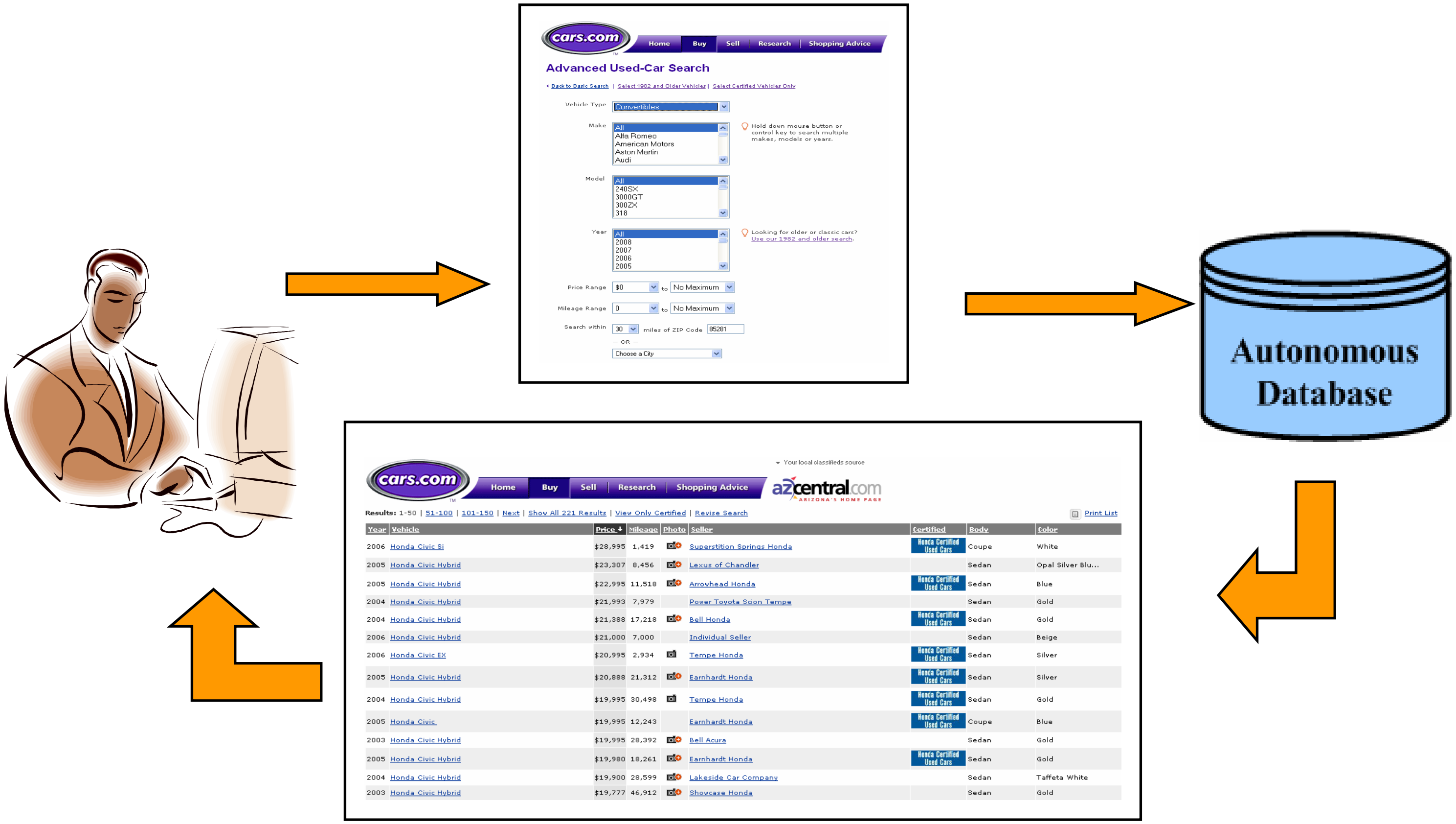
Query Processing over Incomplete Autonomous Databases

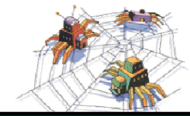
Garrett Wolf (Arizona State University)
Hemal Khatri (MSN Live Search)
Bhaumik Chokshi (Arizona State University)
Jianchun Fan (Amazon)
Yi Chen (Arizona State University)
Subbarao Kambhampati (Arizona State University)

Introduction

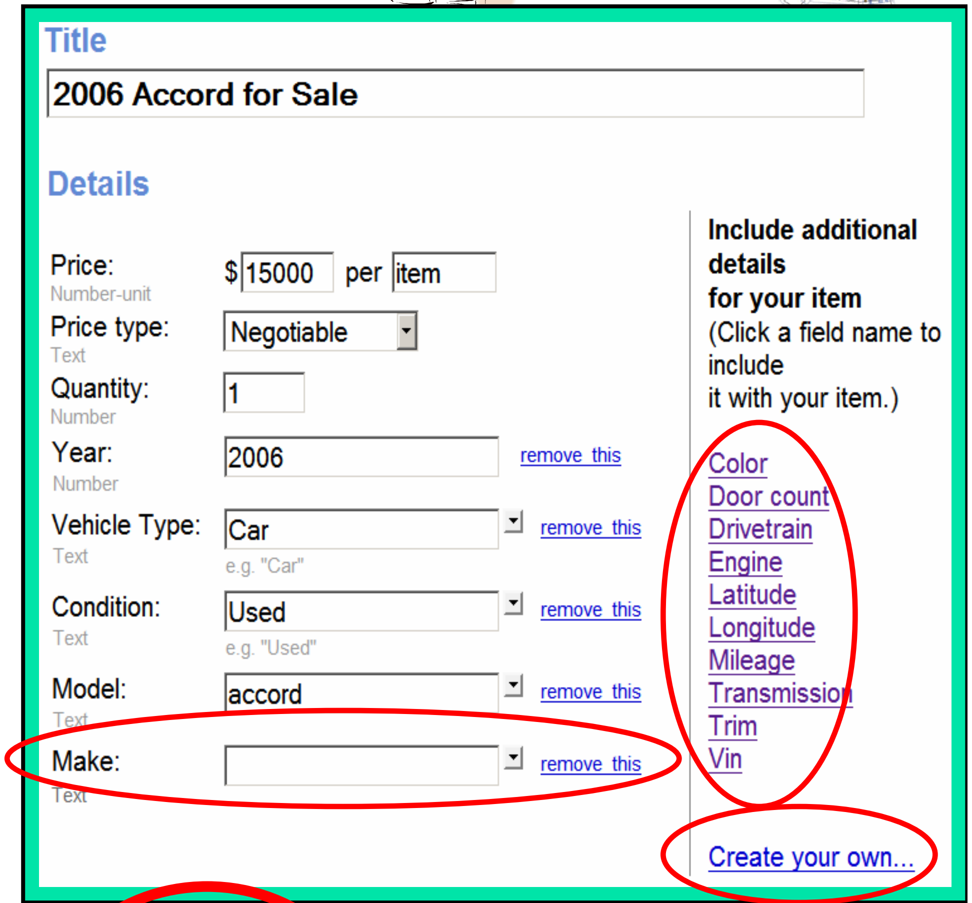
- More and more data is becoming accessible via web servers which are supported by backend databases
  - *E.g. Cars.com, Realtor.com, Google Base, Etc.*



Incompleteness in Web Databases



- **Inaccurate Extraction / Recognition**
- **Incomplete Entry**
- **Heterogeneous Schemas**
- **User-defined Schemas**



Website	# of Attributes	Total Tuples	Incomplete %	Body Style %	Engine %
AutoTrader.com	13	25127	33.67%	3.6%	8.1%
CarsDirect.com	14	32564	98.74%	55.7%	55.8%
Google Base	203+	580993	100%	83.36%	91.98%

Problem

- Current autonomous database systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

How to retrieve relevant uncertain results in a ranked fashion?

Want a '**Honda Accord**' with a '**sedan**' body style for under '**$12,000**'

	Make	Model	Year	Price	Color	Body
						?
						Sedan
	Honda	Accord	1999	?	Green	Sedan

Many entities corresponding to tuples with missing values might be relevant to the user query

Possible Naïve Approaches

Query Q: (Body Style = Convt)

1. **CERTAINONLY:** Return certain answers only as in traditional databases, namely those having Body Style = Convt

Low Recall

2. **ALLRETURNED:** Null matches any concrete value, hence return all answers having Body Style = Convt along with answers having body style as null

Low Precision, Infeasible

3. **ALLRANKED:** Return all answers having Body Style = Convt. Additionally, rank all answers having body style as null by predicting the missing values and return them to the user

Costly, Infeasible

Outline

- **Core Techniques**
- Peripheral Techniques
- Implementation & Evaluation
- Conclusion & Future Work

The QPIAD Solution

Given a query Q:(*Body=Convt*) retrieve all relevant tuples

Id	Make	Model	Year	Body
1	Audi	A4	2001	Convt
2	BMW	Z4	2002	Convt
3	Porsche	Boxster	2005	Convt
4	BMW	Z4	2003	**NULL**
5	Honda	Civic	2004	**NULL**
6	Toyota	Camry	2002	Sedan
7	Audi	A4	2006	**NULL**

Base Result Set

Id	Make	Model	Year	Body
1	Audi	A4	2001	Convt
2	BMW	Z4	2002	Convt
3	Porsche	Boxster	2005	Convt

LEARN

AFD: Model~> Body style

Select Top K Rewritten Queries

Q_1': Model=A4

Q_2': Model=Z4

Q_3': Model=Boxster

REWRITE

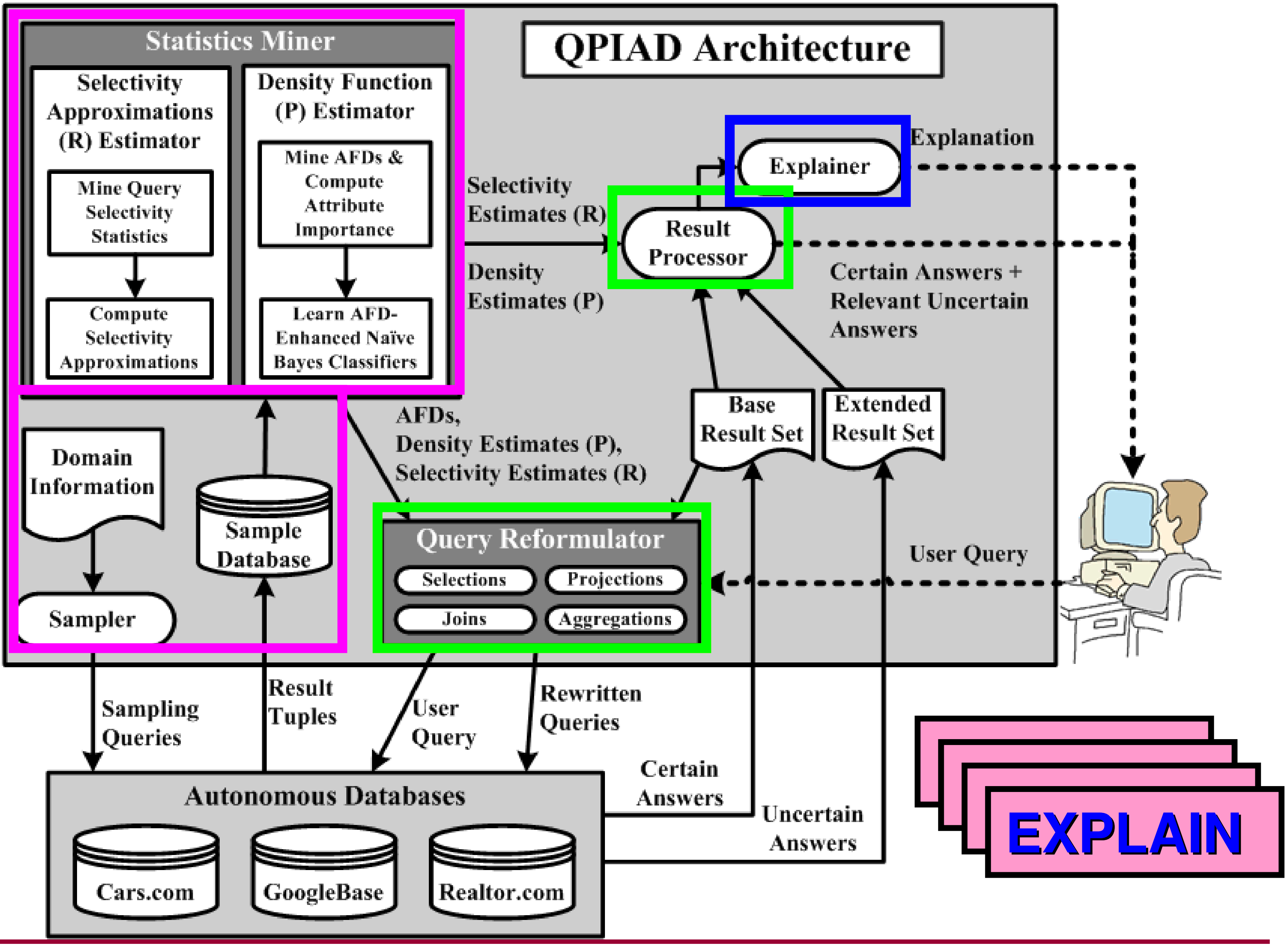
Re-order queries based on Estimated Precision

RANK

Ranked Relevant Uncertain Answers

Id	Make	Model	Year	Body	Confidence
4	BMW	Z4	2003	**NULL**	0.7
7	Audi	A4	2006	**NULL**	0.3

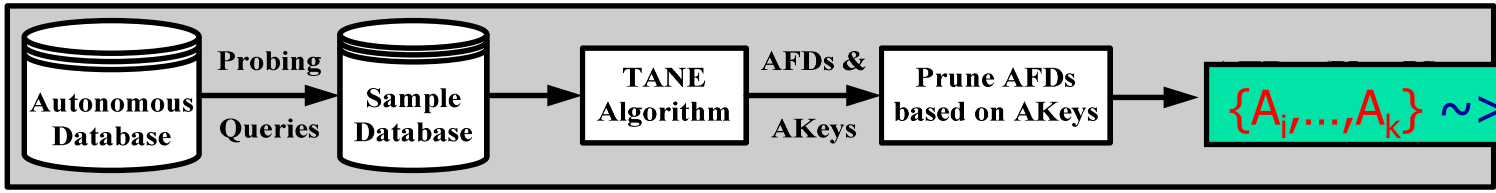
EXPLAIN


QPIAD Architecture
Statistics Miner
Selectivity Approximations (R) Estimator
Mine Query Selectivity Statistics
Compute Selectivity Approximations
Density Function (P) Estimator
Mine AFDs & Compute Attribute Importance
Learn AFD-Enhanced Naïve Bayes Classifiers
Domain Information
Sampler
Sample Database
Selectivity Estimates (R)
Density Estimates (P)
Result Processor
Explainer
Explanation
Certain Answers + Relevant Uncertain Answers
Base Result Set
Extended Result Set
AFDs, Density Estimates (P), Selectivity Estimates (R)
Query Reformulator
Selections
Projections
Joins
Aggregations
User Query
Sampling Queries
Result Tuples
User Query
Rewritten Queries
Certain Answers
Uncertain Answers
Autonomous Databases
Cars.com
GoogleBase
Realtor.com
EXPLAIN

Learning Statistics to Support Ranking & Rewriting

- **What is hard?**
  - Learning correlations useful for rewriting
  - Efficiently assessing the probability distribution
  - Cannot modify the underlying autonomous sources

- **Attribute Correlations** - Approximate Functional Dependencies (AFDs) & Approximate Keys (AKeys)



$\{A_i,\ldots,A_k\} \sim> A_m \quad 0<conf<=1$

Make, Body ~> Model

- **Value Distributions** - Naïve Bayes Classifiers (NBC)

$EstPrec(Q|R) = (A_m=v_m|dtrSet(A_m))$

P(Model=Accord | Make=Honda, Body=Coupe)

Make	Model	Year	Body
Honda	NULL	2001	Coupe

Rewriting to Retrieve Relevant Uncertain Results

- **What is hard?**
  - Retrieving relevant uncertain tuples with missing values
  - Rewriting queries given the limited query access patterns

AFD: Model~> Body

Base Set for Q:(Body=Convt)

Make	Model	Year	Body
Audi	A4	2001	Convt
BMW	Z4	2002	Convt
Porsche	Boxster	2005	Convt
BMW	Z4	2003	Convt

- Given an AFD and Base Set, it is likely that tuples with
  1) Model of A4, Z4 or Boxster
  2) Body of NULL

  are actually convertible.

- Generate rewritten queries for each distinct Model:
  - Q1′: Model=A4
  - Q2′: Model=Z4
  - Q3′: Model=Boxster

Selecting/Ordering Top-K Rewritten Queries

- **What is hard?**
  - Retrieving precise, non-empty result sets
  - Working under source-imposed resource limitations

- Select top-k queries based on **F-Measure**

P – Estimated Precision
R – Estimated Recall

$$F_\alpha = \frac{(1+\alpha)\cdot(P \cdot R)}{(\alpha \cdot P + R)}$$

- Reorder queries based on **Estimated precision**

$$F_0 = \frac{P \cdot R}{R} = P$$

All tuples returned for a single query are ranked equally

- Retrieves tuples in order of their **final ranks**
  - **No need to re-rank tuples after retrieving them!**

Explaining Results to the Users

- **What is hard?**
  - Gaining the user's trust
  - Generating meaningful explanations

Explanations based on AFDs.

Make, Body ~> Model
yields
This car is 83% likely to have Model=Accord given that its Make=Honda and Body=Sedan

Make	Model	Year	Price	Color	Body	Explanation
Honda	Accord	2001	$10,500	Silver	Sedan	
Honda	Accord	2002	$11,200	White	Coupe	
Honda	Accord	1999	$9,000	Green	Sedan	
?	Accord	2001	$11,700	Red	Sedan	This car is 100% likely to have [illegible]ke=Honda given that its Model=Accord
Honda	?	2000	$10,100	Blue	Sedan	This car is 83% likely to have Model=Accord given that its Make=Honda and Body=Sedan
Honda	Accord	1999	?	Black	Sedan	This car is 71% likely to have Price<$12,000 given that its Model=Accord and Year=1999
Honda	?	2002	$10,750	Silver	Coupe	This car is 42% likely to have Model=Accord given that its Make=Honda and Body=Coupe

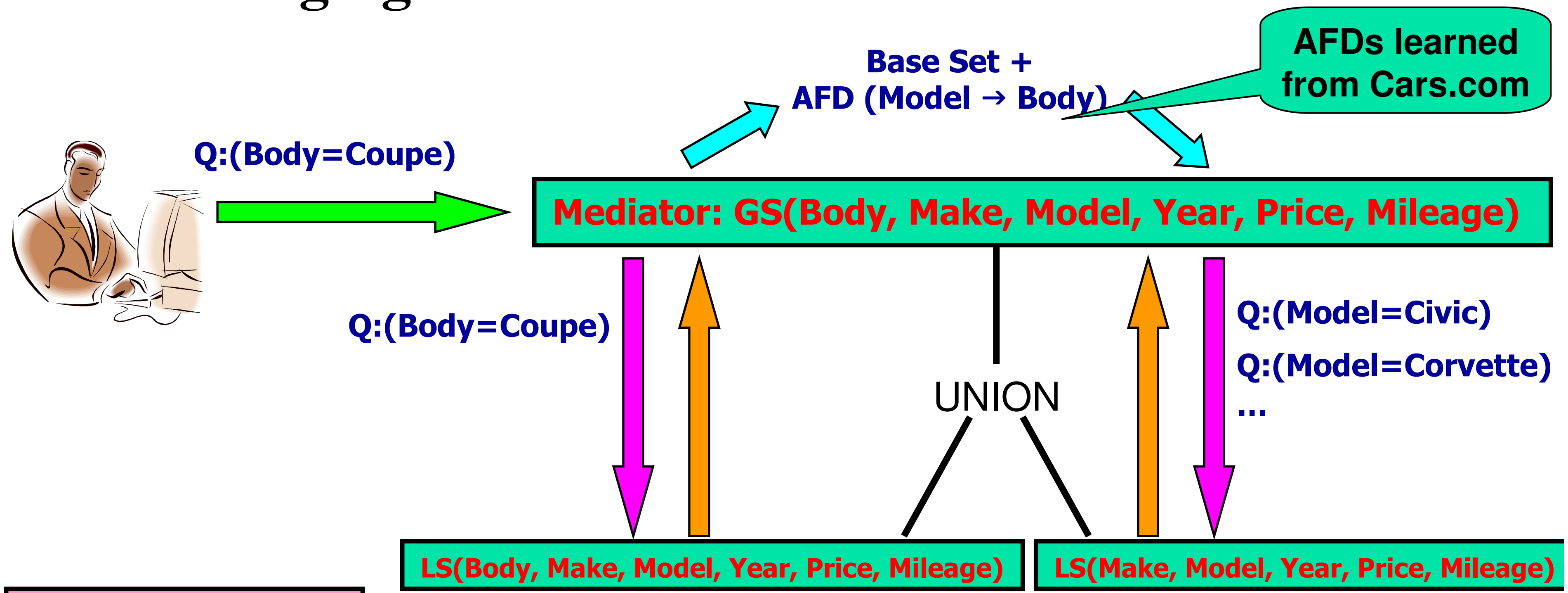
Provide to the user:

- ✓ Certain Answers
- ✓ Relevant Uncertain Answers
- ✓ Explanations

Outline

- ☑ Core Techniques
- ❑ **Peripheral Techniques**
- ❑ Implementation & Evaluation
- ❑ Conclusion & Future Work

Leveraging Correlation between Data Sources



Two main uses:

Source doesn't support all the attributes in GS

Sample/statistics aren't available

Handling Aggregate and Join Queries

- Aggregate Queries

Id	Make	Model	Body	
t1	Audi	A4	Convt	
t2	BMW	Z4	**NULL**	**P(Convt)=.9, P(Coupe)=.1**
t3	Porsche	Boxster	Convt	
t4	BMW	325i	**NULL**	**P(Convt)=.4, P(Coupe)=.6**
t5	Honda	Civic	Coupe	

Q:(Count(*) Where Body=Convt)

t1 + t3 + .9(t2) + .4(t4) =3.3

~~Include a portion of each tuple relative to the probability its missing value matches the query constraint~~

~~Count(*) = 3.3~~

t1 + t3 + t2 = 3

Count(*) = 3

Only include tuples whose most likely missing value matches the query constraint

- Join Queries

Make	Model	Year	Mileage
Honda	Accord	**NULL**	60,400
Toyota	Camry	2004	21,150
NULL	Civic	2002	53,275
Audi	A4	2003	41,650

Make	Year	Part
Honda	2001	Brakes
NULL	2002	Windows
Toyota	2005	Air Bags

$\bowtie$ **Make=Honda**

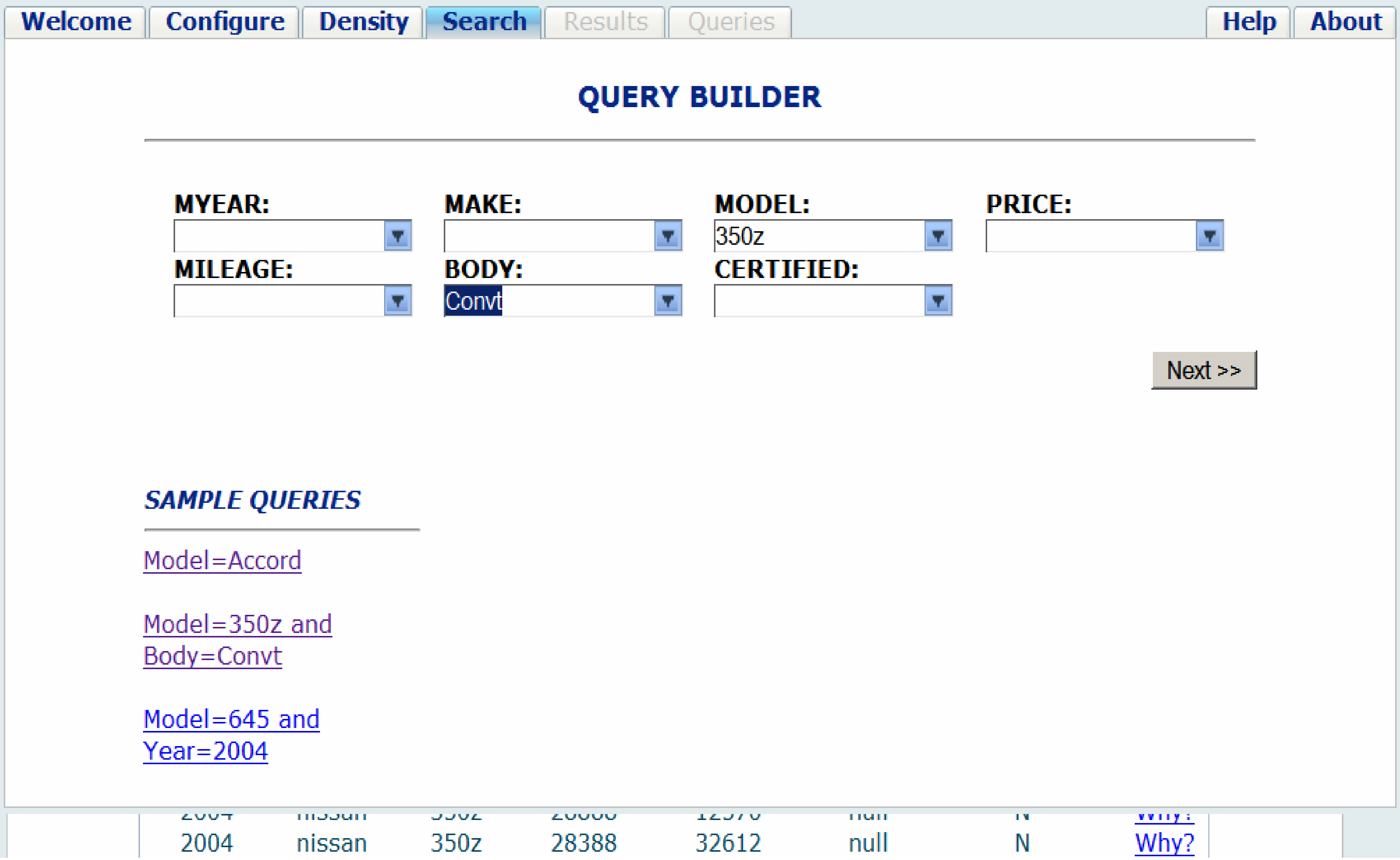
Make	Model	Year	Mileage	Part
Honda	Accord	**NULL/ 2001**	60,400	Brakes
NULL/ NULL	Civic	2002	53,275	Windows
Honda/ NULL	Accord	**NULL/ 2002**	60,400	Windows

Outline

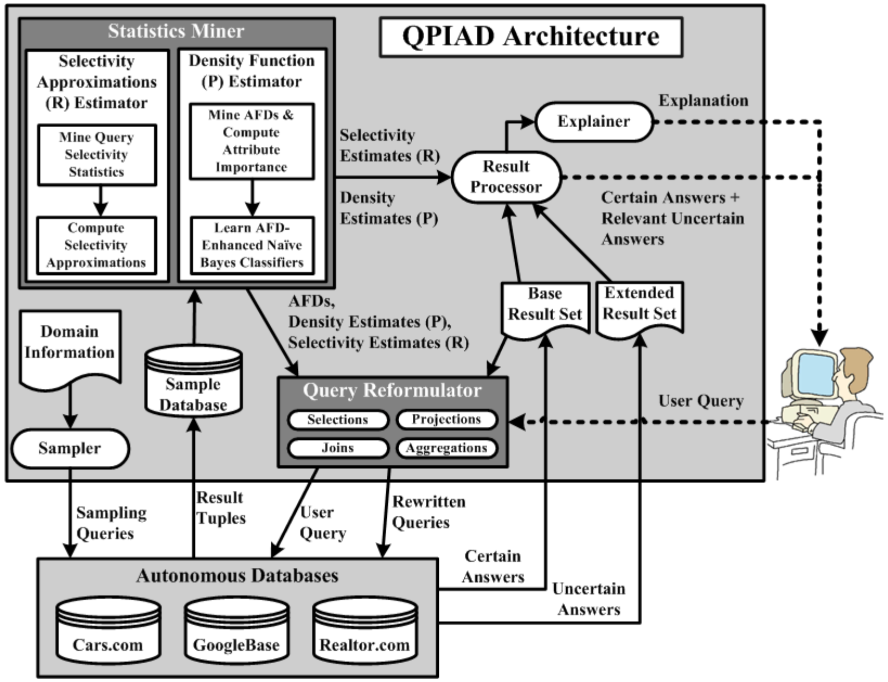
- ☑ Core Techniques
- ☑ Peripheral Techniques
- ❑ **Implementation & Evaluation**
- ❑ Conclusion & Future Work

QPIAD Web Interface

http://rakaposhi.eas.asu.edu/qpiad



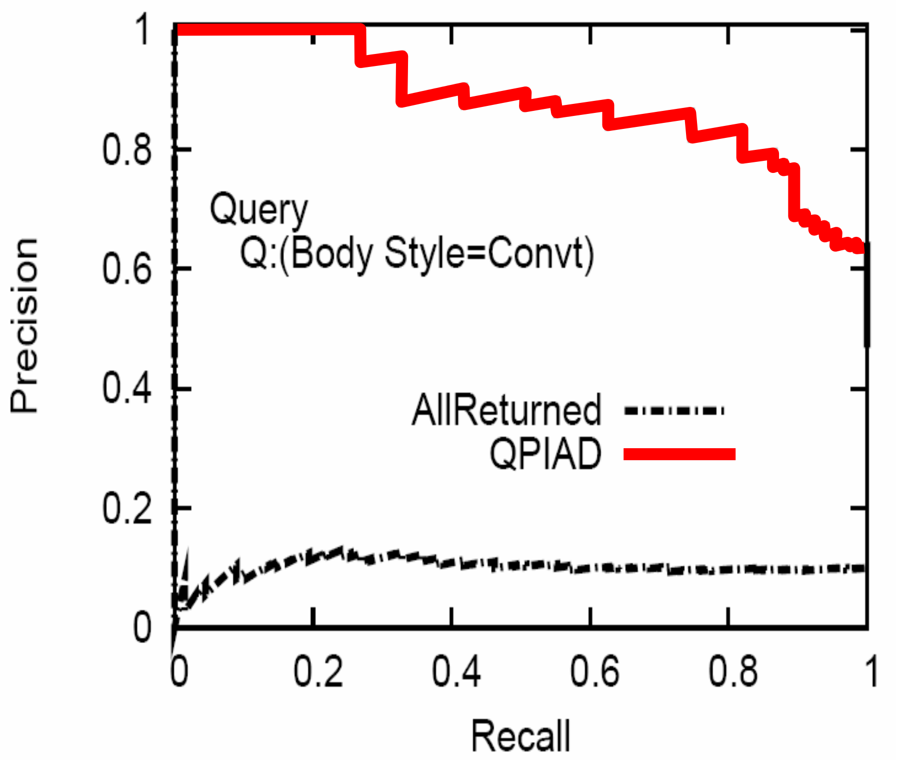
Empirical Evaluation

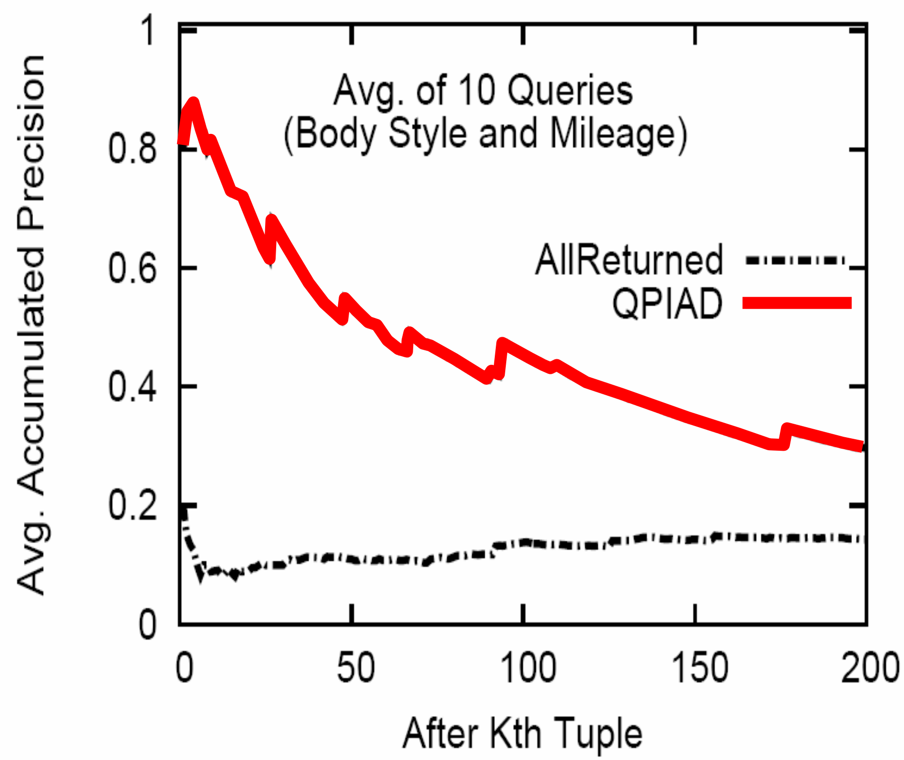


- **Datasets:**
  - **Cars**
    - *Cars.com*
    - 7 attributes, 55,000 tuples
  - **Complaints**
    - *NHSTA Office of Defect Investigation*
    - 11 attributes, 200,000 tuples
  - **Census**
    - *US Census dataset, UCI repository*
    - 12 attributes, 45,000 tuples
- **Sample Size:**
  - 3-15% of full database
- **Incompleteness:**
  - 10% of tuples contain missing values
  - Artificially introduced null values in order to compare with the ground truth
- **Evaluation:**
  - Ranking and Rewriting Methods (e.g. quality, efficiency, etc.)
  - General Queries (e.g. correlated sources, aggregates, joins, etc.)
  - Learning Methods (e.g. accuracy, sample size, etc.)

Experimental Results – Ranking & Rewriting

- QPIAD vs. ALLRETURNED - ***Quality***

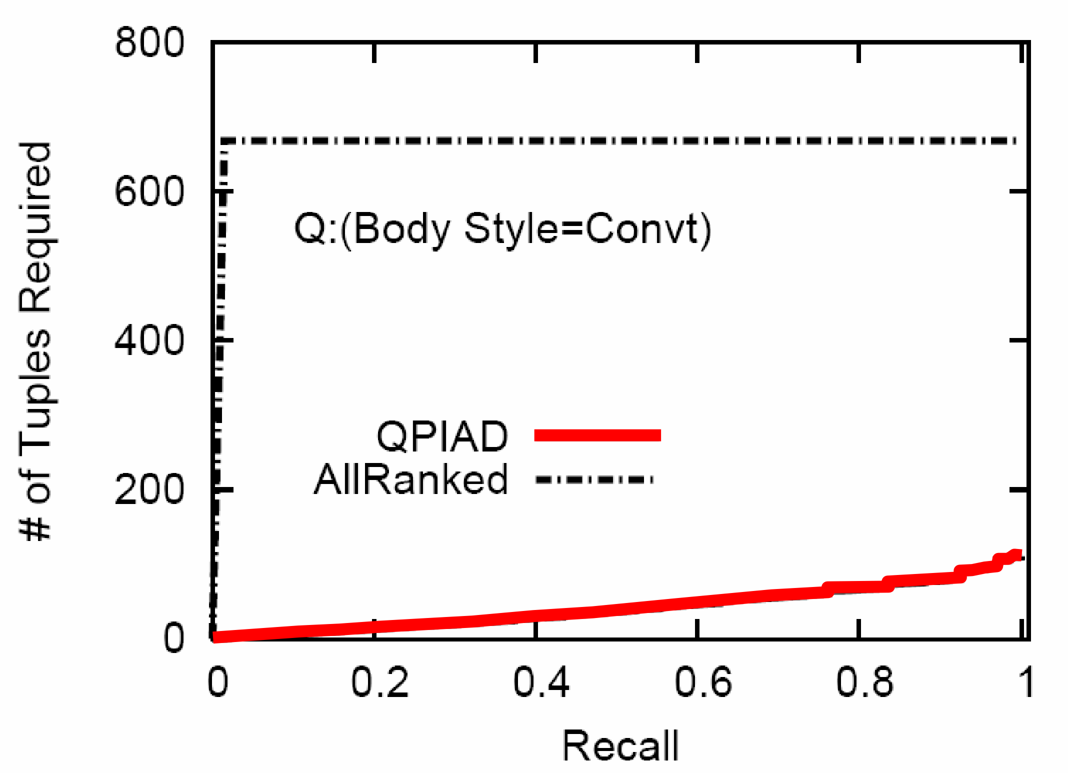




ALLRETURNED – all certain answers + all answers with nulls on constrained attributes

Experimental Results – Ranking & Rewriting

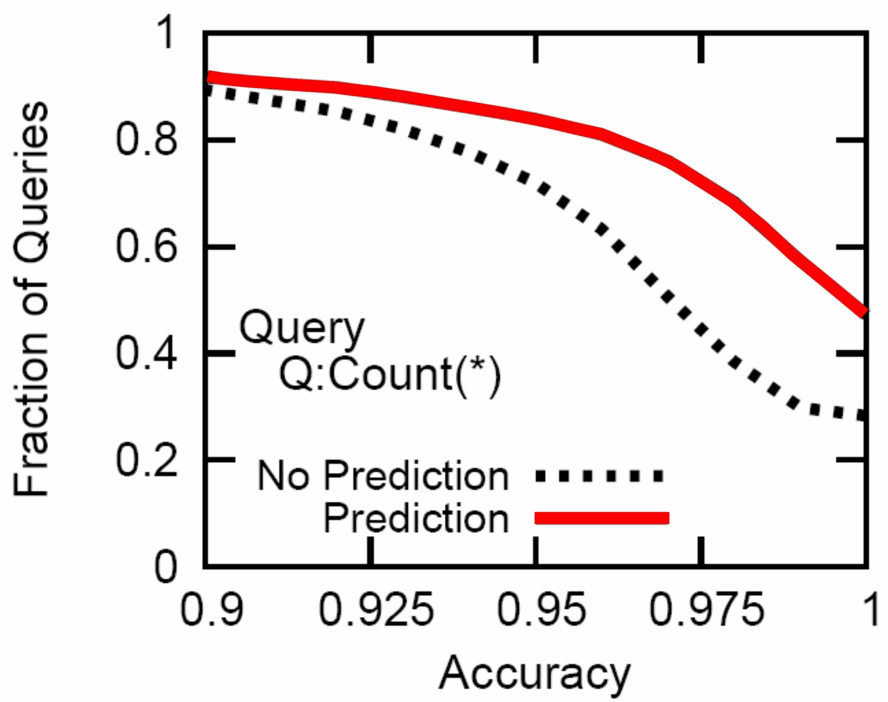
- QPIAD vs. ALLRANKED - ***Efficiency***



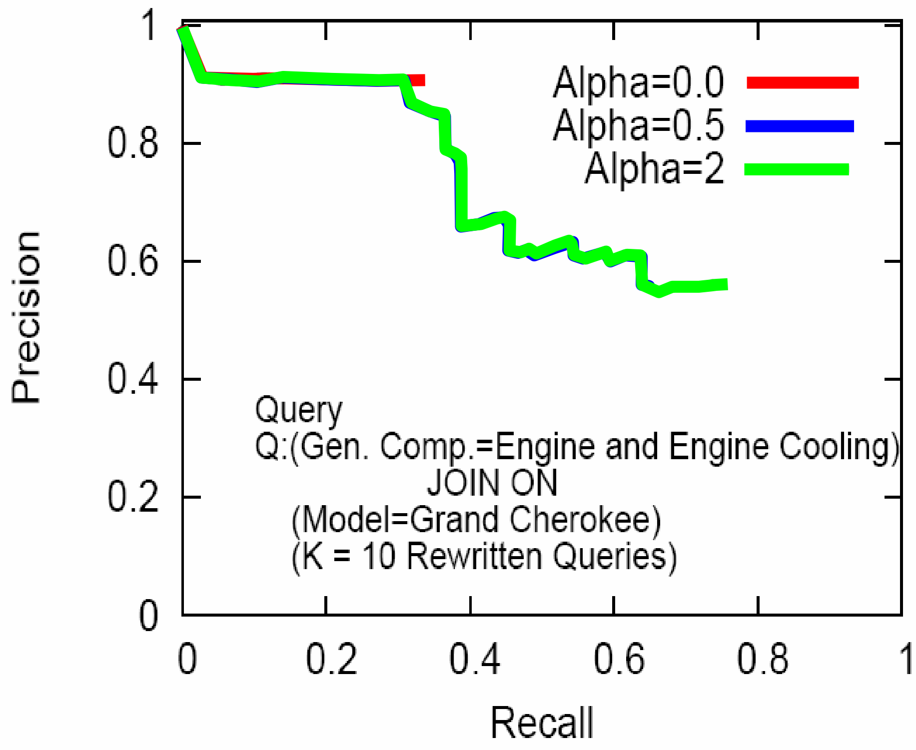
ALLRANKED – all certain answers + all answers with predicted missing value probability above a threshold

Experimental Results – General Queries

- Aggregates

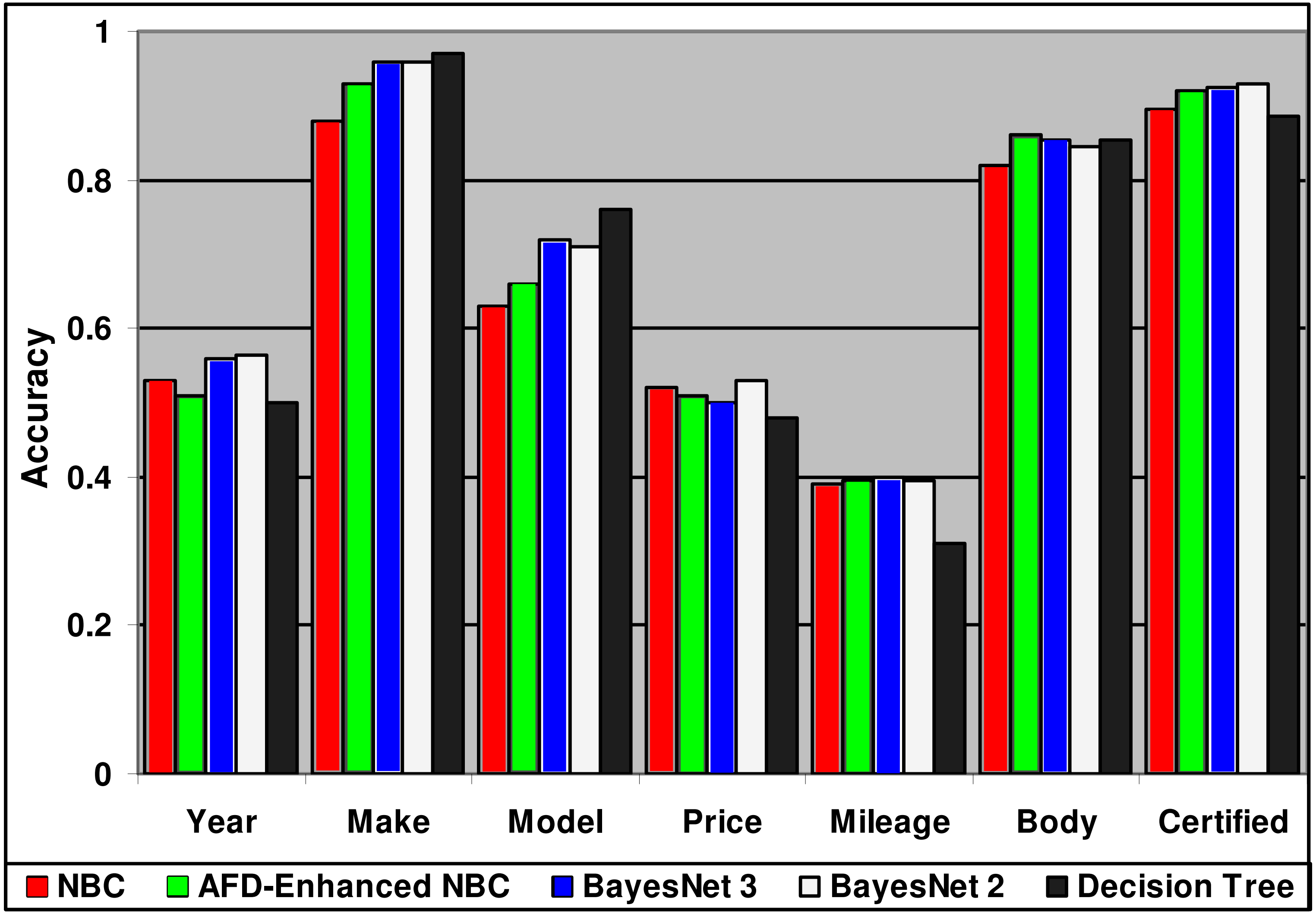


- Joins



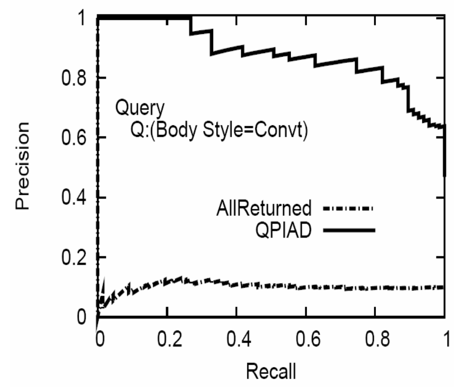
Experimental Results – Learning Methods

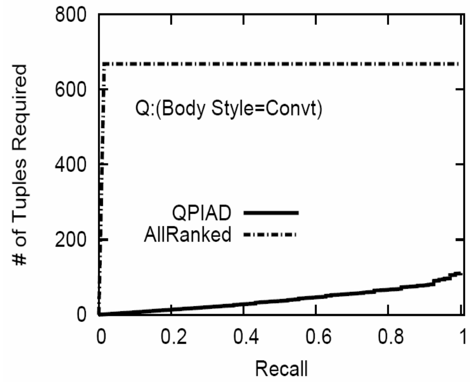
- Accuracy of Classifiers

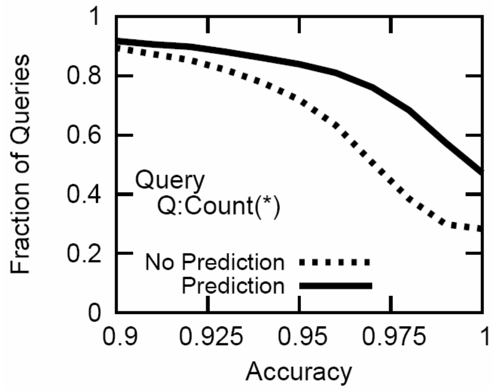


Experimental Summary

- Rewriting / Ranking
  - **Quality** – QPIAD achieves higher precision than ALLRETURNED by only retrieving the relevant tuples
  - **Efficiency** – QPIAD requires fewer tuples to be retrieved to obtain the same level of recall as ALLRANKED
- Learning Methods
  - AFDs for feature selection improved accuracy
- General Queries
  - Aggregate queries achieve higher accuracy when missing value prediction is used
  - QPIAD achieves higher levels of recall for join queries while trading off only a small bit of precision
- Additional Experiments
  - Robustness of learning methods w.r.t. sample size
  - Effect of alpha value on F-measure
  - Effectiveness of using correlation between sources







Outline

- ☑ Core Techniques
- ☑ Peripheral Techniques
- ☑ Implementation & Evaluation
- ❑ **Conclusion & Future Work**

Related Work

All citations found in paper

- Querying Incomplete Databases
  - Possible World Approaches – tracks the completions of incomplete tuples *(Codd Tables, V-Tables, Conditional Tables)*
  - Probabilistic Approaches – quantify distribution over completions to distinguish between likelihood of various possible answers

Our work fits here

- Probabilistic Databases
  - Tuples are associated with an attribute describing the probability of its existence
  - However, in our work, the mediator does not have the capability to modify the underlying autonomous databases
- Query Reformulation / Relaxation
  - Aims to return similar or approximate answers to the user after returning or in the absence of exact answers
  - Our focus is on retrieving tuples with missing values on constrained attributes
- Learning Missing Values
  - Common imputation approaches replace missing values by substituting the mean, most common value, default value, or using kNN, association rules, etc.
  - Our work requires schema level dependencies between attributes as well as distribution information over missing values

Contributions

- Efficiently retrieve relevant uncertain answers from autonomous sources given only limited query access patterns
  - Query Rewriting
- Retrieves answers with missing values on constrained attributes without modifying the underlying databases
  - AFD-Enhanced Classifiers
- Rewriting & ranking considers the natural tension between precision and recall
  - F-Measure based ranking
- AFDs play a major role in:
  - Query Rewriting
  - Feature Selection
  - Explanations

Current Directions – *QUIC (CIDR `07 Demo)*

http://rakaposhi.eas.asu.edu/quic

Incomplete Data

Databases are often populated by:

- **Lay users entering data**
- **Automated extraction**

Imprecise Queries

User's needs are not clearly defined:

- **Queries may be too general**
- **Queries may be too specific**

Density Function

$$\mathcal{P}(t|\hat{t}, D)$$

$$\mathcal{ER}(\hat{t}|Q, U, D) = \sum_{t \in C(\hat{t})} \mathcal{R}(t|Q, U)\mathcal{P}(t|\hat{t}, D)$$

Relevance Function

$$\mathcal{R}(t|Q, U)$$

General Solution: **"Expected Relevance Ranking"**

Challenge: Automated & Non-intrusive assessment of Relevance and Density functions

Estimating Relevance (R):

Learn relevance for user population as a whole in terms of value similarity

- Sum of weighted similarity for each constrained attribute
  - **Content Based Similarity**
  - **Co-click Based Similarity**
  - **Co-occurrence Based Similarity**

$\sigma_{Model \approx Civic}$	$Civic$	$Accord$	$Prelude$
Relevance	1.0	0.78	0.59
Density	0.62	0.21	0.17

Problem

- Current mediator systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results?

Want a '**Honda Accord**' with a '**sedan**' body style for under '**$12,000**'

	Make	Model	Year	Price	Color	Body
						?
						Sedan
	Honda	Accord	1999	?	Green	Sedan

Many entities corresponding to tuples with missing values might be relevant to the user query

Handling Aggregate and Join Queries

- Aggregate Queries

Q:(Count(*) Where Body=Convt)

Id	Make	Model	Body	Prob. Distr.
t1	Audi	A4	Convt	
t2	BMW	Z4	**NULL**	**P(Convt)=.9, P(Coupe)=.1**
t3	Porsche	Boxster	Convt	
t4	BMW	325i	**NULL**	**P(Convt)=.4, P(Coupe)=.6**
t5	Honda	Civic	Coupe	

t1 + .9(t2) + t3 + .4(t4) =3.3

~~Include a portion of each tuple relative to the probability its missing value matches the query constraint~~

~~Count(*) = 3.3~~

t1 + t2 + t3 = 3

Only include tuples whose most likely missing value matches the query constraint

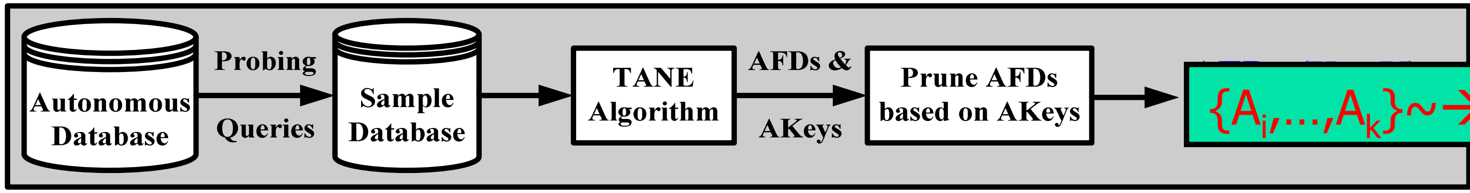
Count(*) = 3

- Join Queries
  - Refer to the paper for details

Learning Statistics to Support Ranking & Rewriting

- **What is hard?**
  - Learning correlations useful for rewriting
  - Efficiently assessing the probability distribution
  - Cannot modify the underlying autonomous sources

- **Attribute Correlations** - Approximate Functional Dependencies (AFDs) & Approximate Keys (AKeys)



$\{A_i,\ldots,A_k\} \sim\!\rightarrow A_m \quad 0<conf<=1$

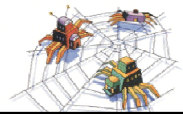
- **Value Distributions** - Naïve Bayes Classifiers (NBC)

$$EstPrec(Q|R) = (A_m=v_m|dtrSet(A_m))$$

- **Selectivity Estimates** – Sample Size, Ratio, Percent Incomplete

$$EstSel(Q|R) = SmplSel(Q) * SmplRatio(R) * PerInc(R)$$

Incompleteness in Web Databases



Inaccurate Extraction/Recognition

- Imperfections in segmenting of web pages or scanning and converting handwritten forms

Incomplete Entry

- User leaves the *Make* attribute blank assuming it is obvious as the *Model* is *Accord*

Heterogeneous Schemas

- Global schema provided by the mediator often contains attributes not present in all the local schemas

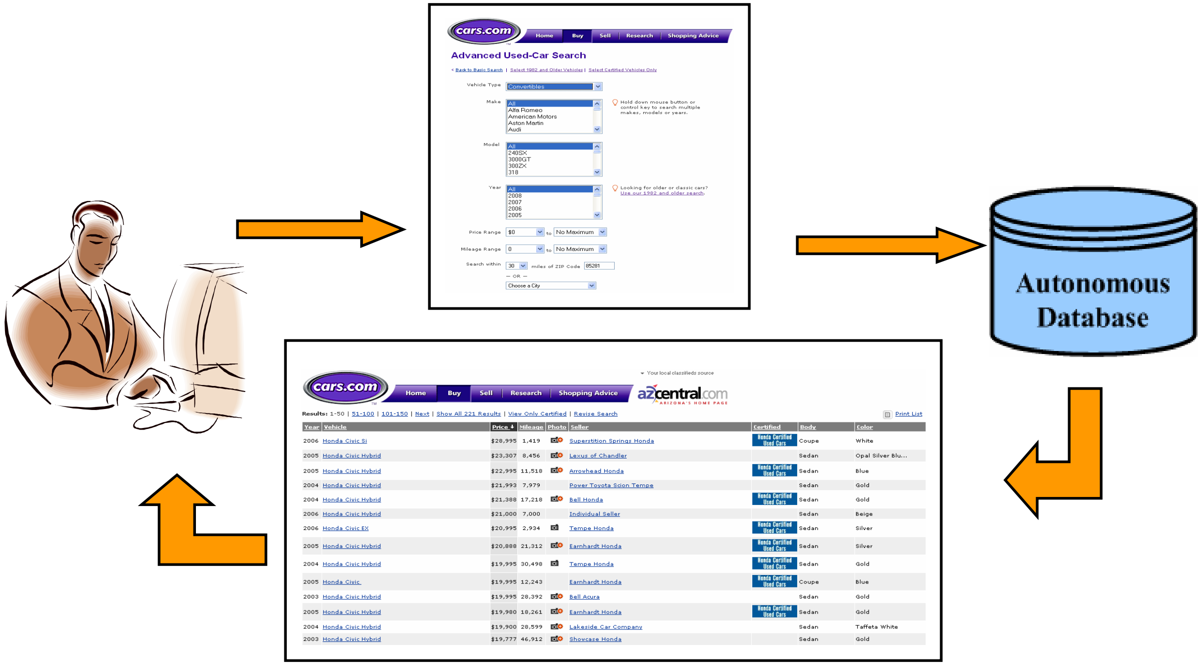
User-defined Schemas

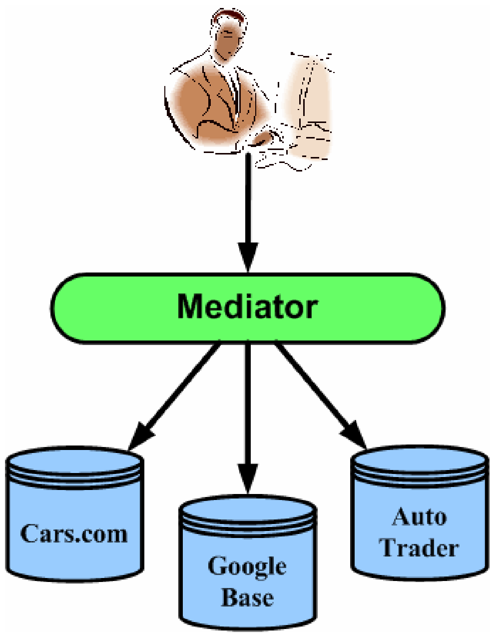
- Redundant attributes (e.g. Make vs. Manufacturer) and the proliferation of null values (e.g. tuples with Make are unlikely to provide Manufacturer)

Website	# of Attributes	Total Tuples	Incomplete %	Body Style %	Engine %
AutoTrader.com	13	25127	33.67%	3.6%	8.1%
CarsDirect.com	14	32564	98.74%	55.7%	55.8%
Google Base	203+	580993	100%	83.36%	91.98%

Introduction

- More and more data is becoming accessible via web servers which are supported by backend databases
  - *E.g. Cars.com, Realtor.com, Google Base, Etc.*
- As a result, mediator systems are being developed to provide a single point of access to multiple databases





Problem

- Current mediator systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

Want a '**Honda Accord**' with a '**sedan**' body style for under '**$12,000**'

	Make	Model	Year	Price	Color	Body
	Honda	Accord	2001	$10,500	Silver	?
	?	Accord	2002	$11,200	White	Sedan
	Honda	Accord	1999	?	Green	Sedan

Problem

- Current mediator systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results?

Want a '**Honda Accord**' with a '**sedan**' body style for under '**$12,000**'

	Make	Model	Year	Price	Color	Body
						?
						Sedan
	Honda	Accord	1999	?	Green	Sedan

Many entities corresponding to tuples with missing values might be relevant to the user query

Selecting Top-K Rewritten Queries using F-Measure

- Sources may impose resource limitations on the # of queries we can issue

- Therefore, we should select only the top-K queries while ensuring the proper balance between precision and recall

P – Estimated Precision

R – Estimated Recall
(based on P & Est. Sel.)

- SOLUTION: Use F-Measure based selection with configurable alpha parameter
  - **α=1** P = R
  - **α<1** P > R
  - **α>1** P < R

$$F_\alpha = \frac{(1+\alpha)\cdot(P\cdot R)}{(\alpha\cdot P+R)}$$

- NOTE: F-Measure is used for selecting the top-K queries but does not determine the order in which they are sent

We still want the most precise tuples first!

Ordering Top-K Queries using Estimated Precision

- Once we've selected the top-K rewritten queries, we must reorder them in order of their **estimated precision**
  - Use the precision estimates we already have

$$F_0 = \frac{P \cdot R}{R} = P$$

- Issuing the queries in order of estimated precision allows us to retrieve tuples in order of their final ranks
  - **No need to re-rank tuples after retrieving them, simply show them to the user!**

NOTE: All tuples returned for a single query are ranked equally

Problem

- Current autonomous database systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

Want a 'Honda Accord' with a 'sedan' body style for under '$12,000'

	Make	Model	Year	Price	Color	Body
	Honda	Accord	2001	$10,500	Silver	?
	?	Accord	2002	$11,200	White	Sedan
	Honda	Accord	1999	?	Green	Sedan

Problem

- Current autonomous database systems only return ***certain answers***, namely those which exactly satisfy all the user query constraints.

High Precision
Low Recall

How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results in a ranked fashion?

Want a '**Honda Accord**' with a '**sedan**' body style for under '**$12,000**'

	Make	Model	Year	Price	Color	Body
						?
						Sedan
	Honda	Accord	1999	?	Green	Sedan

Many entities corresponding to tuples with missing values might be relevant to the user query

Query Rewriting in QPIAD

Given a query Q:(Body Style=Convt) retrieve all relevant tuples

Id	Make	Model	Year	Body
1	Audi	A4	2001	Convt
2	BMW	Z4	2002	Convt
3	Porsche	Boxster	2005	Convt
4	BMW	Z4	2003	Null
5	Honda	Civic	2004	Null
6	Toyota	Camry	2002	Sedan
7	Audi	A4	2006	Null

Base Result Set

Id	Make	Model	Year	Body
1	Audi	A4	2001	Convt
2	BMW	Z4	2002	Convt
3	Porsche	Boxster	2005	Convt

AFD: Model~> Body style

Select Top K Rewritten Queries

Q_1': Model=A4

Q_2': Model=Z4

Q_3': Model=Boxster

Re-order queries based on Estimated Precision

Ranked Relevant Uncertain Answers

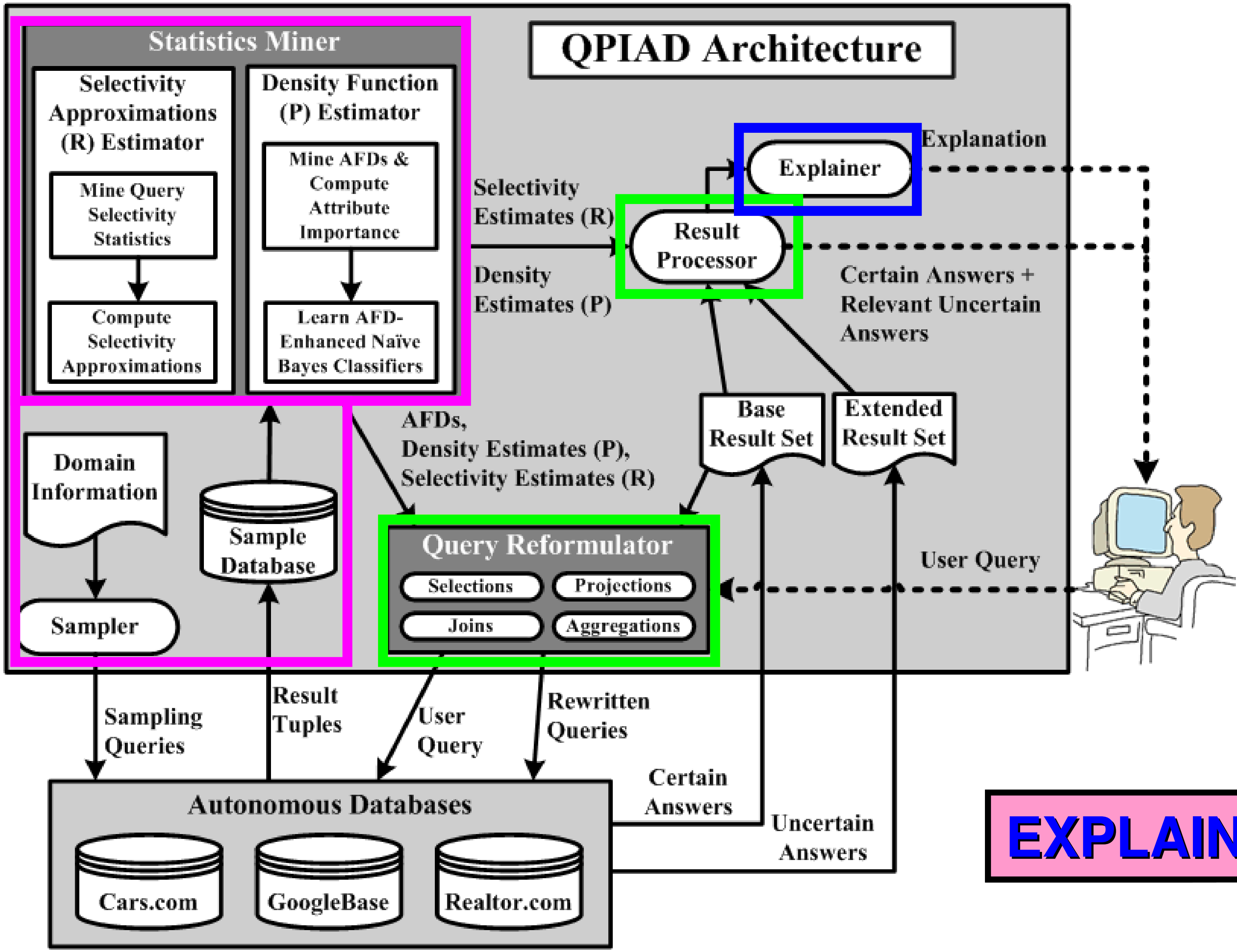
Id	Make	Model	Year	Body	Confidence
4	BMW	Z4	2003	Null	0.7
7	Audi	A4	2006	Null	0.3

We can select top K rewritten queries using F-measure

F-Measure = $(1+\alpha)*P*R/(\alpha*P+R)$

P – Estimated Precision

R – Estimated Recall based on P and Estimated Selectivity


QPIAD Architecture
Statistics Miner
Selectivity Approximations (R) Estimator
Mine Query Selectivity Statistics
Compute Selectivity Approximations
Density Function (P) Estimator
Mine AFDs & Compute Attribute Importance
Learn AFD-Enhanced Naïve Bayes Classifiers
Domain Information
Sample Database
Sampler
Selectivity Estimates (R)
Density Estimates (P)
Result Processor
Explainer
Explanation
Certain Answers + Relevant Uncertain Answers
Base Result Set
Extended Result Set
AFDs, Density Estimates (P), Selectivity Estimates (R)
Query Reformulator
Selections
Projections
Joins
Aggregations
User Query
Sampling Queries
Result Tuples
User Query
Rewritten Queries
Certain Answers
Uncertain Answers
Autonomous Databases
Cars.com
GoogleBase
Realtor.com

EXPLAIN

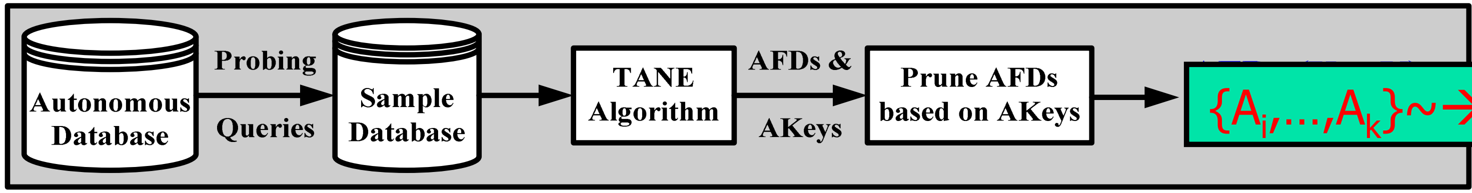
Handling Aggregate Queries

- As the fraction of incomplete tuples increases, the aggregates such as SUM and COUNT become increasingly inaccurate
  - **SOLUTION: Use query rewriting and missing value prediction to improve the accuracy of such aggregates**

1. Issue the original query to the database and retrieve the base set.
2. Compute the certain aggregate over the base set tuples.
3. Use the base set to generate rewritten queries according to the QPIAD algorithm.
4. Issue the rewritten queries and retrieve the extended result set.
5. For each tuple in the extended result set the value most likely to be the tuple's missing value.
6. If the most likely value is equal to the value specified in the original query, then include the tuple in the running uncertain aggregate total.
7. Return to the user the certain aggregate along with the uncertain aggregate.

Learning Statistics to Support Ranking & Rewriting

- Learning attribute correlations in the form of Approximate Functional Dependencies (*AFDs*) and Approximate Keys (*AKeys*)



$\{A_i, \ldots, A_k\} \sim\!\rightarrow A_m \quad 0 < conf <= 1$

- Learning value distributions using Naïve Bayes Classifiers (*NBC*)

$EstPrec(Q|R) = (A_m = v_m | dtrSet(A_m))$

- Learning Selectivity Estimates (*EstSel*) of Rewritten Queries based on:
  - Selectivity of rewritten query issued on sample *SmplSel(Q)*
  - Ratio of original database size over sample *SmplRatio(R)*
  - Percent of incomplete tuples while creating sample *PerInc(R)*

$EstSel(Q|R) = SmplSel(Q) * SmplRatio(R) * PerInc(R)$

Rewriting to Retrieve Relevant Uncertain Results

- An AFD tells us that for some fraction of the tuples, a car's Model can be used to determine its Body

AFD: Model~> Body

Id	Make	Model	Year	Body
1	Audi	A4	2001	Convt
2	BMW	Z4	2002	Convt
3	Porsche	Boxster	2005	Convt
4	BMW	Z4	2003	Convt

Base Set for Q:(Body=Convt)

- Base set tuples are known to have Body=Convt therefore if we:
  1) Encounter a tuple having a Model in the base set
  2) And the tuple has a missing value for Body,

  then it is likely that the tuple's Body is in fact Convt

- Given a query on attribute A, and an AFD B~>A, we generate rewritten queries by:
  – Determine the set of distinct values for the attributes contained in B
  – For each distinct value, generate a rewritten query constraining the corresponding attributes with the values from the distinct set

Selecting/Ordering Top-K Rewritten Queries

P – Estimated Precision
R – Estimated Recall

- Sources may impose resource limitations on the # of queries we can issue
- SOLUTION: Use F-Measure based selection with configurable alpha parameter

$$F_\alpha = \frac{(1+\alpha)\cdot(P \cdot R)}{(\alpha \cdot P + R)}$$

- NOTE: F-Measure is used for selecting the top-K queries but does not determine the order in which they are sent
- Once we've selected the top-K rewritten queries, we must reorder them in order of their **estimated precision**

$$F_0 = \frac{P \cdot R}{R} = P$$

- Issuing the queries in order of estimated precision allows us to retrieve tuples in order of their final ranks
  - **No need to re-rank tuples after retrieving them, simply show them to the user!**

NOTE: All tuples returned for a single query are ranked equally

Explaining Results to the Users

Problem:
How to gain users trust when showing them incomplete tuples?

Q:(Make=Honda and Model=Accord and Price<$12,000)

Make	Model	Year	Price	Color	Body	Explanation
Honda	Accord	2001	$10,500	Silver	Sedan	
Honda	Accord	2002	$11,200	White	Coupe	
Honda	Accord	1999	$9,000	Green	Sedan	
?	Accord	2001	$11,700	Red	Sedan	This car is 100% likely to have Make=Honda given that its Model=Accord
Honda	?	2000	$10,100	Blue	Sedan	This car is 83% likely to have Model=Accord given that its Make=Honda and Body=Sedan
Honda	Accord	1999	?	Black	Sedan	This car is 71% likely to have Price<$12,000 given that its Model=Accord and Year=1999
Honda	?	2002	$10,750	Silver	Coupe	This car is 42% likely to have Model=Accord given that its Make=Honda and Body=Coupe

Provide to the user:

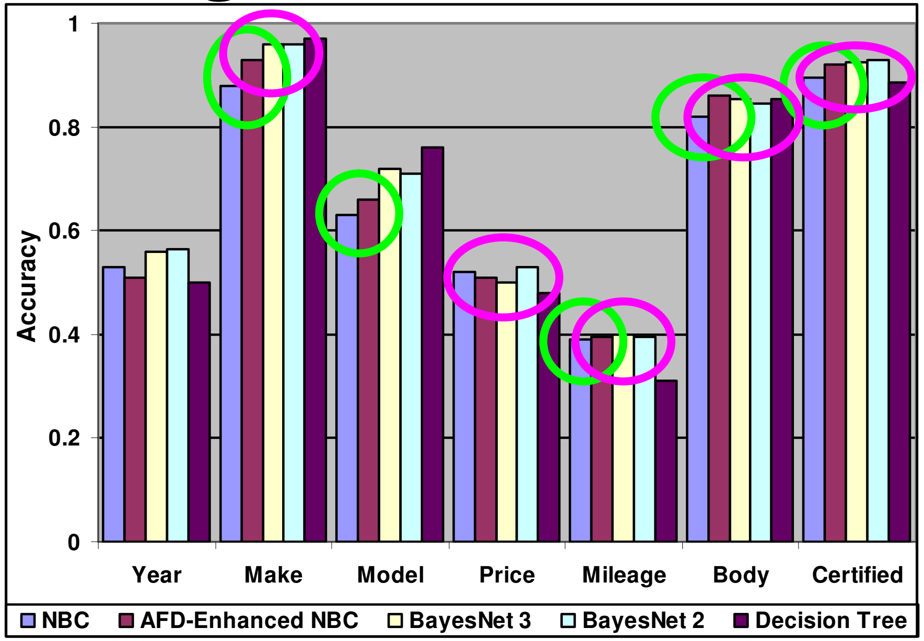
- ✓ Certain Answers
- ✓ Relevant Uncertain Answers
- ✓ Explanations

Experimental Results – Learning Methods

- Accuracy of Classifiers

  Using AFDs during feature selection improves accuracy

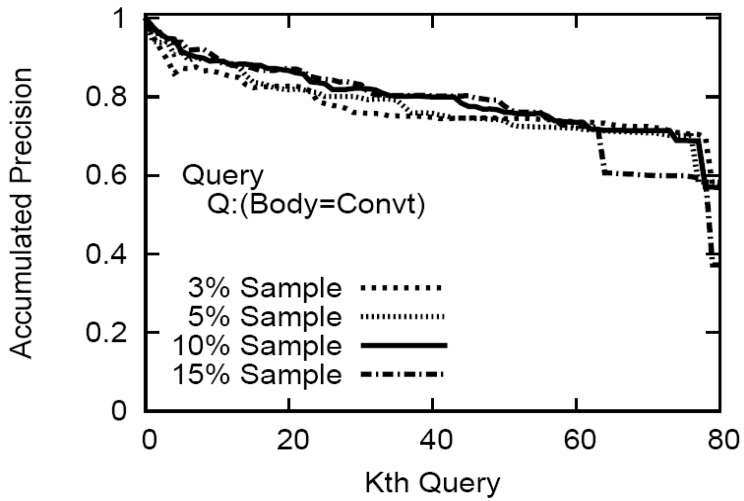
  Accuracy of AFD-Enhanced NBC is comparable with BayesNet



- Robustness w.r.t. Sample Size

  QPIAD is robust even when faced with a relatively small data sample

  Similar results were obtained on the Census database



Handling Aggregate and Join Queries

- Aggregate Queries
  - As the fraction of incomplete tuples increases, the aggregates such as SUM and COUNT become increasingly inaccurate
    - **SOLUTION: Use query rewriting and missing value prediction to improve the accuracy of such aggregates**

- Join Queries
  - A join query can be thought of as individual queries over each source, the results of which are joined at the mediator
  - Estimated precision and estimated selectivity must be considered when deciding which queries to issue
  - When estimating precision/selectivity, estimates should be made for a query pair rather than for each individual query
    - **We must ensure that the results of each of the individual queries agree on their join attribute values**

Experimental Results – Ranking & Rewriting

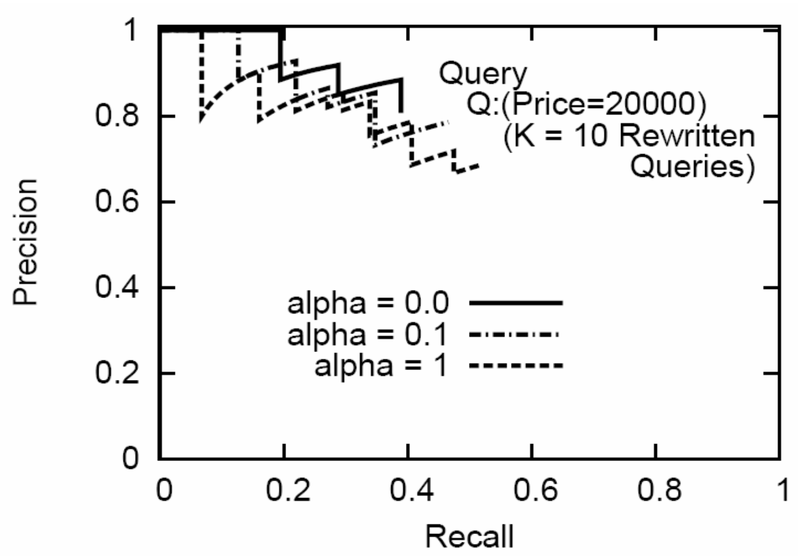
- Effect of α on F-Measure

Assumption: 10 query limit on # of rewritten queries we are allowed to issue

Sets tradeoff of precision & recall

Combined Effect = α + k

Resource limitation on # of rewritten queries



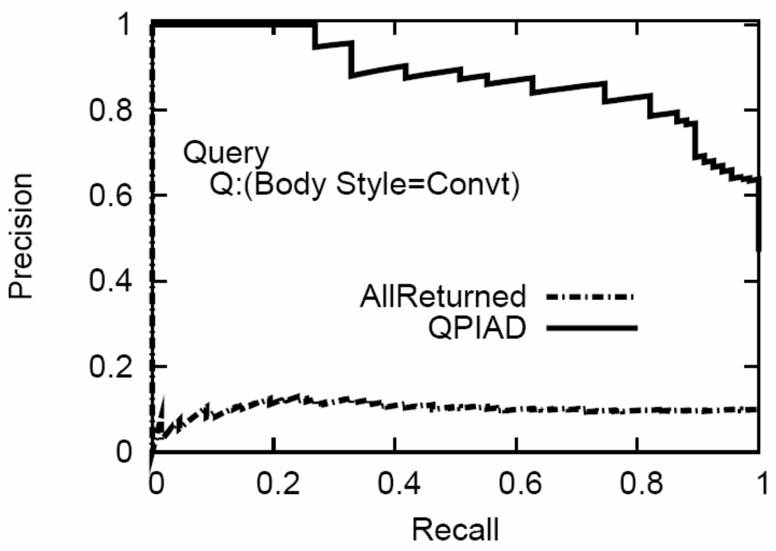
As alpha increases, we allow queries with lower precision to be issued in order to obtain a higher throughput

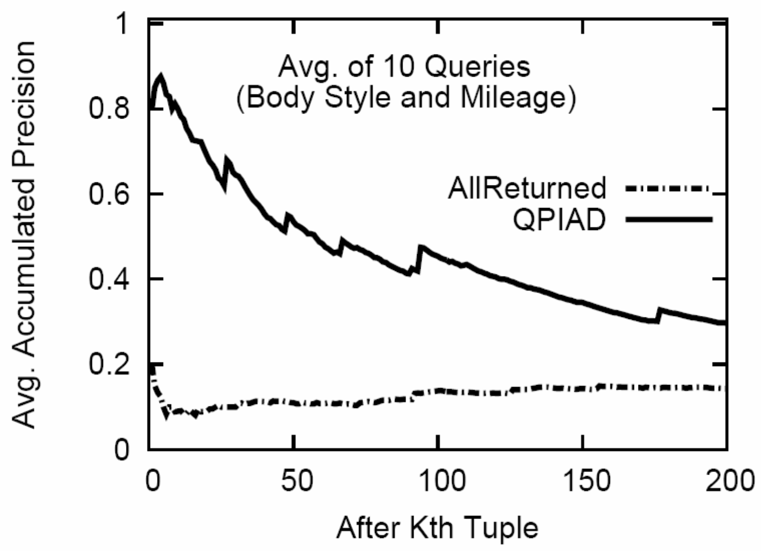
Experimental Results – Ranking & Rewriting

- QPIAD vs. ALLRETURNED - ***Quality***

  ALLRETURNED has **low precision** because not all tuples with missing values on the constrained attributes are relevant to the query

  QPIAD has a much **higher precision** than **ALLRETURNED** as it aims to retrieve tuples with missing values on the constrained attributes which are very likely to be relevant to the query





Experimental Results – Ranking & Rewriting

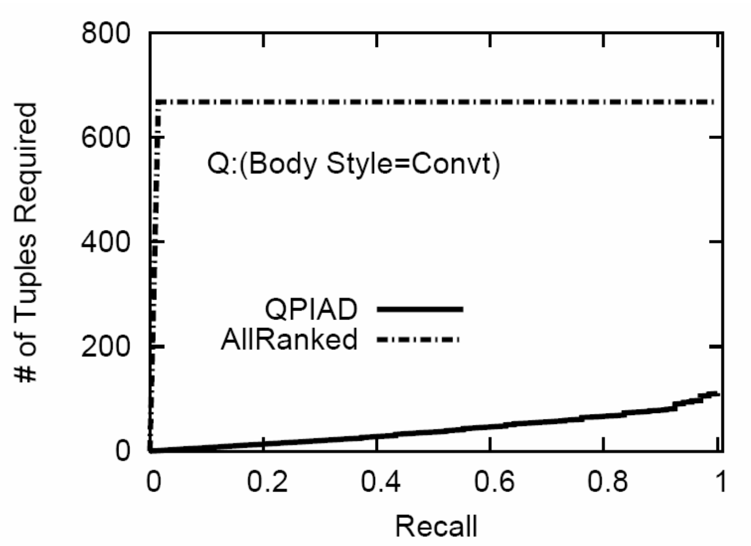
- QPIAD vs. ALLRANKED - ***Efficiency***

ALLRANKED approach is often **infeasible** as direct retrieval of null values is not often allowed

ALLRANKED approach must retrieve all tuples w/ missing Body Style in order to achieve any nonzero recall

QPIAD only retrieves a subset of the tuples having missing values on constrained attributes, namely those which are highly likely to be relevant to the query

QPIAD is able to achieve the same level of recall as **ALLRANKED** while requiring much **fewer tuples** to be retrieved

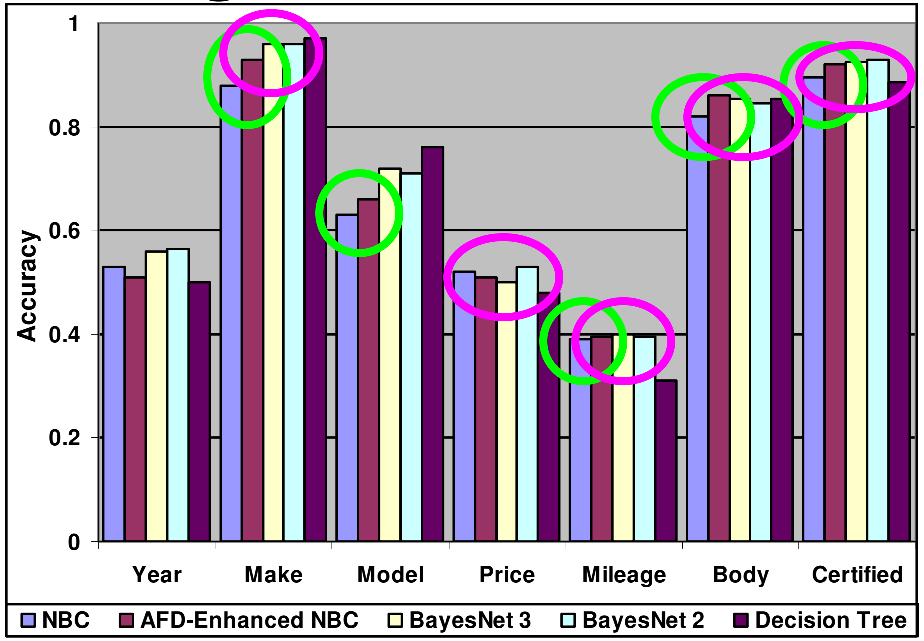


Experimental Results – Learning Methods

- Accuracy of Classifiers

  Using AFDs during feature selection improves accuracy

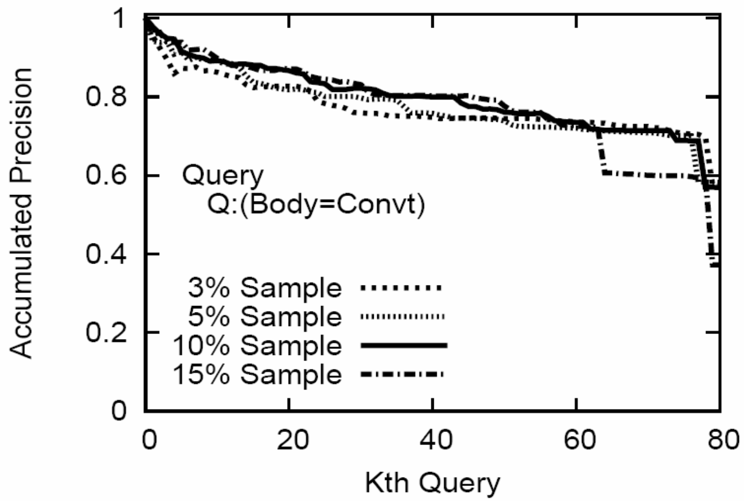
  Accuracy of AFD-Enhanced NBC is comparable with BayesNet



- Robustness w.r.t. Sample Size

  QPIAD is robust even when faced with a relatively small data sample

  Similar results were obtained on the Census database

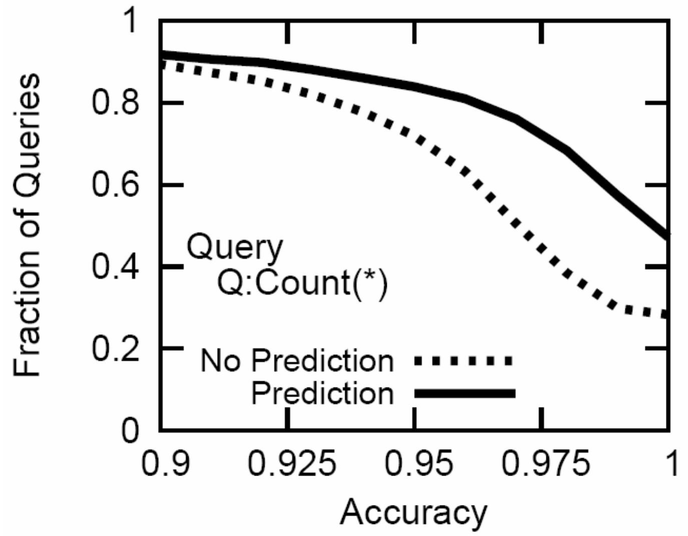


Experimental Results - Extensions

- Aggregates

  Prediction of missing values increases the fraction of queries that achieve higher levels of accuracy

  Approximately 20% more queries achieve 100% accuracy when prediction is used



- Joins

  As alpha is increased, we obtain a higher recall without sacrificing much precision