Efficient Feedback Collection for Pay-as-you-go Source Selection

DOI:
10.1145/2949689.2949690

Document Version
Accepted author manuscript

Efficient Feedback Collection for Pay-as-you-go Source Selection

Julio César Cortés Ríos,
Norman W. Paton,
Alvaro A.A. Fernandes
School of Computer Science
University of Manchester
Manchester M13 9PL, UK
(juliocesar.cortesrios,npaton,
alvaro.a.fernandes)@manchester.ac.uk

Khalid Belhajjame
Université Paris Dauphine
Place du Maréchal de Lattre de Tassigny
75775 Paris Cedex 16, France
Khalid.Belhajjame@dauphine.fr

ABSTRACT

Technical developments, such as the web of data and web data extraction, combined with policy developments such as those relating to open government or open science, are leading to the availability of increasing numbers of data sources. Indeed, given these physical sources, it is then also possible to create further virtual sources that integrate, aggregate or summarise the data from the original sources. As a result, there is a plethora of data sources, from which a small subset may be able to provide the information required to support a task. The number and rate of change in the available sources is likely to make manual source selection and curation by experts impractical for many applications, leading to the need to pursue a pay-as-you-go approach, in which crowds or data consumers annotate results based on their correctness or suitability, with the resulting annotations used to inform, e.g., source selection algorithms. However, for pay-as-you-go feedback collection to be cost-effective, it may be necessary to select judiciously the data items on which feedback is to be obtained. This paper describes OLBP (Ordering and Labelling By Precision), a heuristics-based approach to the targeting of data items for feedback to support mapping and source selection tasks, where users express their preferences in terms of the trade-off between precision and recall. The proposed approach is then evaluated on two different scenarios, mapping selection with synthetic data, and source selection with real data produced by web data extraction. The results demonstrate a significant reduction in the amount of feedback required to reach user-provided objectives when using OLBP.

CCS Concepts

•Information systems → Mediators and data integration;

SSDBM '16, July 18-20, 2016, Budapest, Hungary

DOI: http://dx.doi.org/10.1145/2949689.2949690

Keywords

Data integration; mapping selection; source selection; feedback collection; pay-as-you-go

1. INTRODUCTION

There are ever growing numbers of data sources resulting from technology developments such as the ability to publish to the web of data [19], or to extract data systematically from web forms (e.g. [16]) or tables (e.g. [7]). In addition, given numerous data sources, it is then possible to generate even more numerous views that combine data from different sources to provide an integrated representation of the information of relevance to a user (e.g. [4]).

Abstracting over the precise origin of a data collection, henceforth we use the term *data source* to refer to a structured data collection to which we have access but over which we may have little control; this data resource may either represent a materialized data source or a virtual data source (e.g. produced using schema mappings).

As the available data sources may be of variable quality or relevance, it may be important for the quality or cost of the user experience to select a subset of the available sources that meet their needs. As a result, source selection has been studied by others, where selection criteria have included features such as economic value [12], freshness [31] and quality [29].

Where there may be a large and rapidly changing pool of sources, it may be difficult, or prohibitively expensive, for experts to manually select the sources that meet user needs. As a result, in this paper we focus on pay-as-you-go approaches, particularly in which there is automated data extraction or mapping generation, with user feedback on values from the sources. We envisage that feedback may take the form of true or false positive annotations on data items, an approach that has been followed for different data access and integration tasks (e.g. [3, 6, 34]); such feedback could potentially come from end users or from crowd workers.

Pay-as-you-go approaches show promise when dealing with web scale data [26], but it is clearly necessary to try to maximise the return on investment; obtaining feedback from users or crowds involves costly human effort. Making the best use of sources involves making well informed decisions as to which feedback can yield the greatest improvement in a result. There has been significant recent interest in targeting the most effective feedback, in particular for crowdsourcing

(e.g. [11, 21, 23, 28, 35]).

In this paper, we contribute further to this active line of research into cost-effective feedback collection, by investigating pay-as-you-go data source selection. We assume that the user may be willing to trade off *precision* (the fraction of the data items in the selected sources that are correct) with *recall* (the fraction of the correct data items that are included in the selected sources). For example, the user may want the system to maximise the precision of the data from the selected sources, subject to the constraint that these sources must contain at least half of the correct data (i.e. $recall > 0.5$). However, in general we do not know the precision or recall of the available data sources, so these need to be estimated based on feedback. Assume we have a fixed budget that allows us to collect feedback on f data items. The problem is to identify the f data items on which feedback should be obtained in order to allow the most effective decisions to be made on which sources to use.

Note that in this problem we are interested in obtaining a set of sources that together best meet the user's requirements, and not the single best source; this distinction with some earlier results (e.g. [11, 23]) turns out to be significant in practice. In addressing this problem, we make the following contributions:

1. an investigation into the source selection problem, which identifies the important role that the confidence level in quality estimates plays in the production of suitable results;
2. an algorithmic strategy that can be used to prioritise data items for annotation, and then to efficiently select a suitable subset of the available sources in the light of the feedback received; and
3. an empirical evaluation of the approach in scenarios where sources result from both mapping generation and web data extraction, the latter building on real web data sources.

Our contributions show that our proposed approach achieves high quality source selection with fewer instances than the baseline used for comparison. As such, this approach increases the impact on quality of each requested feedback instance.

The paper is structured as follows. In Section 2, we define the problem more precisely, in particular identifying three variants that are supported within our approach and the role of feedback. In the light of the problem description, in Section 3 we review related work on source selection and on targeted feedback collection. In Section 4 we describe the overall approach, which includes both targeted feedback collection and source selection; the algorithms that describe how the approach is implemented are detailed in Section 5. The approach is evaluated empirically in two different technical settings, with algorithmically generated mappings in Section 6, and with sources from web data extraction in Section 7. Conclusions are presented in Section 8.

2. PROBLEM DESCRIPTION

The source selection problem we consider in this paper can be defined as follows: given a set of sources, and feedback in the form of true positive (TP) or false positive (FP) annotations on the data items they contain, identify the subset of the sources that best meets the user's requirements in terms of *user-provided* precision and recall thresholds.

In the light of this definition, we identify three different ways in which the users can specify their requirements in terms of precision and recall, in the context of a set of sources S:

Variant MaxP: Maximise precision while meeting a recall constraint.

maximise (for some S' ⊆ S) precision(S')
such that recall(S') > recall-threshold

Variant MaxR: Maximise recall while meeting a precision constraint.

maximise (for some S' ⊆ S) recall(S')
such that precision(S') > precision-threshold

Variant MaxPN: Maximise precision while meeting a cardinality constraint.

maximise (for some S' ⊆ S) precision(S')
such that $|\{s|s \in S'\}|$ > cardinality-threshold

In *Variant MaxP*, a set of sources S' is produced that has maximum precision while satisfying the constraint that the recall of S' is greater than a given *recall-threshold*. For example, assume that the user is interested in obtaining an insight into the real estate market in an area. For this, the user doesn't need to look at every property in the area, but may decide that a sample of 10% of the available properties would be sufficient. Then, the user would like to have the highest precision possible in the resulting sample. Hence the request is to maximise precision such that the *recall-threshold* is 0.1.

In *Variant MaxR*, a set of sources S' is produced that has maximum recall while satisfying the constraint that the precision of S' is greater than a given *precision-threshold*. For example, assume that the user is interested in analysing the used cars available in an area, as part of a study on pricing trends. For this, the user requires good quality data, and thus may want to use as many sources as possible (by maximising recall) such that the *precision-threshold* is 0.95.

In *Variant MaxPN*, a set of sources S' is produced that has maximum precision, such that the number of data items within S' is greater than a given *cardinality-threshold*. For example, assume that the user will browse real estate entries manually, and expects only to look at 100 data items. For this, the requirement is to identify the best quality sources that can between them provide 100 distinct items.

These variants depend on *precision* and *recall* functions; however, precision and recall are classically used to measure results quality against a ground truth, to which we do not have access, as the time and/or cost required to label every single data item will be prohibitive in many scenarios. Thus we use feedback to provide estimates of the ground truth.

The precision of a source (or sources) s in the context of user feedback UF, can be estimated by counting true positive (tp) and false positive (fp) annotations of s in UF:

$$precision(s, UF) = \frac{|tp(s, UF)|}{|tp(s, UF)| + |fp(s, UF)|} \quad (1)$$

Similarly, the recall of a source (or sources) s can be estimated as follows:

$$recall(s, UF) = \frac{|tp(s, UF)|}{|tp(S, UF)| + |fn(S, UF)|} \quad (2)$$

where S is the complete set of sources and fn is the number of false negative annotations of S in UF.

Now the problem is to identify the data items on which to obtain feedback that will enable good precision and recall estimates to be calculated, thereby allowing cost-effective solutions to the optimization problems described above as *Variant MaxP*, *Variant MaxR* and *Variant MaxPN*.

On these problem scenarios we consider a setting where the human users providing feedback have been informed of the intent of the source selection, e.g. to get the best available sources for Italian restaurants in the UK, and therefore they are assumed to be able to provide feedback in line with the intent, by being asked, if a given data item is a true positive, against a provided conceptual model. We are also assuming that the feedback provided contains no errors, but several techniques can be applied to allow for unreliable evidence (e.g. [38], [10]).

3. RELATED WORK

In the light of the problem description from Section 2, here we discuss relevant related work, specifically for *targeted feedback selection*, *mapping selection* and *source selection*.

3.1 Targeted feedback selection

This section discusses approaches to targeted feedback selection, emphasising work on data management. We start by reviewing results that build on active learning, which can be seen as a generic strategy, and then describe bespoke techniques developed to exploit knowledge of the task at hand.

Active learning builds on the hypothesis that a machine learning algorithm can perform better if it is allowed to select its own training data [32]. Given an oracle that is assumed to be able to provide training data (e.g. the user of a system or a crowd worker), active learning provides techniques for selecting which questions to put to the oracle.

For example, Isele *et al.* [23] use active learning to obtain feedback on which pairs of instances represent the same real world objects, to inform the generation of linkage rules that can be used for matching such instances [22]. Active learning has developed a range of question selection strategies; Isele *et al.* use *query-by-committee*, where the most suitable question is identified as a result of voting by a committee of candidate solutions – in this case, each candidate solution is a linkage rule. The question put to the user is established following a *query by vote entropy* strategy [32], that selects the value about which the members of the committee disagree the most. Thus each linkage rule identifies a set of pairs of instances that it considers to be equivalent. The question selection strategy then selects a pair that is considered to match by some of the rules but not to match by others; the pair about which there is most disagreement is identified using an entropy measure. The empirical evaluation of the approach showed both that much better f-measures for the best rule could be obtained for a given amount of feedback selected using active learning than when the feedback was selected at random, and that good overall f-measures could be obtained for a range of data sets when obtaining feedback on a small portion of the overall data set.

Active learning has been used in a range of data management problems. Also for record linkage, in Corleone [17], an entropy measure is used to select pairs of records for feedback on which decision tree classifiers disagree the most, with additional results on how to identify sets of pairs for crowdsourcing and on deciding when to stop obtaining feedback. For web data extraction, Crescenzi *et al.* [11], use active learning to distinguish between automatically generated extraction rules, with feedback on the results of the rules obtained using a variant of vote entropy that takes into account the probability of the correctness of the rules. For large scale classification tasks, Mozafari *et al.* [28] investigate the application of active learning for crowdsourcing, to address issues such as obtaining sets of results from crowds, coping with unreliable data, and generalising the approach to arbitrary classification problems.

These approaches normally select a single item from a collection, and therefore definitively discarding items is encouraged (e.g. [11, 22]), which differs from our problem in that the solution may potentially include any item in the collection to satisfy the user requirements.

However, some data management problems may not be amenable to solution using existing active learning strategies, and researchers have developed a number of bespoke techniques for obtaining targeted feedback. A prominent category is that of crowd database systems such as CrowdDB [15] and Quirk [27]; in such systems, access to the crowd takes place in the context of a query, and the query optimizer takes responsibility for minimizing the cost of crowd tasks taking into account the whole query, and possibly other factors such as the reliability of different crowd workers. In other examples, specific features of the problem play a prominent role in targeting feedback. For example, for skyline queries over incomplete data sets, Lofi *et al.* [25] describe an approach that consults the crowd for missing values based on the risk that a missing value poses to the correctness of the skyline. The Crowdsourcing Data Analytics System (CDAS) [24] takes account of properties of both the problem to be solved and the crowdsourcing setting in trying to carry out just enough crowdsourcing. Initially, a prediction is made of the number of crowd tasks needed to obtain the required accuracy, and as results are received from the crowd a verification process takes into account the reliability of the participating workers to determine if more answers are required.

In both the active learning and bespoke approaches, features of the problem guide the feedback collection process. In this paper we contribute a bespoke approach that, like CDAS but for a different problem, aims to collect feedback cost-effectively.

3.2 Mapping selection

Given techniques (e.g. [18]), that support the automated generation of mappings for integrating data sources (for example building on evidence from matchers), there is then the problem of deciding which of the generated mappings to use. There is a significant body of work on techniques, for use in data integration tools, on validation [5] and refinement [36] of mappings, some based on data examples [1]. However, these typically focus on expert-driven enterprise data integration [4], rather than on the pay-as-you-go setting that is the context for this paper.

There have been several proposals for pay-as-you-go mapping selection, for example using explicit feedback on the mappings themselves (e.g. [8]), on mapping results (e.g. [2]), or on implicit feedback via query logs (e.g. [13]). Perhaps the most closely related study to the work in this paper is Yan *et al.* [37], in which active learning is used to obtain

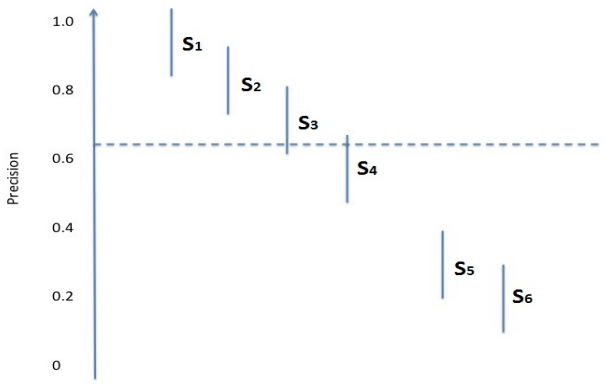


Figure 1: Sources with estimated precisions.

feedback on search results over structured data that is most informative for reducing uncertainty about the validity of the queries that produced the results. The feedback used by Yan *et al.* is of a similar form to that required here, but the objective is to refine understanding of individual mappings, whereas our objective is to identify a set of mappings that, together, meet user-specified requirements. In our approach, mappings are selected based only on estimates of the precision and recall of their results that in turn build on user feedback, as detailed in Section 2.

3.3 Source selection

There are many drivers for the provision of ever more data sources on the web, as diverse as open government initiatives and e-commerce, for which structured data may be published directly (e.g. in the web of data [19]), or extracted programmatically (e.g. from web forms [16]). As these are typically of variable quality, and have significant overlaps, there is a recurring need to identify subsets of the sources that meet user requirements.

In Dong *et al.* [12], sources are added to a collection when it is predicted that the value of the new data outweighs the cost of obtaining it, with the estimated financial value of a source being based on the predicted accuracy of its contents. This is part of a strand of research that seeks to automatically infer the quality of sources [30]. There have also been pay-as-you-go approaches to source selection, in which new or existing feedback on search results is used to inter-relate new and existing sources [33], and in which the user iteratively steers a source selection search by indicating their priorities in terms of the attributes that are required and the priority the user gives to certain quality metrics. This paper shares some objectives with these other results, but selects sources based on different criteria, and emphasises cost-effective targeting of user feedback.

4. TARGETING FEEDBACK USING CONFIDENCE

This section describes the approach taken to the targeting of feedback to inform source selection.
Consider a source selection problem of the form:

maximise (for some S' $\subseteq$ S) precision(S')
such that recall(S') > *recall-threshold*

Having the following set of candidate sources from which a subset will be selected: $S = \{s_1, \cdots, s_6\}$, and assume that feedback has been obtained on all sources in S, so that each s_i has an *estimated precision*, by using Equation 1, each of which has an associated *sampling error* that results from the fact that the feedback is partial. The *sampling error* is the range within which the true value is estimated to be, with some level of confidence. Figure 1 shows the estimated precisions for sources in S, along with their sampling errors, represented by error bars denoting confidence intervals.

To meet the given *recall-threshold* a subset of the sources is selected, and its complement is excluded. The horizontal dashed line in Figure 1 separates *candidate sources* from *excluded sources*. The line can be seen as categorizing two subsets: (i) sources with precisions that may lie above the line (given the confidence associated with their sampling errors) are candidates to be selected (in the figure); and (ii) sources with precisions that lie below the line (with the confidence associated with their sampling errors) are not candidates to be selected ($\{s_5, s_6\}$ in the figure). Sources $\{s_5, s_6\}$ are not candidates because the *recall-threshold* can be met using only the set of sources $\{s_1, s_2, s_3, s_4\}$.

Our source selection approach builds on the following:

1. Obtaining a dependable precision estimate of a set of candidate sources requires reliable estimates of the numbers of true and false positives in each candidate source. This follows from the definition of *precision* in Section 2; the precision estimate of a set of sources s depends only on feedback on the members of s.

2. The recall of a collection of sources depends on having a dependable estimate of the numbers of true positives not only for the candidate sources, but also for the collection as a whole. This follows from the definition of *recall* in Section 2; the precision of a set of sources s depends not only on feedback on the members of s, but also on an estimate of the recall of the complete collection of sources S.

3. For a given problem, it may be appropriate to collect more feedback on some sources than others in order to make well informed decisions on which sources should be included.

In relation to (3) above, in Figure 1, the feedback on sources s_5 and s_6, which are not deemed candidates for inclusion in the solution, does not contribute to good precision estimates for candidate solutions. Furthermore, feedback on sources s_5 and s_6 does not contribute to the numerator of the definition of *recall* in Section 2. Indeed, as discussed below, we can estimate the number of true positives in the complete collection of sources S for the denominator of the definition of *recall* in Section 2 without needing dependable estimates of the numbers of true positives in all the sources contributing to S. We will describe how feedback can be targeted in such a way as to: (i) allow sources to be excluded from further feedback collection as soon as it is established by prior feedback collection that they are not candidates for inclusion in the solution; and (ii) focus feedback in a way that contributes efficiently to the precision and recall estimates that inform mapping selection.

Our strategy is to reduce the sampling error for those sources with higher estimated precision, on which we will need to collect more feedback instances to improve their

precision and recall estimates. The sampling error is the margin of error around our precision or recall estimates; the greater this margin the less confidence we have in those estimates. To compute the sampling error we are assuming a target confidence level of 95%. The associated z-score z for that confidence level is 1.96, assuming a normal distribution.

To obtain the margin of error e we use the classical formulas from statistical theory for standard error se, finite population correction factor fpc and margin of error e [14]:

$$se_s = \sqrt{\frac{q\hat{m}_s \cdot (1 - q\hat{m}_s)}{L_s}} \tag{3}$$

$$fpc_s = \sqrt{\frac{T_s - L_s}{T_s - 1}} \tag{4}$$

$$e_s = z \cdot se_s \cdot fpc_s \tag{5}$$

where s is a source in S, se_s is the standard error, fpc_s is the finite population correction factor, L_s is the number of feedback instances collected for s, T_s is the total number of records produced by s, and $q\hat{m}_s$ is the population proportion or estimated quality measure (estimated precision $\hat{p}$ or recall $\hat{s}$). The result is the margin of error around our estimate, e.g. $\hat{p}_s \pm e_s$, for a given confidence level.

The finite population correction factor is used in those scenarios where the number of records is relatively small, for instance when it is required to collect feedback from less than 5% of the available records.

As our strategy focuses on improving initial estimates for precision and recall, and as shown in Equation 2 recall estimates are based on feedback instances collected from all the candidate sources, we need to obtain an initial sample for bootstrapping for our approach.

Given that the traditional formula for recall in Equation 2 requires the number of false negative annotations, and this can only be obtained by knowing the ground truth, and that the relevant true positive annotations for all the sources in S may be biased by the targeted nature of our algorithm, in which we are collecting more feedback from only a subset of the available sources (those on which we have better estimated quality), we decided to modify the traditional definition of recall with the following that considers both limitations (lack of false negatives and biased sampling):

$$recall(s, iUF) = \frac{|tp(s, iUF)|}{|tp(S, iUF)|} \tag{6}$$

where S is the set of all sources and iUF is an initial random sample of user feedback collected from all the sources in S obtained prior to the process of feedback collection.

A critical element for bootstrapping is to obtain a representative initial sample from the data. To compute a suitable sample size we can rely on the traditional formula for the sample size for estimated proportions ss_0 (in our case estimated precision $\hat{p}$ or recall $\hat{r}$) which is as follows [9, 20]:

$$ss_0 = \frac{z^2 \cdot p \cdot (1 - p)}{e^2} \tag{7}$$

where p represents the estimated proportion (initial estimate for precision $\hat{p}$ or recall $\hat{r}$), e is the required margin of error for our initial sample, and z represents the z-score for a required confidence level assuming a normal distribution of the data.

Equation 7 applies for large populations, but considering that in our case some sources may produce rather few records (e.g. less than 100) we need to apply the finite population correction factor [9, 20] to Equation 7, resulting the following formula to compute the initial sample size for finite populations ss:

$$ss = \frac{T \cdot ss_0}{ss_0 + (T - 1)} \tag{8}$$

where T is the number of distinct items produced by sources in S.

In applying these formulas: we set p, the initial population proportion to 0.5 (in our case this represents the initial estimate for precision $\hat{p}$ or recall $\hat{r}$); the z-score z is set to 1.96, representing a default confidence level of 95%; and the margin of error e is set to 0.05, based on the results from experiments performed over different data sets and quality distributions. We should also assume that all the sources produce a minimum number of records (e.g. 30 instances each), otherwise we will obtain unreliable estimates.

5. ALGORITHM

In this section, we describe our algorithm for source selection using feedback collected in a pay-as-you-go fashion. As the algorithm is based on an ordering of the sources based on their estimated precision, we will refer to the approach as *OLBP* (Ordering and Labelling By Precision). The pseudocode for the algorithm is given in Figure 2.

The algorithm takes as input: S – a collection of sources from which we need to select a subset that together satisfy constraints in the form of one of the variants from Section 2; U – the set of (unlabelled) data items from the sources in S; var – which is either $MaxP$, $MaxR$ or $MaxPN$; thr – the value of the threshold corresponding to var; $step$ – the default number of feedback items that will be obtained in a single interaction with crowd workers or users; $budget$ – the total number of items of feedback that can be obtained; $conf$ – the confidence level required for the estimations; and err – the margin of error also required to compute the estimations. The result of the algorithm is a set of sources $S' \subseteq S$.

To select sources from those available, we require an initial sample of the data therein; this will bootstrap the quality estimates of the candidate sources, and enable recall estimates based on Equation 6. To obtain these estimates we need to compute the initial sample size $iStep$ using Equation 8 (line 2). This sample (and subsequent samples augmented with feedback) are used to compute the margins of error for the precision and recall estimates (line 10).

The feedback collection process is implemented by the *getFeedback* function, which takes as arguments the set of sources S, the set of unlabelled data items U, the number of additional data items on which feedback is required $step$ and the cut-off computed by our approach $lowP$. For experimental evaluation purposes, items may be labelled depending on the required strategy (RND for random selection and OLBP for our approach). The *getFeedback* function randomly selects from U at most $step$ data items to be annotated. The sources considered in this selection depend on required strategy. For random selection, all sources in S are considered. For the OLBP strategy, the cut-off parameter $lowP$ is used to identify the subset of sources in S that are above or overlapping this cut-off; this has the effect of refining the estimates only for those sources that are candidate members of S'. During the first iteration the algorithm always selects $iStep$ data items from all candidate sources in S, to provide enough information for bootstrapping (line 6), and then on

it follows the required strategy (line 8).

After some feedback has been collected by using *getFeedback*, and the quality of the sources estimated (line 10), the sources are sorted by their estimated precision in descending order (line 14). These sorted sources are then combined following a greedy approach (lines 15-26), that gives preference to those sources with the highest estimated precision, and stops when the threshold is met (lines 17-22). If the threshold is not yet reached by adding the new sources to the subset of candidate sources S', we compute a new cut-off, to divide those sources that will be considered in a potential solution from those that will not (lines 16 and 23-25). On the other hand, if the threshold is met by adding the new sources to the collection of candidate sources S', the process finishes but, for the case of *MaxR* we need to remove the last added sources from S' or the estimated precision will drop below the threshold (lines 18-20). This process repeats until we run out of budget (line 27), measured in terms of feedback instances, and the set of selected sources that fulfil the threshold is returned (line 28).

When selecting the sources sorted by their estimated precision, we ensure that only the sources with the highest proportion of true positive annotations are considered for the solution, and we assume that the rest of the sources (with lower estimated precision), if added to the collection, will only diminish its overall quality.

5.1 Maximising precision for a given recall

To illustrate the algorithm in practice for the problem *Variant MaxP* from Section 2, we use the example in Figure 4, which describes the properties of 6 sources with different precisions and recalls, to see how the algorithm selects the feedback based on the defined criteria and on evolving precision and recall estimates.

In this case, the stopping condition for the source selection is when the accumulated estimated recall for the selected sources S' is greater or equal to the required constraint, therefore the portion of method *thresholdMet* for this problem variant is defined as in Figure 3 (lines 2 and 3).

We start considering the candidate sources described in Figure 4 to represent a simple execution of the algorithm collecting *10* feedback instances at a time and defining a hard recall constraint of *0.8*. In Figure 4, there are 6 sources (S_1 to S_6), each of which return 1,000 items, where the precisions and recalls of the sources cover a wide range of values.

Initially, we collect 10 feedback instances for bootstrapping from all sources randomly (lines 5-6 in Figure 2), obtaining the precision and recall estimates indicated in the row labeled "Iteration 0" in Figure 4. In this figure dark gray indicates a source included in the selected subset S', and light gray indicates an unselected but still considered source for feedback collection in current iteration.

If we sort the sources by their estimated precision (line 14), we will include S_1 to S_3 in subset S', as combined they have an estimated precision of 0.666 (20 TPs over 30 total data items collected for these sources) and a recall of 0.869 (20 TPs from selected sources over 23 produced by all sources). The cut-off $lowP$ to separate good candidate sources from the others will be 0.2 as this is the lowest estimated precision for all sources in S'.

Based on the previous selection, in the next iteration of the outer loop (lines 3 and 27) we will collect feedback (line 8) only on those sources above and overlapping the cut-off

Input: set of sources S
Input: set of unlabelled data items U
Input: a variant type var
Input: a threshold value thr
Input: a default step size $step$
Input: a budget size $budget$
Input: a confidence level $conf$
Input: a margin of error err
Output: set of selected sources S'

```
1: L ← {}, iL ← {}
2: iStep ← getSampleSize(S, conf, err)
3: repeat
4:     S' ← {}, lowP ← 1
5:     if size of L = 0 then
6:         iL ← L ← getFeedback(S, U, iStep, RND, conf, 0)
7:     else
8:         L ← getFeedback(S, U, step, OLBP, conf, lowP)
9:     end if
10:    estimateQuality(S, L, iL, conf, err)
11:    if (size of U - size of L) < step then
12:        step ← (size of U - size of L)
13:    end if
14:    S ← sortByEstPrecisionInDescOrder(S)
15:    while s ← getNextAvailableSource(S) do
16:        S' ← S' ∪ s
17:        if thresholdMet(S', var, thr) then
18:            if var = MaxR then
19:                S' ← S' − s
20:            end if
21:            break
22:        end if
23:        if estimatedPrecision(s) < lowP then
24:            lowP ← estimatedPrecision(s)
25:        end if
26:    end while
27: until size of L < budget
28: return S'
```

Figure 2: SelectSources algorithm

Input: subset of sources S'
Input: a variant type var
Input: a threshold value thr
Output: boolean value stating if threshold was met or not

```
1: switch var do
2:     case MaxP
3:         return (estimatedRecall(S') >= thr)
4:     end case
5:     case MaxR
6:         return (estimatedPrecision(S') < thr)
7:     end case
8:     case MaxPN
9:         return (estimatedCardinality(S') >= thr)
10:    end case
11: end switch
```

Figure 3: thresholdMet function

Sources	S_1	S_2	S_3	S_4	S_5	S_6
Data Items Produced	1000	1000	1000	1000	1000	1000
Actual Precision	0.8	0.6	0.4	0.2	0.1	0.0
Actual Recall	0.38	0.29	0.19	0.10	0.05	0.00
Iteration 0 (Precision / Recall / Error)	0.7 / 0.3 / 0.28	0.8 / 0.35 / 0.24	0.5 / 0.22 / 0.30	0.1 / 0.04 / 0.18	0.2 / 0.09 / 0.24	0.0 / 0.0 / 0.1
Iteration 1 (Precision / Recall / Error)	0.75 / 0.35 / 0.18	0.7 / 0.33 / 0.19	0.45 / 0.21 / 0.21	0.15 / 0.07 / 0.18	0.1 / 0.05 / 0.13	0.0 / 0.0 / 0.1
Iteration 2 (Precision / Recall / Error)	0.77 / 0.37 / 0.14	0.67 / 0.32 / 0.16	0.43 / 0.21 / 0.17	0.13 / 0.06 / 0.11	0.1 / 0.03 / 0.13	0.0 / 0.0 / 0.1
Iteration 3 (Precision / Recall / Error)	0.77 / 0.41 / 0.14	0.63 / 0.32 / 0.14	0.4 / 0.2 / 0.14	0.13 / 0.05 / 0.11	0.1 / 0.03 / 0.13	0.0 / 0.0 / 0.1

Figure 4: Example of source selection scenario.

(S_1 to S_5), as S_6 is below the cut-off and therefore is not considered for further feedback collection to avoid wasting effort on it. The new round of collected feedback yields the precision and recall estimates indicated in the row labeled "Iteration 1" in Figure 4.

If we again sort the sources by their revised precision estimates (line 14), we will select again S_1 to S_3 to form the subset S', as combined they have an estimated precision of 0.633 (38 TPs over 60 total items collected for these sources) and a recall of 0.883 (38 TPs out of 43 are produced by the selected sources). The cut-off $lowP$ will now be 0.24.

If we collect feedback one more time, on those sources above and overlapping the cut-off (S_1 to S_4), we have the new estimates for precision and recall indicated in the row labeled "Iteration 2" in Figure 4. At this point we will raise the cut-off to 0.26 leaving the source S_4 out of contention for more feedback collection, and we will focus the feedback on selected sources to refine their estimates.

This process continues until the allocated budget has been exhausted and the final selection of sources (that meet the constraint) is returned to the user.

5.2 Maximising recall for a given precision

In the case of the problem labelled as *Variant MaxR* in Section 2, there is a hard constraint on precision, and the goal is to select a subset of sources that meet this constraint while maximising the associated recall. The steps are as in the previous variant except the stopping condition used to determine when the subset of selected sources has met the required threshold.

The break condition for the source selection in *Variant MaxR* is when the accumulated estimated precision for the subset of selected sources in S', combined with next candidate sources r, is below the required constraint. To support this, the function *thresholdMet* includes this problem variant as shown in Figure 3 (lines 4 and 5); in this case we are adding elements to the collection of candidates just before the estimated precision falls below the required threshold.

5.3 Maximising precision for a given result size

In the case of the problem labelled as *Variant MaxPN* in Section 2, there is a hard constraint on the number of data items required from the selected sources, and the goal is to select a subset of sources that meet this constraint while maximising the associated precision. The steps are as in the previous variants, the only difference being in the stopping condition used to determine when the subset of selected sources has reached the required threshold.

The stopping condition for *Variant MaxPN*, is when the accumulated estimated number of data items returned by the subset of selected sources S' is greater than or equal to the required constraint (lines 6 and 7 in Figure 3).

6. EVALUATION: MAPPING SELECTION

In this section, we evaluate the the OLBP approach from Section 5, for selecting sources that are the result of running schema mappings; as such, the sources are *virtual*. In the experiment, we consider *global-as-view* mappings, which relate one element in the integration schema to a query over the source schemas. We also adopt the relational model for expressing integration and source schemas.

6.1 Experimental setup

For for mapping selection, we used IBM Infosphere Architect[1] to create mappings from the Mondial database[2] to a target schema relating to European cities. The resulting test set contained 100 mappings, producing in total 100,000 data items (tuples). The database and ground truth were created in such a way that the size and quality of the mappings varies. The objective was to make it challenging to find collections of mappings with sufficient precision and recall to fulfil some of the optimisation targets. As a result, for example, the test set does not contain mappings with both high precision and recall, as a single mapping with precision and recall of *1.0* would trivially satisfy all requirements. Each of the 3 mapping selection experiments was repeated 20 times to the impact of the random selection of data items for feedback, and the average value of each property (precision, recall) was used to produce the results. A 95% confidence level with an initial error margin of 0.5 was used for these experiments. OLBP is compared against a random selection of data items because we are not aware of another solution to the problem addressed here, as mentioned in Section 3. In general, the feedback required is relatively low for the mapping selection experiments (1-5% of the available data items), as we are dealing with a large number of tuples (100,000) and require only an small fraction to obtain reliable estimates.

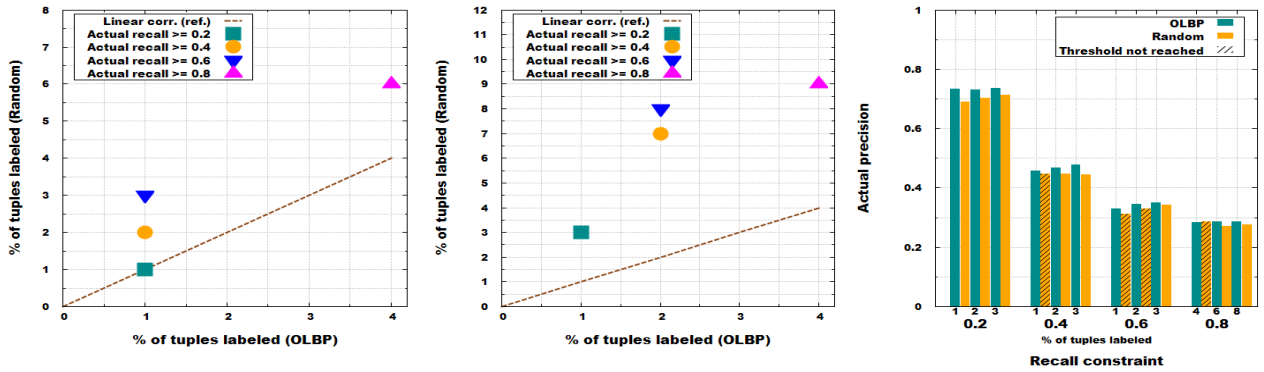
We evaluated the proposed strategy on *Variants MaxP, MaxR and MaxPN* defined in Section 2, by comparing the random selection of data items for feedback collection against the items identified using our OLBP approach, to investigate under what circumstances and to what extent the technique provides an improved return on investment. The random selection of data items is carried out in the context of the *SelectSources* algorithm from Figure 2, except that *getFeedback* selects items at random from the union of the data items in the sources in S.

6.2 Results

Similar experiments have been carried out for each of the three variants. In each case, the control variable is the constraint threshold, and we provide three graphs that:

1. Compare the percentage of data items that must be labelled with OLBP and using random feedback selection to meet the constraint threshold. The constraint threshold is considered to be met when the estimated value for the constrained variable equals or exceeds the threshold for the first time, given that we have labelled a number of tuples equal to or greater than the sample

[1]http://www-03.ibm.com/software/products/en/ibminfodataarch
[2]http://www.dbis.informatik.uni-goettingen.de/Mondial



(a) (b) (c)

Figure 5: Results summary for recall constraint on mapping selection experiments, presenting (a) the feedback required to meet the constraint threshold, (b) the amount of feedback required to get the maximum actual precision after the constraint threshold is reached, and (c) the actual precision for different feedback amounts.

size computed using Equation 7 in line 8 of Figure 2. If the threshold is reached before acquiring this number of labelled tuples the objective has not been met, as the sample collected is not sufficiently representative.

2. Compare the percentage of data items that must be labelled with OLBP and using random feedback selection to obtain the highest value, for the variable to be maximised, once the constraint threshold was reached. The maximum value is considered to be reached when the estimated value for the variable to be maximised reaches a stable point, with variations between feedback collections equal to or less than 0.001 in the scale 0 to 1, given that we have labelled a number of tuples equal to or greater than the sample size computed using Equation 7 in line 8 of Figure 2. If the maximum value is reached before acquiring this number of labelled tuples the objective has not been met, as the sample collected is not sufficiently representative.

3. Compare the values of the variable to be maximised by using different amounts of feedback.

6.2.1 *Maximise precision for a given recall*

For this problem variant the control variable is the recall constraint, which takes on the values 0.2, 0.4, 0.6 and 0.8. The constraints of 0 and 1 are trivially satisfied, the first by choosing no mappings, and the second by selecting all available mappings. Results are given in Figure 5.

Figure 5 (a) compares the amount of feedback required to meet the threshold for each of the recall constraints. The dashed line represents a 1 to 1 correspondence between collection of feedback on randomly selected mapping results and results identified for feedback using OLBP. A result above the line indicates that OLBP has met the threshold with less feedback. OLBP meets the threshold earlier than random, except in the case of recall ≥ 0.2, where both approaches reach the threshold with the same amount of feedback. For the higher thresholds, fewer mappings are selected, and thus OLBP is able to target feedback collection on a smaller number of candidate mappings.

Figure 5 (b) compares the amount of feedback required to get the maximum actual precision after the constraint threshold is reached. The results are favourable for OLBP, and the difference increases as the recall threshold is raised.

Figure 5 (c) compares the precision obtained for the selected subset of mappings, for increasing amounts of feedback and for the 4 constraint levels. In this case, the difference between our strategy and random results is marginal. However, these results need to be seen within the context of Figure 5(a). Where small percentages of the data items have been labelled, random often misses the recall constraint, so OLBP not only meets the constraint with less feedback but also provides comparable (generally better) overall precision. In Figure 5(c), results that have been obtained without meeting the hard constraint are marked with diagonal lines.

6.2.2 *Maximise recall for a given precision*

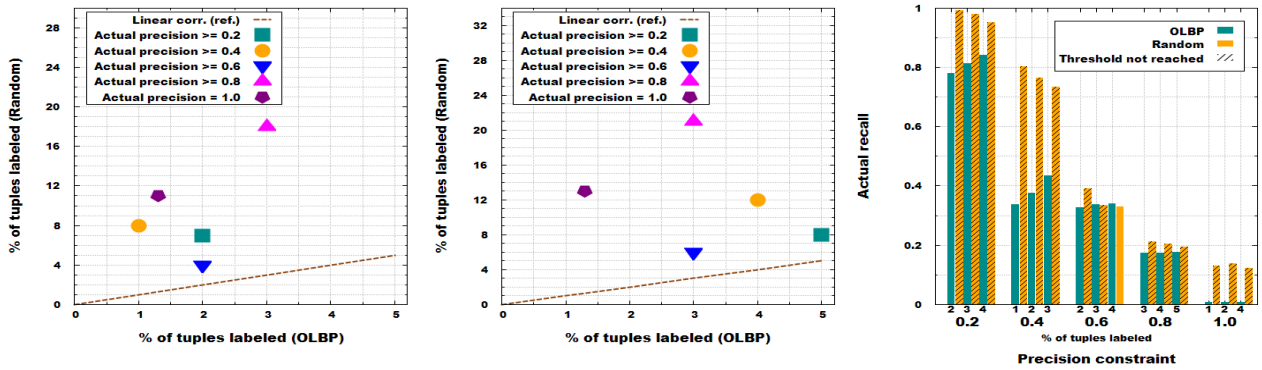
For this variant, the experiments are similar to those for *Variant MaxP*, but now the precision constraint is the control variable, taking the values: 0.2, 0.4, 0.6, 0.8 and 1.0 (or highest precision possible). The constraint of 0 is trivially satisfied by choosing no mappings. Results are in Figure 6.

In Figure 6 we have 3 plots comparing: (a) the amount of feedback required to meet the precision constraint; (b) the amount of feedback needed to reach a maximum stable recall; and (c) the recall for different amounts of feedback collected with each of the different precision constraints.

In Figure 6 (a) and (b) it is evident that OLBP outperforms random by requiring much less feedback to meet the required threshold and to obtain a maximum stable recall.

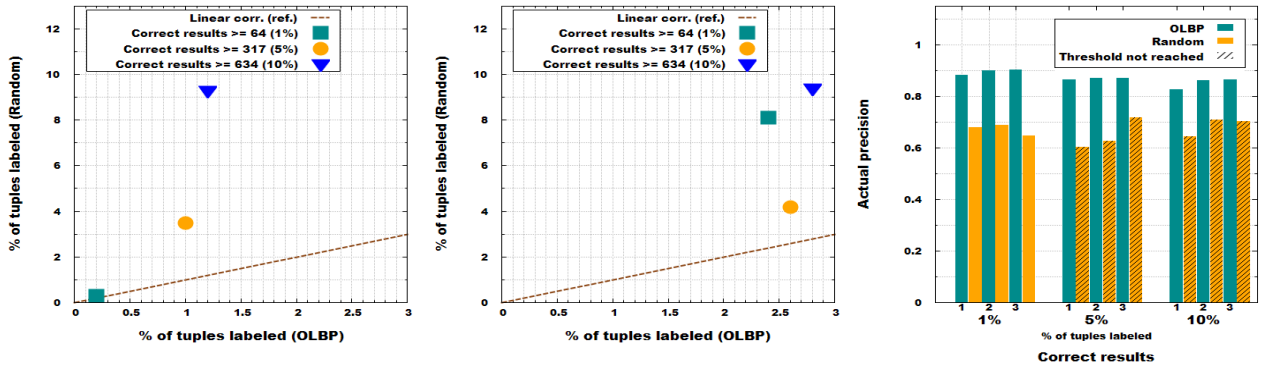
In Figure 6 (c) the apparent recall advantage of the random approach is a consequence of the low precision obtained at these feedback amounts. For example, taking the case of the high precision constraint (≥ 0.8), the random approach requires 6 times more feedback to meet the precision threshold, and before reaching this threshold the precision obtained is well below it. This has the consequence of a higher recall than our approach, but this is only because in the random case we have not yet reached the hard constraint. Thus the untargeted random feedback selection is leading to inappropriate mapping selection decisions being made until a lot of feedback has been collected, and thus the user's requirements are not being met.

6.2.3 *Maximise precision for a given result size*



(a) (b) (c)

Figure 6: Results summary for precision constraint on mapping selection experiments, presenting (a) the feedback required to meet the constraint threshold, (b) the amount of feedback required to get the maximum actual recall after the constraint threshold is reached, and (c) the actual recall for different feedback amounts.



(a) (b) (c)

Figure 7: Results summary for number of data items constraint on mapping selection experiments, presenting (a) feedback required to meet the constraint threshold, (b) amount of feedback required to get the maximum actual precision after constraint threshold is reached, and (c) actual precision for various feedback amounts.

In this case the experiments are similar to the other variants, except that the control variable is number of data items. The target was chosen to be small (1, 5, 10 % of available items), to reflect the case where a user expects to manually inspect a small collection. Results are in Figure 7.

In Figure 7 (a) and (b), as with *Variants MaxP and MaxR*, we again have better results for OLBP than for random, with random often needing many times more feedback to meet the constraint or to reach a maximum precision.

In Figure 7 (c) the precision obtained by OLBP is significantly higher than for random. No initial sample is collected by OLBP as we do not need recall estimates, so OLBP can focus from the beginning on selecting mappings based on their estimated precisions. In so doing, OLBP targets those mappings that produce more correct results (highest precision), this allows OLBP to meet all the constraints tested. Random, in contrast, keeps collecting feedback from every mapping regardless of its quality and, given that in this data set we have a larger proportion of false positives in general, random collects an smaller proportion of true positive labelled tuples which prevent it from reaching those constraints that require more correct results (5 and 10%), indicated in the figure by diagonal lines, even when collecting a similar amount of feedback than OLBP. In this case, unlike the previous graphs, the precision does not decrease as we increment the percentage of required correct results. This is because in this scenario, there is no trade-off between precision and recall, therefore the initial maximum precision is maintained while the approach tries to collect enough data items to fulfill the required threshold.

7. EVALUATION: WEB DATA SOURCE SELECTION

7.1 Experimental setup

In this section we evaluate the OLBP approach described in Section 5 on sources obtained from web data extraction. Specifically, we used real-world data sets produced using OXPath wrappers induced by DIADEM [16] from web sources in the UK Real State domain[3]. The resulting data set has 79 sources, producing in total 7,612 different tuples; the quality and number of records produced by the sources varies. Extracted tuples have properties from the real state

[3]http://diadem.cs.ox.ac.uk/evaluation/14/02/reports/refull

domain such as address, price and number of bedrooms. In contrast, comparing against previous experiments, the feedback required for source selection experiments is high (5-27% of the available data items), as in this case we are dealing with a smaller number of tuples (7,612) and therefore we require a larger fraction to obtain reliable estimates.

We evaluate OLBP on the problem variants defined in Section 2, using experiments analogous to those for mapping selection in Section 6. Each of the 3 source selection experiments was repeated 20 times to reduce variations inherent to the random selection of data items, and the average value of each property (precision, recall) was used for the results. Assuming 95% confidence level and error margin of 0.5.

7.2 Results

7.2.1 Maximise precision for a given recall

For this problem variant, as in mapping selection, 4 experiments were executed to represent scenarios requiring a minimum recall of 0.2, 0.4, 0.6, 0.8 while maximising precision. In these experiments, the constraints of 0 and 1 are trivially satisfied by choosing no sources, or selecting all the available sources, respectively. Results are in Figure 8.

Figure 8 (a) shows the amount of feedback required to reach the threshold for each constraint, with a dashed line representing a 1 to 1 relation between the two approaches. In this scenario the results are closer between the random selection of data items and the OLBP approach, as we need to sample a bigger fraction of smaller sources to yield a given confidence than that required for a larger source. However, OLBP still outperforms random for all tested scenarios.

In Figure 8 (b) the amount of feedback required to reach a maximum precision once the threshold has been met presents the same favourable results for OLBP as it requires less feedback instances to satisfy the constraint and also to meet the optimisation target by maximising the precision. The results are better for OLBP for low recall constraints, as this requires the search to identify fewer high-quality sources, which benefits from the targeted feedback.

Figure 8 (c) shows the actual precision obtained by both approaches for different amounts of feedback. When the value presented corresponds to a feedback amount where the threshold has not been reached the bar is presented with crossing diagonals. Again, OLBP performs better than random in maximising the precision for those feedback amounts where the threshold has been reached, and in many cases the threshold was missed when collecting feedback randomly.

7.2.2 Maximise recall for a given precision

For this problem variant, five thresholds were considered to represent scenarios requiring a minimum precision of: 0.2, 0.4, 0.6, 0.8 and 1.0 (or highest possible precision), while maximising recall. The constraint of 0 is trivially satisfied by choosing no mappings. Results are given in Figure 9

In Figure 9 (a), OLBP reaches the threshold with far less feedback collected than in the random approach; this is more evident for higher precision constraints as they require better estimates to meet increasingly challenging thresholds.

Figure 9 (b) again shows a clear reduction in the number of feedback instances required to achieve maximum stable levels of recall once the threshold constraint is reached; the higher the precision constraint the larger the difference between OLBP and random selection of data items.

In Figure 9 (c) the bars with crossing diagonals represent recall values for random item selection for feedback amounts where the precision constraint was not reached therefore, while both approaches seem to have similar levels of recall in most cases, OLBP achieves these values after fulfilling the required constraint whilst random does not.

7.2.3 Maximise precision for a given result size

To conclude, we investigate the third problem variant, for scenarios that aim to obtain 1, 5 and 10 percent of the available data items while maximising the precision, representing a user search that seeks to obtain a small subset of the available records. Results are shown in Figure 10.

In Figure 10 (a) the feedback required by the random selection of tuples to reach the constraint threshold is several times higher than that required by OLBP, particularly for smaller subsets of data items as this requires a finer source selection while deeply relying on their precision estimates.

Figure 10 (b) shows how OLBP outperforms the random approach by reaching a maximum actual precision once the constraint has been fulfilled with smaller feedback amounts.

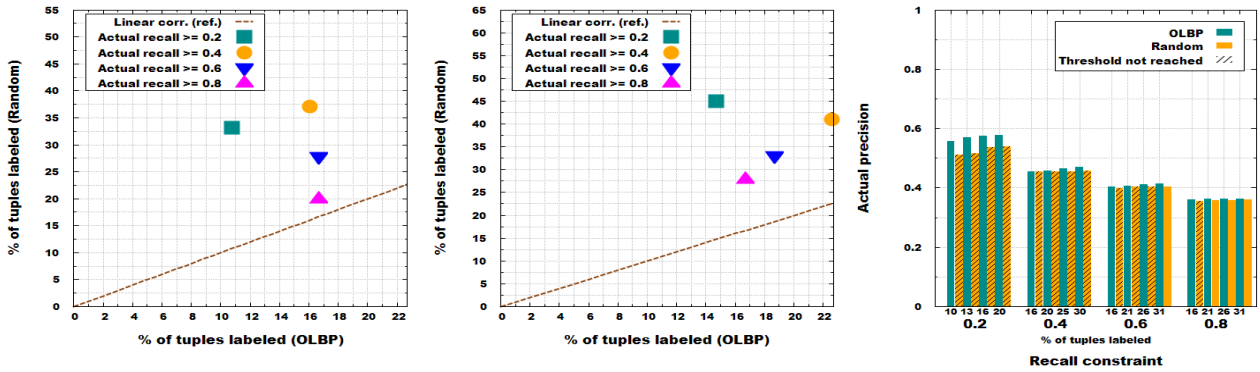
Finally, in Figure 10 (c) the actual precision achieved by both approaches is compared. In this plot the difference between the maximised precision for OLBP and the random selection of data items is especially noteworthy, as only 2 values obtained for the random strategy satisfy the constraint, and in both cases the maximised precision is significantly lower than the corresponding value for OLBP. In this case, unlike previous graphs, the precisions do not decrease when incrementing the threshold, as there is no trade-off between precision and recall, and the maximum precision is maintained while data items are obtained to meet the threshold.

In summary, after analysing the results, we found that OLBP outperforms the random selection in all the scenarios tested, resulting in a significant reduction of the amount of feedback required, particularly where user's requirements can be met using a small portion of the available sources.

8. CONCLUSIONS

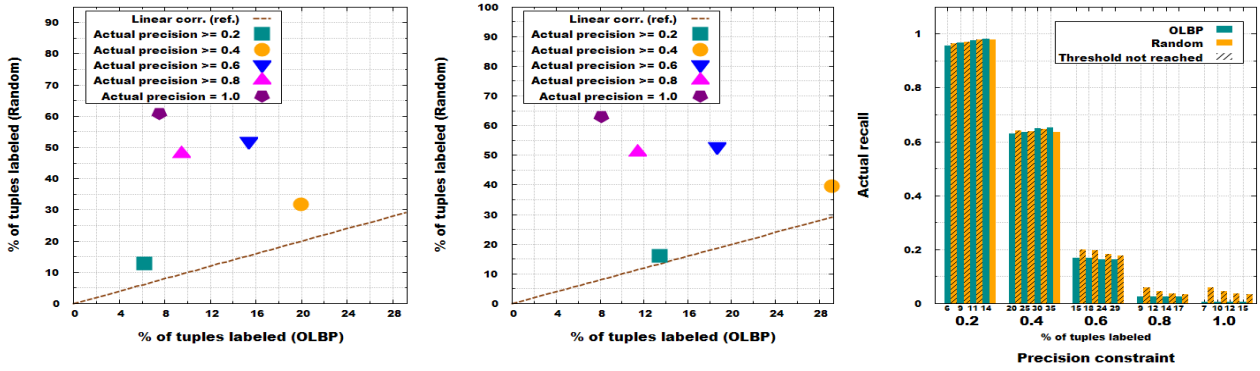
The proliferation of data sources means that it is increasingly important for data consumers to be able to characterise and select subsets of the available sources in a cost-effective manner. In pay-as-you-go source selection, users or crowd workers provide feedback on data from the sources, which in turn informs the selection process. This paper has presented an approach to targeting data items for feedback, with a view to enabling cost-effective source selection; specifically, the contributions include:

- A strategy for efficient feedback collection that takes into account both the estimated precision of sources and the confidence in those estimates.
- The application of the strategy on three variants of the source selection problem that involve trading-off precision and recall, specifically maximising precision for a given recall, maximising recall for a given precision, and maximising precision for a given cardinality.
- The evaluation of the strategy on both materialized and virtual data source selection problems, with results that show the approach can substantially reduce the amount of feedback required. While the benefits are not consistently as large in all cases, they are significant in almost all scenarios.



(a) (b) (c)

Figure 8: Results summary for recall constraint on source selection experiments, presenting (a) the feedback required to meet the constraint threshold, (b) the amount of feedback required to get the maximum actual precision after the constraint threshold is reached, and (c) the actual precision for different feedback amounts.



(a) (b) (c)

Figure 9: Results summary for precision constraint on source selection experiments, presenting (a) the feedback required to meet the constraint threshold, (b) the amount of feedback required to get the maximum actual recall after the constraint threshold is reached, and (c) the actual recall for different feedback amounts.

Acknowledgment

Julio César Cortés Ríos is supported by a grant from the Mexican National Council for Science and Technology (CONACyT). Data integration research at Manchester is supported by the UK Engineering and Physical Sciences Research Council, through the VADA Programme Grant.

9. REFERENCES

[1] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23, 2011.

[2] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, and C. Hedeler. Feedback-based annotation, selection and refinement of schema mappings for dataspaces. In *EDBT*, pages 573–584, 2010.

[3] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, and C. Hedeler. Incrementally improving dataspaces based on user feedback. *Inf. Syst.*, 38(5):656–687, 2013.

[4] P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *CACM*, 51(9):72–79, 2008.

[5] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa. Schema mapping verification: the spicy way. In *EDBT*, pages 85–96, 2008.

[6] A. Bozzon, M. Brambilla, and S. Ceri. Answering search queries with crowdsearcher. *WWW*, pages 1009–1018, 2012.

[7] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *CACM*, 54(2):72–79, 2011.

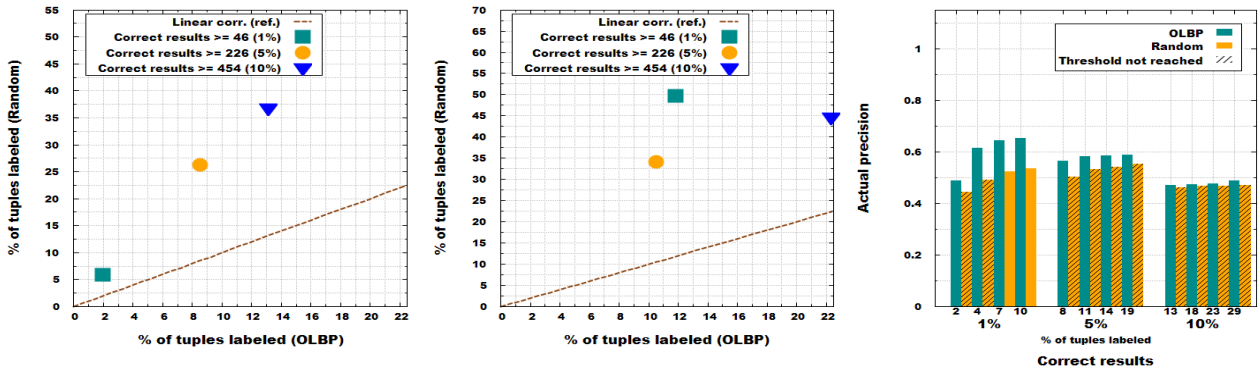
[8] H. Cao, Y. Qi, K. S. Candan, and M. L. Sapino. Feedback-driven result ranking and query refinement for exploring semi-structured data collections. In *EDBT*, pages 3–14, 2010.

[9] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2 edition, 1988.

[10] V. Crescenzi, P. Merialdo, and D. Qiu. Wrapper generation supervised by a noisy crowd. In *VLDB*, pages 8–13, 2013.

[11] V. Crescenzi, P. Merialdo, and D. Qiu. Crowdsourcing large scale wrapper inference. *Distributed and Parallel Databases*, 33(1):95–122, 2015.

[12] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. *PVLDB*, 6(2):37–48, 2012.



(a) (b) (c)

Figure 10: Results summary for number of data items constraint on source selection experiments, presenting (a) feedback required to meet the constraint threshold, (b) amount of feedback required to get the maximum actual precision after constraint threshold is reached, and (c) actual precision for various feedback amounts.

[13] H. Elmeleegy, A. K. Elmagarmid, and J. Lee. Leveraging query logs for schema mapping generation in u-map. In *SIGMOD*, pages 121–132, 2011.

[14] D. H. Foley. Considerations of sample and feature size. *IEEE Trans. on Inf. Theory*, 18(5):618–626, 1972.

[15] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *ACM SIGMOD*, pages 61–72, 2011.

[16] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang. DIADEM: Thousands of websites to a single database. *PVLDB*, 7(14):1845–1856, 2014.

[17] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD Conference*, pages 601–612, 2014.

[18] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *ACM SIGMOD*, pages 805–810, 2005.

[19] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[20] S. B. Hulley, S. R. Cummings, W. S. Browner, D. G. Grady, and T. B. Newman. *Designing Clinical Research*. Lippincott Williams and Wilkins, 3 edition, 2007.

[21] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. Minimizing efforts in validating crowd answers. In *SIGMOD*, pages 999–1014, 2015.

[22] R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *PVLDB*, 5(11):1638–1649, 2012.

[23] R. Isele and C. Bizer. Active learning of expressive linkage rules using genetic programming. *JWS*, 2–15, 2013.

[24] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[25] C. Lofi, K. E. Maarry, and W.-T. Balke. Skyline queries in crowd-enabled databases. In *Proc. 16th EDBT*, pages 465–476, 2013.

[26] J. Madhavan, S. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Halevy. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[27] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[28] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *PVLDB*, 8(2):125–136, 2014.

[29] F. Naumann, J. C. Freytag, and M. Spiliopoulou. Quality driven source selection using data envelope analysis. In *IQ*, pages 137–152, 1998.

[30] T. Rekatsinas, X. L. Dong, L. Getoor, and D. Srivastava. Finding quality in quantity: The challenge of discovering valuable sources for integration. In *CIDR, USA, 2015*, 2015.

[31] T. Rekatsinas, X. L. Dong, and D. Srivastava. Characterizing and selecting fresh data sources. In *SIGMOD*, pages 919–930, 2014.

[32] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[33] P. P. Talukdar, Z. G. Ives, and F. C. N. Pereira. Automatically incorporating new sources in keyword search-based data integration. In *ACM SIGMOD, USA, 2010*, pages 387–398, 2010.

[34] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. C. N. Pereira, and S. Guha. Learning to create data-integrating queries. *PVLDB*, 1(1):785–796, 2008.

[35] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.

[36] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *ACM SIGMOD*, pages 485–496, 2001.

[37] Z. Yan, N. Zheng, Z. G. Ives, P. P. Talukdar, and C. Yu. Active learning in keyword search-based data integration. *VLDB J.*, 24(5):611–631, 2015.

[38] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu. Cleaning uncertain data with a noisy crowd. *ICDE*, 6–17, 2015.