2008.

[16] P. Bosc, L. Litard, O. Pivert, and D. Rocacher, *Base de donnes, Gradualit et Imprcision dans les bases de donnes : Ensembles flous, requłtes flexibles et interrogation de donnes mal connues*. Paris, France: AEditions ellipses, 2004.

[17] I. Muslea, "Machine learning for online query relaxation," in *Proceedings of the national conference of knowledge and discovery and data mining, KDD2004*, 2004, pp. 246–255.

[18] U. Nambiar and S. Kambhampati, "Mining approximate funcitonal dependencies and concept similarities to answer imprecise queries," in *WebDB*, Jun. 2004.

[19] ——, "Answering imprecise queries over web databases," in *VLDB Demonstration*, 2005.

[20] ——, "Answering imprecise queries over autonomous web databases," in *Proc. ICDE 2006, 22 nd International Conference on Data Engineering*, 2006.

[21] M. Boughanem, W. Kraaij, and J. Nie, *Modles de langue pour la recherche dinformations, dans Les systmes de recherche dinformations - Modles conceptuels*. Paris, France: Hermes, 2004.

[22] J. Ponte, W. Bruce, and A. Crift, "Language modeling approach to information retrieval," in *Proc. Research and Development in Information Retrieval, ACM-SIGIR*, 1998, pp. 275–281.