Intensional Data on the Web

ANTOINE AMARILLI
Institut Mines Télécom; Télécom ParisTech; CNRS LTCI, Paris, France
and
SILVIU MANIU
Huawei Noah's Ark Lab, Hong Kong
and
PIERRE SENELLART
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, Paris, France &
National University of Singapore; CNRS IPAL, Singapore

We call data *intensional* when it is not directly available, but must be accessed through a costly interface. Intensional data naturally arises in a number of Web data management scenarios, such as Web crawling or ontology-based data access. Such scenarios require us to model an uncertain view of the world, for which, given a query, we must answer the question "What is the best thing to do next?" Once data has been retrieved, the knowledge of the world is revised, and the whole process is repeated, until enough knowledge about the world has been obtained for the particular application considered. In this article, we give an overview of the steps underlying all intensional data management scenarios, and illustrate them on three concrete applications: focused crawling, online influence maximization in social networks, and mining crowdsourced data.

1. INTRODUCTION

Intensional data management. Many data-centric applications on the Web involve data that is not directly available, but can only be obtained after performing some data accesses, which carry a specific cost. In traditional database querying [Garcia-Molina et al. 2009], the access type may be disk I/O, and the I/O cost will depend on which indexes are available. In Web crawling [Gouriten et al. 2014], accesses are HTTP requests and cost involves bandwidth usage, network latency, and quota use for rate-limited interfaces. In social influence maximization scenarios [Kempe et al. 2003; Lei et al. 2015], accesses are influence campaigns starting at a few seed users, and the cost is that of starting the campaign at the given seeds. In crowdsourcing platforms [Parameswaran et al. 2011; Amsterdamer et al. 2013b], accessing data involves recruiting a worker to provide or extract the data, and the cost includes monetary compensation for workers and latency in obtaining the data. In ontology-based data access [Calvanese et al. 2011], accesses mean applying an ontology reasoning rule, and the cost is the computational cost of such an evaluation.

All these examples can be thought as instances of the abstract problem of *intensional data*

management. This terminology contrasts with *extensional data management* where data is freely accessible (entirely stored in-memory, or locally stored on disk in situations when disk accesses are negligible). *Intensional data* is also the term used to refer to the *schema* of a database, as opposed to extensional facts, especially in the setting of deductive databases [Reiter 1977]; in the same way, in intensional data management, we study how to perform query optimization and other data management tasks when only the schema (and access methods) to some of the data is directly available, and not the facts themselves.

Intensional data management applications share a number of distinguishing features. At every point in time, one has an *uncertain view of the world*, that includes all the data that has already been accessed, together with the schema, access methods, and some *priors* about the data not yet accessed. Given a user's query, the central question in intensional data management is: "What is the best thing to do next?" in order to answer the query, meaning, what is the best access that should be performed at this point, given its cost, its potential gain, and the current uncertain knowledge of the world. Once an access is chosen and performed, some data is retrieved, and the uncertain view of the world must be revised in light of the new knowledge obtained. The process is repeated until the user's query receives a satisfactory answer or some other termination condition is met.

Example use cases. To illustrate, let us give some concrete examples of complex use cases involving intensional data management.

Consider a first application of *mobility in smart cities*, i.e., a system integrating information about transportation options, travel habits, traffic, etc., in and around a city. Various public resources can be used to collect and enrich data related to this application: the Web, deep Web sources, social networking sites, the Semantic Web, annotators and wrapper induction systems, crowdsourcing platforms, etc. Moreover, in such a setting, domain-specific resources, not necessarily public, contribute to the available data: street cameras, red light sensors, air pollution monitoring systems, etc. Users of the system, namely, transport engineers, ordinary citizens, etc., may have many kinds of needs. They can be simple queries expressed in a classical query language (e.g., "How many cars went through this road during that day?" or "What is the optimal way to go from this place to that place at a given time of day?"), certain patterns to mine from the data ("Find an association rule of the form $X \Rightarrow Y$ that holds among people commuting to this district"), or higher-level business intelligence queries ("Find anything interesting about the use of the local bike rental system in the past week").

To be specific, let us imagine what options are available for the query "How many cars went through this road during that day?". One could:

If applicable, use data from electronic toll gates;

Use a computer vision program to analyze the street camera feeds and automatically extract each passage of a vehicle;

Ask crowd workers to perform the same analysis;

Do the same, but only a fraction of the day, and extrapolate the results;

Use traffic data from Bing Maps API, correlated with external data about road characteristics;

Analyze the mentions of this road segment on social media, to determine both its usage rate, and a subjective idea of whether it is overloaded;

Send a team of expert traffic specialists to survey the road;

etc.

Each of these (and each combination of these) has a cost (in terms of manpower, budget, processing time, bandwidth) and a precision. The objective is to obtain an optimal solution given a precision threshold. This example is fairly simple, but imagine that determining the traffic on a road may be just one component of a more complex information need, such as redesigning an entire district.

A second example is *socially-driven Web archives* [Risse et al. 2012]: the goal is to build semantically annotated Web archives on specific topics or events (investment for growth in Europe, the war in Eastern Ukraine, etc.), guiding the process with clues from the social Web as to which documents are relevant. These archives can then be semantically queried by journalists today or historians tomorrow, e.g., to retrieve all resources that mention a given person. The construction of these archives relies on Web crawling, deep Web harvesting, accessing social networking sites such as Twitter or YouTube via their APIs, using tools for information extraction, named entity recognition, opinion mining, etc. Again, these various intensional sources come with very different costs, and an optimal plan for a user's query involves choosing an optimal way to collect, annotate, and query the different relevant sources.

We present next a general and abstract approach for intensional data management in Section 2. We then illustrate in Section 3 three research works which instantiate this general approach, on the subjects of focused crawling, influence maximization, and crowdsourcing.

2. GENERAL APPROACH

We now present the general steps required for intensional data management, and give an overview of the current research for each of them.

A summary of the general approach is presented in Figure 1, with references to corresponding sections. First, once the user query has been formulated, we must design a model to represent the current knowledge of the world, and more generally our uncertainty about its state; this requires inferring (e.g., via wrapping) the access methods and cost of each available source, and building priors on the information that they contain. Second, we must deal with the core optimization problem: decide which access allows us to make the most progress on our query, as a function of our current knowledge and the possible accesses on the various sources; this allows us to build a plan to evaluate the query. The third step is, after one intensional access to the sources, to update our knowledge of the world with the results of the chosen access, ultimately revising the query plan. Once we have repeated this process sufficiently many times, and our knowledge of the world allows us to determine the final query answer, we conclude by returning the answer to the user.

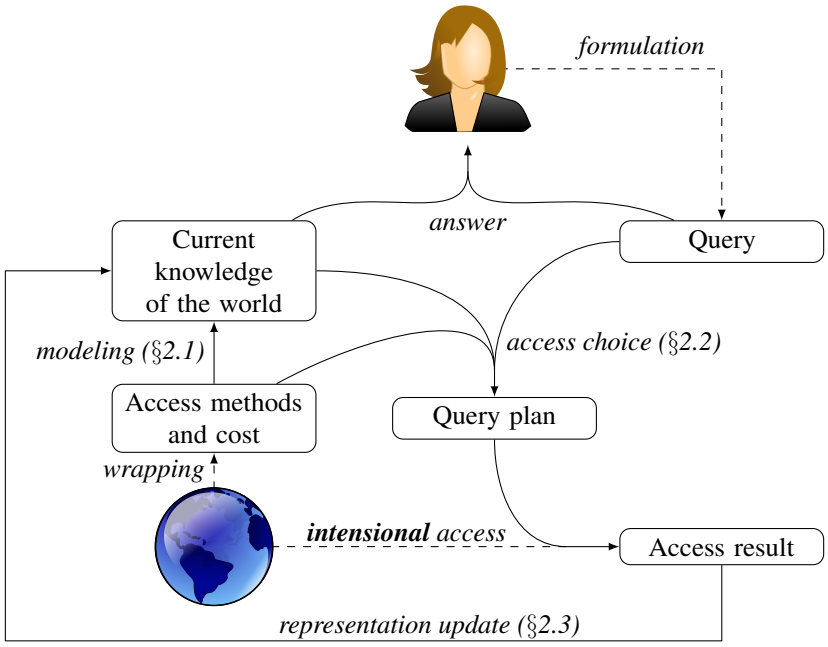

Fig. 1. Approach to Intensional Data Management

2.1 Modeling the World

At any point in time, the system must have a representation of its current state. In general, the state is an *uncertain* description of the world, constructed from the past observations and prior knowledge. Of course, this representation must be *concise*, as we cannot write out explicitly all the (possibly infinite) cases; it must also be *operational*, in that we must have a way to perform efficiently the next steps of the approach with it.

Existing methodologies for such representations can be distinguished first and foremost by the *kind of data* which they attempt to represent. For simple, unstructured data, such as the answers to a yes/no question in a crowdsourcing application, or estimators in a Web crawling scenario, the representation may just be a choice of parameters fitted to the distribution of answers. More interestingly, for structured data, we can turn to probabilistic representations for relational [Suciu et al. 2011] or XML [Kimelfeld and Senellart 2013] data management systems, and probabilistic graphs [Valiant 1979]. These frameworks annotate instances with information about the uncertainty of the data items.

The initial knowledge of the world consists of some initial data, some *priors* on the unknown data, and priors on what data may be returned by the various access methods.

2.2 Choosing Accesses

Once we have represented our uncertain knowledge about the world, the main problem is to choose which access to perform next.

Of course, this choice depends on the kind of accesses that are possible. For instance, in crowdsourcing contexts, the simplest setting is when the possible questions to ask are concerned with separate and independent items [Yang et al. 2013]: in this case, we only need to choose the item on which we want to obtain more information. By contrast, in crowd situations where the answers depend on each other, the problems becomes more complex [Amarilli et al. 2014].

More generally, the possible accesses can be different "views" on the same information. In the setting of Web sources, the language of *binding patterns* [Rajaraman et al. 1995] is used to represent the various possibilities (with different restrictions) to access information about the same relations. Even more generally, in the context of data integration [Lenzerini 2002], or of query answering using views [Halevy 2001], there can be very general dependencies between the views (that are accessed) and the underlying data (on which the query is posed): e.g., equality-generating or tuple-generating dependencies. In such general situations, it can become undecidable to determine, under arbitrary dependencies, whether a given access is relevant to the query [Benedikt et al. 2011; Benedikt et al. 2012].

Having fixed our representation of the accesses, we must now decide which one to perform. We call this the *access choice problem*. We focus here on an *interactive* approach, where each access is chosen depending on the result of the previous accesses. Examples of the access choice problem with interactive solutions include *focused crawling*, to locate interesting information by choosing which pages to query [Menczer et al. 2001]; *deep Web crawling*, where information is located behind Web forms [Nayak et al. 2012]; as well as many crowdsourcing scenarios [Parameswaran et al. 2012].

Another way to see the access choice problem is from the angle of machine learning, namely *reinforcement learning* and *active learning*. Reinforcement learning [Sutton and Barto 1998; Audibert et al. 2009; Puteman 2005] is the study of how to maximize rewards in the following setting: whenever we find ourselves in a state, we must choose an action to perform, which yields a reward and changes the state. This implies an inherent tradeoff between exploration (trying out new actions leading to new states and to potentially high rewards) and exploitation (performing actions already known to yield high rewards). We can model the access choice problem as a reinforcement learning problem on a huge structured state space corresponding to the possible states of our intermediate knowledge, as proposed in [Benedikt et al. 2006] for data cleaning.

Active learning [Settles 2012] deals with the problem of optimally using an oracle, which is costly to access, to provide labels for training data that will be used to build a learning model, e.g., a classifier. This implies a tradeoff between the cost of oracle calls, and the cost of errors on the task. Active learning can be seen as an access choice problem between two accesses: one which is costly but certain, and another which is a cheap but noisy extrapolation based on the current knowledge.

2.3 Updating the Representation

Once an access has been performed and results have been obtained, the representation of the world must be updated accordingly. The simplest possibility is to update (or re-train) some model parameters as new data is retrieved [Gouriten et al. 2014]. A general framework to perform this task is that of *Bayesian inference* [Gelman et al. 2013]: integrate

the observation to our prior knowledge to obtain a posterior representation of the world. This is relevant whenever we are keeping a prior probabilistic model of the world, which is the case, e.g., when using probabilistic influence graphs for influence maximization [Lei et al. 2015].

Updates are especially hard to perform in the common situation when the accesses only return uncertain or incomplete information about the real data. For instance, say an access gave us the *number* of data items of interest (which may be useful, e.g., to decide whether we want to retrieve them one by one or in bulk); we must represent the existence of these items, though we know nothing about them. This problem also occurs in crowdsourcing contexts: constraints on access results may force us to extrapolate additional information [Amarilli et al. 2014].

Updating probabilistic representations has been studied for probabilistic XML documents [Kharlamov et al. 2010], and for probabilistic databases, where it is called *conditioning* [Koch and Olteanu 2008], i.e., restricting the possible worlds of a database with an additional constraint such as a logical rule or an additional observation. In most situations, however, this task is intractable.

3. APPLICATIONS

The intensionality of data, but also the heterogeneity of its structure, and our uncertainty about the true state of the world, are three major challenges of intensional data management. What is more, these challenges are not independent, but interact tightly. For example, if we use a probabilistic modeling of uncertainty, we need to represent, manage, query, probability distributions on structured objects, so that the representation of uncertainty depends on the structure that we use. Likewise, the kinds of intensional accesses which we may perform depend on the structure of the data considered, and structural constraints can be used to restrict the kind of accesses to make. Last, our intensional accesses will depend on our representation of uncertainty, as this representation may be used, e.g., to *predict* the results of accesses which have not been performed yet. Hence, we simply cannot use independent solutions to address each of these challenges.

In this section, we explain how these issues are addressed in three problem settings: focused Web crawling, online influence maximization, and crowdsourcing.

3.1 Focused Web Crawling

Focused Web crawling [Gouriten et al. 2014] is the task of collecting all information on the Web which is *relevant* to a given query. For instance, a media company might only be interested in blog posts related to its business, or in web pages containing music clips, while it may not be interested in general news.

This focused data collection task can be formalized as a *driven* exploration of a graph. This is supported by the current model of the Web as a graph: searching for Web content (nodes) is performed by following hyperlinks (edges). In most of the cases, however, accessing such data is not free. Each time we retrieve Web content, and extract the outgoing hyperlinks, we incur costs. For example, in Twitter, the current crawling API only allows a small number of requests in a 15-minute window. For these reasons, it is very important to

determine effectively which nodes in the Web graph should be crawled next, given our limited resources.

In the following, we summarize how focused Web crawling can be solved within the framework that we described in Section 2.

Model. In Web crawling, we have a directed graph $G = (V, E)$ where the nodes V are the resources to be crawled, and the edges E are the hyperlinks between those resources. Each node and edge has a non-negative *weight*. The weight of a node can be interpreted as its relevance to a topic: it can only be known once it has been crawled. The weight of an edge can be seen as an estimation of the outgoing node's weight. For example, this edge weight may be generated by analyzing anchor text in Web hyperlinks.

A crawl has starting points, or *seed* nodes. From these seed nodes, only one action can be made: crawling nodes which have not been crawled before but can be reached by a hyperlink from a crawled node. This is commonly called the *frontier*. A sequence of such actions is called a *crawl sequence*, and our objective is to find the optimal crawl sequence, so that the sum of the crawled node weights is maximized, while the number of accesses satisfies our budget limit.

The most important operation that must be supported by the model is to estimate the unknown scores of the frontier nodes. In our case, this estimation is made using *neighborhood estimators*, which are arbitrary functions, depending on the type of neighborhood – incoming or outgoing edges or nodes, or a combination thereof.

Choosing accesses. At each step of the crawl, the system must find the next node in the optimal crawl. Finding this optimal sequence of crawls is computationally hard, even when the graph is known in advance.

This hardness result means that the choice of the next node to crawl will be necessarily heuristic. A number of different neighborhood estimators can be built, based on which type of data is included: based on nodes (`n` estimator), based or edges (`e` estimator), or based on both (`ne` estimator). These three can also be combined in a linear regression model to guess the weight of a node (`lr_ne` estimator).

These estimators each have their strengths and weaknesses, and perform better or worse on different graphs or topics. This is a classic situation where multi-armed bandit approaches can help (`mab` estimator). We see each estimator as an arm, whose payoff in each situation is that of the choice that it recommends: each arm will then have a probability of being chosen in the next step of the crawl.

Updating the model. Since the next best access is chosen according to the estimators, we need to update them constantly with new evidence. In the case of focused crawling, this means re-training the parameters of the estimators – especially in the case of linear regression – at every step of the crawl. This leads to an inherent performance versus precision tradeoff. Depending on the application, the time for training might be greater than the allowed latency between accesses, and hence we need to re-train in batches for such scenarios.

To illustrate the performance of this framework for focused crawling, we present exper-

Table I. Global score of our estimators (higher is better) at different stages of a crawl

Estimator	100 nodes	1,000 nodes	10,000 nodes	100,000 nodes
Baseline (`bfs`)	0.147	0.132	0.130	0.207
Node-based (`n`)	0.358	0.280	0.362	0.467
Edge-based (`e`)	0.594	0.560	0.457	0.377
Node- and edge-based (`ne`)	0.583	0.570	0.466	0.378
Linear regression (`lr_ne`)	0.325	0.382	0.466	0.504
Multi-armed bandits (`mab`)	0.427	0.413	0.461	0.458

imental results in Table I, adapted from [Gouriten et al. 2014], on datasets derived from Wikipedia and Twitter. This table shows the global normalized score of the crawl as the number of crawled nodes increases, and gives a general intuition of the performance of the various strategies. We compare with a baseline breadth-first search estimator (`bfs`) which behaves like a vanilla crawler by following the outgoing hyperlinks as they are discovered. Simple estimators like `e`, `n` and `ne` are good in various stages of a crawl, while `lr_ne`'s capability of retraining its parameters pays off in the long term. However, `mab` has the advantage of being more stable at any stage of the crawl.

3.2 Online Influence Maximization

In social networks, and in tasks such as marketing and computational advertising, one often wishes to target a small number of popular users in the network, in a way that maximises the overall number of users influenced by the campaign. We can use propagation models to represent the influence as something that propagates probabilistically along edges of the known social network. Consequently, to solve influence maximization problems, probabilistic influence graphs are helpful to effectively maximize real influence.

In reality, however, these graphs are not known. To illustrate, imagine a scenario where a marketing company is coming into a new university campus, has a budget (monetary or time-wise), and wants to successfully maximize the influence of their campaign without any a priori knowledge about the influence behaviour of the student social network. A successful campaign must carefully make the tradeoff between *exploitation* – in this case, influence maximization on the current knowledge of the social network – and *exploration* – in this case, probing parts of the network in the hope that new knowledge about the influence behaviour is learned.

Model. Similarly to Web crawling, the social network is modeled as a directed graph $G = (V,E)$. Here, the nodes V are users, and edges E are follower relations between them. Each edge $e \in E$ has an unknown *influence probability* p_e associated to it. On this influence graph, the influence propagation is a process occurring in discrete time steps: if at time t an user u is activated, then at time $t+1$ all friends of u along outgoing edges $e \in E_u$ can be independently activated with probability p_e. Once an user has been activated, they remain activated. The *influence maximization problem* then seeks to find k users to maximize the expected influence spread starting from them.

Table II. Influence propagation, at various numbers of steps, for online influence maximization variants

Method	**Update**	10 steps	20 steps	50 steps
Baseline (`random`)	N/A	268	528	1,205
Exploit-only (`exploit`)	None	1,775	2,789	4,032
Exploit-only (`exploit`)	Local and global	1,655	2,785	5,542
Confidence bound (`CB`)	Local and global	1,742	2,920	5,805
Exponentiated Gradient (`EG`)	Local and global	1,715	3,267	6,076

In the *online* setting, only the social network is known, but not the values of the probabilities p_e. In this case, the uncertainty of the influence graph can be modelled by replacing the (unknown) probabilities by distributions on edges; in our case, each edge e is modeled as a Beta distribution: $p_e \sim \text{Beta}(\alpha_e, \beta_e)$. Initially, each edge is associated a prior distribution $\text{Beta}(\alpha_0, \beta_0)$.

Choosing accesses. For each unit of budget, or *step*, the choice of the system is between picking k seeds based on the current uncertain graph (*exploit*), and picking k seeds using other assumptions about the graph. Each edge is modelled by a uncertain distribution, and the assumptions about the graph can be tested by "moving" the influence probabilities by a number of θ standard deviations – when $\theta = 0$, we follow a pure *exploit* strategy. There are various ways in which the value of θ can be chosen. First, we can fix an arbitrary value of θ – say, 1 – and then choose to use this non-zero θ and perform an *explore* step with probability ε (we call this method `CB`, from *confidence bound*). Alternatively, we can think of a set of θ values as arms in a multi-armed bandit; the corresponding ε values can then be re-computed after each step using the Exponentiated Gradient algorithm [Cesa-Bianchi and Lugosi 2006], which has a theoretical guarantee for a constant choice of θ values (we call this method `EG`).

Updating model. Once a step has been chosen, the influence campaign is executed and the activation feedback is retrieved from the real world. The feedback indicates on which edges an activation succeeded, and on which an activation failed. Based on this feedback, two types of updates are performed:

> a Bayesian *local update*, only for the edges which are in the activation feedback, in which we increment the α_e parameter by 1 if the edge was in the successful set, and increment β_e by 1 otherwise – this updates the Beta distribution on the edge, and *decreases* uncertainty; and
>
> a *global update*, based on maximum likelihood methods, in which the prior $\text{Beta}(\alpha_0, \beta_0)$ is recomputed for all edges.

We show experimental results on article collaboration datasets in Table II, adapting them from [Lei et al. 2015]. They suggest that updating the uncertain graph can lead to considerable gains in influence propagation. Combining the update step with methods to choose between θ values, such as `CB` and `EG`, further increases the effectiveness.

3.3 Crowd Data Sourcing

Consider now the task of *crowd data sourcing* [Boim et al. 2012]: using the crowd (e.g., through crowdsourcing platforms such as Amazon Mechanical Turk) to gather or enrich data, in order to satisfy a particular goal. Here again, data accesses have a cost: that of engaging human workers, for instance with a monetary compensation, but also that of the inherent latency caused by crowd requests. Hence, this is again a situation of intensional data management.

In many crowd data sourcing applications [Boim et al. 2012; Parameswaran et al. 2011; Amsterdamer et al. 2013b; Crescenzi et al. 2013], intensional data management is a central question: "What is the best question to ask next to the crowd?" In settings where we can choose whom to ask questions to, the problem becomes: "Who is the best worker to ask this question next?"

Consider, for instance, the application where we wish to leverage the crowd's knowledge to mine association rules about people's daily practices [Amsterdamer et al. 2013b; 2013a]. A system to solve this problem must determine whether to ask an *open question* ("Tell me something you often do?") or a *closed question* which confirms the relevance of an already identified itemset ("Do you often have coffee in the morning?"). In another application, where the crowd helps inferring structured Web wrappers [Crescenzi et al. 2013], one has to decide which attribute of which Web page should the crowd workers be asked to annotate. Should we try to ask a question about an attribute on which many candidate wrapping rules disagree? Should we favor one that would confirm a rule already ranked as very likely? This is, once again, a tradeoff between exploration and exploitation.

Similarly, it is important for such applications to be able to update their current knowledge. This usually relies on Bayesian updating, applied to the score of an item of interest. In the first scenario, for instance, the system would update the support and confidence of an association rule as evidence is gathered. In the second, the updates would affect the probability that a given wrapping rule is correct.

4. CONCLUSION

We have illustrated the prevalence of *intensional data* in Web data management applications. Though a wealth of prior work has investigated specific applications of intensional data management, we expect that tackling specifically the general intensional data access problem could be fruitful, due to the numerous similarities between the tasks that such applications require: modeling an uncertain knowledge of the world, optimizing accesses to costly resources, and updating their knowledge. We also identified possible connections with the fields of reinforcement and active learning; however, we are dealing with data that is often of larger scale, with more structure, and more inherent uncertainty, and with queries that are more structured and complex, than what is commonly studied in these areas.

REFERENCES

AMARILLI, A., AMSTERDAMER, Y., AND MILO, T. 2014. Uncertainty in crowd data sourcing under structural constraints. In *UnCrowd*.

AMSTERDAMER, Y., GROSSMAN, Y., MILO, T., AND SENELLART, P. 2013a. Crowd miner: Mining association rules from the crowd. In *VLDB*. Demonstration.

AMSTERDAMER, Y., GROSSMAN, Y., MILO, T., AND SENELLART, P. 2013b. Crowd mining. In *SIGMOD*.

AUDIBERT, J.-Y., MUNOS, R., AND SZEPESVÁRI, C. 2009. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci. 410,* 19.

BENEDIKT, M., BOHANNON, P., AND BRUNS, G. 2006. Data cleaning for decision support. In *CleanDB*.

BENEDIKT, M., BOURHIS, P., AND SENELLART, P. 2012. Monadic Datalog containment. In *ICALP*.

BENEDIKT, M., GOTTLOB, G., AND SENELLART, P. 2011. Determining relevance of accesses at runtime. In *PODS*.

BOIM, R., GREENSHPAN, O., MILO, T., NOVGORODOV, S., POLYZOTIS, N., AND TAN, W. C. 2012. Asking the right questions in crowd data sourcing. In *ICDE*.

CALVANESE, D., DE GIACOMO, G., LEMBO, D., LENZERINI, M., POGGI, A., RODRIGUEZ-MURO, M., ROSATI, R., RUZZI, M., AND SAVO, D. F. 2011. The MASTRO system for ontology-based data access. *Semantic Web 2,* 1.

CESA-BIANCHI, N. AND LUGOSI, G. 2006. *Prediction, Learning and Games*. Cambridge University Press.

CRESCENZI, V., MERIALDO, P., AND QIU, D. 2013. A framework for learning web wrappers from the crowd. In *WWW*.

GARCIA-MOLINA, H., ULLMAN, J. D., AND WIDOM, J. 2009. *Database systems - the complete book*, 2 ed. Pearson Education.

GELMAN, A., CARLIN, J. B., STERN, H. S., DUNSON, D. B., VEHTARI, A., AND RUBIN, D. B. 2013. *Bayesian Data Analysis*, 3 ed.

GOURITEN, G., MANIU, S., AND SENELLART, P. 2014. Scalable, generic, and adaptive systems for focused crawling. In *Hypertext*.

HALEVY, A. Y. 2001. Answering queries using views: A survey. *VLDBJ 10,* 4.

KEMPE, D., KLEINBERG, J. M., AND TARDOS, É. 2003. Maximizing the spread of influence through a social network. In *KDD*.

KHARLAMOV, E., NUTT, W., AND SENELLART, P. 2010. Updating probabilistic XML. In *Updates in XML*.

KIMELFELD, B. AND SENELLART, P. 2013. Probabilistic XML: Models and complexity. In *Advances in Probabilistic Databases for Uncertain Information Management*. Springer.

KOCH, C. AND OLTEANU, D. 2008. Conditioning probabilistic databases. *PVLDB 1,* 1.

LEI, S., MO, L., MANIU, S., CHENG, R., AND SENELLART, P. 2015. Online influence maximization. Preprint available at `http://i.cs.hku.hk/~lymo/paper/oim.pdf`.

LENZERINI, M. 2002. Data integration: A theoretical perspective. In *PODS*.

MENCZER, F., PANT, G., SRINIVASAN, P., AND RUIZ, M. E. 2001. Evaluating topic-driven web crawlers. In *SIGIR*.

NAYAK, R., SENELLART, P., SUCHANEK, F. M., AND VARDE, A. 2012. Discovering interesting information with advances in Web technology. *SIGKDD Explorations 14,* 2.

PARAMESWARAN, A. G., GARCIA-MOLINA, H., PARK, H., POLYZOTIS, N., RAMESH, A., AND WIDOM, J. 2012. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*.

PARAMESWARAN, A. G., SARMA, A. D., GARCIA-MOLINA, H., POLYZOTIS, N., AND WIDOM, J. 2011. Human-assisted graph search: it's okay to ask questions. *PVLDB 4,* 5.

PUTEMAN, M. L. 2005. *Markov Decision Processes*. Wiley.

RAJARAMAN, A., SAGIV, Y., AND ULLMAN, J. D. 1995. Answering queries using templates with binding patterns. In *SIGMOD*.

REITER, R. 1977. Deductive question-answering on relational data bases. In *Logic and Data Bases*.

RISSE, T., DIETZE, S., PETERS, W., DOKA, K., STAVRAKAS, Y., AND SENELLART, P. 2012. Exploiting the social and semantic Web for guided Web archiving. In *TPDL*. Poster.

SETTLES, B. 2012. *Active Learning*. Morgan & Claypool Publishers.

SUCIU, D., OLTEANU, D., RÉ, C., AND KOCH, C. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers.

SUTTON, R. S. AND BARTO, A. G. 1998. *Reinforcement Learning*. MIT Press.

VALIANT, L. G. 1979. The complexity of enumeration and reliability problems. *SIAM J. Comput. 8,* 3.

YANG, X., CHENG, R., MO, L., KAO, B., AND CHEUNG, D. 2013. On incentive-based tagging. In *ICDE*.

Antoine Amarilli is a PhD candidate at Télécom ParisTech supervised by Pierre Senellart. He obtained his MSc from École normale supérieure in 2012. His PhD research is on the theoretical aspects of the management of uncertain, structured, and intensional data.

Silviu Maniu is a Researcher at the Huawei Noah's Ark Lab in Hong Kong. Between 2012 and 2014, he was a Postdoctoral Fellow in the Department of Computer Science at the University of Hong Kong. He received his PhD from Télécom ParisTech in 2012. His research interests are uncertain databases, and Web data management and mining.

Pierre Senellart is a Professor at Télécom and a Senior Research Fellow at the National University of Singapore. His research interests focus around practical and theoretical aspects of Web data management, including Web crawling and archiving, Web information extraction, uncertainty management, Web mining, and querying under access limitations.