Estimating Recall and Precision for Vague Queries in Databases

Raquel Kolitski Stasiu*, Carlos A. Heuser, and Roberto da Silva

Instituto de Informática, Universidade Federal do Rio Grande do Sul,
Av. Bento Gonçalves, 9500, CEP 91501-970 - Porto Alegre, RS, Brazil
{rkstasiu, heuser, rdasilva}@inf.ufrgs.br

Abstract. In *vague* queries, a user enters a value that represents some real world object and expects as the result the set of database values that represent this real world object even with not exact matching. The problem appears in databases that collect data from different sources or databases were different users enter data directly. Query engines usually rely on the use of some type of similarity metric to support data with inexact matching. The problem of building query engines to execute vague queries has been already studied, but an important problem still remains open, namely that of defining the threshold to be used when a similarity scan is performed over a database column. From the bibliography it is known that the threshold depends on the similarity metrics and also on the set of values being queried. Thus, it is unrealistic to expect that the user supplies a threshold at query time. In this paper we propose a process for estimation of recall/precision values for several thresholds for a database column. The idea is that this process is started by a database administrator in a pre-processing phase using samples extracted from database. The meta-data collected by this process may be used in query processing in the optimization phase. The paper describes this process as well as experiments that were performed in order to evaluate it.

1 Introduction

In *vague* queries, the problem is to find all database values that represent the same real world as the one represented by the value entered by the user in the query. A vague query accept variation in spelling to consider not exact match in query argument compared to the values in the database. This type of query is usual in databases that collect data from different sources or are generated by different users. As an example consider a query like "Authors that have published at the 'Intl. Conf. on Very Large Databases' in 2002". In the database the name of this conference may be spelled in different ways, like 'International Conference on Very Large Databases' or simply 'VLDB'.

For this kind of problem a typical Information Retrieval (IR) solution is to *rank* the values in the database using some type of similarity metric (e.g. an edit distance [1, 2]

* On leave from Pontifícia Universidade Católica do Paraná (www.pucpr.br) and Centro Federal Tecnológico do Paraná (www.cefetpr.br).

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 187–200, 2005.

and display them to the user ordered according to this ranking. The values that are more similar to the query value should be shown first. The user pages through the output identifying the database values that represent the real object that is being searched.

However, not all queries can be solved by applying a single similarity search over the database. In more complex queries the values resulting from one similarity search are used as query values for another similarity search. For example, consider a database that contains two tables with data collected from the Web: `ConfPaper (ConfName, PaperTitle)` and `PaperAuthor (PaperTitle, AuthorName)`. If both tables are fed with data from different sources it may happen that the title of one specific paper is spelled differently in both tables. In order to process the example query above, two similarity searches must be executed: (1) over table ConfPaper retrieving the paper titles for the conference name given in the query and (2) over table PaperAuthor retrieving the author names for the paper titles found in step (1) (actually this last step is a similarity join operation [3, 4, 5, 6]).

The classical database solution for processing such types of queries is to build a query *execution plan*. An execution plan defines the query operators that are used in each step as well as the order of execution of those operators. In the context of similarity queries additionally to the classical database operators (table scan, index scan, join,...), similarity operators like a similarity table scan [3], similarity index scan [7] and a similarity join [4, 5] are involved. The problem of building query execution plans for this type of queries has already been addressed in query systems that handle vague queries, like Vague and Query Refinement System Architecture [8, 9].

An open problem in these systems is to automatically determine the results that are relevant for a query. The IR approach described above presents all results to a user and leaves to him the task of picking up relevant results. This approach is not feasible if we are handling large data sets. The entire set of database values would appear as result of each step of the execution plan, leading to unacceptable performance. Thus, *thresholds* must be established for each similarity operator involved in the query execution plan.

The threshold to be applied depends on factors like the specific similarity metric that is being applied and the set of values to be queried. The threshold will depend also on the quality of the result that the user expects stated, for example, in classical IR measures like recall or precision [10].

In this paper we focus on the problem of semi-automatically estimating recall and precision for queries on a specific database column when a specific similarity metric is applied. The user intervention required in our approach is small. The user must just provide an approximation of the number of different real world objects that are represented in a small sample (typically 50 values) of the database column. Using this number as input and applying the process described in this paper the database system may generate meta-data that contains estimations of recall and precision for queries on the column, when different thresholds and different similarity metrics are considered. This meta-data may be subsequently used during query optimization phase. It may help the query optimizer in the choice of similarity operators, similarity metrics, as well as thresholds to be used when processing a specific query.

It should be noted that the idea of gathering information about the values in the columns in a database is central to query optimization [11, 12]. The query optimizers

of many commercial DBMS depend on meta-data gathered at specific time points determined by a database administrator. In the case of the process described in this paper the database administrator would have to additionally provide the information for the recall and precision estimation process.

This paper is organized as follows. In Section 2 related work is discussed. Section 3 describes our method to estimate recall and precision values. Section 4 presents the experiments that were performed in order to validate the proposed approach. Section 5 presents the conclusion and discusses future work.

2 Related Work

The idea of vague or inexact queries over databases is not new and has been studied from several points of view.

Vague or Imprecise Queries Over Database. The Vague System [8] and the Query Refinement System Architecture [9] are examples of vague query database processing. The former discusses a query language (an extension of QUEL) and a general model of vague queries implementation over a relational database. In this system the similarity metrics (called data metrics) are user provided. The latter discusses the problem of query refinement in the presence of similarity queries. Both cited approaches assume the availability of a user provided distance measure.

Another system that implements vague queries is the Imprecise Query Engine (IQE)[13]. Here the query engine is implemented over a classical query engine to handle vague queries. The similarity query engine converts the vague query into equivalent precise queries that appear in an existing query workload.

Probabilistic Algebra. Dey [14] has proposed an extended relational model and new algebraic operators supporting probabilistic aspects. Fuhr [15] presents the PRA (Probabilistic Relational Algebra) which is a generalization of the relational algebra. PRA represents a logical data model allowing close integration of IR and database to model probability values, but makes no assumptions about the underlying physical data model. The WHRIL [16] system is also based on Fuhrs work and uses text-based similarity and logic-based data access as known from Datalog to integrate data from heterogeneous sources. The work of several other authors follow the same line[17, 18, 19, 20]. These proposals include the use of redefined relational operators as select, join, etc. with a probability value associated to each attribute or tuple. This value is a measure of uncertainty obtained from a probabilistic model based on preprocessing of stored data.

These approaches use probabilistic models to compute similarity values in a preprocessing phase. This is similar to our pre-processing phase to create meta-data. However, in these approaches probability values must be stored associated with tuples or attributes whereas in our approach similarity values are dynamically computed.

Probabilistic Model and XML. Several proposals explore the probabilistic IR model. An example considering XML documents is the TIJAH system [21], an XML-IR system

where XML documents are treated as 'flat-text'. This query model extends XPath with a special function called *about*. TIJAH is based on region algebra [22] and it is used to rank node-set trees. The idea behind region algebra [22] is the representation of text documents as a set of extents where each one is defined by its starting and end position. Another similar approach is XIRQL [23], which develops an algebra that implements the querying capabilities found in XPath extended with probabilistic functions. This approach is different from ours because we apply specific similarity metrics for each column.

IR-Style Ranking. Several IR-style systems implement *k-top* queries instead of considering a rank that is cut by a threshold value [24, 25, 26].

Additionally to the study of similarity searches over data sets, the problem of *proximity joins* or *similarity joins* has also received attention. Gravano [4, 5] describes an approach for similarity based on joins on string attributes. This work is based on the identification of all string pairs (or set of strings) similar to each other using cosine similarity metric [10] with weights derived from term frequency-inverse document frequency (tf-idf) to join similar data. Cohen [27] describes WHRIL, which is also based on cosine similarity to integrate information from structured information sources that contain textual information. Cohen describes efficient algorithms do compute the top scoring matches of a ranked result set.

Cohen [28] presents a survey comparing several similarity metrics for specific domains. This work shows that the quality of the result of a query may be improved if specific domain similarity metrics are used.

Schallehn [3, 29] presents a set of redefined relational operators to process vague queries. His work shows how the operators can be used to evaluate the query using these redefined operators.

What can be generally observed in related work is that a critical point is how to specify a threshold to meet requirements regarding efficiency and accuracy [29]. This should not be done by the user because it would lead to several trial-and-errors cycles. Our approach differs also in evaluate the quality of intermediate results. None of related work studied refers to evaluate the result set produced by similarity functions specific for domains.

3 The Estimation Process

In this section we describe the process by which a vague query engine can estimate recall and precision for a database column.

At specific time points defined by a database administrator (DBA), traditional DBMS gathers statistics about the database (number of different values in a column, distribution of the values in a column,...). These statistics are used by the query engine during so called *cost optimization* [30].

In our case, the aim of this preprocessing phase is to estimate recall/precision tables for approximate queries on specific database columns. A recall/precision table contains estimated precision and recall values for several different threshold values.

For each similarity metric that may be applied to the column, a recall/precision table will be generated. These values can be used to optimize the query in the query processing phase.

The specific metrics that can be used depend on the column domain [28]. For example, a column containing **author names** and a column containing **dates** probably would require different similarity metrics. The association of similarity metrics to database columns could be established in the database schema.

Further for the same domain several different similarity metrics may be applied. For example, in a column with person names if person names are always written in the same order (e.g. name, surname) but may spelled in different ways, an edit distance metric like Levenshtein [1] is adequate. However if the words that comprise the name may appear in different orders another kind of similarity metric may be used.

Therefore, in our approach we allow several different metrics for each column and estimate recall/precision for each of them. This information may be used by the query engine to decide which of them is better suited for the specific set of values that appear in the database.

Notice that recall/precision tables need to be generated only for those database columns that may appear as arguments in vague queries.

The process of estimation is executed once for each database column and comprises the steps described below.

1. *Sampling*
   A random sample of the values in the database column is generated. Our experiments have shown that a sample of 50 values is enough.
2. *DBA intervention*
   The values in the sample are displayed to the database administrator (DBA). The DBA counts the number of different real world objects that are represented by the values in the sample. Remember that different values (e.g. "VLDB" and "Very Large Databases") may represent the same real world object.
   The DBA enters the number n_o of real world objects that appear in the sample in the system.
3. For each similarity metric that may be applied to the database column the following steps are performed.
   (a) *Clustering*
   The values in the sample are clustered. Clustering begins with a predetermined threshold and is repeated iteratively with different thresholds until the number of clusters n_c is equal to the number of real world objects (n_o) that were identified by the user in the sample.
   The underlying idea is that, if the similarity metric behaves correctly, each cluster will contain values that represent a single real world object.
   (b) *Recall/precision computation*
   The usual approach to compute precision and recall requires user intervention. In this approach queries are stated against the database and the user identifies false positives and false negatives in the result set.
   In our approach we aim at minimizing user intervention. Recall/precision will be automatically computed by the procedure described below.

Each value in the sample is used as the query value. This means that with a sample of size 50, we will execute 50 queries. Each query will result in a set of ranked values.
In order to estimate recall/precision the set of objects that should result from the query must be identified. This would usually require user intervention. In our approach we will use the cluster instead in which the query value is contained as the set of values that should be returned. Therefore, we will use the clustering result instead of users intervention.
Our approach is based on the assumption that the clustering process has partitioned the sample correctly in sets such that each one contains exactly those values that represent one and only one real world object. Thus, the clusters are used as the set of values that should be returned.
This way, recall and precision are computed for several thresholds. The average value of recall/precision considering all queries is regarded as the recall/precision for the similarity metric in several thresholds.

(c) *Storage of meta-data*
Recall/precision values for each threshold and each similarity metric are stored as meta-data for usage during query optimization phase.

4 Experiments

In this section we describe the experiments performed in order to empirically evaluate that the estimated recall/precision values hold also for the entire database.

4.1 Data Sets

For the experiments two data sets were chosen. Data set *City-DS* contains city names and data set *Street-DS* contains street names. These sets were taken from a real world database that contains information about students that are candidates to enrolment in a Brazilian University. Both data sets refer to the student's address. Most of these candidates come from a single Brazilian state. In both data sets data was entered directly by the candidates themselves. Thus the names of cities and streets may appear spelled in several ways due to several factors, like misspelling, different ways of abbreviation, etc.

The main characteristics of these data sets are shown in Table 1. The number of real world objects in each data set was counted by an human expert.

Table 1. Main characteristics of the data sets used for the experiments

Data set	Number of instances in the database	Number of real world objects	Average number of instances in a cluster
City	10180	387	26.3049
Street	3500	2377	1.4724

The value distribution in both sets presents several differences:

- The *City-DS* contains relatively few (387) real world objects represented. As most of the students come from the same Brazilian state, their addresses concentrate city names of this state. Approximately 45% of the values correspond to a single real world object, the largest city in the state. In the average each city appears 26 times in the database.
- The *Street-DS* contains many different real world objects (2377) as the number of different street names is much bigger than the number of cities. In average, each street is represented 1.4 times in the database.

4.2 Similarity Functions and Clustering Algorithm

As similarity metrics we have applied three well known metrics for comparing strings: Levenshtein or Edit Distance (*Edit*) [1], *Guth* [31] and *N-grams* [1] with 3 characters in each gram. Additionally we have applied a similarity metric (*Acronyms*) developed in our group that is adequate for the comparison of strings that contain abbreviations and acronyms [32].

The results of all similarity metrics applied are normalized between 0 and 1. Clustering was performed using the Hierarchical Agglomerative Clustering Method [33]. The SLINK [34] Algorithm was used to implement clustering process.

4.3 Sample Generation

Due to the fact that the user must count the real world objects represented by the values in a sample we need to be careful with the sample size. Very small samples are not trustworthy to represent the database content but large samples are inappropriate to user interaction.

We have experimented with two sample sizes: 50 instances for the *City-DS* and 15 instances for the *Street-DS*. As the experimental results show, a sample size around 50 values leads to correct results compared to the whole database. The values in each sample were randomly selected in the database.

4.4 Clustering Results

As mentioned above, our approach is based on the assumption that the clustering process has partitioned the sample correctly in sets such that each one contains exactly those values that represent one and only one real world object.

To empirically validate this assumption we extracted 40 samples (each with 50 instances) from the *City-DS*.

In each sample each cluster was compared to the set of values that a user would consider as representing a single real world object. In these 40 samples 352 cities were represented. Two types of errors of the clustering process could be identified:

- The number of clusters does not converge to the number of real world objects. Even with small variations in the threshold (0.01) either the number of clusters exceeds the number of objects in the sample or the number of clusters is lower than the

Table 2. Clustering **errors** found considering 40 samples clustered and validated by the user

Metric	Number of clusters is incorrect	Content of cluster is incorrect
Edit	none	none
Guth	1.6%	2%
N-Grams	0.4%	0.4%
Acronyms	none	none

number of objects in the sample. This probably is an indication that the similarity metric is not adequate for the set of values.
- The correct number of clusters has been identified but their content is not correct, i.e., values that should appear in one cluster appear in the other.

The number of errors found in this process was very low and is summarized in Table 2.

For *Edit* and *Acronyms* similarity metrics no errors were found. The clustering process gave exactly the same results as expected by the user.

For *Guth* and *N-gram* similarity metrics a small percentage (around 1%) of clusters were incorrectly identified.

Those results show an high percentage of correct clusters. We can conclude that it is acceptable to use the clusters for recall/precision evaluation.

4.5 Recall/Precision Estimation

The other premise in which our approach is founded is that the values of recall/precision that were calculated for the sample apply also to the entire database. In this section we will show experimental results to validate this premise.

We have performed experiments with both data sets.

Experiments with *City-DS*. Using the *City-DS* we first executed the estimation of recall/precision by applying the aforementioned process (Section 3). More specifically the following steps were executed:

1. We took 4 samples each containing 50 values from the cities data set.
2. For each sample an human expert determined the number of real world objects (cities in this case) represented by the values in the sample. This corresponds to the user intervention that should be executed by the database administrator during the pre-processing phase.
3. The samples were clustered as described above.
4. Each value in a sample was used as a query value in that sample. For each query, recall and precision were computed by the procedure aforementioned. We took the following values for the thresholds 0.9, 0.8, 0.7, 0.5 and 0.3. This resulted in 4 tables, one for each sample, containing recall/precision values for each threshold.
5. We took the average of the samples resulting in a single table with estimated recall/precision values for each threshold.

In order to evaluate these results we compared them to the results of queries against the complete data set. Each of the values in the samples (4 samples $*$ 50 values per sample $=$ 200 values) was taken as a query value against the database. Again a table plotting recall/precision for each of the thresholds above was computed. This procedure was repeated for each of the four similarity metrics.

The results are shown in Figure 1. In this figure each graph corresponds to one similarity metric. The x-axis corresponds to the thresholds and the y-axis corresponds to similarity values. The values plotted are recall and precision. Dotted lines represent actual recall/precision values obtained for the entire data set, whereas continuous lines represent the estimated recall/precision values obtained from the samples.

As can be seen in the figure, estimated recall/precision follows similar curves to actual recall/precision for all four similarity metrics.

The results show also that some similarity metrics are more adequate than others. In this case, Edit and N-Grams are better metrics, since the values of recall and precision tend to be higher and less dependent from the threshold values. Guth and Acronyms are less adequate as precision decreases faster whit smaller thresholds.

In order to evaluate how close the estimated results for the sample are to the actual results calculated over the database, we computed the Mean Square Deviation (MSD) as defined by Equation (1).

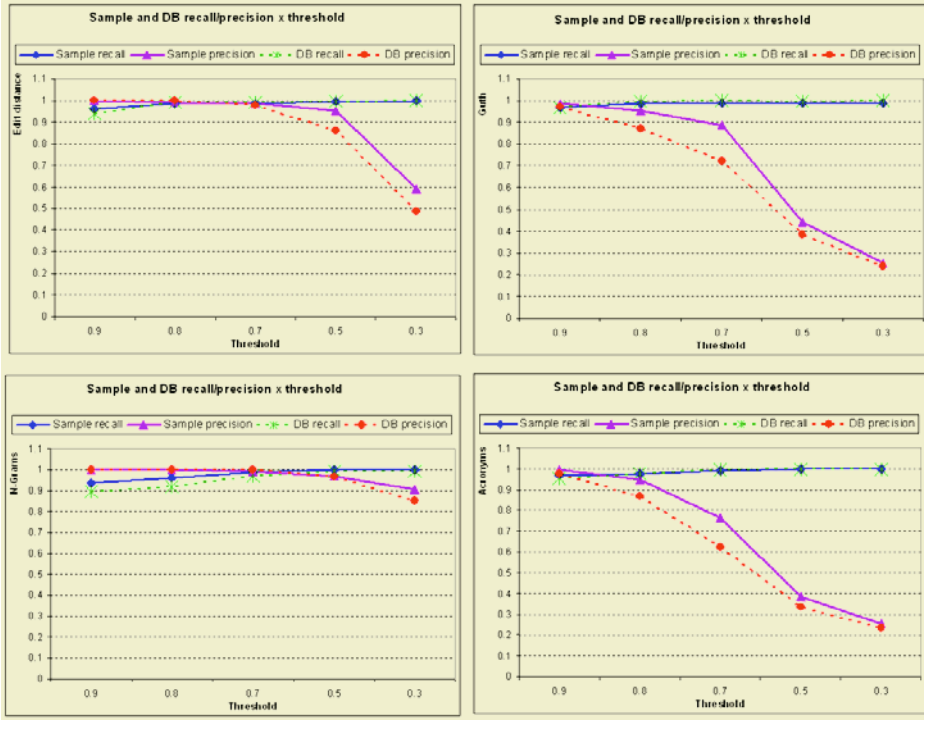

Fig. 1. City-DS – Comparing sample and database recall/precision

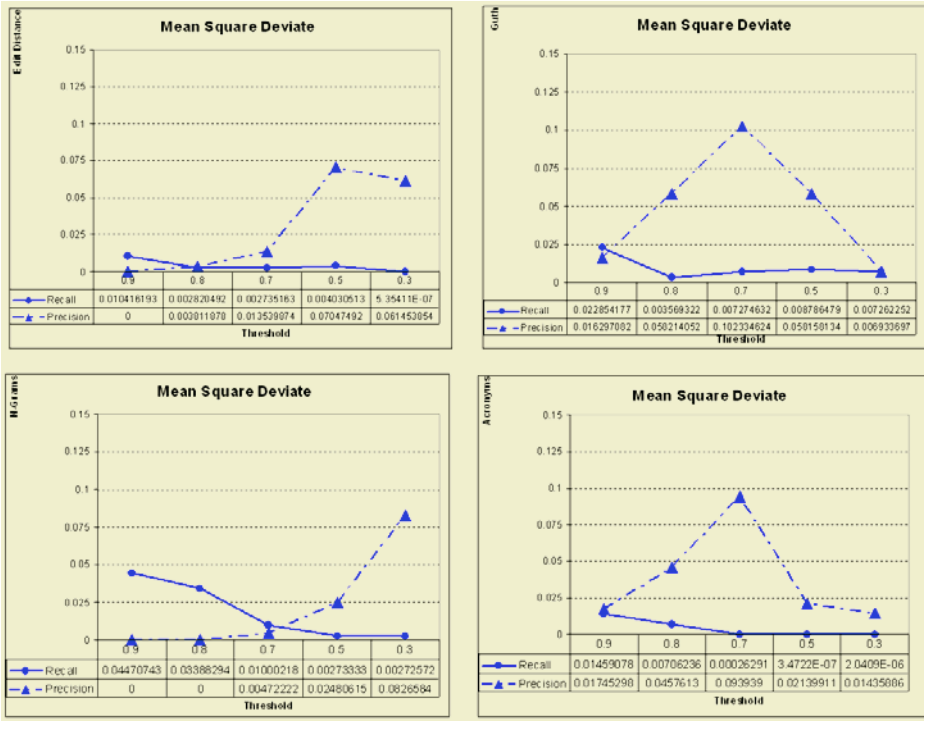

Fig. 2. City-DS: Mean Square Deviation between samples and database

$$f(x^b_M, x^s_M) = \frac{1}{n}\sum_{i=1}^{n}(x^b_{M,i} - x^s_{M,i})^2, \tag{1}$$

where $x^b_M = (x^b_{M,1}, x^b_{M,2}, ..., x^b_{M,n})$ is a similarity value of the database and $x^s_M = (x^s_{M,1}, x^s_{M,2}, ..., x^s_{M,n})$ is a similarity value of the sample.

In Figure 2 the values of MSD obtained from Equation 1 using thresholds 0.9, 0.8, 0.7, 0.5 and 0.3 with each similarity function are shown. The values shown in that figure present the mean MSD (MSDm) for all four samples.

As can be seen the values are low showing that the estimated values are close to the actual values for the database.

We have observed that some clusters contain many duplicate values. This leads to higher similarity values. We repeated the experiment described above removing the duplicate values from the clusters. In this case similarity values are lower but still the estimated recall/precision values are similar to the actual recall/precision values for the database. Due to space restrictions the detailed results of this experiment are not shown here.

Experiments with *Street-DS*. To test the limits of our approach we also evaluated a data set much harder to handle, the *Street-DS*.

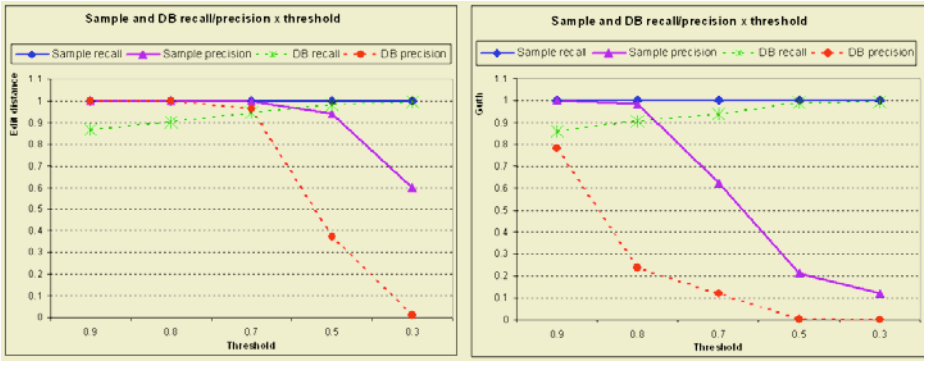

Fig. 3. Street-DS – Comparing sample and database recall/precision

The *Street-DS* was chosen because the relation between data values and real world objects is different from that in the *City-DS*. The number of real world objects is similar to that of values in the database, i.e. clusters tend to be small, many of them containing just an instance. In this case the quality of the results depends much more on the ability of the similarity metric to handle data from this domain. As few values represent each real world object every false positive or false negative changes the values of recall and precision by a considerable amount.

The results are shown in figure 3. We have used two metrics, Edit, that gave the best results in the previous experiment and Guth that gave the worst results. We have also used smaller samples (15 values) than in the previous experiment.

As can be seen in the figure, in this example the precision that was estimated is much smaller than the actual precision measured on the database. This difference is due to the inability of these similarity metrics to correctly identify which values represent the same real world object and which do not. This inability appears more clearly in the database than in the sample, because the sample contains less instances and the query value is part of those instances.

5 Concluding Remarks

This paper presents a contribution to the problem of query execution and optimization in a query engine that handles vague queries. Specifically we have presented an approach for estimating recall/precision values for queries with several thresholds. These values are important for query engines like that described in [8, 9].

The estimation process is to be started by a database administrator when he estimates that the distribution of values in the database has changed. We tried to minimize user intervention. The database administrator enters just a single information, namely the number of different real world objects that are represented in a small sample of the database. Based on our experiments, we can improve the sampling process through learning methods in future works.

We have described the experiments that empirically validate our approach. Specifically the experiments corroborate two premises.

- In order to estimate recall and precision we need to identify the set of values in a sample that represent a single real world object. In our approach these sets correspond to the result of the clustering process. The experiments show that the result of the clustering process may be used instead of the identification of values by a user.
- When the sample is big enough and similarity metrics are adequate for the column domain the recall/precision results are very similar to the actual recall/precision values obtained when querying the database.

However, several problems are still open.

As identified by the experiments (and also by other authors [28, 3]) some similarity metrics are more adequate than others for handling a specific column. We are working on heuristics that use the recall/precision estimations to identify what the best similarity metric for a column is.

Further, the size of the samples obviously affects the results of the estimation process. We are working on the problem of identifying what the minimum sample size for a given data set is.

References

1. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys **33** (2001) 31–88
2. Santini, S., Jain, R.: Similarity measures. IEEE Transaction on Pattern Analysis and Machine Intelligence **21** (1999) 871–883
3. Schallehn, E., Sattler, K.U., Saake, G.: Efficient similarity-based operations for data integration. Data Knowl. Eng. **48** (2004) 361–387
4. Gravano, L., Ipeirotis, P.G., Koudas, N., Srivastava, D.: Text joins in an RDBMS for web data integration. In: Proceedings of the Twelfth International Conference on World Wide Web, ACM Press (2003) 90–101
5. Gravano, L., Ipeirotis, P.G., Koudas, N., Srivastava, D., Muthukrishnan, S.: Approximate string joins in a database (almost) for free. In: Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, VLDB 2001, Morgan Kaufmann (2001) 491–500
6. Schallehn, E., Geist, I., Sattler, K.U.: Supporting similarity operations based on approximate string matching on the web. In Meersman, R., Tari, Z., eds.: CoopIS/DOA/ODBASE (1). Volume 3290 of Lecture Notes in Computer Science., Springer (2004) 227–244
7. Navarro, G., Baeza-Yates, R.A., Sutinen, E., Tarhio, J.: Indexing methods for approximate string matching. IEEE Data Engineering Bulletin **24** (2001) 19–27
8. Motro, A.: Vague: a user interface to relational databases that permits vague queries. ACM Trans. Inf. Syst. **6** (1988) 187–214
9. Ortega-Binderberger, M.: Integrating Similarity Based Retrieval and Query Refinement in Databases. Phd thesis, UIUC - University of Illinois at Urbana-Champaign, Urbana, Illinois (2002)
10. Baeza-Yates, R.A., Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)

11. Ullman, J.D., Garcia-Molina, H., Widom, J.: Database Systems: The Complete Book. Prentice Hall Inc., Upper Saddle River, New Jersey, USA (2002)
12. Chaudhuri, S.: An overview of query optimization in relational systems. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, ACM Press (1998) 34–43
13. Nambiar, U., Kambhampati, S.: Answering imprecise database queries: a novel approach. In: Proceedings of the fifth ACM international workshop on web information and data management, ACM Press (2003) 126–133
14. Dey, D., Sarkar, S.: A probabilistic relational model and algebra. ACM Trans. Database Syst. **21** (1996) 339–369
15. Fuhr, N., Rolleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst. **15** (1997) 32–66
16. Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, ACM Press (1998) 201–212
17. de Keijzer, A., van Keulen, M.: A possible world approach to uncertain relational data. In: 15th International Workshop on Database and Expert Systems Applications (DEXA 2004) Workshops. SIUFDB-04 1st International Workshop on Supporting Imprecision and Uncertainty in Flexible Databases, Zaragoza, Spain, September 3, 2004, IEEE Computer Society (2004)
18. Lakshmanan, L.V.S., Leone, N., Ross, R., Subrahmanian, V.S.: Probview: a flexible probabilistic database system. ACM Trans. Database Syst. **22** (1997) 419–469
19. Fuhr, N.: A probabilistic relational model for the integration of ir and databases. In: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press (1993) 309–317
20. Barbara, D., Garcia-Molina, H., Porter, D.: A probabilistic relational data model. In: Proceedings of the international conference on extending database technology on Advances in database technology, Springer-Verlag New York, Inc. (1990) 60–74
21. List, J., Mihajlovic, V., de Vries, A.P., Ramirez, G., Hiemstra, D.: The TIJAH XML-IR system at INEX 2003. Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX 2003), ERCIM Workshop Proceedings (2003)
22. Consens, M.P., Milo, T.: Algebras for querying text regions: expressive power and optimization. J. Comput. Syst. Sci. **57** (1998) 272–288
23. Fuhr, N., Grossjohann, K.: XIRQL: a query language for information retrieval in XML documents. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press (2001) 172–180
24. Fagin, R., Kumar, R., Sivakumar, D.: Efficient similarity search and classification via rank aggregation. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM Press (2003) 301–312
25. Ortega, M., Chakrabarti, K., Mehrotra, S.: Efficient evaluation of relevance feedback for multidimensional all-pairs retrieval. In: Proceedings of the 2003 ACM Symposium on Applied computing, SAC 2003, ACM Press (2003) 847–852
26. Chakrabarti, K., Ortega-Binderberger, M., Mehrotra, S., Porkaew, K.: Evaluating refined queries in top-k retrieval systems. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 256–270
27. Cohen, W.W.: Data integration using similarity joins and a word-based information representation language. ACM Trans. Inf. Syst. **18** (2000) 288–321
28. Cohen, W.W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico, Morgan Kaufmann (2003) 73–78

29. Schallehn, E., Sattler, K.U.: Using similarity-based operations for resolving data-level conflicts. In: BNCOD. (2003) 172–189
30. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In Bernstein, P.A., ed.: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, May 30 - June 1, ACM (1979) 23–34
31. Guth, G.J.: Surname spellings and computerized record linkage. Historical Methods Newsletter **10** (1976) 10–19
32. Dorneles, C.F., Lima, A.E.N., Heuser, C.A., da Silva, A., Moura, E.: Measuring similarity between collection of values. In: Proceedings of 6th ACM International Workshop on Web Information and Data Management (WIDM 2004), Washington DC , USA, ACM Press (2004) 56 – 63
33. Hartigan, J.A.: Clustering Algorithms. John Wiley and Sons, Inc., New York, NY, USA (1975)
34. Sibson, R.: SLINK: an optimally efficient algorithm for the single-link cluster method. The Computer Journal **16** (1973) 30–34