Deep Learning Methods for Query Auto Completion

https://aka.ms/dl4qac

Manish Gupta, Puneet Agrawal

{gmanish, punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
AutoSuggest Examples

Prefix

Block

Conversation

DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Conceptual difficulty of the AS problem

Intention gap is very high in AS
- Guessing user intent with very short prefixes.
- Guessing user language using very short prefix.
- Guessing incorrectly can lead to defects/misspellings/inappropriate suggestions, freshness/local tail intent problems.

Goal:
- Suggest the user’s intended query after minimal input keystrokes
- Rank the user’s intended query highly in completion suggestions
Important Components in a QAC system

• Ranking suggestions
  • Most popular completion
  • Time sensitive suggestions
  • Location sensitive suggestions
  • Personalization

• Ghosting, Session co-occurrences

• Online spell correction, Defect handling

• Non-prefix matches, Generating suggestions

• Mobile QAC, Enterprise QAC
Ranking suggestions: Most Popular Completion (MPC)

• “Wisdom of the crowds” MPC solution
  • A trie indexes historical queries along with popularity values.
  • Candidates=suggestions from trie that match the prefix.
  • Rank candidates by a function of its past popularity

• Language specific popularity

• Region specific popularity

• Vary the query-log aggregation period
  • For shorter prefix lengths, a shorter query-log aggregation period is optimal, and vice-versa [Whiting 2013]

• Can also rank by clicks
  • But click data is sparse


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Ranking suggestions: Time-sensitive suggestions

• Predictably vs unpredictably popular
  • Predictably popular queries: temporally recurring (e.g. at Christmas, in January, etc.) or known/foreseeable events and phenomena (e.g. TV episodes, sporting events, expected weather etc.).
    • Ranking of candidates must be adjusted with time. “halloween” might be the right suggestion after typing “ha” in October, “harry potter” might be better any other time.
  • Unpredictably popular queries: unforeseeable current events and phenomena (e.g. breaking news).
    • “sarah burke” that gained high popularity in Jan 2012, but was not queried as often in the past, might get lower ranking if compared to “sarah palin”, which has high volume, since it was queried for many years, despite being relatively less popular in Jan 2012.

• Instead of past popularity, can we rank candidates based on forecasted frequencies?
  • Can use typical time series forecasting methods like ARIMA, exponential smoothing, ... [Shokouhi, 2012]
  • Model as a ranked Multi-armed Bandit problem [Wang, 2017]
Ranking suggestions: Location sensitive suggestions

- A user in San Diego types “Uni”.
  - “Univ of California, San Diego” is ok.
  - “Univ of California, Los Angeles” is not ok at the top.

- Location sensitivity of queries
  - Local interest queries [Backstrom, 2008]
    - Queries only interested by users at particular location
      - e.g., name of local high school, newspaper
    - Find center of geographic focus for query
    - Determine if query is tightly concentrated or spread diffusely geographically
      - Given query, what is center and dispersion?
  - Localizable queries
    - Users at different locations may issue the same query, but referring to different things
      - e.g., pizza hut, house for rent.
    - A localizable query is likely to appear as a sub query in other queries, associating with different locations. “car rental california”, “car rental new york”, etc


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Ranking Suggestions: Personalization

Using short-term/long-term user history, location, other signals

Search/Click History:
- tender is the night
- f. scott fitzgerald
- this side of paradise
- the silent patient
- the great gatsby book

Search/Click History:
- the wolf of wall street
- the revenant
- inception
- gangs of new york
- pain & gain

Search/Click History:
- the wolf of wall street
- the revenant
- inception
- gangs of new york
- pain & gain

Search/Click History:
- beijing travel advisory
- forbidden city

Session history
- the great wall of china
- the great gatsby film 2013
- the great gatsby trailer
- the great gatsby film 1974
- the greatest showman

{gmanish,punagr}@microsoft.com
Ghosting, Session co-occurrences

- Ghosting: auto-completing a search recommendation by highlighting the suggested text inline i.e., within the search box.
- Session-context ghosting increased the acceptance of offered suggestions by 6.18% and reduced misspelled searches by 4.42% [Ramachandran et al, 2019]

![Figure 1: Default QAC experience (left) and QAC with ghosting (right) for prefix "wireless bl"

<table>
<thead>
<tr>
<th>wireless bl</th>
<th>wireless bluetooth headphones</th>
</tr>
</thead>
<tbody>
<tr>
<td>speakers wireless bluetooth</td>
<td>wireless bluetooth earbuds</td>
</tr>
<tr>
<td>wireless bluetooth earbuds</td>
<td>wireless bluetooth speakers portable</td>
</tr>
<tr>
<td>wireless bluetooth mouse</td>
<td>wireless bluetooth car kit</td>
</tr>
<tr>
<td>wireless Bluetooth head set</td>
<td>wireless bluetooth microphone</td>
</tr>
<tr>
<td>wireless Bluetooth earbuds for iphone</td>
<td>wireless bluetooth keyboard</td>
</tr>
</tbody>
</table>

- Context sensitive AS [Bar-Yossef et al. 2011]
  - If after the query “richard nixon” the most popular successive query starting with “am” is “american presidents”, the search engine will suggest “american presidents” as its top completion.
  - Based on existence of reoccurring query sequences in search logs.
  - Handle sparsity of co-occurrences
    - clustering similar query sequences together
    - similarity may be syntactic (e.g., american airlines → american airlines flight status) or only semantic (e.g., american airlines → continental).

Small portions of the prefix can be corrected at trie exploration time paying a penalty cost. E.g. “cbo” → “ceboo”

- More flexible than Offline Speller because small portions of the prefix can be changed
- More coverage
- Key idea: it is possible to jump to a different node in the search trie paying a cost dictated from the Conversion table

Defects
- Spelling mistakes
- Offensive suggestions
- Partial suggestions
- Rare intents
- Non-sensical suggestions/hallucinations
- Gibberish
- Bad URL


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Non-prefix matches, Generating suggestions

- Non-prefix matches
  - If Q is “shrimp dip rec”, then a plausible completion found by prefix-search could be “shrimp dip recipes”.
  - A multiterm prefix-search could return, instead, “shrimp bienville dip recipe” or “recipe for appetizer shrimp chipolte dip”.
  - Use inverted index.

- A significant proportion of queries issued daily have never been seen previously.

- Generate suggestions
  - Improves recall in tail.
  - Deep learning NLG
  - FST: Finite state transducers
  - N-gram models

- Issues
  - Latency
  - Partial suggestions, offensive suggestions, hallucinations, grammatically-incorrect suggestions
  - Personalization


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Mobile QAC, Enterprise QAC

Figure 1: A commercial mobile QAC. The *Real Madrid* app is installed and recently opened. Given prefix “real”, popular queries on real estate (“real estate” and “realtor.com”) are suggested at higher positions than query “real madrid”.

Figure 2: Suggestions from the extended query auto-completion system at www.bankwest.com.au on 26 Aug 2013.

Zhang, Aston, Amit Goyal, Ricardo Baeza-Yates, Yi Chang, Jiawei Han, Carl A. Gunter, and Hongbo Deng. "Towards mobile query auto-completion: An efficient mobile application-aware approach." In WWW, pp. 579-590. 2016.

Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
  • Traditional Machine Learning methods for ranking suggestions
  • Convolutional Latent Semantic Model
  • LSTM encoder
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
  • Traditional Machine Learning methods for ranking suggestions
  • Convolutional Latent Semantic Model
  • LSTM encoder
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Prefix and Pairwise Features

- **Prefix features**
  - Does it end with a space character
  - Prefix length

- **Suggestion features**
  - Suggestion length (characters and words)
  - Frequency in the background set.
    - Time scales: 1/2/3/4 weeks, 1 month, 1 year.
  - Is it a navigational query?
  - Overall impressions of homologous queries: (1) queries with the same terms as the candidate query but in a different order and (2) queries that extend the candidate query.
  - Number of queries with this suggestion as prefix.

---


Suggestion and Contextual Features

• Pairwise features (using anchor query from session data)
  • For each candidate suggestion, count how many times it follows the anchor query in the background data.
  • Frequency of the anchor query in the background data.
  • Levenshtein distance between the anchor and the suggestion.

• Contextual Features
  • 10 features corresponding to the character n-gram similarity between the suggestion and the 10 most recent queries in the context.
  • Average Levenshtein distance between the suggestion and each query in the context.
  • Scores estimated using the context-aware Query Variable Markov Model (QVMM). QVMM models the context with a variable memory Markov model able to automatically back-off shorter query n-grams if the exact context is not found in the background data.


Reformulation Features

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature Class</th>
<th>Description</th>
<th>Formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>Term Combination</td>
<td>(16 features)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number of terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>term keeping</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>term adding</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>term removing</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>number of used terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>ratio of used terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>number of repeat times</td>
<td></td>
<td>$\text{Rep}(q_T)$, $\text{Rep}(q_T)/T$, $\text{Rep}(q_T)/</td>
</tr>
<tr>
<td>Query</td>
<td>Query Similarity</td>
<td>(10 features)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cosine similarity</td>
<td></td>
<td>$\text{sim}<em>{\text{cos}}(q</em>{T-1}, q_T)$</td>
</tr>
<tr>
<td></td>
<td>average cosine similarity</td>
<td></td>
<td>$\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{cos}}(q_i, q</em>{i+1})$, $\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{cos}}(q_i, q</em>{T})$</td>
</tr>
<tr>
<td></td>
<td>trends of cosine similarity</td>
<td></td>
<td>$\text{sim}<em>{\text{lev}}(q</em>{T-1}, q_T)$</td>
</tr>
<tr>
<td></td>
<td>Lev. similarity</td>
<td></td>
<td>$\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{lev}}(q_i, q</em>{i+1})$, $\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{lev}}(q_i, q</em>{T})$</td>
</tr>
<tr>
<td></td>
<td>average Lev. similarity</td>
<td></td>
<td>$\text{sim}<em>{\text{lev}}(q</em>{T-1}, q_T)$</td>
</tr>
<tr>
<td></td>
<td>trends of Lev. similarity</td>
<td></td>
<td>$\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{lev}}(q_i, q</em>{i+1})$, $\frac{1}{T-1} \sum_{i=1}^{T-1} \text{sim}<em>{\text{lev}}(q_i, q</em>{T})$</td>
</tr>
<tr>
<td>Query</td>
<td>Query Length</td>
<td>(6 features)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number of terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>average number of terms</td>
<td></td>
<td>$\frac{1}{T-1} \sum_{i=1}^{T-1}</td>
</tr>
<tr>
<td></td>
<td>trends of term number</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>Query Frequency</td>
<td>(2 features)</td>
<td>pairwise frequency</td>
<td>$P((q_{T-1}, q_T)</td>
</tr>
<tr>
<td>Click-through Data</td>
<td>(6 features)</td>
<td>previous clicks</td>
<td>$c_{T-1}$, $\text{sgn}(c_{T-1})$</td>
</tr>
<tr>
<td></td>
<td>number of effective terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>ratio of effective terms</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>Session</td>
<td>Time Duration</td>
<td>(2 features)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>average time duration</td>
<td></td>
<td>$\frac{1}{T-2} \sum_{i=1}^{T-2} (t_{i+1} - t_i)$</td>
</tr>
<tr>
<td></td>
<td>trends of time duration</td>
<td></td>
<td>$\frac{1}{T-2} \sum_{i=1}^{T-2} (t_{i+1} - t_i)$</td>
</tr>
<tr>
<td></td>
<td>Position Number</td>
<td>(1 feature)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>position in the session</td>
<td></td>
<td>$(T)$</td>
</tr>
</tbody>
</table>

**Formulas**

- $S(q_i)$: set of terms in query $q_i$
- If $x > 0$, then $\text{sgn}(x) = 1$. If $x = 0$, then $\text{sgn}(x) = 0$.
- For each term in $q_T$, if it has been used in some of the previous queries, we count the number of clicks on the search results of that query and then sum up these counts by $C_{\text{eff}}(q_T)$.
User Features

- Typing speed at this keystroke
- Number of times the suggestion is issued by the user in the past.
- Suggestion frequency over queries submitted by users of same gender.
- Suggestion frequency over queries submitted by users of same age group.
- Average length of queries the user clicked in the past.
- Average number of words in queries the user clicked in the past.
- Sim between suggestion words and words in previous queries in same session.
  - Cosine similarity, Jaro Winkler edit distance, WordNet similarity, N-Gram similarity, SERP-Similarity

Implicit Negative Feedback from Previous Prefixes in same conversation

- User wants “facetime”.
- With prefix “fac”, “facebook” is ranked at the top.
- User dwells for a long time to examine “facebook” but does not select it.
- In the next keystroke “e”, popularity-based QAC still makes “facebook” top in the list.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DwellT-M</td>
<td>The maximum dwell time when ( q ) is suggested.</td>
</tr>
<tr>
<td>DwellT</td>
<td>Total dwell time where ( q ) is suggested.</td>
</tr>
<tr>
<td>WordBound</td>
<td>No. of the keystrokes at word boundaries when ( q ) is suggested.</td>
</tr>
<tr>
<td>SpaceChar</td>
<td>No. of the keystrokes at space characters when ( q ) is suggested.</td>
</tr>
<tr>
<td>OtherChar</td>
<td>No. of the keystrokes at non-alphanumeric. char. when ( q ) is suggested.</td>
</tr>
<tr>
<td>IsPrevQuery</td>
<td>1 if ( q ) is the immediately previous query; 0 otherwise.</td>
</tr>
<tr>
<td>Pos@i</td>
<td>No. of the keystrokes when ( q ) is at Position ( i ) of a suggestion list (( i = 1, 2, \ldots, 10 )).</td>
</tr>
</tbody>
</table>

*Dwell time greater than 3 seconds at one suggestion list is set to 3 seconds.*

Zhang, Aston, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A. Gunter, and Jiawei Han. "adaqac: Adaptive query auto-completion via implicit negative feedback." In SIGIR, pp. 143-152. 2015.

DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22
Agenda

- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
  - Traditional Machine Learning methods for ranking suggestions
  - Convolutional Latent Semantic Model
  - LSTM encoder
- Natural Language Generation [20 min]
- Personalization [20 min]
- Handling defective suggestions and prefixes [20 min]
- Summary and Future Trends [5 min]
Convolutional latent semantic model for rare prefixes

- Trained on a prefix-suffix pairs dataset
- Training data
  - Split each query at every possible word boundary.
  - "breaking bad cast" \(\rightarrow ("breaking", "bad cast")\) and ("breaking bad", "cast").
- Test time
  - Given a prefix P and a suggestion candidate C, extract \(\bar{p}\) by removing the end-term.
  - \(\bar{s}\) is extracted by removing \(\bar{p}\) from the query C.
  - Use the trained CLSM model to project the normalized prefix and the normalized suffix to a common 128D

\[
\text{clsmsim}(\bar{p}, \bar{s}) = \text{cosine}(y_1, y_2) = \frac{y_1^\top y_2}{\|y_1\| \|y_2\|}
\]

- Suffix based candidate generation
  - We match all the suffixes that start with the end-term from our precomputed set (10K/100K set).
  - These selected suffixes are appended to the prefix to generate synthetic suggestion candidates.

\[
\begin{tabular}{|c|c|c|}
\hline
Top suffixes & Top 2-word suffixes & Top 3-word suffixes \\
\hline
com & for sale & federal credit union \\
org & yahoo com & new york city \\
net & myspace com & in new york \\
gov & google com & or no deal \\
pictures & new york & disney channel com \\
lyrics & real estate & my space com \\
edu & of america & in new jersey \\
sale & high school & homes for sale \\
games & new jersey & department of corrections \\
florida & space com & chamber of commerce \\
for sale & aol com & bath and beyond \\
us & s com & in las vegas \\
\hline
\end{tabular}
\]


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22

\{gmanish,punagr\}@microsoft.com
Table 1: Synthetic QAC candidates generated by the suffix-based approach and ranked using only the CLSM similarity feature. The CLSM model projects both the prefix and the suffix to a common 128-dimensional space allowing us to rank according to prefix-suffix cosine similarity. One of the lower quality synthetic candidates "cheapest flights from seattle to airport" is ranked seventh in the second list.

- what to cook with chicken and broccoli and bacon
- what to cook with chicken and broccoli and brown sugar
- what to cook with chicken and broccoli and garlic
- what to cook with chicken and broccoli and orange juice
- what to cook with chicken and broccoli and beans
- what to cook with chicken and broccoli and ham soup

- cheapest flights from seattle to dc
- cheapest flights from seattle to washington dc
- cheapest flights from seattle to bermuda
- cheapest flights from seattle to bahamas
- cheapest flights from seattle to aruba
- cheapest flights from seattle to punta cana
- cheapest flights from seattle to airport
- cheapest flights from seattle to miami

Figure 2: The CLSM model architecture. The model has a convolutional-pooling structure and a 128-dimensional output.
Results with CLSM

Figure 3: MRR improvements by historical popularity of the input prefix on the AOL testbed. The LambdaMART model uses n-gram and CLSM features and includes suffix-based suggestion candidates. Any prefix in the top 100K most popular prefixes from the background data is considered as Frequent. There are 7622, 6917 and 14,135 prefix impressions in the Frequent, Rare and Unseen segments, respectively. All reported differences in MRR with the MPC model are statistically significant by the t-test (p < 0.01).

Table 3: Comparison of all models on the AOL and the Bing testbeds. Due to the proprietary nature of the Bing dataset, we only report MRR improvements relative to the MPC model for this testbed. Statistically significant differences by the t-test (p < 0.01) are marked with "**". Top three highest MRR values per testbed are bolded.

<table>
<thead>
<tr>
<th>Models</th>
<th>AOL MRR</th>
<th>% Improv.</th>
<th>Bing MRR</th>
<th>% Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-query based candidates only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MostPopularCompletion</td>
<td>0.1446</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = no, CLSM feature = no)</td>
<td>0.1445</td>
<td>-0.1</td>
<td>-1.7*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = no)</td>
<td>0.1427</td>
<td>-1.4*</td>
<td>-1.2*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = no, CLSM feature = yes)</td>
<td>0.1445</td>
<td>-0.1</td>
<td>-1.2*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = yes)</td>
<td>0.1432</td>
<td>-1.0*</td>
<td>-1.5*</td>
<td></td>
</tr>
<tr>
<td>Full-query based candidates + Suffix based candidates (Top 100K suffixes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MostPopularCompletion</td>
<td>0.1446</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = no, CLSM feature = no)</td>
<td>0.2116</td>
<td>+46.3*</td>
<td>+32.8*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = no)</td>
<td>0.2324</td>
<td>+60.8*</td>
<td>+42.6*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = no, CLSM feature = yes)</td>
<td>0.2249</td>
<td>+55.5*</td>
<td>+40.1*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = yes)</td>
<td>0.2339</td>
<td>+61.7*</td>
<td>+43.8*</td>
<td></td>
</tr>
<tr>
<td>Full-query based candidates + Suffix based candidates (Top 100K suffixes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MostPopularCompletion</td>
<td>0.1446</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = no, CLSM feature = no)</td>
<td>0.2105</td>
<td>+45.5*</td>
<td>+39.9*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = no)</td>
<td>0.2441</td>
<td>+68.7*</td>
<td>+54.2*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = yes)</td>
<td>0.2248</td>
<td>+55.4*</td>
<td>+48.9*</td>
<td></td>
</tr>
<tr>
<td>LambdaMART Model (n-gram features = yes, CLSM feature = yes)</td>
<td>0.2453</td>
<td>+69.6*</td>
<td>+55.3*</td>
<td></td>
</tr>
</tbody>
</table>


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
  • Traditional Machine Learning methods for ranking suggestions
  • Convolutional Latent Semantic Model
  • LSTM encoder
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Efficient Generation and Ranking for Neural QAC

• Candidate generation
  • We aim to increase recall of candidates with more context utilization.
  • 3 ways
    • MPC
    • Maximum Context Generation (MCG) i.e., ngram word tries: use as much context as possible.
    • LastWordGeneration (LWG) selects from a list of 100k most frequent suffixes

• Candidate Ranking
  • Two major components
    • The unnormalized language model layer computes “the sequence probability as the query scores” for a query candidate efficiently.
      • For efficiency, softmax normalization is approximated by 1 learnable scalar parameter.
    • Then pairwise learning-to-rank (LTR) objective functions are applied on the scores of the clicked and non-clicked query pairs.
  • These two components are trained together in an end-to-end fashion.

Our neural ranking model architecture. On top of it is a Learning-To-Rank layer that takes in multiple candidate scores. The input query has a special token “<sos> research scientist”; the probability of "research scientist <eos>" is computed based on LSTM hidden states.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Efficient Generation and Ranking for Neural QAC

The average time cost of ranking a candidate list with 10 candidates is measured for each model. The average number of words in candidates is 3.20. The hidden vector size and embedding size of LM is 100 and the LSTM layer number is 1.

Table 2: Performance of different candidate generation methods on AOL. For each method, candidates are generated in the same order as the ranking order described in Section 4.3.1. Recall@10 is computed for all prefixes, seen prefixes and unseen prefixes separately. † indicates statistically significant improvements over LastWordGeneration through a paired t-test with p < 0.05.

<table>
<thead>
<tr>
<th>Candidate Generation Methods</th>
<th>Recall@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td>MostPopularCompletion (MPC)</td>
<td>0.2075</td>
</tr>
<tr>
<td>LastWordGeneration (LWG)</td>
<td>0.3884</td>
</tr>
<tr>
<td>MaximumContextGeneration (MCG)</td>
<td>0.3992†</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaximumContextGeneration</td>
<td>0.18ms</td>
</tr>
<tr>
<td>CLSM</td>
<td>2.15ms</td>
</tr>
<tr>
<td>Unnormalized LM</td>
<td>3.01ms</td>
</tr>
<tr>
<td>Normalized LM</td>
<td>53.32ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation</th>
<th>Ranking</th>
<th>All</th>
<th>Seen</th>
<th>Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC</td>
<td>Frequency</td>
<td>0.1805</td>
<td>0.4431</td>
<td>0.0000</td>
</tr>
<tr>
<td>LWG</td>
<td>Frequency</td>
<td>0.3147</td>
<td>0.4465</td>
<td>0.2241</td>
</tr>
<tr>
<td>MCG</td>
<td>Frequency</td>
<td>0.3283</td>
<td>0.4469</td>
<td>0.2467</td>
</tr>
<tr>
<td>CLSM + Frequency</td>
<td>0.3369</td>
<td>0.4472</td>
<td>0.2610</td>
<td></td>
</tr>
<tr>
<td>LSTMEmbed + Frequency</td>
<td>0.3379‡ (+0.297%)</td>
<td>0.4472</td>
<td>0.2628‡</td>
<td></td>
</tr>
<tr>
<td>UnnormalizedLM + Frequency</td>
<td>0.3402‡ (+0.980%)</td>
<td>0.4473</td>
<td>0.2665‡</td>
<td></td>
</tr>
<tr>
<td>NormalizedLM + Frequency</td>
<td>0.3404‡ (+1.039%)</td>
<td>0.4473</td>
<td>0.2669‡</td>
<td></td>
</tr>
</tbody>
</table>

LSTMEmbed: The final hidden state vector from LSTM is used as the semantic representation of the sequence.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 (gmanish,punagr)@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• **Natural Language Generation** [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • LSTMs with subword embeddings
  • Hierarchical RNN Encoder-decoder
  • Next Phrase Prediction with T5
  • Problems with NLG
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
NLG for QAC

- Technical challenges
  - Handling partial words in the input
  - Optimization of computation requirements and throughput
  - Model compression/distillation
  - Beam search vs greedy decoding

- Considerations
  - Multi-language support
  - Inappropriate leakage
  - Suggestion quality
  - Coverage
  - Latency
Query Blazer: NLG without deep learning

- n-gram language model at a subword-level
- Exploits the n-gram model’s inherent data structure to precompute completions prior to runtime.

Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • LSTMs with subword embeddings
  • Hierarchical RNN Encoder-decoder
  • Next Phrase Prediction with T5
  • Problems with NLG
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Character-level Neural Language Model.

- **Char-LM**
  - Can handle OOV words
  - Can use the last incomplete word in prefix.

Top suggested queries by our char-LM. Phrases such as “afraid of the dead” and “afraid of the dog” and all prefixes do not exist in the data. Note that there is also a low-quality suggestion “when is a good time to buy a lyrics.

Architecture of our language model for an example query where ‘\n’ indicates the end of the query. Green cells contain one-hot encoded vectors of characters, blue cells contain character-embedded vectors, and red cells contain word-embedded vectors. <INC> means incomplete word token.
Character-level Neural Language Model.

- Mitra10K+MPC+λMART and Mitra100K+MPC+λMART: use 10K and 100K synthetic candidates using suffixes.
- NQLM: LM not using word-embedded character space
- NQLM+WE: uses word embeddings.
- NQLM(S): models with a small network using 512 hidden LSTM units
- NQLM(L): large network using 1,536 units
- +MPC: Append our LM-generated candidates to the end of MPC candidates, if there are any.
- +λMART: Employ LambdaMART and the same features as Mitra et al., except that CLSM scores are replaced by NQLM scores.

New metric: Partial-matching MRR (PMRR)
- Partial-match rank is the rank of the first candidate that is the same as the original query or that extends the prefix by one or more complete words.
- Partial-match rank <= full match rank.

<table>
<thead>
<tr>
<th>Model</th>
<th>Seen MRR</th>
<th>Unseen MRR</th>
<th>All MRR</th>
<th>Seen PMRR</th>
<th>Unseen PMRR</th>
<th>All PMRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC [1]</td>
<td>0.428</td>
<td>0.000</td>
<td>0.171</td>
<td>0.566</td>
<td>0.000</td>
<td>0.225</td>
</tr>
<tr>
<td>Char. n-gram (n=7)</td>
<td>0.363</td>
<td>0.236</td>
<td>0.287</td>
<td>0.550</td>
<td>0.376</td>
<td>0.445</td>
</tr>
<tr>
<td>Mitra10K+MPC+λMART [12]</td>
<td>0.427</td>
<td>0.179</td>
<td>0.278</td>
<td>0.586</td>
<td>0.297</td>
<td>0.412</td>
</tr>
<tr>
<td>Mitra100K+MPC+λMART [12]</td>
<td>0.428</td>
<td>0.212</td>
<td>0.298</td>
<td>0.588</td>
<td>0.368</td>
<td>0.455</td>
</tr>
</tbody>
</table>

Proposed models

- NQLM(S)
- NQLM(S)+WE
- NQLM(L)+WE
- NQLM(S)+MPC
- NQLM(S)+WE+MPC
- NQLM(L)+WE+MPC
- NQLM(S)+MPC+λMART
- NQLM(S)+WE+MPC+λMART
- NQLM(L)+WE+MPC+λMART


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]

• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • **LSTMs with subword embeddings**
  • Hierarchical RNN Encoder-decoder
  • Next Phrase Prediction with T5
  • Problems with NLG

• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Subword Language Model for QAC

- Representing queries with subwords shorten decoding length significantly compared to char-LM.

- Problem with subword LM
  - If we segment prefix as given to encode it using neural networks, the segmentation of prefix may not match with that of ground truth query because the prefix is an incomplete substring of the original desired query.
  - This enforced segmentation is less likely to appear in training
  - The model starting from this segmentation is unlikely to generate ground truth query

- Two ways of segmentation of prefix
  - BPE algorithm is deterministic because it segments greedily from left to right.
  - Subword regularization (SR): stochastically samples multiple segmentations by utilizing a unigram LM.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
For SR, due to the stochasticity of segmentation, we should marginalize over all possible segmentations to calculate the likelihood of a query.

The number of possible segmentations is exponentially large. Marginalization over all possible segmentations of very long sequences is intractable.

Hence, decode for the best token sequence.

Since finding best token sequence is also intractable, beam search decoding is used but only results in suboptimal predictions.

Solution: To consider every possible segmentation of target completion, retrace algorithm goes a few characters back from the end and generates candidates with the restriction that they should match with retraced characters.

Figure 1: Illustration of retrace algorithm with the example of “restaurants.” The gray area means given prefix (“res”) of the query. The solid line indicates the boundary of the segmentation. GT is the segmentation of ground truth query. Possible examples of the generated sequence of tokens belonging to the case $R_r$ are visualized. Blue boxes indicate a fixed segmentation with retrace algorithm at the end of the prefix.
Subword Language Model for QAC

• Subword LM is \sim 2.5\times faster while maintaining a similar quality of generated results compared to the character-level LM.

• New evaluation metric, mean recoverable length (MRL)
  • measures how many upcoming characters the model could complete correctly.
  • useful for additive QAC which suggests one word at a time instead of a whole query completion.
  • does not care about the order of candidates and check whether they contain the target query or not.

Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • LSTMs with subword embeddings
  • **Hierarchical RNN Encoder-decoder**
  • Next Phrase Prediction with T5
  • Problems with NLG
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Given a session $S = \{Q_1, \ldots, Q_M\}$, we aim to predict the target query $Q_M$ given the context $Q_1, \ldots, Q_{M-1}$.

- HRED generates synthetic suggestions sampled one word at a time.
- Useful for rare, or long-tail, queries.

Hierarchical recurrent encoder-decoder (HRED)

The user types “cleveland gallery → lake erie art”. During training, the model encodes “cleveland gallery”, updates the session-level recurrent state and maximizes the probability of seeing “lake erie art”. The process is repeated for all queries in the session. During testing, a contextual suggestion is generated by encoding the previous queries, by updating the session-level recurrent states accordingly and by sampling a new query from the last obtained session-level recurrent state. Here, the generated contextual suggestion is “cleveland indian art”.

**HRED with LambdaMART**

<table>
<thead>
<tr>
<th>Context</th>
<th>Synthetic Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ace series drive</td>
<td>ace hardware, ace hard drive, hp office jet drive, ace hardware series</td>
</tr>
<tr>
<td>cleveland gallery → lake erie art</td>
<td>cleveland indian art, lake erie art gallery, lake erie picture gallery, sandusky ohio art gallery</td>
</tr>
</tbody>
</table>

Table 1: HRED suggestions given the context.

- \( Q_{M-1} \) is anchor query.
- BaselineRanker: 17 features
  - Pairwise and Suggestion Features.
  - Contextual Features.
- HRED Score (log-likelihood of the suggestion given the context) can also be used for ranking.
- LambdaMART

- **Test Scenario 1: Next-Query Prediction**
  - For each session, extract 20 candidate queries that most likely follow the anchor query in background data, i.e. with the highest ADJ score.
  - Take instances where target is in top 20 candidate set.

- **Test Scenario 2: Robust Prediction**
  - Label 100 most frequent queries in background set as noisy.
  - For each entry in the previous next-query prediction task, corrupt its context by inserting a noisy query at a random position.
  - The candidates and the target are unchanged.
  - The probability of sampling a noisy query is proportional to its frequency in the background set.
  - E.g., given context “airlines → united airlines” and target “delta airlines”, the noisy sample “google” is inserted at a random position. Thus, corrupted context is “airlines → united airlines → google”.

- **Test Scenario 3: Long-Tail Prediction**
  - Retain the sessions for which the anchor query has not been seen in the background set, i.e., it is a long-tail query.
  - For each session, iteratively shorten the anchor query by dropping terms until we have a query that appears in the background data.
  - If a match is found, we proceed as described in the next-query prediction setting, i.e., ensure that target is in top 20 candidate set.

Comparison of HRED with BaselineRanker and ADJ

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>Δ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>0.5334</td>
<td>-</td>
</tr>
<tr>
<td>Baseline Ranker</td>
<td>0.5563</td>
<td>+4.3%</td>
</tr>
<tr>
<td>+ HRED</td>
<td>0.5749</td>
<td>+7.8%/+3.3%</td>
</tr>
</tbody>
</table>

Table 3: Next-query prediction results. All improvements are significant by the t-test ($p < 0.01$).

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>Δ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>0.3830</td>
<td>-</td>
</tr>
<tr>
<td>Baseline Ranker</td>
<td>0.6788</td>
<td>+77.2%</td>
</tr>
<tr>
<td>+ HRED</td>
<td>0.7112</td>
<td>+85.3%/+5.6%</td>
</tr>
</tbody>
</table>

Table 5: Long-tail prediction results. The improvements are significant by the t-test ($p < 0.01$).

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>Δ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>0.4507</td>
<td>-</td>
</tr>
<tr>
<td>Baseline Ranker</td>
<td>0.4831</td>
<td>+7.2%</td>
</tr>
<tr>
<td>+ HRED</td>
<td>0.5309</td>
<td>+17.8%/+9.9%</td>
</tr>
</tbody>
</table>

Table 4: Robust prediction results. The improvements are significant by the t-test ($p < 0.01$).

Figure 8: User study results, which compare the effectiveness of HRED with the baseline techniques.
RIN: Reformulation Inference Network for Context-Aware Query Suggestion

• Homomorphic embedding = average of node2vec embeddings for every word.

• Reformulation $r_i$ from $q_i$ to $q_{i+1}$ can be represented as the difference between embeddings as $v_{q_{i+1}} - v_{q_i}$

Figure 2: An example of the heterogeneous network constructed by a search session of four queries for deriving term embeddings. Note that the queries in the graph are auxiliary nodes connecting the domains of terms and websites.

RIN: Reformulation Inference Network for Context-Aware Query Suggestion

- Query Session encoder
  - To capture the structure of the session context, a RNN with the attention mechanism is employed to encode the search session by reading the homomorphic query and reformulation embeddings.
  - It enables the model to explicitly captures the former reformulation for each query in the search session and directly learn user reformulation behaviors.

\[
\begin{align*}
\mathbf{u}_i &= \tanh(\mathcal{F}_s(\mathbf{h}_i)), \\
\alpha_i &= \frac{\exp(\mathbf{u}_i^T \mathbf{u}_s)}{\sum_{i'} \exp(\mathbf{u}_{i'}^T \mathbf{u}_s)} \\
\mathbf{c} &= \sum_i \alpha_i \mathbf{h}_i
\end{align*}
\]
RIN: Reformulation Inference Network for Context-Aware Query Suggestion

• Decoder part
  • Both question suggestion and reformulation prediction can be simultaneously optimized by multi-task learning.
  • 3 parts
    • Reformulation Inferencer
    • Query Discriminator
    • Query Generator

• Reformulation Inferencer
  • A model that accurately predicts the next reformulation can also correctly forecast the next query.
  • Predict the next reformulation $r_L = v_{q_{L+1}} - v_{q_L}$
  • Apply a FC hidden layer on context vector $c$,

• Query Discriminator
  • Given a candidate query $q_{can}$ and the context vector $c$, the goal is to assess how likely $q_{can}$ is the intended query.

• Query Generator
  • Without any candidate query, the query generator aims to produce a sequence of terms as the generated query.

$$s_t = RNN(s_{t-1}, [w_{t-1}; c_t]),$$

$$u_{t,i} = \tanh(F_g([s_{t-1}; h_i])),$$

$$\alpha_{t,i} = \frac{\exp(u_{t,i}^T u_g)}{\sum_{i'} \exp(u_{t,i'}^T u_g)},$$

$$c_t = \sum_i \alpha_{t,i} h_i,$$
RIN: Reformulation Inference Network for Context-Aware Query Suggestion

- **Optimization**
  \[
  \text{loss}_R = \frac{1}{2} || \mathbf{r}_L - \hat{\mathbf{r}}_L ||_F^2 \\
  \text{loss}_D = - (y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \\
  \text{loss}_G = - \sum_{w_t} \log P(w_t | S_t) \\
  \text{loss} = \text{loss}_R + \text{loss}_{\text{task}}
  \]

- **Task could be D or G.**

- Most Popular Suggestion (MPS): ranks queries by the co-occurrence to the last query in the context.

- Query-based Variable Markov Model (QVMM): learns the probability of query transitions over sessions with the variable memory Markov model implemented by a suffix tree.

- Hybrid Suggestion (Hybrid): ranking candidate queries based on a linear combination between the popularity (i.e., MPS) and the similarity to recent queries.

- Personalized Completion (PC): Personalized LambdaMART ranking model using MPS as well as user long-term history signals.

- Reformulation-based Completion (RC): LambdaMART with 43 reformulation-based features

---

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Context</td>
<td>0.5471</td>
<td>0.5823</td>
<td>0.5150</td>
<td>0.5671</td>
<td>0.6202</td>
<td>0.6207</td>
<td>0.6559</td>
<td><strong>0.8254</strong></td>
</tr>
<tr>
<td>Short Context (1 query)</td>
<td>0.5680</td>
<td>0.5822</td>
<td>0.5343</td>
<td>0.5862</td>
<td>0.5960</td>
<td>0.6100</td>
<td>0.6471</td>
<td><strong>0.8361</strong></td>
</tr>
<tr>
<td>Medium Context (2 to 3 queries)</td>
<td>0.5167</td>
<td>0.5841</td>
<td>0.4865</td>
<td>0.5338</td>
<td>0.6689</td>
<td>0.6489</td>
<td>0.6542</td>
<td><strong>0.8190</strong></td>
</tr>
<tr>
<td>Long Context (4 or more queries)</td>
<td>0.4826</td>
<td>0.5768</td>
<td>0.4575</td>
<td>0.5026</td>
<td>0.6704</td>
<td>0.6122</td>
<td>0.6669</td>
<td><strong>0.7611</strong></td>
</tr>
</tbody>
</table>


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 (gmanish,punagr)@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • LSTMs with subword embeddings
  • Hierarchical RNN Encoder-decoder
  • Next Phrase Prediction with T5
  • Problems with NLG
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Next Phrase Prediction for QAC for Emails/Academic Writings

• Next Phrase Prediction (NPP)
  • Encourages a language model to complete the partial query with enriched phrases
  • 2 steps
    • Phrase Extraction
      • extracts qualitative phrases by constituency parsing
    • Generative Question Answering
      • Start with a pre-trained T5 model
      • The pre-trained LM is guided to choose the correct next phrase among other phrases of the same type (e.g., NP, VP, etc.) in the sentence.
    • Finetune on QAC task.

GPT-2 can generate syntactically sound, and semantically general sentence from partial query. However, it still needs to be fine-tuned a lot to generate semantically expert domain (e.g. Computer Science) focused sentence.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
## NPP Results

<table>
<thead>
<tr>
<th>Model / Metrics</th>
<th>Emails</th>
<th></th>
<th>Academic Writing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLEU-4</td>
<td>METEOR</td>
<td>CIDEr</td>
<td>SPICE</td>
</tr>
<tr>
<td>GPT-2 (Radford et al., 2019)</td>
<td>1.1</td>
<td>6.6</td>
<td>26.4</td>
<td>3.3</td>
</tr>
<tr>
<td>T5 (Raffel et al., 2020)</td>
<td>2.8</td>
<td>6.8</td>
<td>39.8</td>
<td>4.2</td>
</tr>
<tr>
<td>NSP+T5</td>
<td>3.0</td>
<td>6.9</td>
<td>41.1</td>
<td>4.4</td>
</tr>
<tr>
<td>NPP+T5 (Ours)</td>
<td><strong>3.2</strong></td>
<td><strong>7.1</strong></td>
<td><strong>43.0</strong></td>
<td><strong>4.5</strong></td>
</tr>
</tbody>
</table>

### Partial Query

<table>
<thead>
<tr>
<th>Building large OCR databases is a time consuming task. The effort involves tedious work. A language interface standard is developed. A filter is established. Of the tools and techniques used in this paper, detection is a key feature of this approach.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original: consuming and tedious work. language interface standard. filter is developed. of code for the systems. recognition is an important task.</td>
</tr>
<tr>
<td>T5: challenging. system. et al. of the apl libraries. selection is based on a set of criteria.</td>
</tr>
<tr>
<td>NPP+T5: consuming task. language. filter is established. of the tools and techniques used in this paper. detection is a key feature of this approach.</td>
</tr>
</tbody>
</table>

---


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22. {gmanish,punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
  • RNNs with character and word embeddings
  • LSTMs with subword embeddings
  • Hierarchical RNN Encoder-decoder
  • Next Phrase Prediction with T5
  • Problems with NLG
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
When Are Search Completion Suggestions Problematic?

- 15% to 47% of problematic suggestions were flagged as problematic due to the query prefix they were surfaced for.
- Search voids: rare query prefixes to be up to 3 times more likely to be linked to problematic suffixes.
- Problematic suggestions can also affect those issuing the search query if e.g., their dignity is compromised
  - Suggesting ... [bed bugs by yourself] when a user typed how to kill [...] may imply the system deduced the user has a bed bugs infestation issue.
  - Suggesting ... [my arrest records] when a user typed how to find [...] may hint the system assumed that such records may exist

Fig. 1. Examples of search query completion suggestions for different query prefixes.

- Racist or sexist
- Containing profanity or violence
- Subtle ways
  - A certain phrase may be bothersome for some group of users, but not for others.
  - A suggestion such as “... is evil” or “... is huge” may be acceptable if the user input was “heinous crime ...” or “the universe ...,” but likely problematic if it was a person’s name
## When Are Search Completion Suggestions Problematic?

<table>
<thead>
<tr>
<th>Problem Category</th>
<th>Working definitions (and sub-categories)</th>
<th>Keywords (p/s: query)</th>
</tr>
</thead>
</table>
| Harmful speech   | • Hate speech: suggestions that could be perceived as hateful or that intend to intimidate or promote violence, against a group or its members.  
• Intimidates & promotes violence: suggestions that may steer users towards acting violently or that aim to intimidate certain individuals.  
• Offensive speech: suggestions that dehumanize, insult, or ridicule, actively seeking to embarrass or harm reputation.  
• Discriminatory speech: suggestions showing known or existing bias, prejudice, or intolerance, perpetuating, employing negative stereotypes, or encouraging feelings of fear or disgust towards a group or individual.  
• Defamation & derogatory speech: suggestions that defame someone by suggesting negative associations, including suggestions of dishonesty or involvement in illicit activities.  
• Profane language: suggestions including any sort of slurs, expletives, swear or curse words. | • punch (p: should i punch [my mother])  
• hit (s: should women be [hit by men])  
• deported (s: arabs should be [deported])  
• poison (p: which poison can kill [an adult fast]) |
| Potentially illicit | • Facilitates illicit activities: suggestions condoning & constituting illicit speech, infringing on intellectual property, copyright rights or trademark agreements, or that facilitate or nudge users towards illicit activities.  
• Privacy breaching: suggestions revealing unwanted details from someone’s past or anything that may be construed as sensitive or personal information.  
• Terrorist or extremist propaganda: suggestions that may steer or help users find extremist content related to terrorist or extremist activities like recruiting or sponsoring.  
• Defamation & derogatory speech: See above.  
• Child abuse & pornography: suggestions related to child abuse or child pornography | • heroin (s: trustworthy website to [buy heroin])  
• fake passports (s: how to get [fake passports])  
• beat child (s: how to [beat my child]) |
## When Are Search Completion Suggestions Problematic?

<table>
<thead>
<tr>
<th>Problem Category</th>
<th>Working definitions (and sub-categories)</th>
<th>Keywords (p/s: query)</th>
</tr>
</thead>
</table>
| Controversy, Misinformation, and Manipulation         | • Controversial topics: suggestions that seem to endorse one side of a known controversial debate.  
• Misinfo., disinfo. or misleading content: suggestions that promote information that is factually incorrect, or that reinforce or nudge users towards conspiracy theories.  
• Coordinated attacks & suggestions manipulation: suggestions that occur as a result of attempts to manipulate the search or suggestions results, such as by promoting certain businesses or by trying to affect someone’s reputation. | • hoax (s: climate change is [a hoax])  
• staged (s: 911 was [staged])  
• vaccines (p: vaccines are [dangerous])  
• divorce lawyer (p: divorce lawyer [nashville LAW_- FIRM_NAME])                                                                 |
| Stereotypes & Bias                                    | • Ideological bias: suggestions that validate or endorse views that belong to certain ideological groups, or that promote stereotypical beliefs about an ideological group.  
• Systemically biased suggestions: suggestions about certain topics that are systematically biased towards a group, reinforcing sensitive associations between the group & negative attributes or stereotypical beliefs.  
• Discriminatory speech, Defamation & derogatory speech, Offensive Speech: See above. | • Refugees (p: refugees are [taking jobs])  
• women (p: women need [to dress modestly])  
• girl (s: running like [a girl])  
• black men (p: black men [are lazy])                                                                 |
| Adult queries                                         | • Adult content: suggestions that contain pornography-related terms or steer users towards pornographic/obscene content.  
• Child abuse: See above. | • naked (p: naked girls [videos])                                                                                                                                        |
| Other types                                            | • Animal cruelty: suggestions that may steer users towards info about how to harm animals.  
• Self-harm and suicidal content: suggestions that may steers someone towards hurting themselves.  
• Sensitive topics: suggestions that may trigger memories of traumatic events or be considered sensitive or emotionally charged by certain groups due to historic or cultural reasons | • strangle dog (p&s: how to strangle [a dog])  
• hitler (p: hitler is [my god])  
• hurt myself (s: I want to [hurt myself])                                                                 |

## When Are Search Completion Suggestions Problematic?

<table>
<thead>
<tr>
<th>Target Category</th>
<th>Working definitions (and sub-categories)</th>
<th>Keywords (p/s: query)</th>
</tr>
</thead>
</table>
| **Individuals** | • References to a public or private person, who may or may not be explicitly named | • ruth ginsburg (p: ruth ginsburg [dead yet])  
• my dad (s: should I kill [my dad]) |
| **Groups** | • References to a group of individuals that share at least a common characteristic, such as race, gender, age, occupation, appearance, disability, or country of origin | • muslims (p: muslims try to [conquer through numbers])  
• children with adhd (s: how to punish [adhd child]) |
| **Businesses** | • References to a specific business | • Macy’s (p: macy’s is [scamming shoppers])  
• CNN (s: should we punish [cnn])  
• Starbucks (s: should I boycott [starbucks]) |
| **Organizations** | • References to an organization, institution or agency, which can be governmental or non-governmental (but not a business); or a group of for-profit organizations if they are not specifically identified (e.g., news media instead of CNN, social media instead of Twitter) | • mainstream media (p: mainstream media is [destroying america])  
• UNICEF (p: UNICEF is running [a scam])  
• travel companies (s: don’t waste money on [travel companies]) |


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22  
{gmanish,punagr}@microsoft.com
When Are Search Completion Suggestions Problematic?

<table>
<thead>
<tr>
<th>Target Category</th>
<th>Working definitions (and sub-categories)</th>
<th>Keywords (p/s: query)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animals &amp; objects</td>
<td>• References to an animal, a group of animals, or anything that may be construed as an object or a group of objects</td>
<td>• cat (s: how to poison [a cat])&lt;br&gt;• knife (p: how to use a knife [to kill])</td>
</tr>
<tr>
<td>Activities &amp; ideas</td>
<td>• References to a specific activity, action, or idea</td>
<td>• cutting yourself (p: cutting yourself is [stupid])&lt;br&gt;• crying (p: crying is [emotional blackmail])</td>
</tr>
<tr>
<td>Other targets</td>
<td>• References to concepts like ideologies, religions, programs, health issues, a situation someone may find themselves in, or other types that do not fit other categories</td>
<td>• bipolar disorder (p: bipolar disorder is [fraud])&lt;br&gt;• vaccination (p: vaccination [herd mentality])&lt;br&gt;• science (p: science should [stay out of faith])</td>
</tr>
<tr>
<td>Generic, no target</td>
<td>• There is no identifiable target or subject</td>
<td>• (what [the heck])&lt;br&gt;• (damn damn [damn])</td>
</tr>
</tbody>
</table>
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Personalization for QAC

Using short-term/long-term user history, location, other signals

**Search/Click History:**
- tender is the night
- f. scott fitzgerald
- this side of paradise
- the silent patient
- the great gatsby book

**Search/Click History:**
- the wolf of wall street
- the revenant
- inception
- gangs of new york
- pain & gain

**Search/Click History:**
- beijing travel advisory
- forbidden city

**Search/Click History:**
- the great wall of china
- the great gatsby film 2013
- the great gatsby trailer
- the great gatsby film 1974
- the greatest showman
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
  • Traditional Machine Learning methods
  • Hierarchical RNN Encoder-decoder
  • GRUs with user and time representations
  • Transformer-based hierarchical encoder
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
  • Traditional Machine Learning methods
  • Hierarchical RNN Encoder-decoder
  • GRUs with user and time representations
  • Transformer-based hierarchical encoder
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Motivation for personalization

• Prefix="i"
  • Young (age<25) female users: suggestion=instagram
  • Male users (25<age<44): suggestion=imdb

• Demography and history features can be used for personalizing auto-completion rankings
  • Age {Below 20, 21-30, 31-40, 41-50, and above 50}
  • Gender
  • Location (zip-codes)
  • Short- and long-history: n-gram similarity

(Top) The likelihood of instagram and imdb in queries submitted by different demographics according to Yahoo Clues. (Bottom) The likelihood of instagram and imdb in queries submitted by the logged-in users of Bing.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
# Features for personalizing auto-completion

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrevQueryNgramSim</td>
<td>Short history</td>
<td>n-gram similarity with the previous query in the session ($n = 3$).</td>
</tr>
<tr>
<td>AvgSessionNgramSim</td>
<td>Short history</td>
<td>Average n-gram similarity with all previous queries in the session ($n = 3$).</td>
</tr>
<tr>
<td>LongHistoryFreq</td>
<td>Long history</td>
<td>The number of times a candidate is issued as query by the user in the past.</td>
</tr>
<tr>
<td>LongHistorySim</td>
<td>Long history</td>
<td>Average n-gram similarity with all previous queries in the user’s search history.</td>
</tr>
<tr>
<td>SameAgeFrequency</td>
<td>Demographics</td>
<td>Candidate frequency over queries submitted by users in the same age group.</td>
</tr>
<tr>
<td>SameAgeLikelihood</td>
<td>Demographics</td>
<td>Candidate likelihood over queries submitted by users in the same age group.</td>
</tr>
<tr>
<td>SameGenderFrequency</td>
<td>Demographics</td>
<td>Candidate frequency over queries submitted by users in the same gender group.</td>
</tr>
<tr>
<td>SameGenderLikelihood</td>
<td>Demographics</td>
<td>Candidate likelihood over queries submitted by users in the same gender group.</td>
</tr>
<tr>
<td>SameRegionFrequency</td>
<td>Demographics</td>
<td>Candidate frequency over queries submitted by users in the same region group.</td>
</tr>
<tr>
<td>SameRegionLikelihood</td>
<td>Demographics</td>
<td>Candidate likelihood over queries submitted by users in the same region group.</td>
</tr>
<tr>
<td>SameOriginalPosition</td>
<td>MPC</td>
<td>The position of candidate in the MPC ranked list.</td>
</tr>
<tr>
<td>SameOriginalScore</td>
<td>MPC</td>
<td>The score of candidate in the MPC ranked list computed based on past popularity.</td>
</tr>
</tbody>
</table>


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22  
{gmanish,punagr}@microsoft.com
LTR framework for personalization

- Supervised framework for personalizing auto-completion ranking.
- Labeled data
  - The query which was eventually submitted by the user is considered as the only right (relevant) candidate and is assigned a positive label.
  - The other candidates are all regarded as non-relevant and get zero labels.
- User’s long-term search history and location are the most effective for personalizing auto-completion rankers.
- Supervised rankers enhanced by personalization features can significantly outperform the existing popularity-based baselines, in terms of MRR by up to 9%.

<table>
<thead>
<tr>
<th>Below 20</th>
<th>21-30</th>
<th>31-40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taylor Swift</td>
<td>Pier Morgan</td>
<td>Bank of America</td>
</tr>
<tr>
<td>Justin Bieber</td>
<td>Richard Nixon</td>
<td>Worldstarhiphop</td>
</tr>
<tr>
<td>DeviantArt</td>
<td>Weather</td>
<td>Alex Jones</td>
</tr>
<tr>
<td>Full House</td>
<td>Beyoncé</td>
<td>Indeed</td>
</tr>
<tr>
<td>Harry Styles</td>
<td>Movies</td>
<td>National Weather Service</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>41-50</th>
<th>Above 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>National Cathedral</td>
<td>Mapquest</td>
</tr>
<tr>
<td>Target</td>
<td>Fedex Tracking</td>
</tr>
<tr>
<td>Chase</td>
<td>Florida Lottery</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Pogo</td>
</tr>
<tr>
<td>Traductor Google</td>
<td>Jigsaw Puzzles</td>
</tr>
</tbody>
</table>

The biggest movers in personalized autocompletion rankings when the ranker is trained by age features. Each column includes the candidates that were boosted most frequently in the personalized autocompletion rankings for users of the specified age groups.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22  
{gmanish,punagr}@microsoft.com
Location for personalization

The top movers in each region. These are queries that their average positions in rankings with and without personalization differ the most in each region. The regions are specified by collapsing the first zip-code digits and the users in each region are grouped accordingly. Each map shows the distribution of query popularity across different US states according to Google Trends, and the colors range between light blue (rare) and dark blue (popular).


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com

65
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
  • Traditional Machine Learning methods
  • Hierarchical RNN Encoder-decoder
    • GRUs with user and time representations
    • Transformer-based hierarchical encoder
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
RNNs for personalization

• Hierarchical Contextual Attention RNN (HCAR-NN)
  • For map query suggestion in an encoding-decoding manner.
  • Learns the local temporal correlation among map queries in a query session
  • Captures global longer range contextual dependencies among map query sessions in query logs (e.g., how a sequence of queries within a short-term interval has an influence on another sequence of queries).

• Three LSTM layers
  • Encode each query session into a vector using an encoding LSTM, $LSTM^e$.
  • Capture the contextual dependencies among query sessions to jointly learn the encoding vector of subsequent query session in a soft attention mechanism (contextual LSTM, $LSTM^c$).
  • Decoding LSTM ($LSTM^d$) predicts map queries according to the fed encoding vectors.

Mission queries.

Attend, Copy, Generate (ACG)

• Co-occurrence based models
  • Suffer from data sparsity and lack of coverage for rare or unseen queries.
  • Dealing with these highly diverse sessions makes using co-occurrence based model almost impossible.

• ACG
  • Query-aware attention mechanism to capture the structure of the session context.
  • Automatically detects session boundaries.

• Within a single session a large portion of query terms is retained from the previously submitted queries and consists of mostly infrequent or unseen terms that are usually not included in the vocabulary.
  • ~62% of the terms in a query are retained from their preceding queries
  • >39% of the users repeat at least one term from their previous query
  • Based on statistics from the AOL query log, >67% of the retained terms in the sessions are from the bottom 10% of terms ordered by their frequency.

Example of generating a suggestion query given the previous queries in the session. The suggestion query is generated during three time steps. The heatmap indicates the attention, red for query-level attention and blue for word-level attention. The pie chart shows if the network decides to copy or to generate.
• Word level hidden layer output: $h_i$
• Decoder hidden layer output: $s_{t-1}$
• Word-level attention:
  • $l_{t,i} = \eta(s_{t-1}, h_i)$
  • $a_{t,i} = \frac{\exp(l_{t,i})}{\sum_j \exp(l_{t,j})}$

• Query level hidden layer output: $g_j$

• Query-level attention:
  • $l_{t,j} = \eta(s_{t-1}, g_j, y_{t-1})$
  • $a_{t,j} = \frac{\exp(l_{t,j})}{\sum_i^n \exp(l_{t,i}^q)}$

• Overall attention:
  • $a_{t,i} = \frac{a_{t,i}^w a_{t,j}^q}{\sum_{i',j'} a_{t,i}^w a_{t,j'}^q}$


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Multi-Objective Training

- Loss of the generator is averaged cross entropy.
- We should choose a target label for the switch gate to copy as much as possible from the input and let the generator handle the rest.
- We update the parameters of the network with respect to the losses in three separate steps.
  - Use $\text{loss}_{\text{copy}}$ to update all parameters of the network, except those for switch gate and the generator.
  - Use $\text{loss}_{\text{generate}}$ to update all parameters of the network except the parameters of the switch gate and the copier.
  - Update the parameters of the network using the gradients from $\text{loss}_{\text{switch}}$, while the parameters of copy and generator are fixed.

\[
\text{loss}_{\text{generate}} = \frac{1}{|V|} H(p,q) = \frac{1}{|V|} \sum_{v \in V} p_v \log q_v
\]
\[
\text{loss}_{\text{copy}} = \frac{1}{|X|} H(p,q) = \frac{1}{|X|} \sum_{x \in X} p_x \log q_x
\]

1. Target copier is $\langle \text{UNK} \rangle$ and target generator is not $\langle \text{OOV} \rangle$: the switch gate shall choose generation ($t_{\text{switch}} = 0$).
2. Target copier is not $\langle \text{UNK} \rangle$ and target generator is $\langle \text{OOV} \rangle$: the switch gate shall choose copying ($t_{\text{switch}} = 1$).
3. Target copier is $\langle \text{UNK} \rangle$ and target generator is $\langle \text{OOV} \rangle$: the switch gate shall choose generation ($t_{\text{switch}} = 0$).
4. Target copier is not $\langle \text{UNK} \rangle$ and target generator is not $\langle \text{OOV} \rangle$: the switch gate shall choose copying ($t_{\text{switch}} = 1$).

\[
\text{loss}_{\text{switch}} = (p(\text{copy}) - t_{\text{switch}})^2
\]
\[
p(q|X) = \prod_{t=1}^{n} \left( p(\text{generate}|y_{<t},X) p(y_t|y_{<t},X,\text{generate}) + p(\text{copy}|y_{<t},X) p(y_t|y_{<t},X,\text{copy}) \right)
\]
Evaluation

- Evaluation based on Discrimination
  - As a feature to score the candidate queries and use it within L2R framework.
  - Use LambdaMART. Metric: MRR

- Evaluation based on Generation
  - Word Overlap Based Query Similarity
  - Embedding Based Query Similarity
  - Retrieval Based Query Similarity
    - Retrieves similar search results ($sim_{ret}$)
    - PRF based for target query vs actual results for generated query ($sim_{ret}^+$)
    - $sim_{ret}^{++}$
      - Take all sessions with length l>2 from the test data.
      - Use first l/2 queries in the session as the context for generating the next query.
      - Retrieve the ranked lists of documents for each of the next l/2 queries in the session and merge.
      - Merged list is used as the reference ranked list. Calculate its agreement with the retrieved results given the generated query.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22
{gmanish,punagr}@microsoft.com
Comparisons

- Methods:
  - Most Popular Suggestions (MPS): Rank by historical co-occurrence count with last query in current session.
  - BaseRanker: 17 feature L2R method by Sordoni et al.
  - seq2seq model and one with query-aware attention only (seq2seq + QaA)
  - seq2seq model with copy mechanism (seq2seq + CM) only

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPS</td>
<td>0.5216</td>
</tr>
<tr>
<td>2</td>
<td>BaseRanker</td>
<td>0.5530</td>
</tr>
<tr>
<td>3</td>
<td>BaseRanker + Seq2Seq</td>
<td>0.5679</td>
</tr>
<tr>
<td>4</td>
<td>BaseRanker + HRED [42]</td>
<td>0.5727</td>
</tr>
<tr>
<td>5</td>
<td>BaseRanker + (Seq2Seq + QaA)</td>
<td>0.5744</td>
</tr>
<tr>
<td>6</td>
<td>BaseRanker + (Seq2Seq + CM)</td>
<td>0.5851</td>
</tr>
<tr>
<td>7</td>
<td>BaseRanker + ACG</td>
<td>0.5941</td>
</tr>
</tbody>
</table>

Table 1: Performance of the different methods as discriminative models. *(x)* indicates that the improvements with respect to the method in row x is statistically significant, at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

Performance of the generated queries at different ranks.

Figure 4: Performance of ACG compared to HRED on sessions with different lengths.
Agenda

- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
- Personalization [20 min]
  - Traditional Machine Learning methods
  - Hierarchical RNN Encoder-decoder
  - **GRUs with user and time representations**
  - Transformer-based hierarchical encoder
- Handling defective suggestions and prefixes [20 min]
- Summary and Future Trends [5 min]
Personalized neural Language Model

- Char level 2 layer GRUs.
- Input: User embedding+char embedding+word embedding+time embedding.
- If $\hat{P}(c_{i+1})$ is reference probability of character $c_{i+1}$ across all queries, loss is

$$
\mathcal{L} = - \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{|q|-1} \hat{P}(c_{i+1}) \times \log P(c_{i+1}|c_1, \ldots, c_i).
$$
Personalized neural Language Model

• User representation
  • Each query $q_i$ is a set of words \{${w_1, ..., w_n}$\}.
  • User $u \in U$ is characterized by the union of words in their $k$ past queries, i.e., $Q_u$
  • PV-DBOW
    • At each training iteration, a random word $w_i$ is sampled from $Q_u$.
    • The model is trained by maximizing the probability of predicting the user $u$ given the word $w_i$

• Time representation
  • For each query, time corresponding to hour $x$, minute $y$ and second $z$, is encoded using these features.
  • We proceed the same way to encode weekdays and we end up with four time features.

\[
\sin \left( \frac{2\pi(3600x + 60y + z)}{86400} \right) \\
\cos \left( \frac{2\pi(3600x + 60y + z)}{86400} \right)
\]
Diverse beam search

- Decodes diverse lists by dividing the beam budget into groups and enforcing diversity between groups of beams.
- We optimize an objective that consists of two terms – the sequence likelihood under the model and a dissimilarity term that encourages beams across groups to differ.
- This diversity-augmented model score is optimized in a doubly greedy manner – greedily optimizing along both time and groups.

Figure 2: Diverse beam search operates left-to-right through time and top to bottom through groups. Diversity between groups is combined with joint log-probabilities, allowing continuations to be found efficiently. The resulting outputs are more diverse than for standard approaches.

MPC: Most Popular Completion

NQAC: word embeddings and the one-hot encoding of characters only.

Subscript U: language model is enriched with user vectors

Subscript T: language model integrates time features.

+D: use of the diverse beam search

Also study the impact of adding MPC and LambdaMART (+MPC, +λMART).

NQLM(L): Neural Query LM with LSTMs.
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
  • Traditional Machine Learning methods
  • Hierarchical RNN Encoder-decoder
  • GRUs with user and time representations
  • **Transformer-based hierarchical encoder**
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Multi-view Multi-task Attentive framework for Personalized QAC

- Input: query prefix and two behavior sequences, and each behavior is a token sequence, which can be a searched query or a browsed item title.
- Transformer-based hierarchical encoder to model different kinds of sequential behaviors, which can be seen as multiple distinct views of the user’s searching history.
  - Obtains context-free behavior-level and context-aware context-level representations of each behavior.
- A prefix-to-history attention mechanism is used to select the most relevant information from two views with the prefix representation from the middle part as key.
- The prefix representation are combined with the weighted view presentations as the final intention representation to feed into two specific tasks, including CTR prediction and query generation.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Examples of query candidates recommended by different models for the same prefix and history behavior.

Ablation Study of query generation models.

$V_q$: the view of searched queries, $V_i$: the view of browsed items, $L_c$: the CTR prediction task, $L_u$: the query generation task for typed queries.

UMRR: Unbiased MRR (to eliminate position bias.)

\[
\text{Unbiased MRR} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \frac{1}{\text{rank}_i}
\]


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
  • LSTMs for inappropriate query suggestion detection
  • Offline Spell Correction
  • Online Spell Correction
• Summary and Future Trends [5 min]
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
  • LSTM for inappropriate query suggestion detection
  • Offline Spell Correction
  • Online Spell Correction
• Summary and Future Trends [5 min]
Inappropriate query suggestion detection

- Inappropriate: if it may cause anger, annoyance to certain users or exhibits lack of respect, rudeness, discourteousness towards certain individuals/communities or may be capable of inflicting harm to oneself or others.

• Randomly initialize the DSSM word vectors for these padded unknown words from the uniform distribution [-0.25, 0.25].

• Use three $3 \times 25$ filters.
Results

- Pattern and Keyword based Filtering (PKF)
- Support Vector Machine (SVM)
- Gradient Boosted Decision Trees (BDT)

Fig. 3. Statistics of Inappropriate Categories in our Evaluation Dataset.

<table>
<thead>
<tr>
<th>Category</th>
<th>No. of Queries</th>
<th>Sample Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme Violence</td>
<td>1619</td>
<td>woman beheaded video how many pills does it take to kill yourself growing marijuana indoors for beginners</td>
</tr>
<tr>
<td>Self Harm</td>
<td>2241</td>
<td>new zealanders hate americans anti islam shirts</td>
</tr>
<tr>
<td>Illegal Activity</td>
<td>1124</td>
<td>gay are destroying this countrybatch clothing for women</td>
</tr>
<tr>
<td>Race</td>
<td></td>
<td>jokes about short people</td>
</tr>
<tr>
<td>Religion</td>
<td></td>
<td>anti gohmer stupid quotes</td>
</tr>
<tr>
<td>Sexual Orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Offensive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Celebrity</td>
<td>74057</td>
<td>20 adjectives that describe chocolate what is the order of the planets</td>
</tr>
<tr>
<td>Clean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>79041</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inappropriate</td>
<td>4594</td>
<td>212</td>
<td>178</td>
<td>4984</td>
</tr>
<tr>
<td>Clean</td>
<td>65447</td>
<td>4788</td>
<td>3822</td>
<td>74057</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>70041</strong></td>
<td><strong>5000</strong></td>
<td><strong>4000</strong></td>
<td><strong>79041</strong></td>
</tr>
</tbody>
</table>

Table 2. Evaluation Dataset Label Distribution across Train, Validation and Test Sets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKF</td>
<td>0.625</td>
<td>0.2142</td>
<td>0.3190</td>
</tr>
<tr>
<td>BDT</td>
<td>0.7926</td>
<td>0.2784</td>
<td>0.4120</td>
</tr>
<tr>
<td>BDT-DSSM</td>
<td>0.9474</td>
<td>0.3051</td>
<td>0.4615</td>
</tr>
<tr>
<td>SVM</td>
<td>0.8322</td>
<td>0.3593</td>
<td>0.5019</td>
</tr>
<tr>
<td>SVM-DSSM</td>
<td>0.9241</td>
<td>0.4101</td>
<td>0.5680</td>
</tr>
<tr>
<td>CNN</td>
<td>0.7148</td>
<td>0.8952</td>
<td>0.7949</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.8862</td>
<td>0.7047</td>
<td>0.7850</td>
</tr>
<tr>
<td>BLSTM</td>
<td>0.8018</td>
<td>0.8285</td>
<td>0.8149</td>
</tr>
<tr>
<td>C-BiLSTM</td>
<td>0.9246</td>
<td>0.8251</td>
<td><strong>0.8720</strong></td>
</tr>
</tbody>
</table>

Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
  • LSTMs for inappropriate query suggestion detection
  • **Offline Spell Correction**
  • Online Spell Correction
• Summary and Future Trends [5 min]
Spell correction using soft-masked BERT

- $X = (x_1, x_2, \ldots, x_n) \rightarrow Y = (y_1, y_2, \ldots, y_n)$
- Can be modelled as sequential labeling
- Usually no or only a few characters need to be replaced and all or most of the characters should be copied.
- Naïve method
  - Select a character from a list of candidates for correction (including non-correction) at each position of the sentence using BERT.
    - Sub-optimal because BERT does not have sufficient capability to detect whether there is an error at each position, due to the way of pre-training it using MLM.
- Hence, soft-masked BERT has
  - A network for error detection based on bi-GRU (predicts the probabilities of errors)
  - A network for error correction based on BERT (predicts the probabilities of error corrections)
  - The former is connected to the latter with soft-masking technique.


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Soft-masked BERT

- Detection Network
  - Input: sum of char embedding, position embedding, and segment embedding of the character, as in BERT.
  - Output is a sequence of labels \( G = (g_1, g_2, \ldots, g_n) \), where \( g_i = 1 \) means the character is incorrect and 0 means it is correct.
  - For \( i \)-th character, soft-masked embedding is \( e'_i = p_i \cdot e_{mask} + (1 - p_i) e_i \)

- Correction network
  - BERT model whose final layer consists of a softmax function for all characters.
  - There is also a residual connection between the input embeddings and the embeddings at the final layer.

\[
\mathcal{L}_d = - \sum_{i=1}^{n} \log P_d(g_i|X) \\
\mathcal{L}_c = - \sum_{i=1}^{n} \log P_c(y_i|X) \\
\mathcal{L} = \lambda \cdot \mathcal{L}_c + (1 - \lambda) \cdot \mathcal{L}_d
\]

Detection loss
Correction loss


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
  • LSTMs for inappropriate query suggestion detection
  • Offline Spell Correction
  • Online Spell Correction
• Summary and Future Trends [5 min]
Need for online spell correction

- Offline spell corrections are very confident spell corrections.
- Issue: Low coverage.
- Online spell correction
  - Small portions of the prefix can be corrected at trie exploration time paying a penalty cost. Eg change “cbo” with “ceboo”
  - More flexible than Offline spell corrections because small portions of the prefix can be changed
  - More coverage
- Key idea: it is possible to jump to a different node in the search trie paying a cost dictated from the Conversion Table

<table>
<thead>
<tr>
<th>Cause</th>
<th>Misspelling</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typing quickly</td>
<td>exxit</td>
<td>exit</td>
</tr>
<tr>
<td></td>
<td>mispell</td>
<td>misspell</td>
</tr>
<tr>
<td>Keyboard adjacency</td>
<td>importantmt</td>
<td>important</td>
</tr>
<tr>
<td>Inconsistent rules</td>
<td>conceive</td>
<td>conceive</td>
</tr>
<tr>
<td></td>
<td>conceirge</td>
<td>concierge</td>
</tr>
<tr>
<td>Ambiguous word breaking</td>
<td>silver light</td>
<td>silverlight</td>
</tr>
<tr>
<td>New words</td>
<td>kinnect</td>
<td>kinect</td>
</tr>
</tbody>
</table>

Trie with online spell correction


### Conversion Table

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>5000</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>5000</td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td>6000</td>
</tr>
</tbody>
</table>

- Each node stores the minimum score in the subtree
- The Conversion Table stores the cost of a correction

Exploration: Path are explored with exact matching or paying some conversion cost

Example for Prefix “a”: the explored trie is highlighted in yellow and this is the resulting priority queue:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>d</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>1000</td>
<td>1000</td>
<td>5000</td>
</tr>
</tbody>
</table>
Learning conversion rules

- Utilizing spelling correction pairs, we train a Markov n-gram transformation model that captures user spelling behavior in an unsupervised fashion.
- Joint-sequence modeling framework to define the probability of transforming the original query into the observed character sequence.
- Treat the desired and realized queries as a sequence of substring transformation units, or transfemes.
  - E.g., the transformation Britney→britny can be segmented into the transfeme sequence \{br→br, i→i, t→t, ney→ny\}, where only the last transfeme, , involves a correction.
- We can decompose the probability of the overall transformation sequence as a product of the transfeme probabilities, each conditioned on the previous transfemes.
- By applying the Markov assumption and experimenting with the length of the transfeme units, we can build transformation models of varying complexities.
- To find the top spell-corrected completion suggestions in real-time, we adapt the A* search algorithm with various pruning heuristics to dynamically expand the search space efficiently.

![Figure 1: Example segmentation into transfemes](image)


DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 (gmanish,punagr)@microsoft.com
Given a sequence of transfemes $s = t_1, t_2, \ldots, t_l$, we can expand the probability of the sequence using the chain rule.

As there are multiple ways to segment a transformation in general, we further model the transformation probability $p(c \rightarrow q)$ as the sum of all possible segmentations.

$S(c \rightarrow q)$ is the set of all possible joint segmentations of $c$ and $q$.

Solution using EM.

\[
p(c \rightarrow q) = \sum_{s \in S(c \rightarrow q)} p(s) = \sum_{s \in S(c \rightarrow q)} \prod_{i \in [1, l]} p(t_i | t_1, \ldots, t_{i-1})
\]
Error correction with char RNNs

• Char RNN model
  • The completion must be error correcting, able to handle small errors in the user’s initial input and provide completions for the most likely “correct” input.
  • The completion must be real-time

• Propose a modified beam search process which integrates with a completion distance-based error correction model, combining the error correction process (as a potential function) together with the language model

• Efficiently perform modified beam search to complete the queries with error correction in real time, by exploiting the greatly overlapped forward propagation process and conducting amortized dynamic programming on the search tree.

Wang, Po-Wei, Huan Zhang, Vijai Mohan, Inderjit S. Dhillon, and J. Zico Kolter. “Realtime query completion via deep language models.” In eCOM@ SIGIR. 2018.

DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22 {gmanish,punagr}@microsoft.com
Completion vs Error Correction

\[ \text{cand} := \{s_{1:m} : 0\}, \quad \text{result} := \{\} \]

For \( t = m; \) cand is not empty; \( t++:\)

\[ \text{cand}_{\text{new}} := \left\{ s_{1:t+1} : \log P(s_{1:t+1} | s_{1:m}) \quad \text{for all } s_{t+1} \in C, \text{ for all } s_{1:t} \in \text{cand} \right\}; \]

\( \text{cand} := \) the most probable \((r - |\text{result}|)\) candidates in \( \text{cand}_{\text{new}}; \)

Move \( s_{1:t+1} \) from \( \text{cand} \) to \( \text{result} \) if \( s_{t+1} \) is EOS symbol;

General Beam Search

\[
\begin{align*}
\arg \max_{\hat{s}_{1:n}} P(\hat{s}_{1:n} | s_{1:m}) & \\
\arg \max_{\hat{s}_{1:n}} & \frac{P(s_{1:m} | \hat{s}_{1:n})P(\hat{s}_{1:n})}{P(s_{1:m})} \quad \text{arg max } \log P(s_{1:m} | \hat{s}_{1:n}) + \log P(\hat{s}_{1:n}).
\end{align*}
\]

log \( P(s_{1:m} | \hat{s}_{1:n}) \) is proportional to the edit distance
second part log \( P(\hat{s}_{1:n}) \) models the prior

Edit Distance v.s. Completion Distance

- we should not count the edit distance for adding words after the last character (of terms) from the user input.
- we change the penalty to an indicator when dealing with the “add” operation in the edit distance algorithm.

\[
\text{dist}_{\text{new}}(j) = \min \begin{cases} 
\text{dist}_{\text{new}}(j-1) + I(s_{j-1} \neq \text{last char}) & \text{add;} \\
\text{dist}_{\text{compl}}(j-1) + 1 & \text{substitute;} \\
\text{dist}_{\text{compl}}(j) + 1 & \text{delete;}
\end{cases}
\]

Wang, Po-Wei, Huan Zhang, Vijai Mohan, Inderjit S. Dhillon, and J. Zico Kolter. "Realtime query completion via deep language models." In eCOM@ SIGIR. 2018.

DL for QAC: Manish Gupta, Puneet Agrawal. IJCAI 22

gmanish,punagr@microsoft.com
Beam Search with Edit Distance

• Dynamic programming algorithm of edit (completion) distance costs $O(m \cdot t)$ to compare two strings of length $m$ and $t$.

• If we apply the algorithm to every candidate in the beam search for the incremental length $t$ which ranges from 1 to $n$, it would add $O(|C|rmn^2)$ overhead to the beam search procedure, where $|C|$ is the size of character set, $r$ is the number of candidates we keep, and $n$ is the length of the final completion.

• We can exploit the fact that every new candidate in the beam search procedure originates incrementally from a previous candidate. That is, only one character is changed.

\[
\text{cand} := \{s_{1:m} : 0\}, \quad \text{result} := \{\}
\]

For $t = m$; cand is not empty; $t++$:

\[
\text{cand}_{\text{new}} := \left\{ s_{1:t+1} : \log P(s_{1:t+1} | s_{1:m}) \right\} \quad \text{for all } s_{t+1} \in C, \text{ for all } s_{1:t} \in \text{cand}
\]

\[
\text{cand} := \text{the most probable } (r - |\text{result}|) \text{ candidates in } \text{cand}_{\text{new}};
\]

Move $s_{1:t+1}$ from cand to result if $s_{t+1}$ is EOS symbol;

\[
\text{cand} := \{\text{empty string "": 0}\}, \quad \text{result} := \{\}
\]

For $t = 0$; cand is not empty; $t++$:

\[
\text{cand}_{\text{new}} := \left\{ \hat{s}_{1:t+1} : \log P(s_{1:m} | \hat{s}_{1:t+1}) + \log P(\hat{s}_{1:t+1}) \right\} \quad \text{for all } \hat{s}_{t+1} \in C, \text{ for all } \hat{s}_{1:t} \in \text{cand}
\]

\[
\text{cand} := \text{the most probable } (r - |\text{result}|) \text{ candidates in } \text{cand}_{\text{new}};
\]

Move $\hat{s}_{1:t+1}$ from cand to result if $s_{t+1}$ is EOS symbol;

Maintain the last col of dist$_{\text{new}}$ for $P(s_{1:m} | \hat{s}_{1:t}) \forall \hat{s}_{1:t} \in \text{cand};$
Agenda

• Components in Query Auto Completion systems [20 min]
• Ranking [20 min]
• Natural Language Generation [20 min]
• Personalization [20 min]
• Handling defective suggestions and prefixes [20 min]
• Summary and Future Trends [5 min]
Summary

• Components in Query Auto Completion systems
  • Ranking suggestions: Most popular completion, Time sensitive suggestions, Location sensitive suggestions, Personalization; Ghosting, Session co-occurrences; Online spell correction, Defect handling; Non-prefix matches, Generating suggestions; Mobile QAC, Enterprise QAC

• Ranking
  • Traditional Machine Learning methods for ranking suggestions; Convolutional Latent Semantic Model; LSTM encoder

• Natural Language Generation
  • RNNs with character and word embeddings; LSTMs with subword embeddings; Hierarchical RNN Encoder-decoder; Next Phrase Prediction with T5; Problems with NLG

• Personalization
  • Traditional Machine Learning methods; Hierarchical RNN Encoder-decoder; GRUs with user and time representations; Transformer-based hierarchical encoder

• Handling defective suggestions and prefixes
  • LSTMs for inappropriate query suggestion detection; Offline Spell Correction; Online Spell Correction
Extreme Multi-label Classification (XC/XMR) for QAC

• Given a data point, tag it with the most relevant subset of labels from a very large set of \( L \) labels

• Aspects
  • Set of labels very large – \( L \) in the 1 billion+. For any data point, few e.g. \( O(\log L) \) labels relevant
  • A few labels are “head” labels, have lots of training points. Most labels are “tail” labels, very few (often < 5) training points.
  • Main challenge: predict tail labels (where diversity lies) accurately

• Session-aware QAC can be framed as a multi-label ranking task where the input is the user’s prefix and previous queries, and the observed next query is the ground-truth label.

• Multiple methods exist: Parabel, XFC, NGAME, DeepXML, etc.
Personalized NLG

- Obtaining clean training data is difficult.
  - Not all sessions are personalizable
- Better ways of using session embeddings or user context signals as input.
- Multi-lingual support
- Lookup + generation: How to leverage trie signals to improve generation?
References


References


network for map query suggestion. IEEE Transactions on Knowledge and Data Engineering, 29(9):1888–1901, 2017.

[15] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-
decoder for generative context-aware query suggestion. In proceedings of the 24th ACM international on conference on information and knowledge

[16] Po-Wei Wang, Huan Zhang, Vijai Mohan, Inderjit S Dhillon, and J Zico Kolter. Realtime query completion via deep language models. In eCOM@
SIGIR, 2018.


[18] Harish Yenala, Manoj Chinnakotla, and Jay Goyal. Convolutional bi-directional lstm for detecting inappropriate query suggestions in web search. In

[19] Di Yin, Jiwei Tan, Zhe Zhang, Hongbo Deng, Shujian Huang, and Jiajun Chen. Learning to generate personalized query auto-completions via a multi-
view multitask attentive approach. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages

Thanks!

• Questions: {gmanish,punagr}@microsoft.com

• Connect with us:
  • http://aka.ms/manishgupta, https://sites.google.com/view/manishg/
  • https://www.linkedin.com/in/puneet-agrawal-2291808/