Top-k Queries over Uncertain Scores

Qing Liu, Debabrota Basu, Talel Abdessalem, Stéphane Bressan
Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
Introduction

▶ Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.

▶ Crowdsourcing Platforms
  ▶ easily announce their needs to the crowd / get access to the information they need
  ▶ choose the highest quality / most competitively priced
Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
- Crowdsourcing Platforms
  - easily announce their needs to the crowd / get access to the information they need
  - choose the highest quality / most competitively priced
- Examples: TripAdvisor

“An extraordinary hotel with a wonderful atmosphere and a lovely team.”

5 stars Reviewed 1 week ago

From the first moment on you feel very comfortable and heartily welcome in that familiar and elegant hotel. The team is very friendly and attentive. The location is perfect for a city trip to relax after an exciting day or night in London. Although the hotel is in the heart of London it is located in a very quiet and...
Introduction

- Modern recommendation systems leverage some forms of collaborative user (crowd) sourced collection of information.
- Crowdsourcing Platforms
  - easily announce their needs to the crowd / get access to the information they need
  - choose the highest quality / most competitively priced
- Examples: TripAdvisor
  - collaborative user or crowdsourced collection of information, e.g., user generated ratings and reviews, to recommend travel plans and hotels, vacation rentals and restaurants.
Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services
Introduction

- Crowdsourcing and Collaborative Economy:
  - communities or crowds rent, share, sell products or services

Spinlister helps active people connect with trusted bike owners around the world
Crowdsourcing and Collaborative Economy:
- communities or crowds rent, share, sell products or services
Introduction

Crowdsourcing and Collaborative Economy:

- communities or crowds rent, share, sell products or services
Introduction

- Independent collection of information $\rightarrow$ uncertainty and diversity.
Introduction

- Independent collection of information $\rightarrow$ uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).
Introduction

- Independent collection of information $\rightarrow$ uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).
Introduction

- Independent collection of information → uncertainty and diversity.
- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).
Introduction

- Independent collection of information → uncertainty and diversity.

- Objects (services, vacation rentals and restaurants...) have uncertain scores (quality, price...).
Introduction

- Ranking is one of the building blocks of recommendation.
- A **top-k query** returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.
Introduction

- Ranking is one of the building blocks of recommendation.
- A top-k query returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.
- Price of the apartments.
Introduction

- Ranking is one of the building blocks of recommendation.
- A **top-k query** returns the sequence of the k objects with the highest scores, given a database of objects ranked by their scores for the feature of interest.

Price of the apartments.

- With uncertain scores, a top-k query can only return an uncertain result.
Related Work

- Soliman, Hyas and Ben-David [Soliman and Ilyas, 2009] study top-k queries over objects with uncertain scores given as probability distributions.
Soliman, Hyas and Ben-David [Soliman and Ilyas, 2009] study top-$k$ queries over objects with uncertain scores given as probability distributions.

In this paper, we consider probabilistic top-$k$ queries under the top-k semantics as in [Soliman and Ilyas, 2009].
Problem Definition

- **O**: a set of \( n \) objects;
Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
Problem Definition

- \( O \): a set of \( n \) objects;
- \( s(o_i) \): the score of an object \( o_i \in O \);
- \( X_i \): a random variable, equals to \( s(o_i) \);
Problem Definition

- $O$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in O$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi(k) = [o_1, \cdots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;
Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi^{(k)} = [o_1, \cdots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;
- $Pr(\pi^{(k)})$: probability of $\pi^{(k)}$ be the top-$k$ sequence;

$$Pr(\pi^{(k)}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_1(x_1) \cdots f_n(x_n) \, dx_n \cdots dx_1$$  \hspace{1cm} (1)
Problem Definition

- $\mathcal{O}$: a set of $n$ objects;
- $s(o_i)$: the score of an object $o_i \in \mathcal{O}$;
- $X_i$: a random variable, equals to $s(o_i)$;
- $f_i$: bounded continuous probability density function of $X_i$;
- $\pi^{(k)} = [o_1, \ldots, o_k]$: sequence of $k$ objects in $\mathcal{O}$;
- $Pr(\pi^{(k)})$: probability of $\pi^{(k)}$ be the top-$k$ sequence;

$$Pr(\pi^{(k)}) = \int_{-\infty}^{\infty} \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_k} f_1(x_1) \cdots f_n(x_n) \, dx_n \cdots dx_1$$

- (Objective:) **Probabilistic top-$k$ sequence**: the $\pi^{(k)}$ that maximizes $Pr(\pi^{(k)})$. 

(page 7 of 19)
Solutions

- Naive: calculate $Pr(\pi(k))$ for every possible sequence $\pi(k)$ and returning the $\pi(k)$ with the highest $Pr(\pi(k))$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.
Solutions

- **Naive**: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.

- **Branch-and-Bound** [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.
Solutions

- **Naive**: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.

- **Branch-and-Bound** [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.

- **Soliman’s Algorithm** [Soliman et al., 2010]: searches the space of candidate probabilistic top-$k$ sequences using a Markov chain Monte Carlo algorithm.
Solutions

- **Naive**: calculate $Pr(\pi^{(k)})$ for every possible sequence $\pi^{(k)}$ and returning the $\pi^{(k)}$ with the highest $Pr(\pi^{(k)})$.
  - $\frac{n!}{(n-k)!}$ possible sequences to examine.

- **Branch-and-Bound** [Soliman et al., 2010]: Prune some $\pi^{(k)}$s.
  - Worst case: $\frac{n!}{(n-k)!}$ possible sequences to examine.

- **Soliman’s Algorithm** [Soliman et al., 2010]: searches the space of candidate probabilistic top-$k$ sequences using a Markov chain Monte Carlo algorithm.

- In this paper, we explore the variants of Markov chain Monte Carlo algorithms.
Markov chain Monte Carlo Algorithms

- Soliman’s Algorithm
  - Initial state: a rank over the $n$ objects

Acceptance Probability: $\alpha = \min\left(\frac{\Pr(\pi(t+1)_{k}) \cdot \Pr(\pi(t+1)|\pi(t))}{\Pr(\pi(t)_{k}) \cdot \Pr(\pi(t)|\pi(t+1))}, 1\right)$
Markov chain Monte Carlo Algorithms

Soliman's Algorithm

- Initial state: a rank over the $n$ objects
- Candidate State:

\[
\begin{align*}
&\text{Initial State: } o_1, o_2, o_3, o_4, o_5, o_6, o_7 \\
&\text{Candidate State: } o_1, o_2, o_3, o_4, o_5, o_6, o_7 \\
&\text{Acceptance Probability: } \alpha = \min \left( \frac{\Pr(\pi_k(t+1) \cdot \Pr(\pi_t|\pi_{t+1}))}{\Pr(\pi_k(t)) \cdot \Pr(\pi_{t+1}|\pi_t)} \right), 1 \right)
\end{align*}
\]
Markov chain Monte Carlo Algorithms

- Soliman’s Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:
    - Acceptance Probability: $\alpha = \min\left(\frac{Pr(\pi^{(k)}_{t+1}) \cdot Pr(\pi_t | \pi_{t+1})}{Pr(\pi^{(k)}_t) \cdot Pr(\pi_{t+1} | \pi_t)}, 1\right)$
Markov chain Monte Carlo Algorithms

- Swap and SwapEXP Algorithm
  - Initial state: a rank over the \( n \) objects
Markov chain Monte Carlo Algorithms

- **Swap and SwapEXP Algorithm**
  - Initial state: a rank over the $n$ objects
  - Candidate State:

  ![Diagram showing the initial and candidate states]

  - **Acceptance Probability:**
    - **Swap:**
      $$\alpha = \min\left(\frac{\Pr(\pi(k)^{t+1})}{\Pr(\pi(k)^{t})} \cdot 1\right)$$
    - **SwapEXP:**
      $$\alpha = \min\left(\hat{\Pr}(\pi(k)^{t+1}) \div \hat{\Pr}(\pi(k)^{t}) = \exp(\beta(\Pr(\pi(k)^{t+1}) - \Pr(\pi(k)^{t})))\right)$$

  - **SwapEXP** is more likely to reject the "worse" candidate state.
Markov chain Monte Carlo Algorithms

- **Swap and SwapEXP Algorithm**
  - Initial state: a rank over the $n$ objects
  - Candidate State:

- **Acceptance Probability:**
  
  **Swap:**
  \[ \alpha = \min\left( \frac{Pr(\pi_{t+1}^{(k)}) \cdot \frac{1}{kn}}{Pr(\pi_t^{(k)}) \cdot \frac{1}{kn}} = \frac{Pr(\pi_{t+1}^{(k)})}{Pr(\pi_t^{(k)})} , 1 \right) \]

  **SwapEXP:**
  \[ \alpha = \min\left( \frac{\hat{Pr}(\pi_{t+1}^{(k)})}{\hat{Pr}(\pi_t^{(k)})} = \exp(\beta(Pr(\pi_{t+1}^{(k)}) - Pr(\pi_t^{(k)}))), 1 \right) \]
  \[ (\hat{Pr}(\pi^{(k)}) = C^{-1}_\beta \exp(\beta Pr(\pi^{(k)}))) \]
Markov chain Monte Carlo Algorithms

- **Swap and SwapEXP Algorithm**
  - **Initial state:** a rank over the $n$ objects
  - **Candidate State:**

- **Acceptance Probability:**
  
  **Swap:**
  $$\alpha = \min\left( \frac{Pr(\pi_{t+1}^{(k)}) \cdot \frac{1}{kn}}{Pr(\pi_t^{(k)})}, 1 \right)$$

  **SwapEXP:**
  $$\alpha = \min\left( \frac{\hat{Pr}(\pi_{t+1}^{(k)})}{\hat{Pr}(\pi_t^{(k)})} = \exp(\beta(Pr(\pi_{t+1}^{(k)}) - Pr(\pi_t^{(k)}))), 1 \right)$$

  $$(\hat{Pr}(\pi^{(k)}) = C_\beta^{-1} \exp(\beta Pr(\pi^{(k)})))$$

  **SwapEXP** is more likely to reject the “worse” candidate state.
Markov chain Monte Carlo Algorithms

- ReSample and ReSampleEXP Algorithm
  - Initial state: a rank over the $n$ objects
Markov chain Monte Carlo Algorithms

- **ReSample and ReSampleEXP Algorithm**
  - Initial state: a rank over the \( n \) objects
  - Candidate State:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( o_1 ): 9</td>
</tr>
<tr>
<td>2</td>
<td>( o_2 ): 8</td>
</tr>
<tr>
<td>3</td>
<td>( o_3 ): 6</td>
</tr>
<tr>
<td>4</td>
<td>( o_4 ): 5</td>
</tr>
<tr>
<td>5</td>
<td>( o_5 ): 4</td>
</tr>
<tr>
<td>6</td>
<td>( o_6 ): 3</td>
</tr>
<tr>
<td>7</td>
<td>( o_7 ): 2</td>
</tr>
</tbody>
</table>

Acceptance Probability:

- **ReSample**:
  \[
  \alpha = \min \left( \frac{\Pr(\pi(k)_{t+1}) \cdot \Pr(\pi_{t+1} | \pi_t)}{\Pr(\pi_{t+1}) \cdot \Pr(\pi_t | \pi_{t+1})} \right)
  \]

- **ReSampleEXP**:
  \[
  \alpha = \min \left( \frac{\Pr(\pi_{t+1} | \pi_t) \cdot \exp(\beta(\Pr(\pi(k)_{t+1}) - \Pr(\pi(k)_t)))}{\Pr(\pi_{t+1}) \cdot \Pr(\pi_t | \pi_{t+1})} \right)
  \]
Markov chain Monte Carlo Algorithms

- ReSample and ReSampleEXP Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:

- Acceptance Probability:
  
  ReSample: $\alpha = \min\left( \frac{Pr(\pi^{(k)}_{t+1}) \cdot Pr(\pi_t|\pi_{t+1})}{Pr(\pi^{(k)}_t) \cdot Pr(\pi_{t+1}|\pi_t)} , 1 \right)$
  
  ReSampleEXP:
  
  $\alpha = \min\left( \frac{Pr(\pi_t|\pi_{t+1})}{Pr(\pi^{(k)}_{t+1}|\pi_t)} \cdot \exp(\beta(Pr(\pi^{(k)}_{t+1}) - Pr(\pi^{(k)}_t))) , 1 \right)$. 
Markov chain Monte Carlo Algorithms

- ReSampleAll Algorithm
  - Initial state: a rank over the $n$ objects
Markov chain Monte Carlo Algorithms

- ReSampleAll Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:

<table>
<thead>
<tr>
<th>$o_1$</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_2$</td>
<td>8</td>
</tr>
<tr>
<td>$o_3$</td>
<td>6</td>
</tr>
<tr>
<td>$o_4$</td>
<td>5</td>
</tr>
<tr>
<td>$o_5$</td>
<td>4</td>
</tr>
<tr>
<td>$o_6$</td>
<td>3</td>
</tr>
<tr>
<td>$o_7$</td>
<td>2</td>
</tr>
</tbody>
</table>

- Top-$k$:
  - $o_3$: 10
  - $o_2$: 9
  - $o_5$: 8
  - $o_1$: 6
  - $o_7$: 4
  - $o_6$: 3
  - $o_4$: 2

Acceptance Probability: ReSample $\alpha = 1$
Markov chain Monte Carlo Algorithms

- ReSampleAll Algorithm
  - Initial state: a rank over the $n$ objects
  - Candidate State:

- Acceptance Probability: ReSample: $\alpha = 1$
Performance Evaluation

- Datasets: synthetic datasets

**Table: Distributions**

<table>
<thead>
<tr>
<th></th>
<th>Setting 1</th>
<th>Setting 2</th>
<th>Setting 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>median score</td>
<td>( G(0.5, 0.05) )</td>
<td>( G(0.5, 0.2) )</td>
<td>( U[0, 1] )</td>
</tr>
<tr>
<td>width</td>
<td>( G(0.5, 0.05) )</td>
<td>( G(0.5, 0.2) )</td>
<td>( U[0, 1] )</td>
</tr>
</tbody>
</table>
Performance Evaluation

- Datasets: synthetic datasets

<table>
<thead>
<tr>
<th>Table: Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting 1</td>
</tr>
<tr>
<td>median score</td>
</tr>
<tr>
<td>width</td>
</tr>
</tbody>
</table>

- default: uniform score distributions, median score of $o_i$: $\frac{l_i + u_i}{2}$, width: $u_i - l_i$
Performance Evaluation

- Datasets: synthetic datasets

<table>
<thead>
<tr>
<th></th>
<th>Setting 1</th>
<th>Setting 2</th>
<th>Setting 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>median score</td>
<td>$G(0.5, 0.05)$</td>
<td>$G(0.5, 0.2)$</td>
<td>$U[0, 1]$</td>
</tr>
<tr>
<td>width</td>
<td>$G(0.5, 0.05)$</td>
<td>$G(0.5, 0.2)$</td>
<td>$U[0, 1]$</td>
</tr>
</tbody>
</table>

- default: uniform score distributions, median score of $o_i$: $\frac{l_i + u_i}{2}$, width: $u_i - l_i$

- Metrics
Performance Evaluation

- Datasets: synthetic datasets

<table>
<thead>
<tr>
<th>Table: Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting 1</td>
</tr>
<tr>
<td>median score</td>
</tr>
<tr>
<td>width</td>
</tr>
</tbody>
</table>

- default: uniform score distributions, median score of $o_i$: $\frac{l_i+u_i}{2}$, width: $u_i - l_i$

- Metrics
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)
Performance Evaluation

- Datasets: synthetic datasets

<table>
<thead>
<tr>
<th>Table: Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting 1</td>
</tr>
<tr>
<td>median score</td>
</tr>
<tr>
<td>width</td>
</tr>
</tbody>
</table>

- default: uniform score distributions, median score of $o_i$: $\frac{l_i + u_i}{2}$, width: $u_i - l_i$

- Metrics
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)
  - Convergence of the Markov chains (Gelman-Rubin Convergence Diagnostic)
Performance Evaluation

- **Datasets:** synthetic datasets

<table>
<thead>
<tr>
<th>Table: Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting 1</td>
</tr>
<tr>
<td>median score</td>
</tr>
<tr>
<td>width</td>
</tr>
</tbody>
</table>

- **default:** uniform score distributions, median score of $o_i$: $\frac{l_i + u_i}{2}$, width: $u_i - l_i$

- **Metrics**
  - Probability of the Probabilistic top-$k$ sequence (higher $\rightarrow$ better)
  - Convergence of the Markov chains (Gelman-Rubin Convergence Diagnostic)
  - Efficiency (Complexity and runtime)
Effectiveness of Six Algorithms (Probability)

(a) Dataset5

(b) Dataset21

(c) Dataset5

(d) Dataset21
Convergence of the Markov Chains

(e) Dataset5

(f) Dataset21

(g) Dataset5

(h) Dataset21
Table: Worst Case Time Complexity of Generating Next State

<table>
<thead>
<tr>
<th>Time Complexity</th>
<th>Soliman</th>
<th>Swap(EXP)</th>
<th>ReSample(EXP)</th>
<th>ReSampleAll</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(nk)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(n\log k)$</td>
<td></td>
</tr>
</tbody>
</table>

Table: Runtime Per Step of the Algorithms (seconds)

<table>
<thead>
<tr>
<th>Runtime Per Step</th>
<th>Soliman</th>
<th>Swap</th>
<th>SwapEXP</th>
<th>ReSample</th>
<th>ReSampleEXP</th>
<th>ReSampleAll</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0058</td>
<td>1.9128</td>
<td>0.1163</td>
<td>0.0523</td>
<td>0.0071</td>
<td>0.9056</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

▶ We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.
Conclusion

- We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.
- We verify through extensive experiments that the proposed algorithms are more effective than the state of the art approach.
Conclusion

- We explore the design space for Metropolis-Hastings Markov chain Monte Carlo algorithms.
- We verify through extensive experiments that the proposed algorithms are more effective than the state of the art approach.
- ReSampleAll is the best, since it samples directly from the target distribution instead of depending on “local” information.
Thank you! Questions?
Top-k Queries
Uncertain Scores
MCMC
liuqing@u.nus.edu
References I
