Unsupervised Transactional Query Classification Based on Webpage Form Understanding

Yuchen Liu
Computer Science Department
University of California, Los Angeles

Xiaochuan Ni, Jian-Tao Sun, Zheng Chen
Machine Learning Group
Microsoft Research Asia

Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Motivation

- Search Query Type Classification [Broder-2002]:
  - Informational Queries: (~39%) “canon 5d review”
  - Navigational Queries: (~25%) “myspace”
  - Transactional Queries: (~36%) “book Marriott hotel for my Seattle visit”

Problem Definition

- Transactional Query Classification: binary classification problem
  - Common transactions:
    - book flights
    - purchase movie tickets
    - find car dealers
  - Query Examples:
    - “book flight from Los Angeles to Glasgow”
    - “buy movie tickets for Lion King 3D”
    - “find dealers for Honda around Seattle”

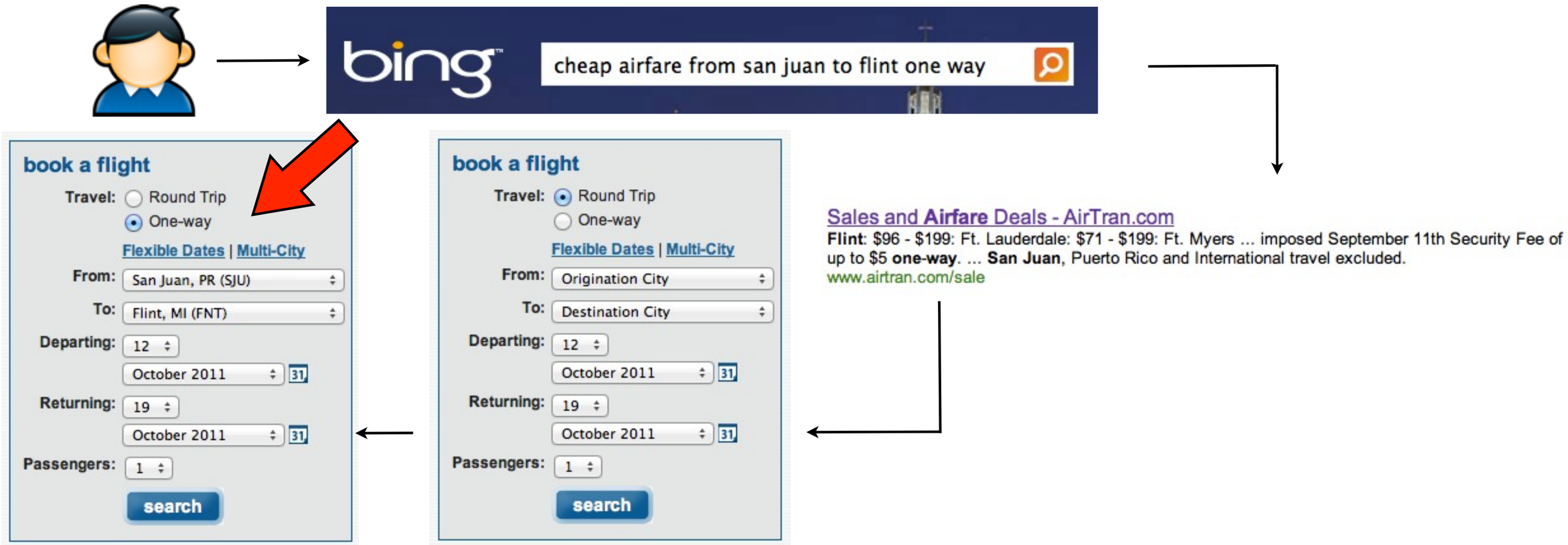
Previous Works

- Query Type Classification
  - Learning-based: focus on informational and navigational queries
    - Transactional queries not addressed well
    - Require cumbersome human labeling efforts
  - Rule-based: require human-compiled rules, not scalable

Observation

- What do users do after typing a transactional query into search box?

Observation (cont'd)



Want to simplify users' transaction by "auto-completion and delivery of forms"?

Our Focus

- Unsupervised Approach for Transactional Query Classification
  - Making use of form clicking signal for queries
  - Build classifier with the help of information in webpage forms
  - Reduce the human labeling effort for training classifier

Challenges: Identifying Transactional Queries

- Not all queries which are often associated with form clicking are transactional queries.
  - Some navigational queries: "american airlines"

  ➡ Ranking queries to estimate how transactional they are.

- Not all forms are served as an entrance to a transaction.
  - "Search box" issue

  ➡ Mutual reinforcement principle

Challenges: Building the Classifier

- How to deal with queries which we haven't seen in the log?

➡ Generate query patterns for further classification.

Pattern Generalization Example

cheap airfare from san juan to flint one way

cheap airfare from [from] to [to] [travel type]

[from]: Atlanta GA, Boston MA, ...
[to]: Allentown PA, Cancun Mexico, ...
[travel type]: round trip, one-way

Content

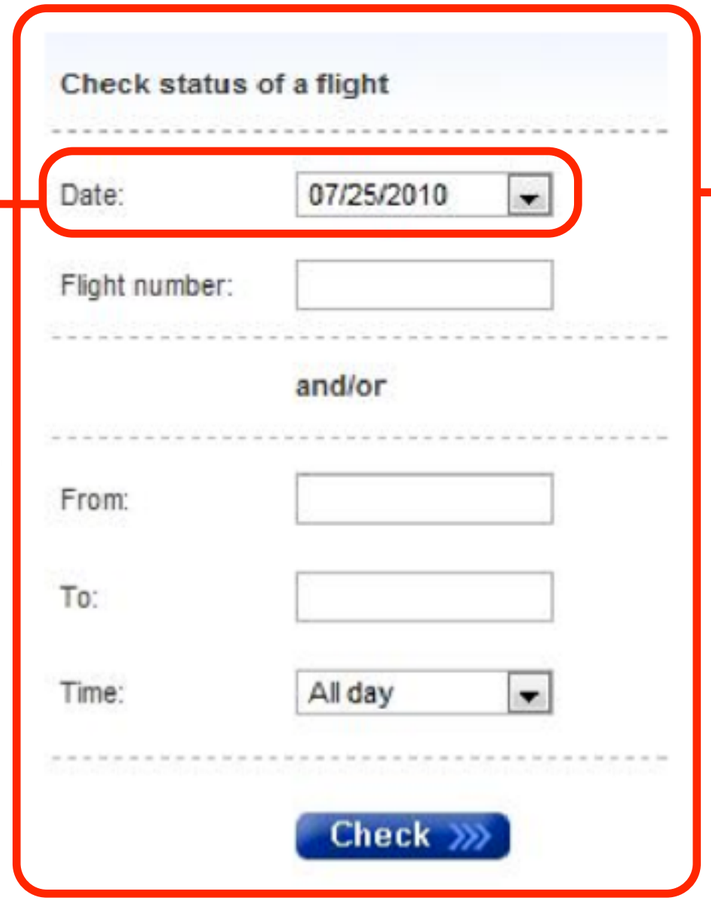
- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Webpage Form Modeling

- Everything between html tag “<form>” and </form>”.

Slot:
(label, type, values)

label: description of the slot
type: *text, select, radio, checkbox, button*
values: pre-set value list



Form:
(label, slots, from-url, to-url)

label: description of the form
slots: collection of all slots
from-url: the url of webpage containing this form
to-url: the url which users will visit after submission

Webpage Form Example

Form1

From-url: http://www.united.com/

To-url: http://www.ua2go.com/flifo/FlightSummary.do

Label: check status of a flight

Slot1 **Type**: Select **Label**: date **Values**: (07/25/2010, ...)	*Slot2* **Type**: Text **Label**: flight number **Values:** none
Slot3 (Text, from, none)	*Slot4* (Text, to, none)
Slot5 **Type**: Select **Label**: time **Values**: (all day, ...)	*Slot6* **Type**: Button **Label**: check **Values**: none

Associate Query with Form

- Toolbar Log
  - Collected by IE toolbar, recording users' behavior.
  - Format: {*userid, search query, clicked result page URL, further clicked page URL in result page*}
- Building **query-form** graph: bipartite graph
  - matching of {*from-url, to-url*} in form model with {*clicked result page URL, further clicked page URL*} in toolbar log

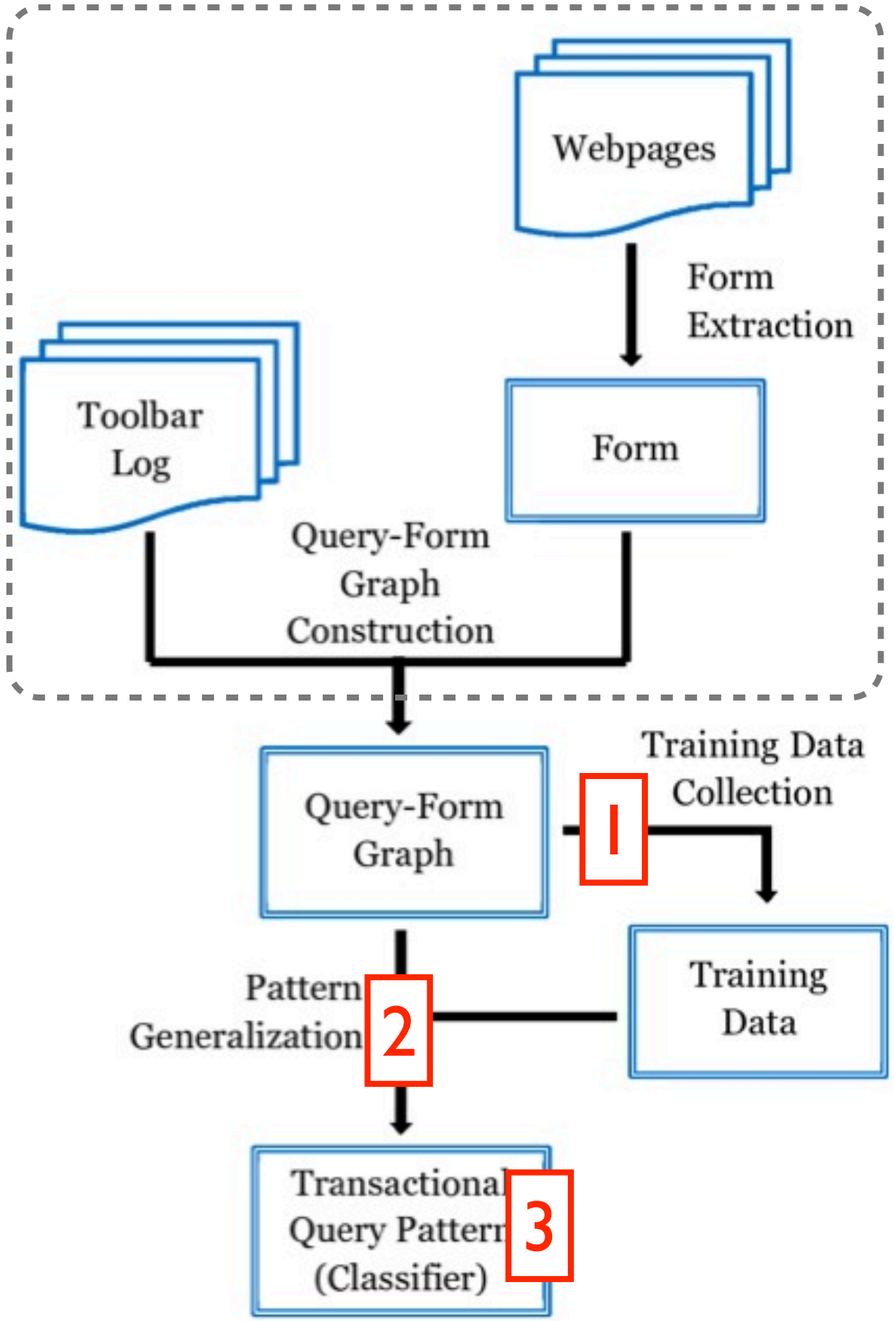
Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Framework

- Three Components:

1. Analyze query-form graph to get high-quality transactional queries as **training data**.
2. **Generalize query patterns** using training data with corresponding web forms.
3. **Predict query types** adopting query patterns for new queries.



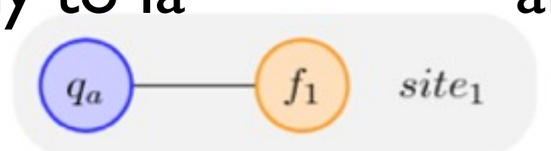
STEP1: Collecting Training Data

- Idea: ranking queries by how transactional they are.
  - two scores: click entropy & click ratio

STEP1: Collecting Training Data (cont'd)

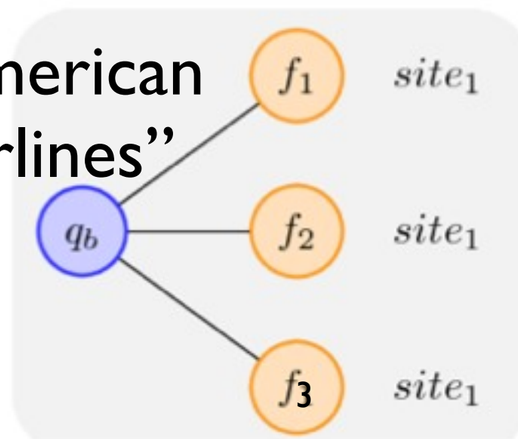
- Click Entropy
  - Consider click distribution over both websites and forms.

"flights from ny to la"



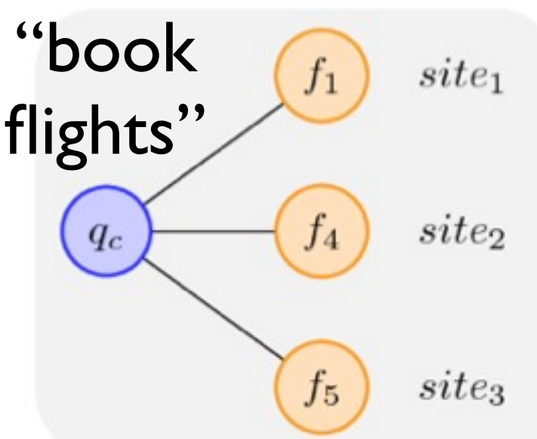
typical transactional

"american airlines"



navigational

"book flights"



good transactional

  - We prefer giving higher score to *qc*, then *qa*, than *qb*.

STEP1: Collecting Training Data (cont'd)

- Click Entropy

$$ClickEntropy(q) = (1 + E(p_{site})) \sum_{s_i \in Site(q)} \frac{Click(q, s_i)}{Click(q)} \frac{1}{2^{E(p_{s_i})}} \quad (1)$$

- E(p) is entropy function for distribution *p*.

STEP1: Collecting Training Data (cont'd)

- Click Ratio
  - how often do users click on forms after submitting this query?

$$ClickRatio(q) = \log_2 \left(\frac{Click(q)}{Impression(q)} + 1 \right) \qquad (3)$$

- Impression(q) is total appearance of *q* in log.

STEP1: Collecting Training Data (cont'd)

- Ranking queries

$$Score(q) = ClickEntropy(q_i) \times ClickRatio(q_i) \quad (4)$$

  - to estimate how likely a query is transactional.

STEP2: Query Pattern Generalization

- **Query pattern**: a sequence of terms (including at least one slot)
  - example: "used car [make] [model]"
- Use query-form pair to generalize the query into a query pattern
  - Input: query (q) & a list of slots from form (f)
  - Output: a query pattern if applicable

STEP2: Query Pattern Generalization (cont'd)

- Idea: if we find a value of a slot appears in the query, replace the value with the slot in the query to generalize a query pattern.
- Mis-spelling - fuzzy string match for value matching.
- Greedy algorithm for replacing values with slots.
- Use redundancy to rank query patterns by score:

$$ScoreQP(p) = |Slot(p)| \sum_{q_i \in Q(p)} Score(q_i) \qquad (6)$$

Pattern Generalization Example

cheap airfare from san juan to flint one way

cheap airfare from [from] to [to] [travel type]

[from]: Atlanta GA, Boston MA, ...
[to]: Allentown PA, Cancun Mexico, ...
[travel type]: round trip, one-way

STEP3: Query Classification

- Idea: Use query patterns to classify a new query
  - Calculate how a query (*p*) fits a query pattern (*p*)
    1. conduct pattern generalization algorithm to *q* with slots in pattern *p*
    2. calculate fit score by what they have left $Fit(p,q) = \frac{SlotMatch(p,q)}{|Slot(p)|} Cosine(p\prime, q\prime)$
  - Find the query pattern which fits the query best
  - Compute the transactional score for the query (q)

$$ScoreQ(q) = Fit(p_{fit}, q) \times ScoreQP(p_{fit}) \qquad (8)$$

Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Improving Training Data Quality

- "mutual reinforcement principle":
  - a transactional query is usually associated with transactional forms, a transactional form is usually associated with transactional queries.
- iterative algorithm for updating transactional score for both queries and forms

$$Score_{k+1}(q_i) = ClickEntropy(q_i) \times ClickRatio(q_i) \times \sum_{f_j \in F(q_i)} Click(q_i, f_j) Score_k(f_j) \quad (10)$$

$$Score_{k+1}(f_j) = \sum_{q_i \in Q(f_j)} Click(q_i, f_j) Score_k(q_i) \quad (11)$$

Improving Transactional Query Pattern Quality

- **quality of slots** (completeness of values, slots) affects pattern quality
- grouping forms with same functionality in different websites together could give us **more slots**, and **more values** in each slot.
- two-step clustering algorithm for forms

Improving Transactional Query Pattern Quality (cont'd)

- Two-step clustering algorithm for forms:

  1. clustering slots with same functionality together.

$$Sim(s_i, s_j) = \alpha LSim(s_i, s_j) + \beta VSim(s_i, s_j) + (1 - \alpha - \beta) CoSim(s_i, s_j) \quad (12)$$

  2. clustering forms using slot clustering result.

$$Sim(f_i, f_j) = \gamma LSim(f_i, f_j) + (1 - \gamma) SSim(f_i, f_j) \quad (13)$$

Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Experiments

- Experimental Settings
  - Toolbar Log: Dec 1 - 31, 2009.
    - 100 million log records, 30 million uni. queries, 7 million uni. users
  - Crawled forms from 3 categories “flight”, “hotel”, “car”.
    - our solution could apply for any other domain.
  - Query-form Graph: 155 forms with 20,934 queries.

Training Data Collection Evaluation

- Manually labeled 3000 queries pooled by both methods
- Top K Precision

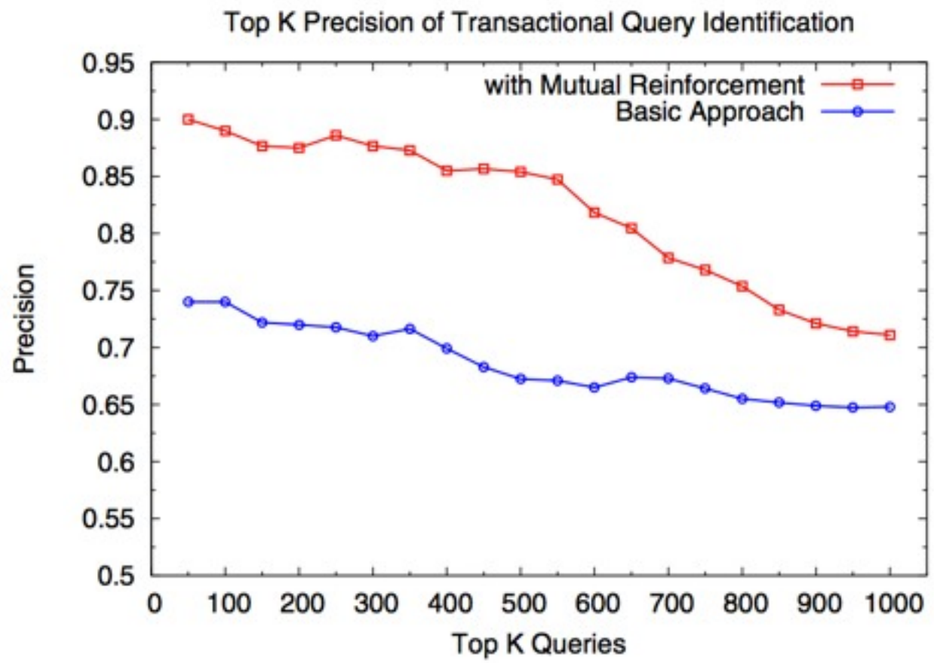


Figure 5: Top K Precision of Transactional Query Identification

- Convergence of Iterative Algorithm

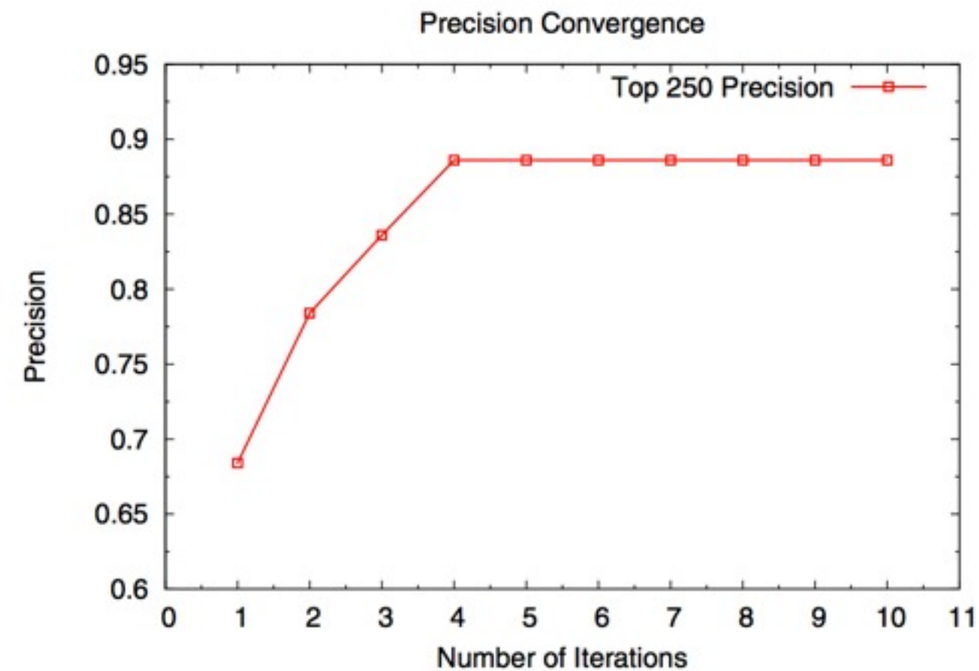
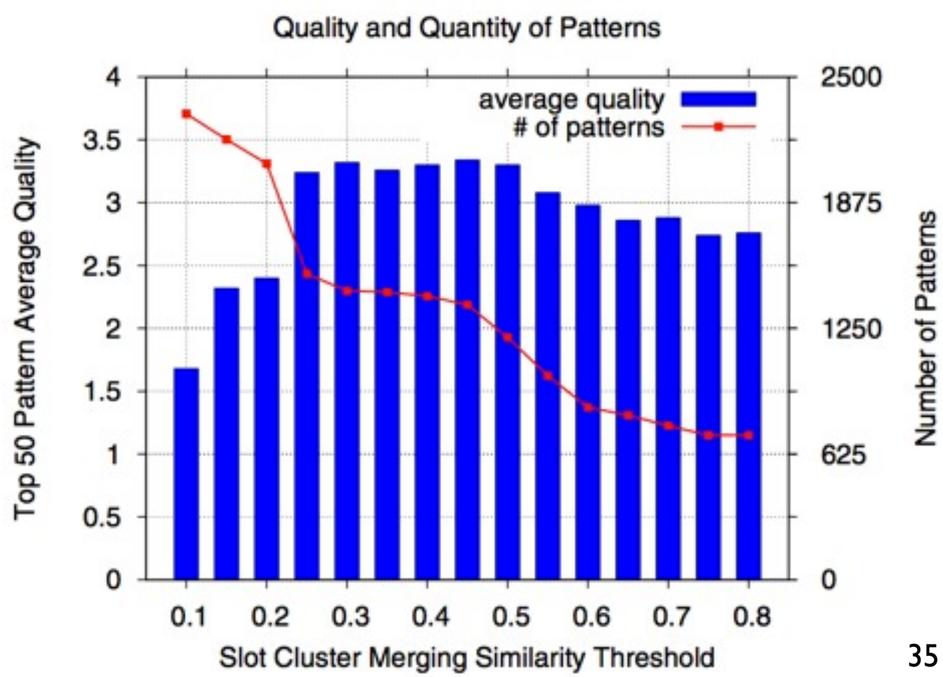


Figure 6: Top 250 Precision with Iteration Number

Query Pattern Generation Evaluation

- Manually labeled top 50 query patterns generated by each methods into 4 ratings.
  - How slot clustering threshold affects query pattern quality and quantity:



  - How form clustering threshold affects query pattern quality and quantity:

Query Pattern Generation Evaluation (cont'd)

- Comparison of different methods:

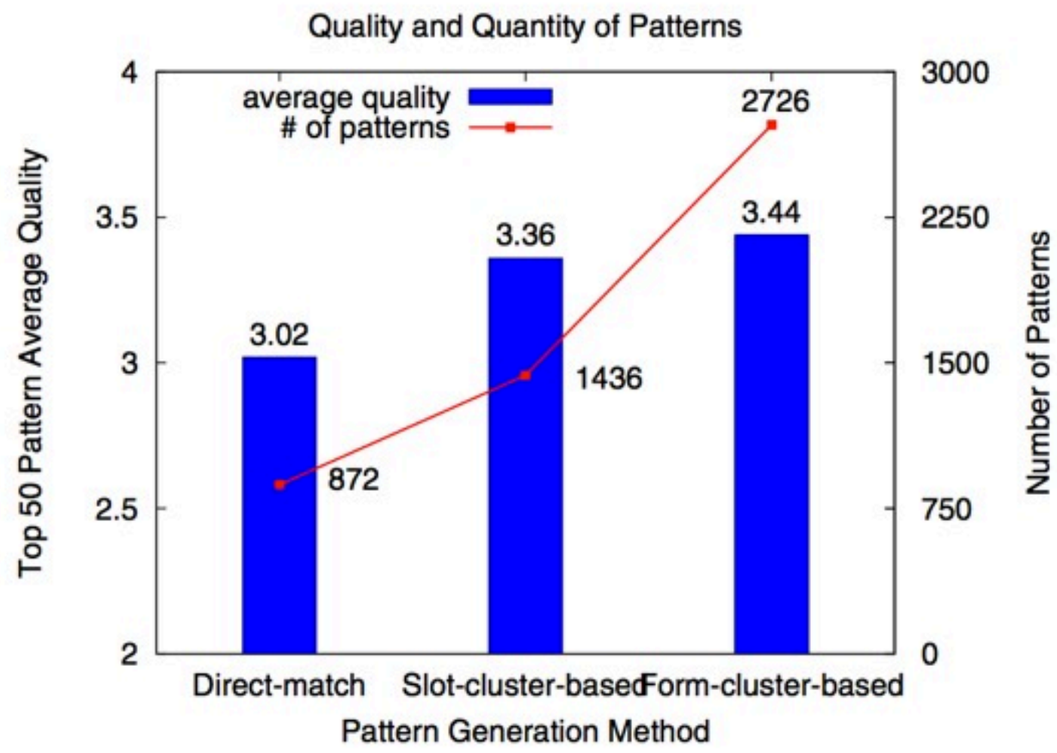


Figure 9: Comparison of Different Methods

Query Pattern Generation Evaluation (cont'd)

- Generalized query pattern examples:

Generalized Query Patterns with High Confidence Score		
Flight Domain	Car Domain	Hotel Domain
flights from [departure] to [destination]	[year] [make] [model]	[city] [state province] hotels
[departure] to [destination]	[make] [model] [year]	[city] hotels
flights to [destination]	[year] [make] [model] reviews	comfort inn [city] [state province]
cheap flights from [departure] to [destination]	[car type] [make] [model]	cheap hotels [state province] [city]
cheap flights to [destination]	[make] [model] for sale	[city] [state province] motels
flights [departure] to [destination]	[year] [make] [model] for sale	downtown [city] hotels
[departure] airport	car rental [pickup location]	[state province] quinta inn [city]
flights to [destination] from [departure]	reviews on [year] [make] [model]	omni hotel [city]
airlines that fly to [destination]	rent [state province] [car type]	hilton [city]
[maximum stops] flights to [destination]	[year] [make] [model] [body style]	hotels [state province] [city] [country]

Table 3: Query Patterns Generalized using Form Clusters

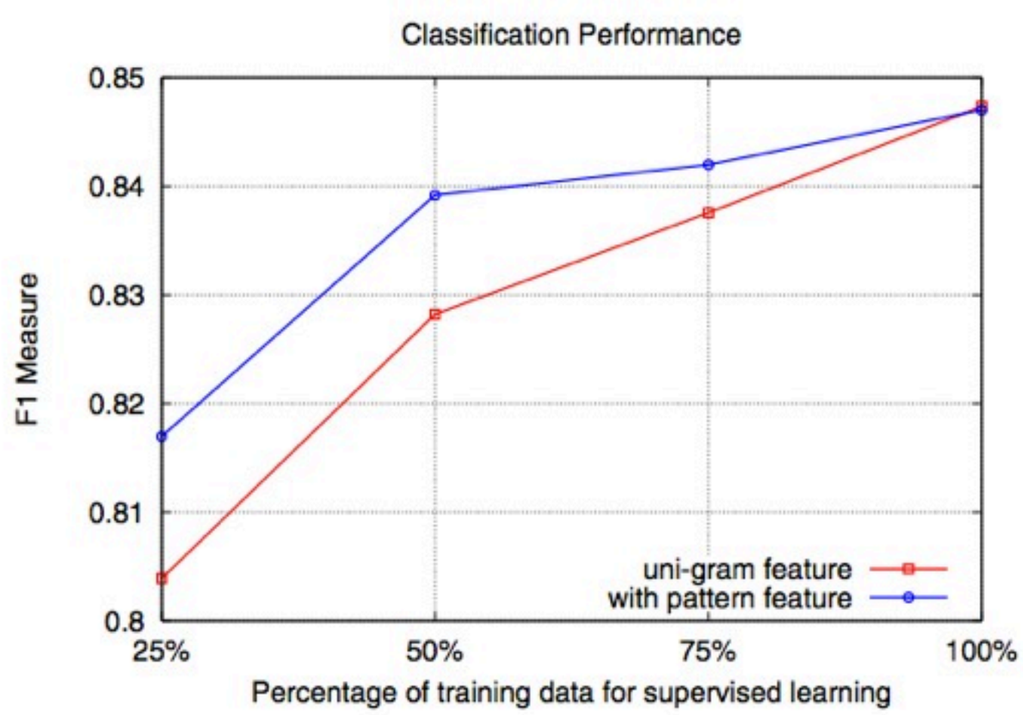
Transactional Query Prediction Evaluation

- We compare our method with a supervised learning approach
  - trained a SVM binary classifier with uni-gram features
- F1: 83% - comparable F-Measure with supervised learning approach
  - but without any human labeling process

	Precision	Recall	F_1
Pattern-based	0.832	0.830	0.830
Supervised-approach	0.864	0.848	0.847

Feature for Supervised Learning

- Using pattern-matching as a feature in supervised learning method
  - different percentage of training data for training process



Content

- Motivation and Problem Definition
- Data Explanation
- Solution
- More Discussion
- Experiments
- Conclusion and Future Works

Conclusion

- This unsupervised classification method could provide a comparable performance with simple supervised learning approach without any human labeling efforts.
- Pattern matching could provide help for supervised learning approach, especially when training dataset is small.
- Query patterns provide us a better understanding of how users formulate their transactional queries.

Future Works

- Optimize search engines' ranking for transactional queries by considering users' intent to conduct transactions on webpage forms.
- Directly provide users with the forms they might click for transactional queries instead of ten blue links.
- Auto form-filling based on a better structured understanding of users' transactional queries.

Thanks!

Any Questions?