Deep Learning Methods for Query Auto Completion

https://aka.ms/dl4qac

Manish Gupta, Puneet Agrawal

{gmanish, punagr}@microsoft.com

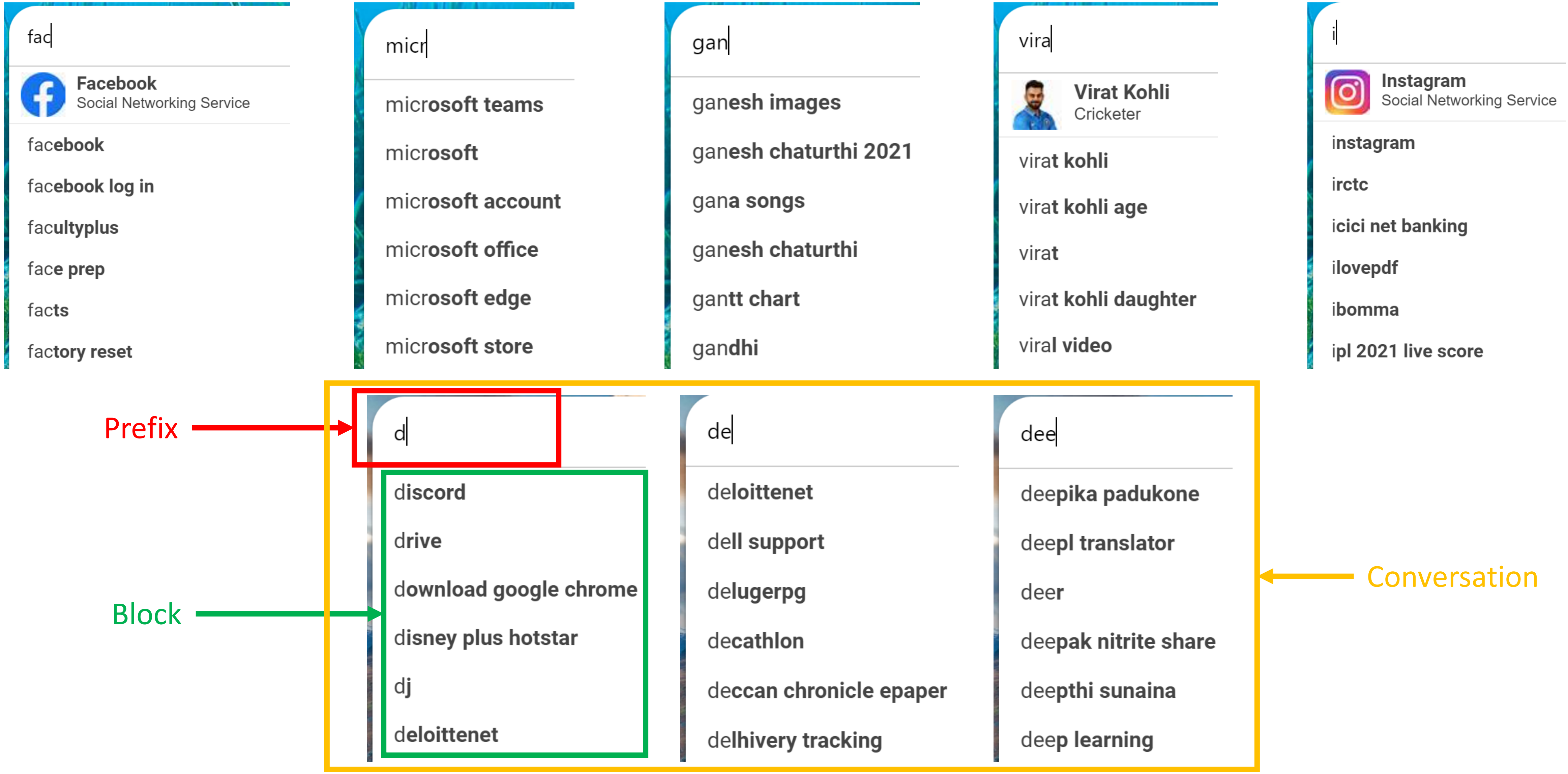
Agenda

- 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

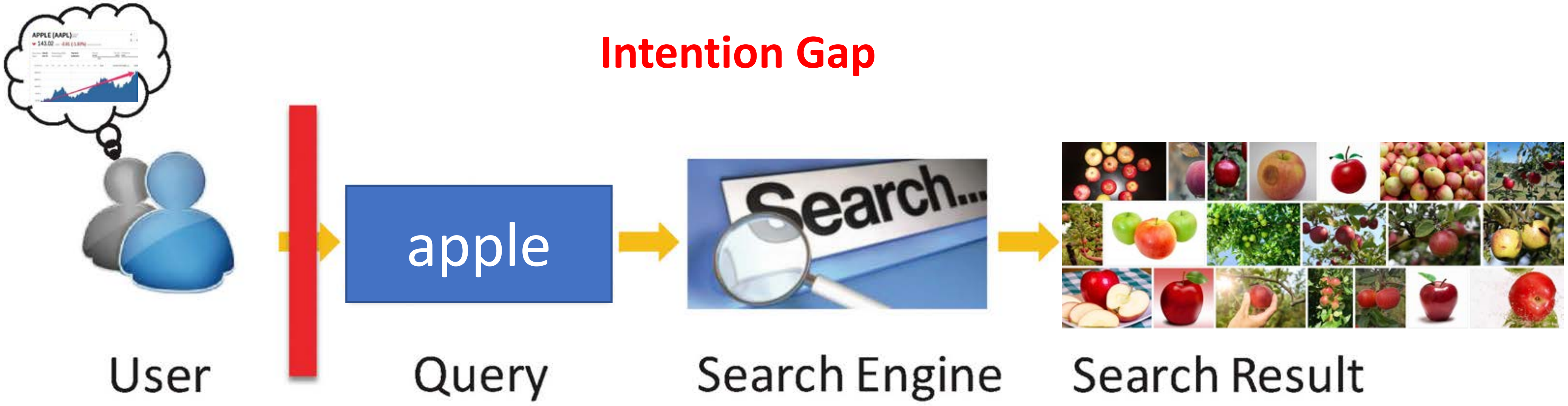
- **Components in Query Auto Completion systems [20 min]**
- Ranking [20 min]
- Natural Language Generation [20 min]
- Personalization [20 min]
- Handling defective suggestions and prefixes [20 min]
- Summary and Future Trends [5 min]

AutoSuggest Examples



 {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

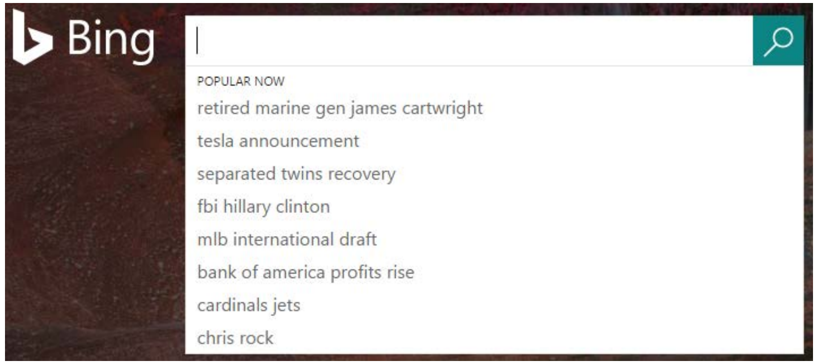
Whiting, Stewart, Andrew James McMinn, and Joemon M. Jose. "Exploring Real-Time Temporal Query Auto-Completion." In *DIR*, pp. 12-15. 2013.

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]

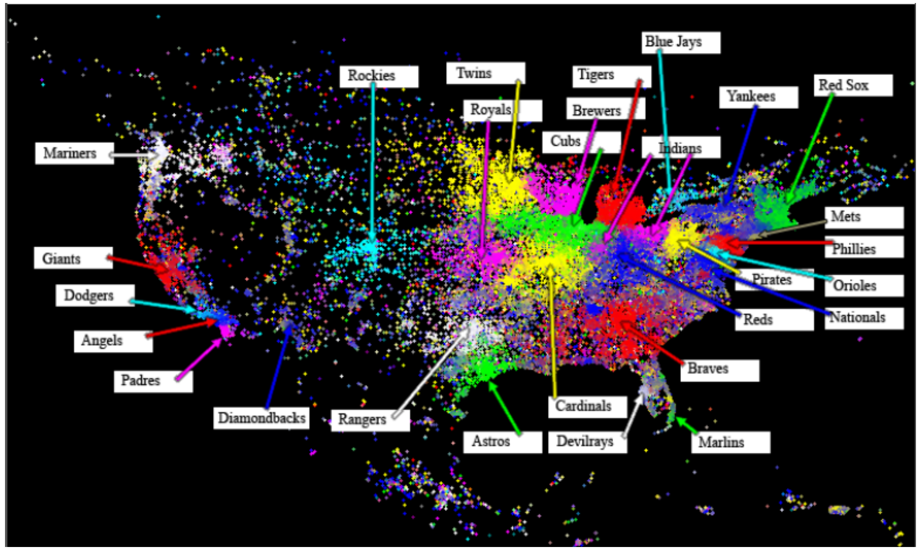
Figure 1: Google auto-completion candidates after typing *di* on Sunday, February 13th, 2012.

Daily frequencies for queries *dictionary* (red) and *disney* (blue) during January 2012 according to Google Trends (the snapshot was taken on Monday, 13-Feb-2012). Among the two queries, *disney* is more popular on weekends, while *dictionary* is issued more commonly by users on weekdays.



Shokouhi, Milad, and Kira Radinsky. "Time-sensitive query auto-completion." In *SIGIR*, pp. 601-610. 2012.
Wang, Yingfei, Hua Ouyang, Hongbo Deng, and Yi Chang. "Learning online trends for interactive query auto-completion." *IEEE Transactions on Knowledge and Data Engineering* 29, no. 11 (2017): 2442-2454.

Ranking suggestions: Location sensitive suggestions

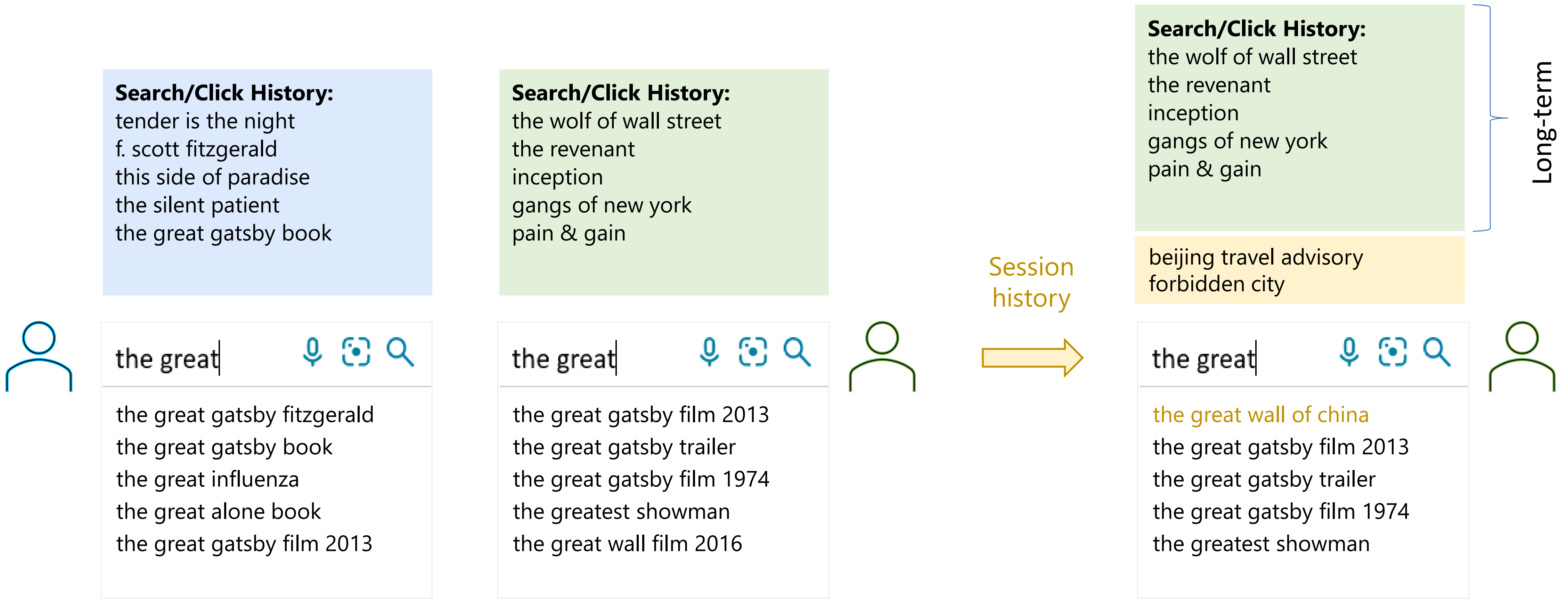


- A user in San Diego types “Uni”.
  - “Univ of California, San Diego” is ok.
  - “Univ of California, Los Angeles” is not ok at the top.
- Location sensitivity of queries
  - Local interest queries [Backstrom, 2008]
    - Queries only interested by users at particular location
      - e.g., name of 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

Backstrom, Lars, Jon Kleinberg, Ravi Kumar, and Jasmine Novak. "Spatial variation in search engine queries." In *WWW*, pp. 357-366. 2008.
Welch, Michael J., and Junghoo Cho. "Automatically identifying localizable queries." In SIGIR, pp. 507-514. 2008.

Ranking Suggestions: Personalization

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



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]

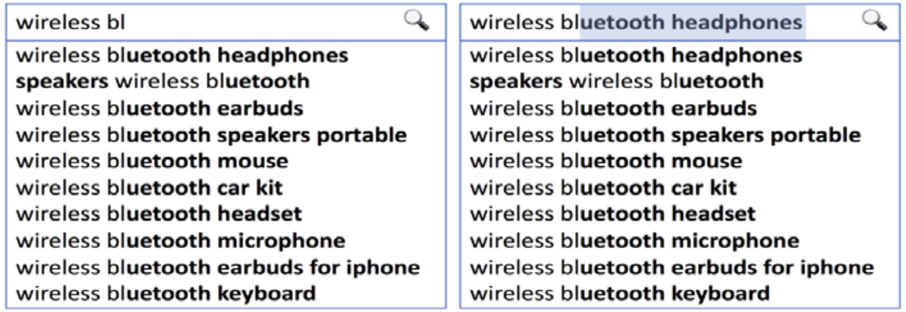


Figure 1: Default QAC experience (left) and QAC with ghosting (right) for prefix "wireless bl"

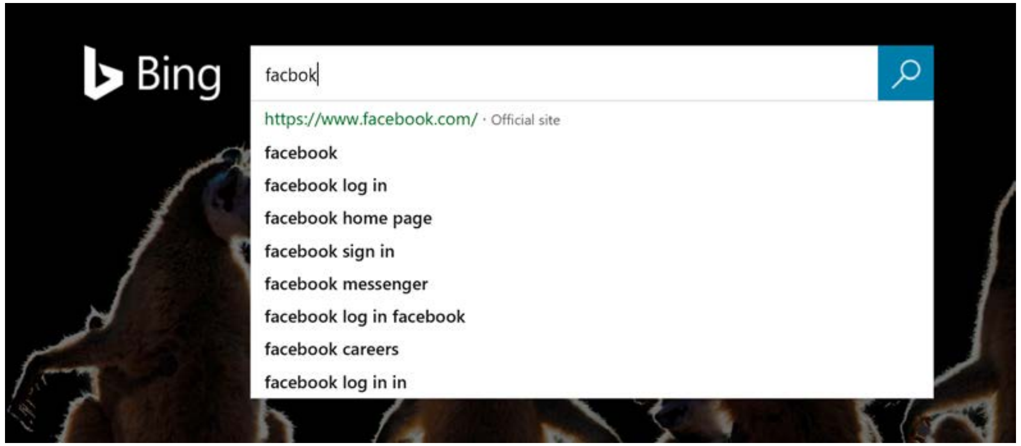
- Context sensitive AS [Bar-Yossef et al. 2011]
  - If after the query “richard nixon” the most popular successive query starting with “am” is “american presidents”, the search engine will suggest “american presidents” as its top completion.
  - Based on existence of reoccurring query sequences in search 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).

Ramachandran, Lakshmi, and Uma Murthy. "Ghosting: contextualized query auto-completion on Amazon search." In *SIGIR*, pp. 1377-1378. 2019.
Bar-Yossef, Ziv, and Naama Kraus. "Context-sensitive query auto-completion." In *WWW*, pp. 107-116. 2011.

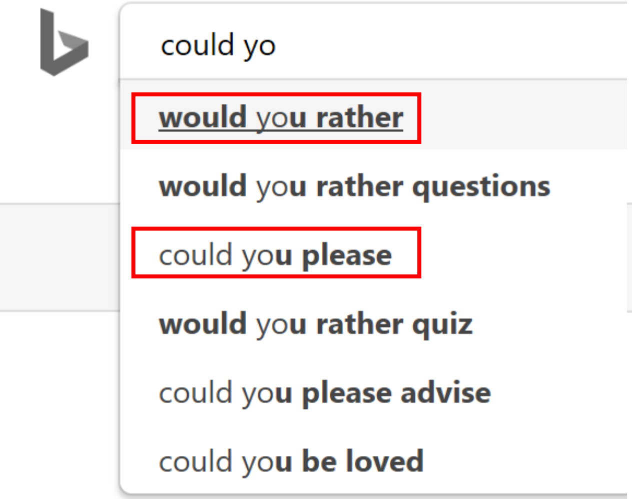
Online spell correction, Defect handling

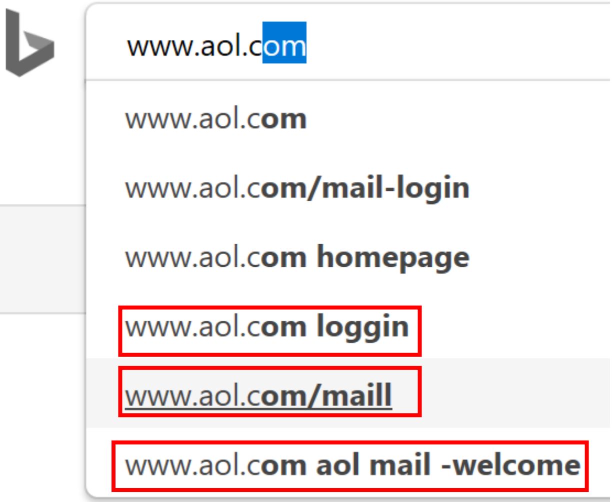
Small portions of the prefix can be corrected at trie exploration time paying a penalty cost. E.g. "cbo" → "ceboo"

- 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

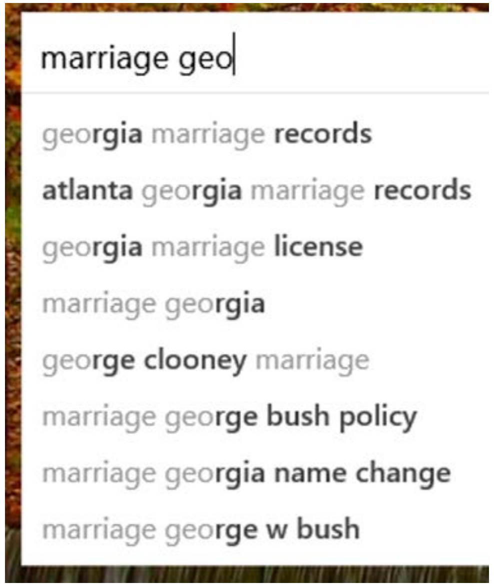




Duan, Huizhong, and Bo-June Hsu. "Online spelling correction for query completion." In *Proceedings of the 20th international conference on World wide web*, pp. 117-126. 2011.

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.

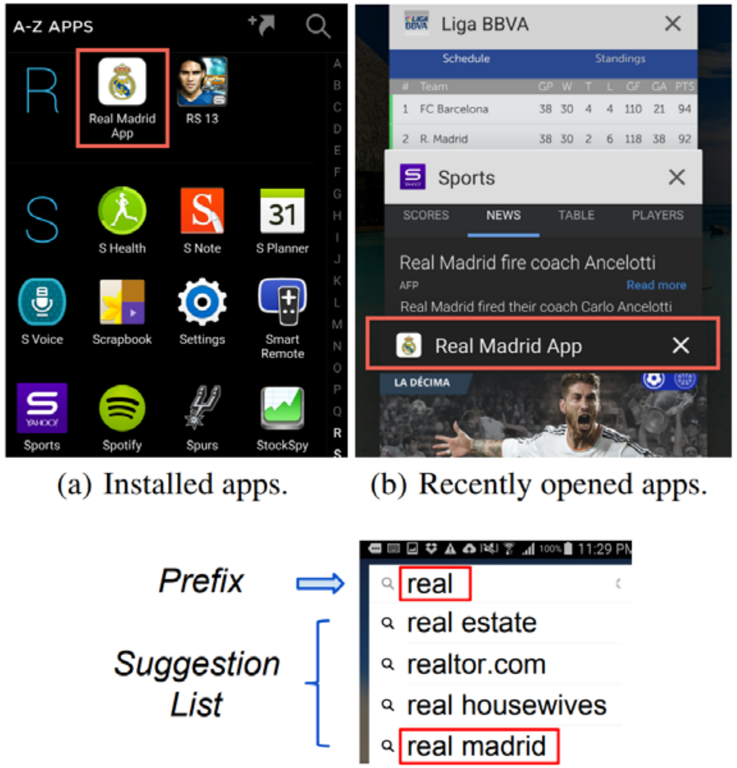


- A significant proportion of queries issued daily have never been seen previously.
- Generate suggestions
  - Improves recall in tail.
  - Deep learning NLG
  - FST: Finite state transducers
  - N-gram models
- Issues
  - Latency
  - Partial suggestions, offensive suggestions, hallucinations, grammatically-incorrect suggestions
  - Personalization

Gog, Simon, Giulio Ermanno Pibiri, and Rossano Venturini. "Efficient and effective query auto-completion." In *SIGIR*, pp. 2271-2280. 2020.

{gmanish,punagr}@microsoft.com

Mobile QAC, Enterprise QAC



(a) Installed apps. (b) Recently opened apps.

(c) Mobile query auto-completion.

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".

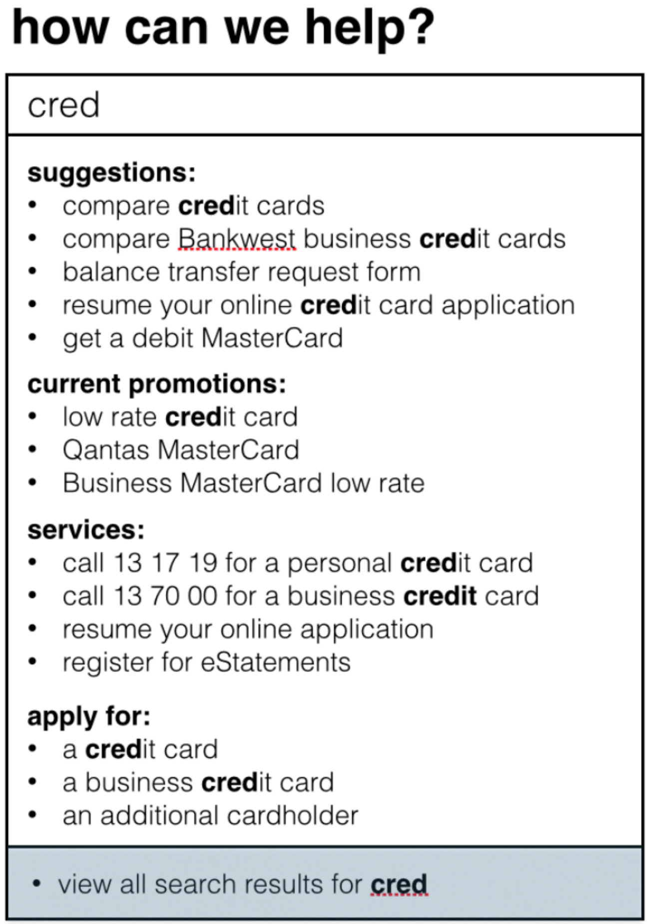


Figure 2: Suggestions from the extended query auto-completion system at `www.bankwest.com.au` on 26 Aug 2013.

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.
Hawking, David, and Kathy Griffiths. "An enterprise search paradigm based on extended query auto-completion: do we still need search and navigation?." In *Proceedings of the 18th Australasian Document Computing Symposium*, pp. 18-25. 2013.

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.

Sordoni, Alessandro, 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 *CIKM*, pp. 553-562. 2015.
Cai, Fei, and Maarten de Rijke. "Learning from homologous queries and semantically related terms for query auto completion." *Information Processing & Management* 52, no. 4 (2016): 628-643.

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.

Sordoni, Alessandro, 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 *CIKM*, pp. 553-562. 2015.
Cai, Fei, and Maarten de Rijke. "Learning from homologous queries and semantically related terms for query auto completion." *Information Processing & Management* 52, no. 4 (2016): 628-643.

Reformulation Features

Category	Feature Class	Description	Formulas
Term	Term Combination (16 features)	number of terms	$\lvert \cup_{i=1}^{T} S(q_i) \rvert$, $\lvert S(q_{T-1}) \cup S(q_T) \rvert$
		term keeping	$\lvert \cap_{i=1}^{T} S(q_i) \rvert$, $\lvert S(q_{T-1}) \cap S(q_T) \rvert$, $\mathrm{sgn}(\lvert S(q_{T-1}) \cap S(q_T) \rvert)$
		term adding	$\lvert S(q_T) - S(q_{T-1}) \rvert$, $\mathrm{sgn}(\lvert S(q_T) - S(q_{T-1}) \rvert)$
		term removing	$\lvert S(q_{T-1}) - S(q_T) \rvert$, $\mathrm{sgn}(\lvert S(q_{T-1}) - S(q_T) \rvert)$
		number of used terms	$\lvert S_{\mathrm{used}}(q_T) \rvert$, $\lvert S(q_T) - S_{\mathrm{used}}(q_T) \rvert$
		ratio of used terms	$\lvert S_{\mathrm{used}}(q_T) \rvert / \lvert S(q_T) \rvert$, $1 - \lvert S_{\mathrm{used}}(q_T) \rvert / \lvert S(q_T) \rvert$
		number of repeat times	$\mathrm{Rep}(q_T)$, $\mathrm{Rep}(q_T)/T$, $\mathrm{Rep}(q_T)/\lvert S(q_T) \rvert$
Query	Query Similarity (10 features)	cosine similarity	$\mathrm{sim}_{\cos}(q_{T-1}, q_T)$
		average cosine similarity	$\frac{1}{T-1}\sum_{i=1}^{T-1} \mathrm{sim}_{\cos}(q_i, q_{i+1}))$, $\frac{1}{T-1}\sum_{i=1}^{T-1} \mathrm{sim}_{\cos}(q_i, q_T))$
		trends of cosine similarity	$\mathrm{sim}_{\cos}(q_{T-1}, q_T)/\frac{1}{T-2}\sum_{i=1}^{T-2} \mathrm{sim}_{\cos}(q_i, q_{i+1}))$
			$\mathrm{sim}_{\cos}(q_{T-1}, q_T)/\frac{1}{T-2}\sum_{i=1}^{T-2} \mathrm{sim}_{\cos}(q_i, q_T))$
		Lev. similarity	$\mathrm{sim}_{\mathrm{Lev}}(q_{T-1}, q_T)$
		average Lev. similarity	$\frac{1}{T-1}\sum_{i=1}^{T-1} \mathrm{sim}_{\mathrm{Lev}}(q_i, q_{i+1}))$, $\frac{1}{T-1}\sum_{i=1}^{T-1} \mathrm{sim}_{\mathrm{Lev}}(q_i, q_T))$
		trends of Lev. similarity	$\mathrm{sim}_{\mathrm{Lev}}(q_{T-1}, q_T)/\frac{1}{T-2}\sum_{i=1}^{T-2} \mathrm{sim}_{\mathrm{Lev}}(q_i, q_{i+1}))$
			$\mathrm{sim}_{\mathrm{Lev}}(q_{T-1}, q_T)/\frac{1}{T-2}\sum_{i=1}^{T-2} \mathrm{sim}_{\mathrm{Lev}}(q_i, q_T))$
	Query Length (6 features)	number of terms	$\lvert S(q_T) \rvert$
		average number of terms	$\frac{1}{T-1}\sum_{i=1}^{T-1} \lvert S(q_i) \rvert$, $\frac{1}{T}\sum_{i=1}^{T} \lvert S(q_i) \rvert$, $\lvert S(q_{T-1}) \rvert + \lvert S(q_T) \rvert$
		trends of term number	$\lvert S(q_T) \rvert / \frac{1}{T-1}\sum_{i=1}^{T-1} \lvert S(q_i) \rvert$, $\lvert S(q_{T-1}) \rvert - \lvert S(q_T) \rvert$
	Query Frequency (2 features)	pairwise frequency	$P((q_{T-1}, q_T) \mid q_T)$, $P((q_{T-1}, q_T) \mid q_{T-1})$
Session	Click-through Data (6 features)	previous clicks	c_{T-1} , $\mathrm{sgn}(c_{T-1})$
		number of effective terms	$\lvert C_{\mathrm{eff}}(q_T) \rvert$
		ratio of effective terms	$\lvert C_{\mathrm{eff}}(q_T) \rvert / T$, $\lvert C_{\mathrm{eff}}(q_T) \rvert / \lvert S(q_T) \rvert$, $\lvert C_{\mathrm{eff}}(q_T) \rvert / \lvert S_{\mathrm{used}}(q_T) \rvert$
	Time Duration (2 features)	average time duration	$\frac{1}{T-1}\sum_{i=1}^{T-1}(t_{i+1} - t_i)$
		trends of time duration	$(t_T - t_{T-1})/\frac{1}{T-2}\sum_{i=1}^{T-2}(t_{i+1} - t_i)$
	Position Number (1 feature)	position in the session	(T)

$$\underbrace{q_1 \to q_2 \to \cdots \to \cdot q_{T-1}}_{context} \to q_T$$

- $S(q_i)$: set of terms in query q_i
- If x > 0, then sgn(x) = 1. If x = 0, then 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 $C_{eff}(q_T)$.

Jiang, Jyun-Yu, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. "Learning user reformulation behavior for query auto-completion." In SIGIR, pp. 445-454. 2014.

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

Li, Yanen, Anlei Dong, Hongning Wang, Hongbo Deng, Yi Chang, and ChengXiang Zhai. "A two-dimensional click model for query auto-completion." In SIGIR, pp. 455-464. 2014.
Di Santo, Giovanni, Richard McCreadie, Craig Macdonald, and Iadh Ounis. "Comparing approaches for query autocompletion." In *SIGIR*, pp. 775-778. 2015.

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.

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

*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.

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" → ("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

$$clsmsim(\bar{p}, \bar{s}) = cosine(y_1, y_2) = \frac{y_1^\intercal y_2}{\|y_1\|\|y_2\|}$$

- 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.

Table 2: Most popular query suffixes extracted from the publicly available AOL logs.

Top suffixes	Top 2-word suffixes	Top 3-word suffixes
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

cheapest flight fro	End-term: "fro"
cheapest flight from	End-term: "from"
cheapest flight from	End-term: "from "
cheapest flight from n	End-term: "n"

Mitra, Bhaskar, and Nick Craswell. "Query auto-completion for rare prefixes." In CIKM, pp. 1755-1758. 2015.

CLSM architecture

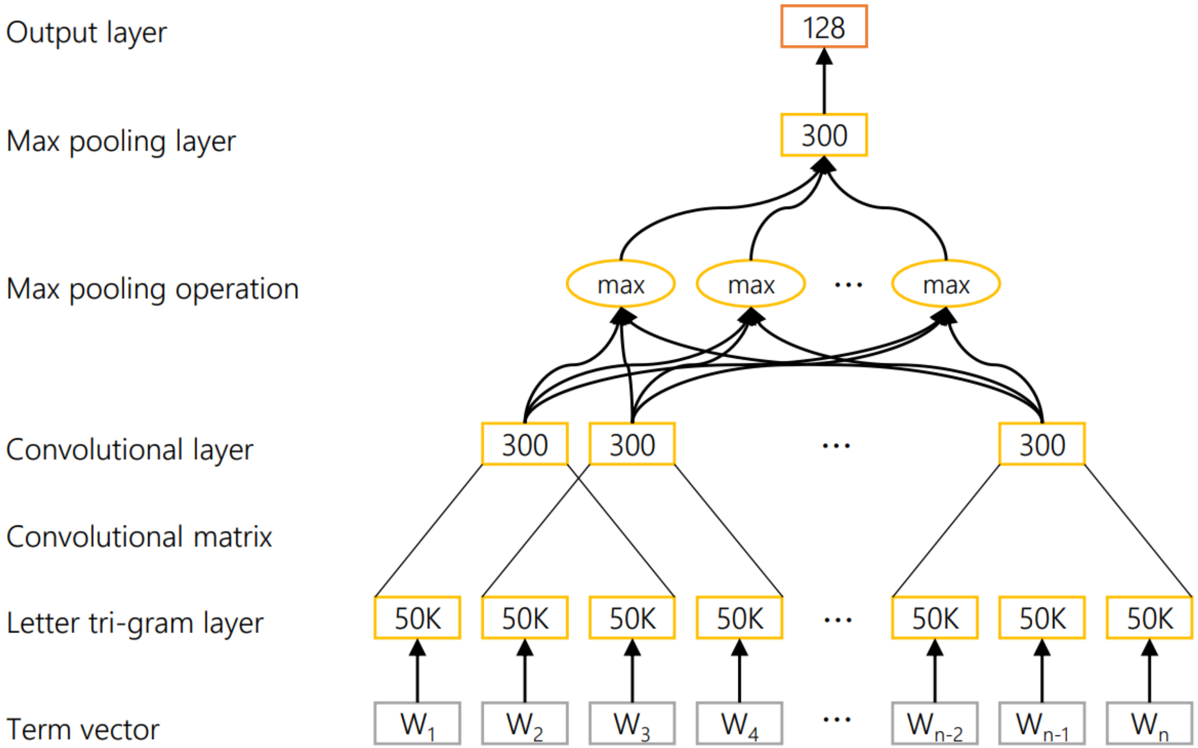


Figure 2: The CLSM model architecture. The model has a convolutional-pooling structure and a 128-dimensional output.

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
what to cook with chicken and broccoli *and bacon*
what to cook with chicken and broccoli *and noodles*
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 onions*
what to cook with chicken and broccoli *and ham soup*

cheapest flights from seattle to
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*

Mitra, Bhaskar, and Nick Craswell. "Query auto-completion for rare prefixes." In CIKM, pp. 1755-1758. 2015.

Results with CLSM

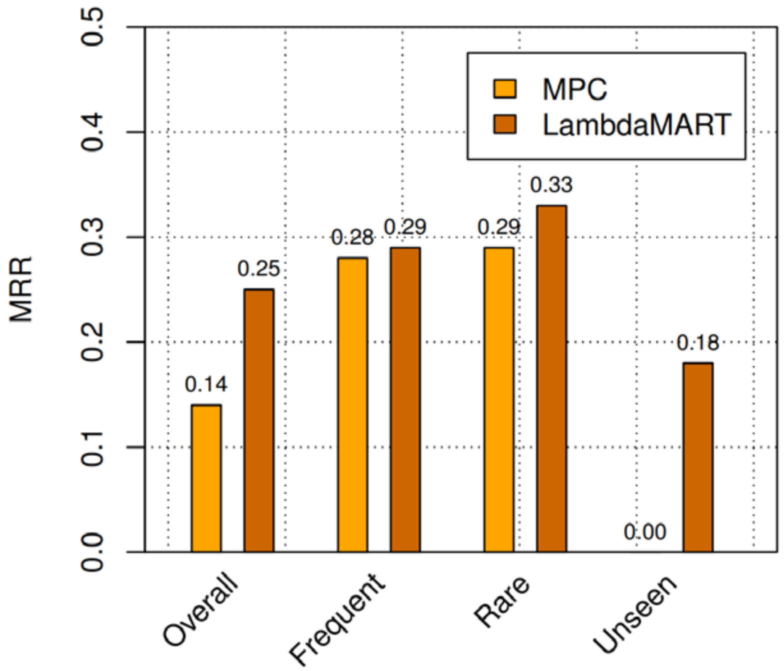


Figure 3: MRR improvements by historical popularity of the input prefix on the AOL testbed. The LambdaMART model uses n-gram and CLSM features and includes suffix-based suggestion candidates. Any prefix in the top 100K most popular prefixes from the background data is considered as *Frequent*. There 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.

	AOL		Bing
Models	MRR	% Improv.	% Improv.
Full-query based candidates only			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.1445	-0.1	-1.7*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.1427	-1.4*	-1.2*
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.1445	-0.1	-1.2*
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.1432	-1.0*	-1.5*
Full-query based candidates + Suffix based candidates (Top 10K suffixes)			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.2116	+46.3*	+32.8*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.2326	+60.8*	+42.6*
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.2249	+55.5*	+40.1*
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.2339	**+61.7***	+43.8*
Full-query based candidates + Suffix based candidates (Top 100K suffixes)			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.2105	+45.5*	+39.9*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.2441	**+68.7***	**+54.2***
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.2248	+55.4*	**+48.9***
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.2453	**+69.6***	**+55.3***

Mitra, Bhaskar, and Nick Craswell. "Query auto-completion for rare prefixes." In CIKM, pp. 1755-1758. 2015.

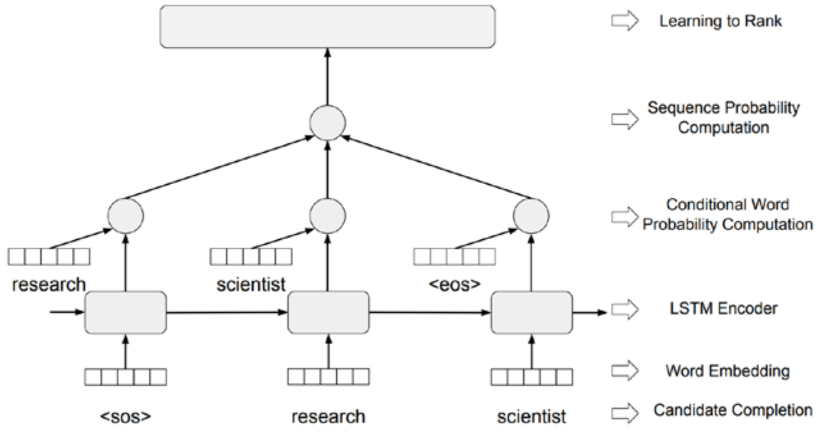
Agenda

- 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.

cheapest flights from seattle to
cheapest *flights from seattle to sfo*
cheapest flights from *seattle to vancouver*
cheapest flights from seattle *to airport*
cheapest flights from seattle *to study*



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.

[Wang, Sida, Weiwei Guo, Huiji Gao, and Bo Long. "Efficient Neural Query Auto Completion." In *CIKM*, pp. 2797-2804. 2020.]

Efficient Generation and Ranking for Neural QAC

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 $p < 0.05$.

Candidate Generation Methods	Recall@10 All	Recall@10 Seen	Recall@10 Unseen
MostPopularCompletion (MPC)	0.2075	0.5091	0.0000
LastWordGeneration (LWG)	0.3884	0.5207	0.2973
MaximumContextGeneration (MCG)	**0.3992†**	**0.5219†**	**0.3147†**

Methods	Latency
MaximumContextGeneration	0.18ms
CLSM	2.15ms
Unnormalized LM	3.01ms
Normalized LM	53.32ms

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.

Generation	Ranking	MRR@10 All	MRR@10 Seen	MRR@10 Unseen
MPC	Frequency	0.1805	0.4431	0.0000
LWG	Frequency	0.3147	0.4465	0.2241
MCG	Frequency	0.3283	0.4469	0.2467
	CLSM	0.3270	0.4229	0.2610
	LSTMEmbed	0.3278† (+0.244%)	0.4224	0.2628†
	UnnormalizedLM	0.3328† (+1.769%)	0.4293†	0.2665†
	NormalizedLM	0.3331† (+1.865%)	0.4293†	0.2669†
	CLSM + Frequency	0.3369	0.4472	0.2610
	LSTMEmbed +Frequency	0.3379‡ (+0.297%)	0.4472	0.2628‡
	UnnormalizedLM +Frequency	**0.3402‡ (+0.980%)**	**0.4473**	**0.2665‡**
	NormalizedLM +Frequency	**0.3404‡ (+1.039%)**	**0.4473**	**0.2669‡**

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

Wang, Sida, Weiwei Guo, Huiji Gao, and Bo Long. "Efficient Neural Query Auto Completion." In *CIKM*, pp. 2797-2804. 2020.

Agenda

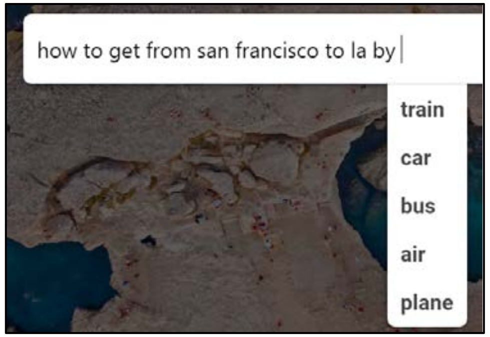
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- **Natural Language Generation [20 min]**
- Personalization [20 min]
- Handling defective suggestions and prefixes [20 min]
- Summary and Future Trends [5 min]

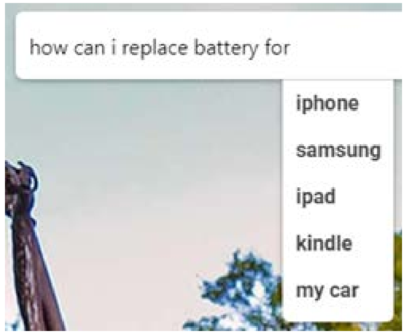
Agenda

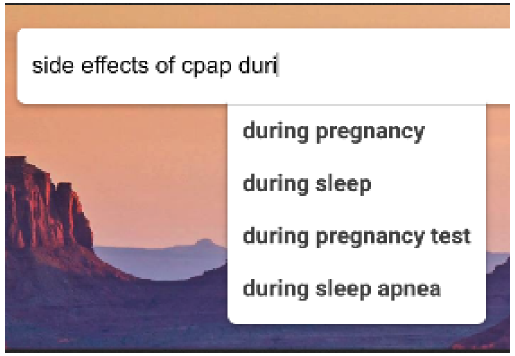
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
  - RNNs with character and word embeddings
  - LSTMs with subword embeddings
  - Hierarchical RNN Encoder-decoder
  - Next Phrase Prediction with T5
  - Problems with NLG
- Personalization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

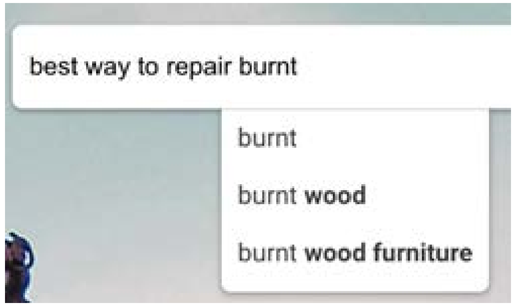
NLG for QAC

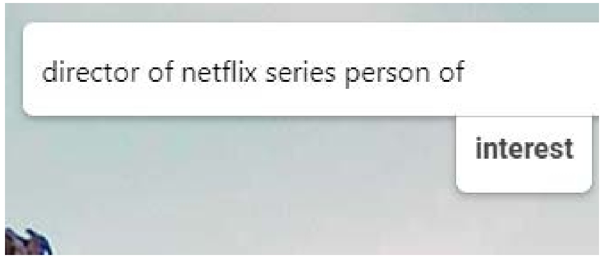
- Technical challenges
  - Handling partial words in the input
  - Optimization of computation requirements and throughput
  - Model compression/distillation
  - Beam search vs greedy decoding
- Considerations
  - Multi-language support
  - Inappropriate leakage
  - Suggestion quality
  - Coverage
  - Latency





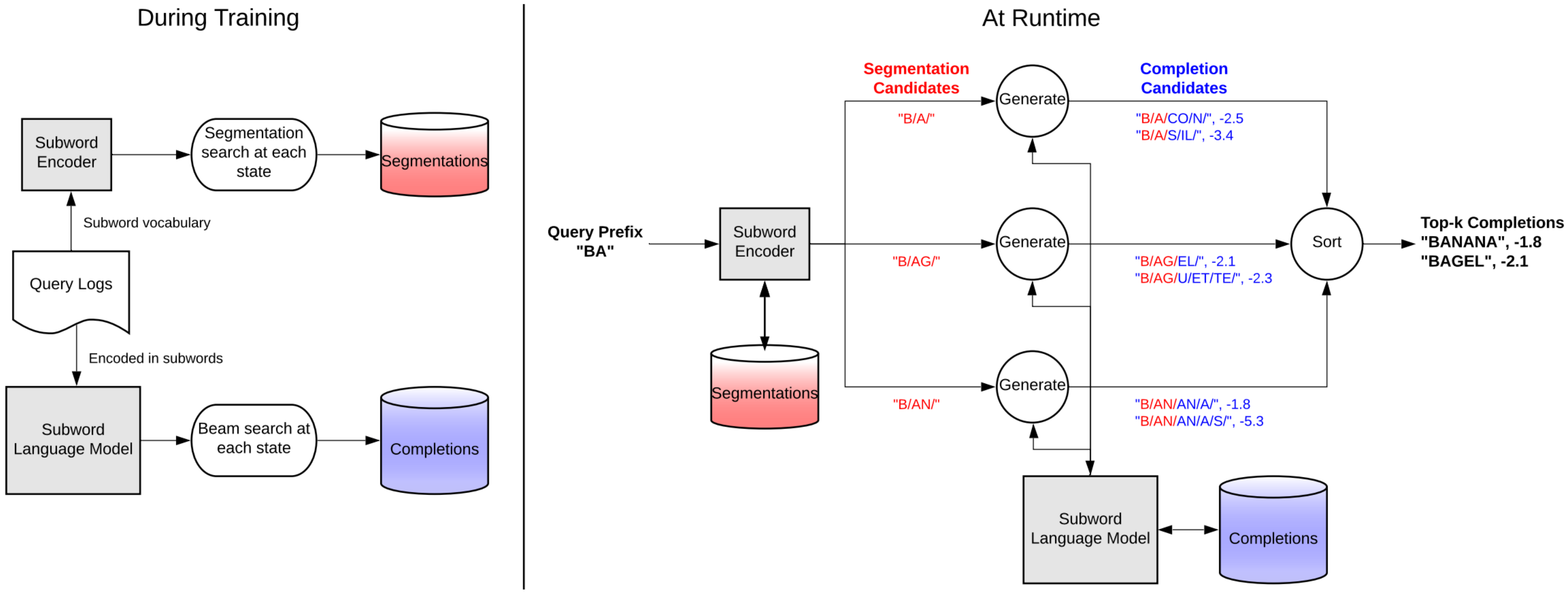






Query Blazer: NLG without deep learning

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



Kang, Young Mo, Wenhao Liu, and Yingbo Zhou. "QueryBlazer: Efficient Query Autocompletion Framework." In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 1020-1028. 2021.

Agenda

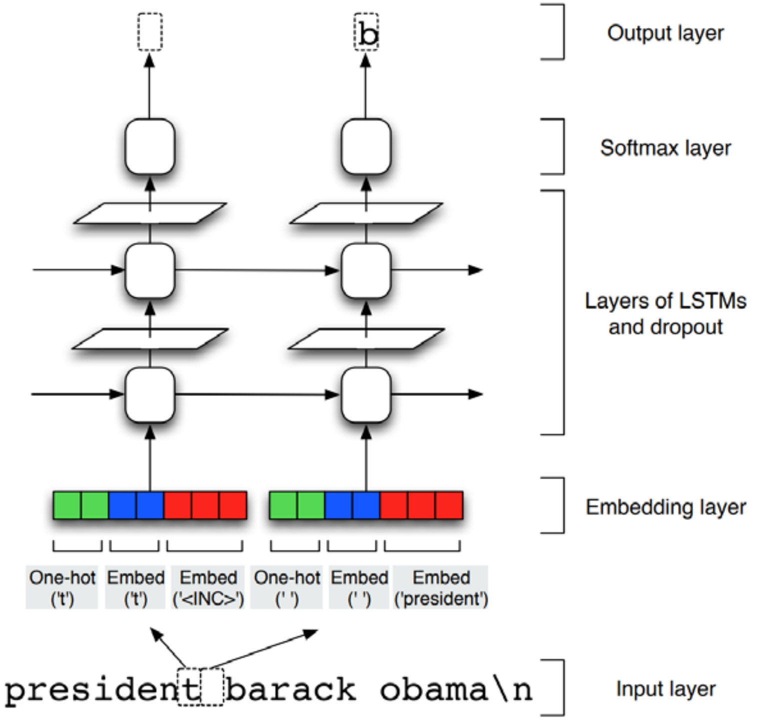
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
  - **RNNs with character and word embeddings**
  - LSTMs with subword embeddings
  - Hierarchical RNN Encoder-decoder
  - Next Phrase Prediction with T5
  - Problems with NLG
- Personalization [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.

when is a good time to buy a	
when is a good time to buy a	**house**
when is a good time to buy a	**home**
when is a good time to buy a	**lyrics**
when is a good time to buy a	**car**
why am i afraid of	
why am i afraid of	**the dark**
why am i afraid of	**the dead**
why am i afraid of	**the dog**
president donald	
president donald	**trump**

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.

Park, Dae Hoon, and Rikio Chiba. "A neural language model for query auto-completion." In SIGIR, pp. 1189-1192. 2017.

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.

	MRR			PMRR		
Model	Seen	Unseen	All	Seen	Unseen	All
MPC [1]	0.428	0.000	0.171	0.566	0.000	0.225
Char. n-gram (n=7)	0.363	0.236	0.287	0.550	0.376	0.445
Mitra10K+MPC+λMART [12]	0.427	0.179	0.278	0.586	0.297	0.412
Mitra100K+MPC+λMART [12]	0.428	0.212	0.298	0.588	0.368	0.455
Proposed models						
NQLM(S)	0.381	0.287	0.325	0.557	0.460	0.499
NQLM(S)+WE	0.406	0.286	0.334	0.582	0.445	0.500
NQLM(L)+WE	0.419	0.303	0.349	0.589	0.465	0.514
NQLM(S)+MPC	0.433	0.287	0.346	0.580	0.460	0.508
NQLM(S)+WE+MPC	**0.434**	0.286	0.345	0.580	0.445	0.499
NQLM(L)+WE+MPC	**0.434**	0.303	**0.355**	0.580	0.465	0.511
NQLM(S)+MPC+λMART	0.428	0.288	0.344	**0.594**	0.465	0.516
NQLM(S)+WE+MPC+λMART	0.428	0.288	0.344	0.590	0.454	0.508
NQLM(L)+WE+MPC+λMART	0.428	**0.305**	0.354	0.593	**0.475**	**0.522**

Park, Dae Hoon, and Rikio Chiba. "A neural language model for query auto-completion." In SIGIR, pp. 1189-1192. 2017.

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.

Kim, Gyuwan. "Subword language model for query auto-completion." *arXiv preprint arXiv:1909.00599* (2019).

Subword Language Model for QAC

- 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.

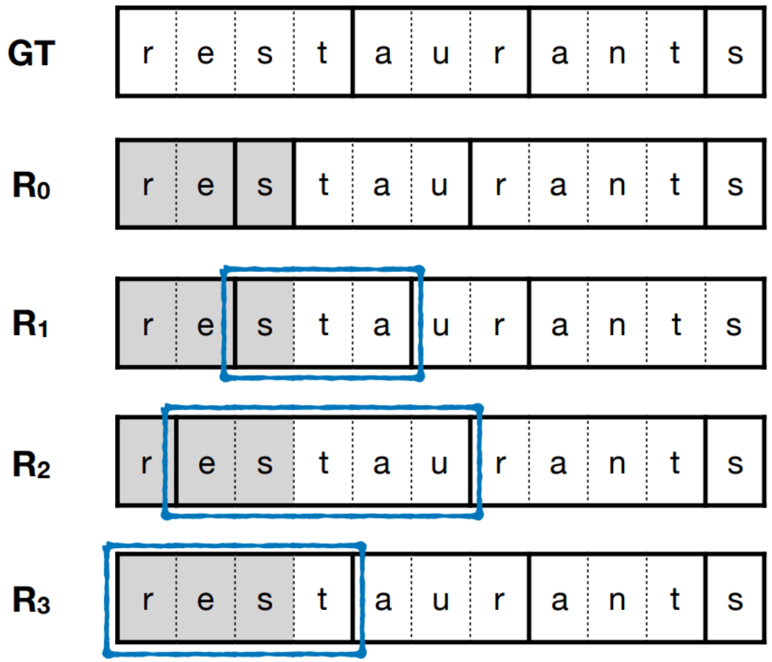


Figure 1: Illustration of retrace algorithm with the example of "restaurants." The gray area means given prefix ("res") of the query. The solid line indicates the boundary of the segmentation. GT is the segmentation of ground truth query. Possible examples of the generated sequence of tokens belonging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

Kim, Gyuwan. "Subword language model for query auto-completion." *arXiv preprint arXiv:1909.00599* (2019).

 {gmanish,punagr}@microsoft.com

Subword Language Model for QAC

- Subword LM is ~2.5x 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.

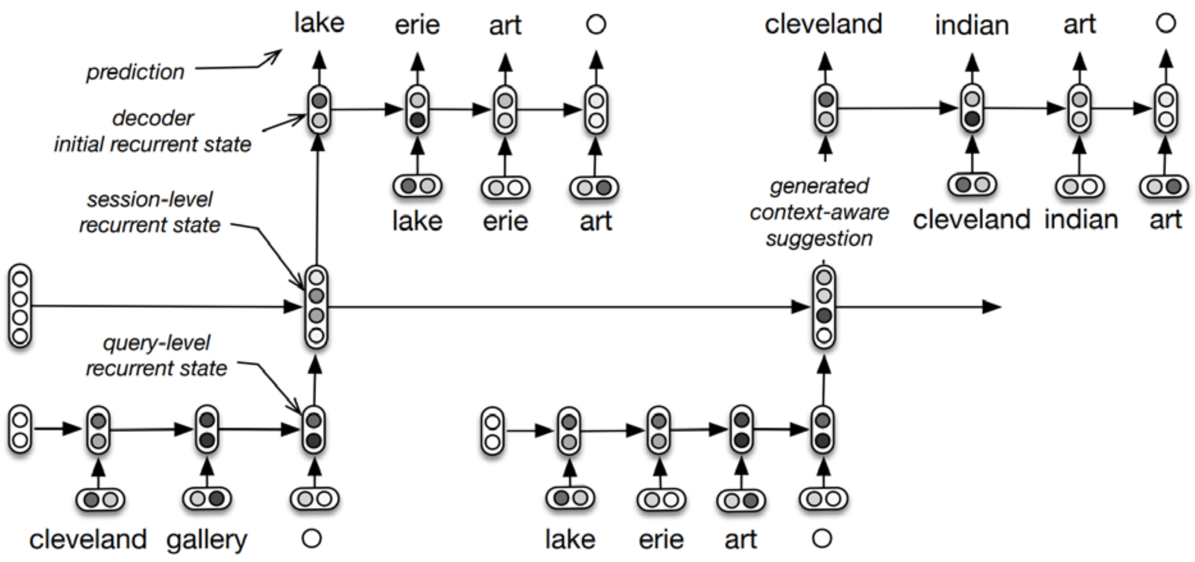
Kim, Gyuwan. "Subword language model for query auto-completion." *arXiv preprint arXiv:1909.00599* (2019).

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]

Hierarchical recurrent encoder-decoder (HRED)

- Given a session S={Q_1,...,Q_M}, we aim to predict the target query Q_M given the context Q_1,...,Q_{M-1}.
- HRED generates synthetic suggestions sampled one word at a time.
- Useful for rare, or long-tail, queries.



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".

Sordoni, Alessandro, 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 *CIKM*, pp. 553-562. 2015.

HRED with LambdaMART

Context	Synthetic Suggestions
ace series drive	ace hardware ace hard drive hp officejet drive ace hardware series
cleveland gallery $\rightarrow$ lake erie art	cleveland indian art lake erie art gallery lake erie picture gallery sandusky ohio art gallery

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.

Sordoni, Alessandro, 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 *CIKM*, pp. 553-562. 2015.

Comparison of HRED with BaselineRanker and ADJ

Method	MRR	Δ%
ADJ	0.5334	-
Baseline Ranker	0.5563	*+4.3%*
+ HRED	**0.5749**	*+7.8%/+3.3%*

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

Method	MRR	Δ%
ADJ	0.3830	-
Baseline Ranker	0.6788	*+77.2%*
+ HRED	**0.7112**	*+85.3% / +5.6%*

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

Method	MRR	Δ%
ADJ	0.4507	-
Baseline Ranker	0.4831	*+7,2%*
+ HRED	**0.5309**	*+17,8%/+9.9%*

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

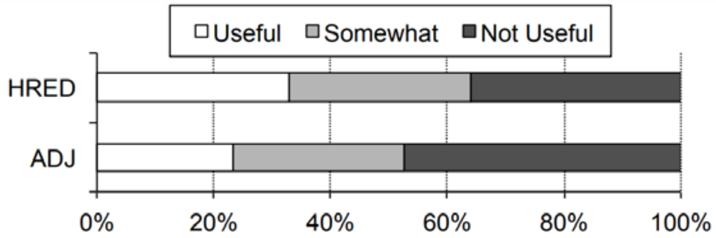


Figure 8: User study results, which compare the effectiveness of HRED with the baseline techniques.

Sordoni, Alessandro, 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 *CIKM*, pp. 553-562. 2015.

RIN: Reformulation Inference Network for Context-Aware Query Suggestion

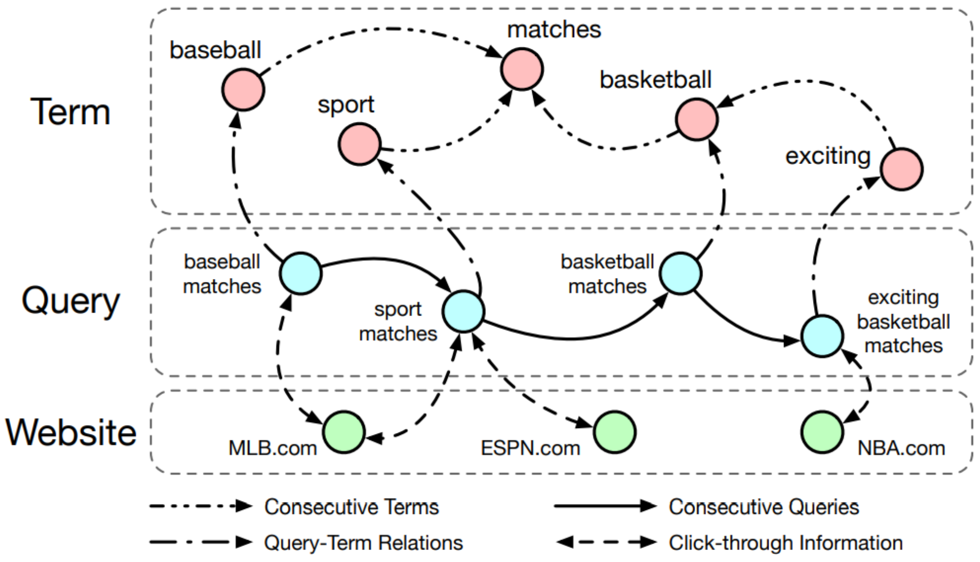
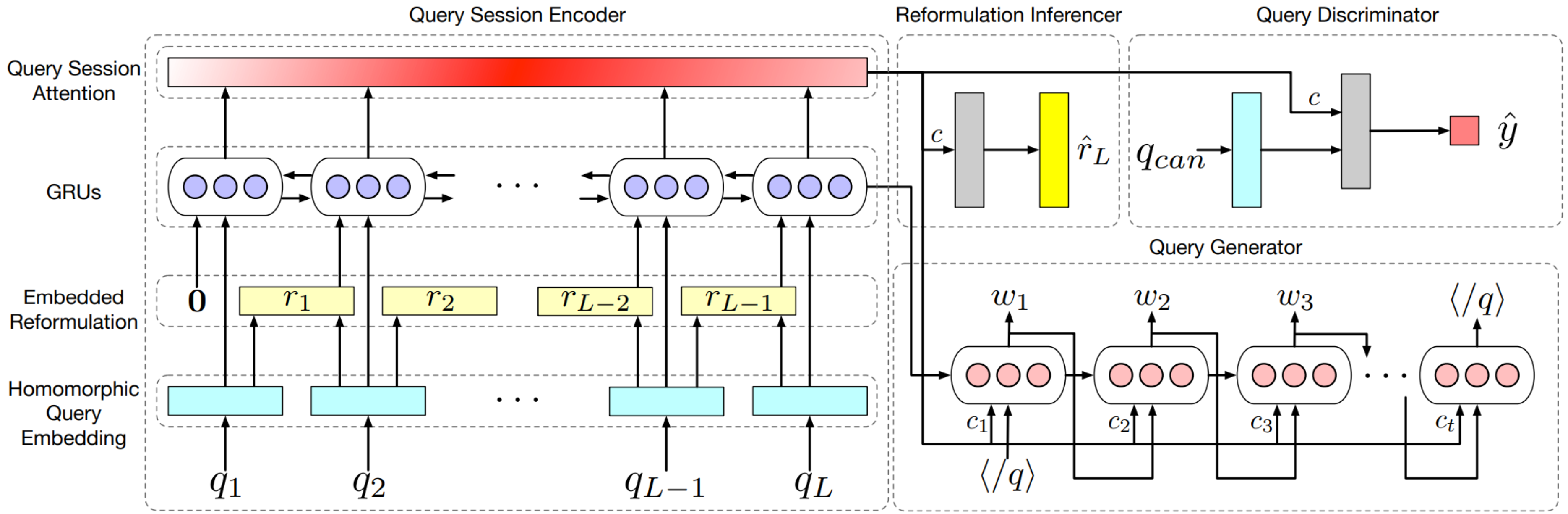


Figure 2: An example of the heterogeneous network constructed by a search session of four queries for deriving term embeddings. Note that the queries in the graph are auxiliary nodes connecting the domains of terms and websites.

- 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 $v_{q_{i+1}} - v_{q_i}$

[Jiang, Jyun-Yu, and Wei Wang. "RIN: Reformulation inference network for context-aware query suggestion." In CIKM, pp. 197-206. 2018.]

RIN: Reformulation Inference Network for Context-Aware Query Suggestion



Self attention

$$u_i = \tanh(\mathcal{F}_s(h_i)),$$

$$\alpha_i = \frac{\exp(u_i^T u_s)}{\sum_{i'} \exp(u_{i'}^T u_s)}$$

$$c = \sum_i \alpha_i h_i$$

- 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

Jiang, Jyun-Yu, and Wei Wang. "RIN: Reformulation inference network for context-aware query suggestion." In CIKM, pp. 197-206. 2018.

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.

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

$$\boldsymbol{u_{t,i}} = \tanh(\mathcal{F}_g([\boldsymbol{s_{t-1}}; \boldsymbol{h_i}])),$$

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

$$\boldsymbol{c_t} = \sum_i \alpha_{t,i} \boldsymbol{h_i},$$

Jiang, Jyun-Yu, and Wei Wang. "RIN: Reformulation inference network for context-aware query suggestion." In CIKM, pp. 197-206. 2018.

RIN: Reformulation Inference Network for Context-Aware Query Suggestion

- Optimization

$$\text{loss}_R = \frac{1}{2}||\boldsymbol{r_L} - \hat{\boldsymbol{r_L}}||_F^2$$

$$\text{loss}_D = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$\text{loss}_G = -\sum_{w_t} \log P(w_t \mid S_t)$$

$$\text{loss} = \text{loss}_R + \text{loss}_{\text{task}}$$

- 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

Dataset	MPS [14, 46]	Hybrid [4]	PC [44]	QVMM [20]	RC [27]	HRED [46]	ACG [14]	RIN
Overall Context	0.5471	0.5823	0.5150	0.5671	0.6202	0.6207	0.6559	**0.8254**
Short Context (1 query)	0.5680	0.5822	0.5343	0.5862	0.5960	0.6100	0.6471	**0.8361**
Medium Context (2 to 3 queries)	0.5167	0.5841	0.4865	0.5338	0.6689	0.6489	0.6542	**0.8190**
Long Context (4 or more queries)	0.4826	0.5768	0.4575	0.5026	0.6704	0.6122	0.6669	**0.7611**

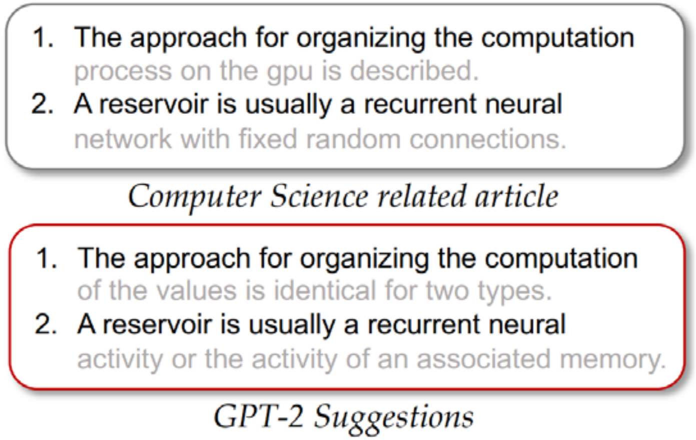
Jiang, Jyun-Yu, and Wei Wang. "RIN: Reformulation inference network for context-aware query suggestion." In CIKM, pp. 197-206. 2018.

Agenda

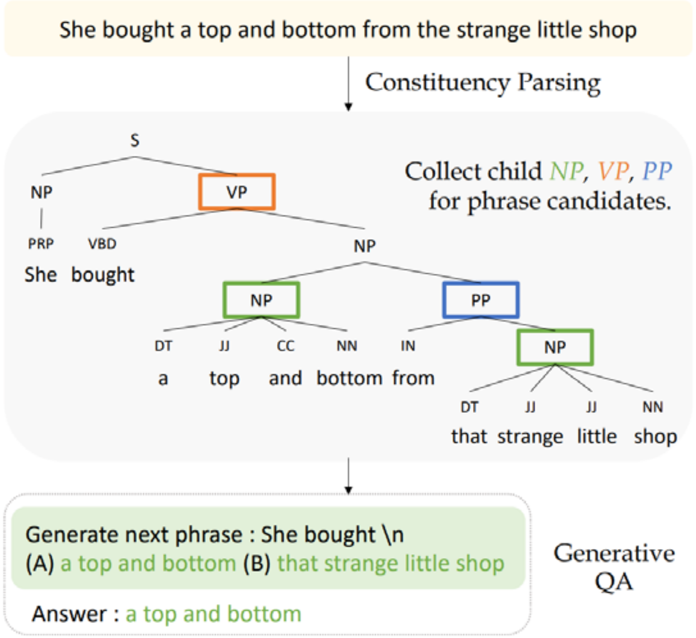
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
    - RNNs with character and word embeddings
    - LSTMs with subword embeddings
    - Hierarchical RNN Encoder-decoder
    - **Next Phrase Prediction with T5**
    - Problems with NLG
- 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.



Lee, Dong-Ho, Zhiqiang Hu, and Roy Ka-Wei Lee. "Improving Text Auto-Completion with Next Phrase Prediction." *arXiv preprint arXiv:2109.07067* (2021).

NPP Results

Model / Metrics	Emails				Academic Writing			
	BLEU-4	METEOR	CIDEr	SPICE	BLEU-4	METEOR	CIDEr	SPICE
GPT-2 (Radford et al., 2019)	1.1	6.6	26.4	3.3	0.6	6.0	23.6	2.6
T5 (Raffel et al., 2020)	2.8	6.8	39.8	4.2	2.2	7.5	50.3	3.9
NSP+T5	3.0	6.9	41.1	4.4	2.3	7.5	51.1	4.0
NPP+T5 (Ours)	**3.2**	**7.1**	**43.0**	**4.5**	**2.5**	**7.8**	**53.5**	**4.2**

Partial Query	**Original**	**T5**	**NPP+T5**
Building large OCR databases is a time	consuming and tedious work .	challenging .	consuming task .
vpi is part of the ieee programming	language interface standard .	system .	language .
a connection between the kalman	filter is developed .	et al .	filter is established .
appendix provides a complete listing	of code for the systems .	of the apl libraries .	of the tools and techniques used in this paper .
automatic target	recognition is an important task .	selection is based on a set of criteria .	detection is a key feature of this approach .

Lee, Dong-Ho, Zhiqiang Hu, and Roy Ka-Wei Lee. "Improving Text Auto-Completion with Next Phrase Prediction." *arXiv preprint arXiv:2109.07067* (2021).

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?

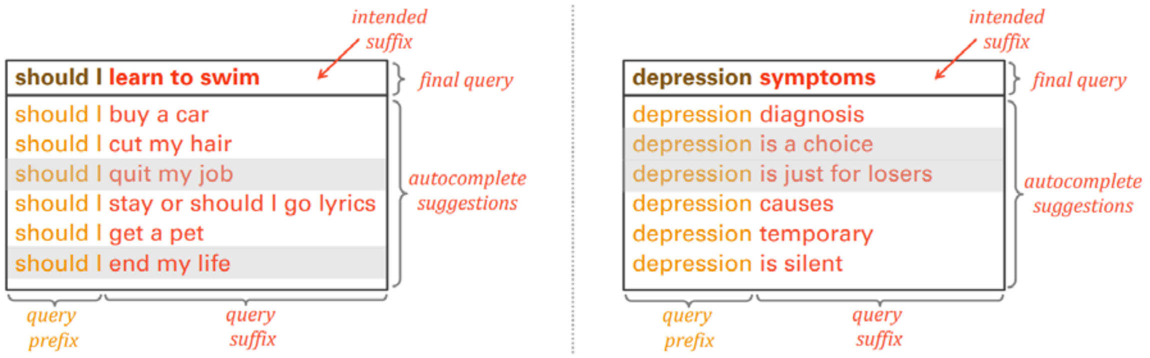


Fig. 1. Examples of search query completion suggestions for different query prefixes.

- 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

- 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

Olteanu, Alexandra, Fernando Diaz, and Gabriella Kazai. "When Are Search Completion Suggestions Problematic?." *Proceedings of the ACM on Human-Computer Interaction* 4, no. CSCW2 (2020): 1-25.

When Are Search Completion Suggestions Problematic?

Problem Category	Working definitions (and sub-categories)	Keywords (p/s: query)
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])

Olteanu, Alexandra, Fernando Diaz, and Gabriella Kazai. "When Are Search Completion Suggestions Problematic?." *Proceedings of the ACM on Human-Computer Interaction* 4, no. CSCW2 (2020): 1-25.

When Are Search Completion Suggestions Problematic?

Problem Category	Working definitions (and sub-categories)	Keywords (p/s: query)
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])

Olteanu, Alexandra, Fernando Diaz, and Gabriella Kazai. "When Are Search Completion Suggestions Problematic?." *Proceedings of the ACM on Human-Computer Interaction* 4, no. CSCW2 (2020): 1-25.

When Are Search Completion Suggestions Problematic?

Target Category	Working definitions (and sub-categories)	Keywords (p/s: query)
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])

Olteanu, Alexandra, Fernando Diaz, and Gabriella Kazai. "When Are Search Completion Suggestions Problematic?." *Proceedings of the ACM on Human-Computer Interaction* 4, no. CSCW2 (2020): 1-25.

When Are Search Completion Suggestions Problematic?

Target Category	Working definitions (and sub-categories)	Keywords (p/s: query)
Animals & objects	• References to an animal, a group of animals, or anything that may be construed as an object or a group of objects	• cat (s: how to poison [a cat]) • knife (p: how to use a knife [to kill])
Activities & ideas	• References to a specific activity, action, or idea	• cutting yourself (p: cutting yourself is [stupid]) • crying (p: crying is [emotional blackmail])
Other targets	• 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	• bipolar disorder (p: bipolar disorder is [fraud]) • vaccination (p: vaccination [herd mentality]) • science (p: science should [stay out of faith])
Generic, no target	• There is no identifiable target or subject	• (what [the heck]) • (damn damn [damn])

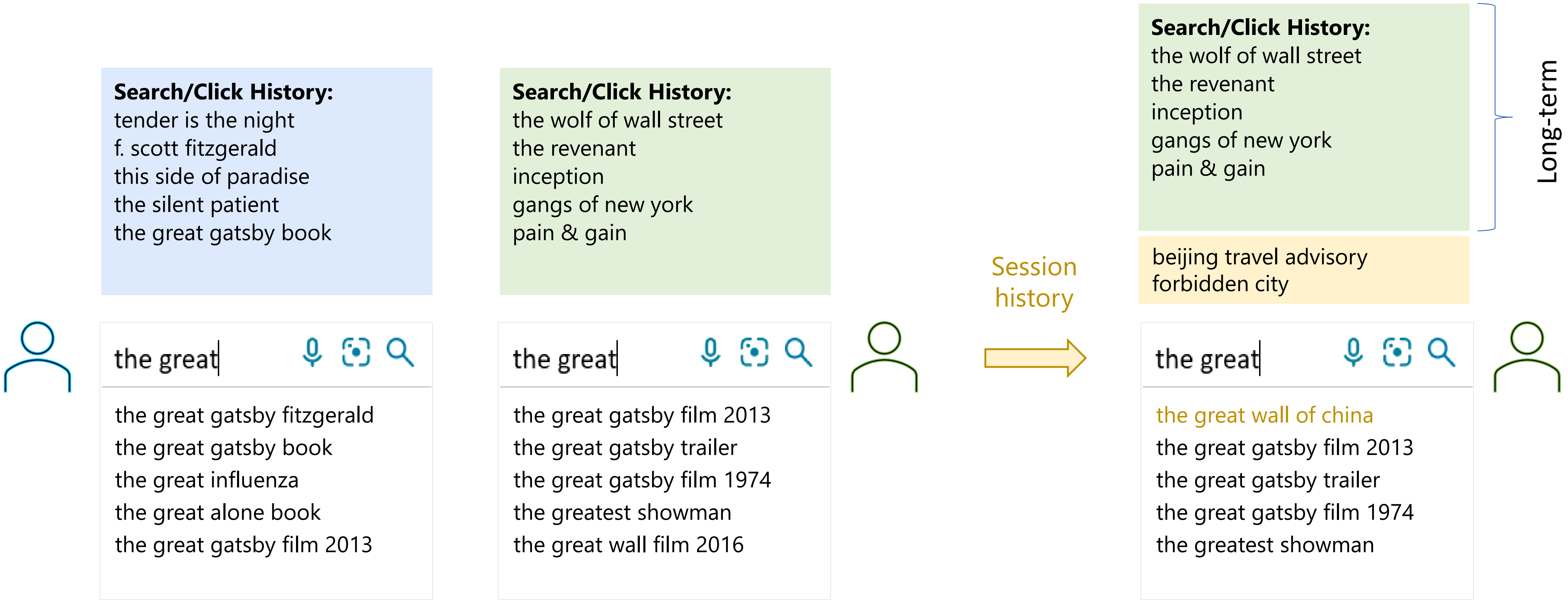
Olteanu, Alexandra, Fernando Diaz, and Gabriella Kazai. "When Are Search Completion Suggestions Problematic?." *Proceedings of the ACM on Human-Computer Interaction* 4, no. CSCW2 (2020): 1-25.

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



{gmanish,punagr}@microsoft.com

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]

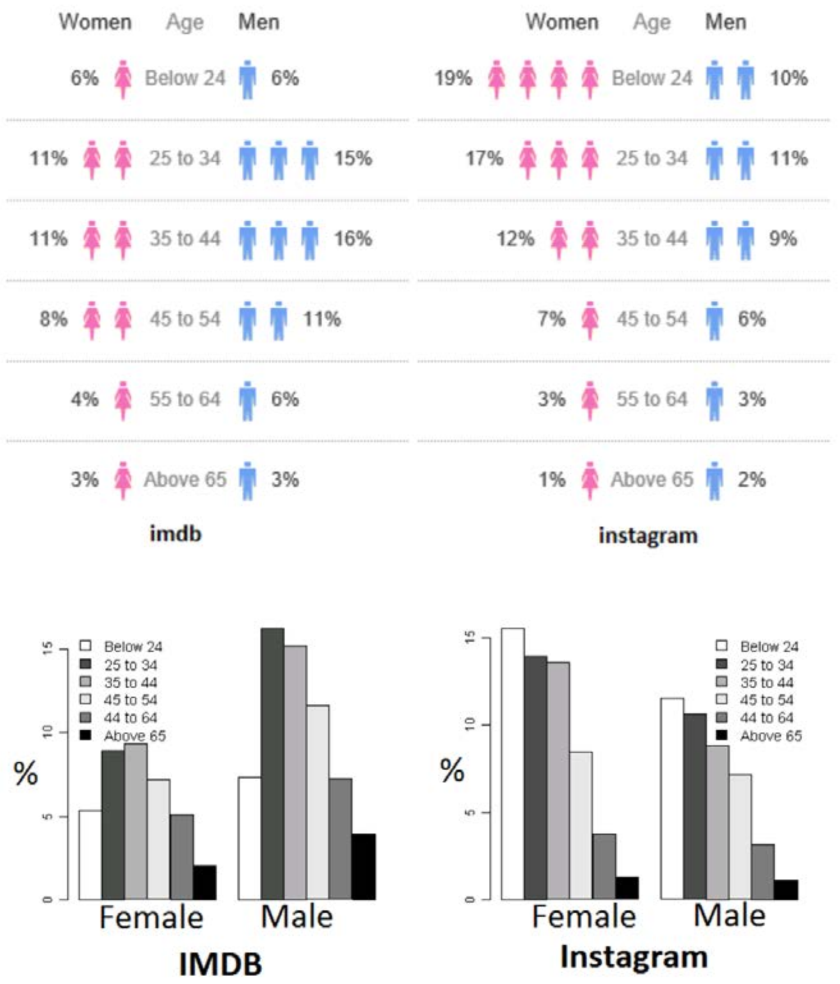
{gmanish,punagr}@microsoft.com

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.

Shokouhi, Milad. "Learning to personalize query auto-completion." In SIGIR, pp. 103-112. 2013.

Features for personalizing auto-completion

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

Shokouhi, Milad. "Learning to personalize query auto-completion." In SIGIR, pp. 103-112. 2013.

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%.

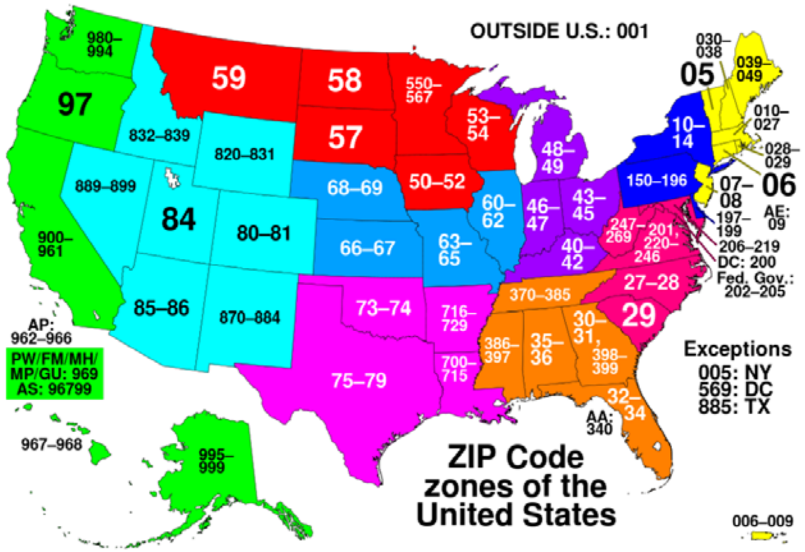
Below 20	**21-30**	**31-40**
taylor swift	piers morgan	bank of america
justin bieber	richard nixon	worldstarhiphop
deviantart	weather	alex jones
full house	beyonce	indeed
harry styles	movies	national weather service
41-50		**Above 50**
national cathedral		mapquest
target		fedex tracking
chase		florida lottery
microsoft		pogo
traductor google		jigsaw puzzles

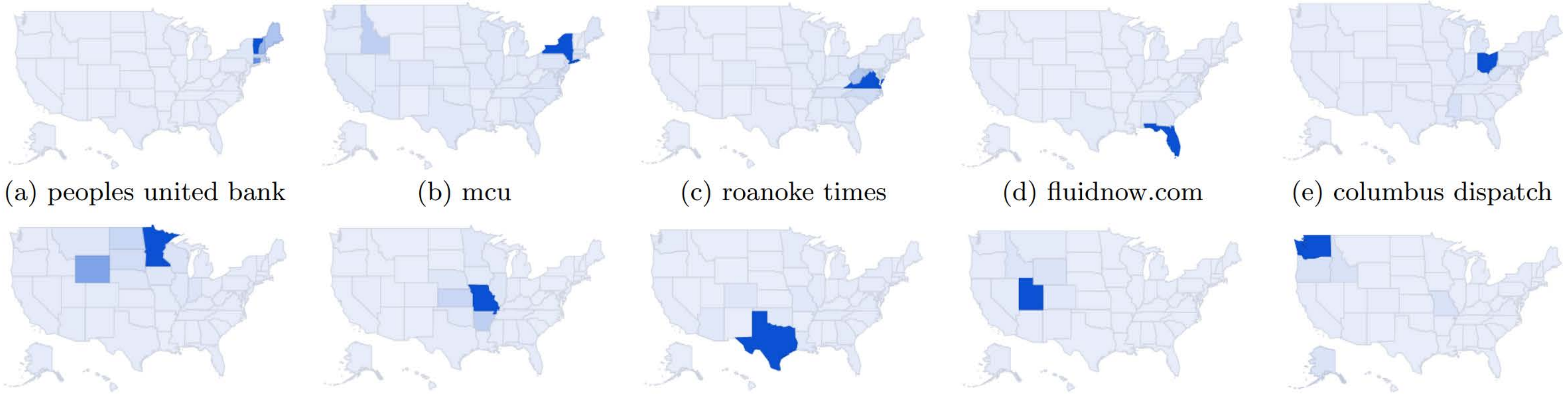
The biggest movers in personalized autocompletion rankings when the ranker is trained by age features. Each column includes the candidates that were boosted most frequently in the personalized auto-completion rankings for users of the specified age groups.

Shokouhi, Milad. "Learning to personalize query auto-completion." In SIGIR, pp. 103-112. 2013.

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).





(a) peoples united bank (b) mcu (c) roanoke times (d) fluidnow.com (e) columbus dispatch

(f) star tribune (g) missouri lottery (h) sacu (i) salt lake tribune (j) wenatchee world

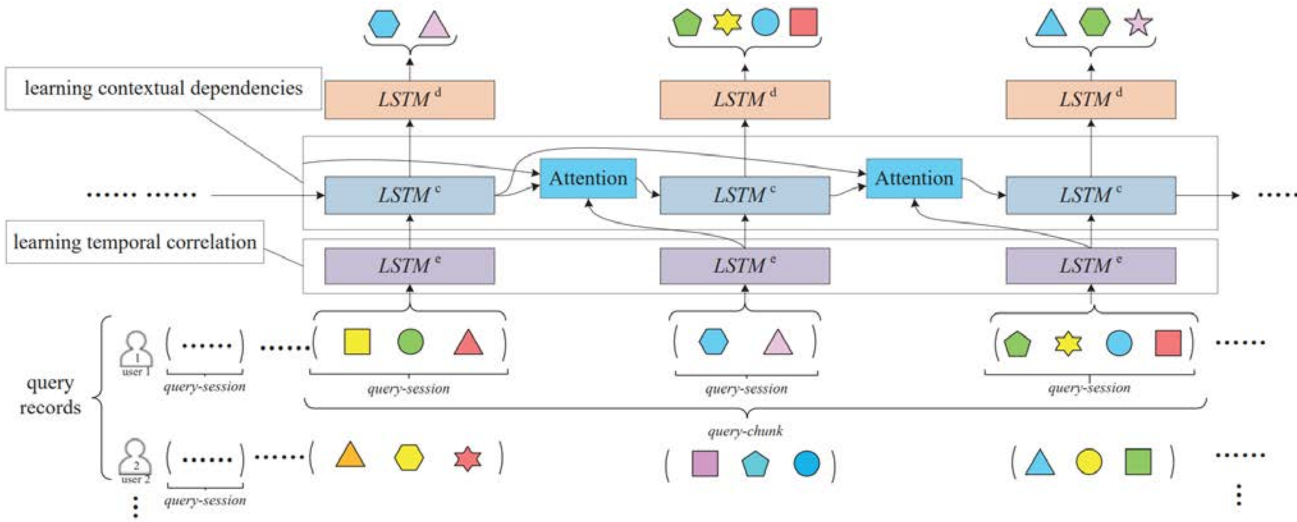
Shokouhi, Milad. "Learning to personalize query auto-completion." In SIGIR, pp. 103-112. 2013.

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.



	User Input Examples	Ground Truth Query	Top Candidates
1	Zuojia Village → No.379 Bus Stop → Building Materials Market → ?	Yuxin District	Yuxin District, Zuojia Village, Guomen Building
2	Liuli Bridge → Beijing Electric Hospital → Wumart Supermarket → ?	MerryMart	WuMart, MerryMart Supermarket, MerryMart
3	Chaoqinghui → Nanxincang Building → KFC → ?	McDonald's	McDonald's, KFC, Starbucks
4	Fengtai Technology Park → 7 Days Inn → ?	Hai You Hotel	Home Inns, Hai You Hotel, Fengtai South Road
5	PetroChina → Sinopec → ?	PetroChina Gas Station	PetroChina, PetroChina Gas Station, Sinopec Gas Station

Song, Jun, Jun Xiao, Fei Wu, Haishan Wu, Tong Zhang, Zhongfei Mark Zhang, and Wenwu Zhu. "Hierarchical contextual attention recurrent neural network for map query suggestion." *IEEE TKDE* 29, no. 9 (2017): 1888-1901.

Attend, Copy, Generate (ACG)

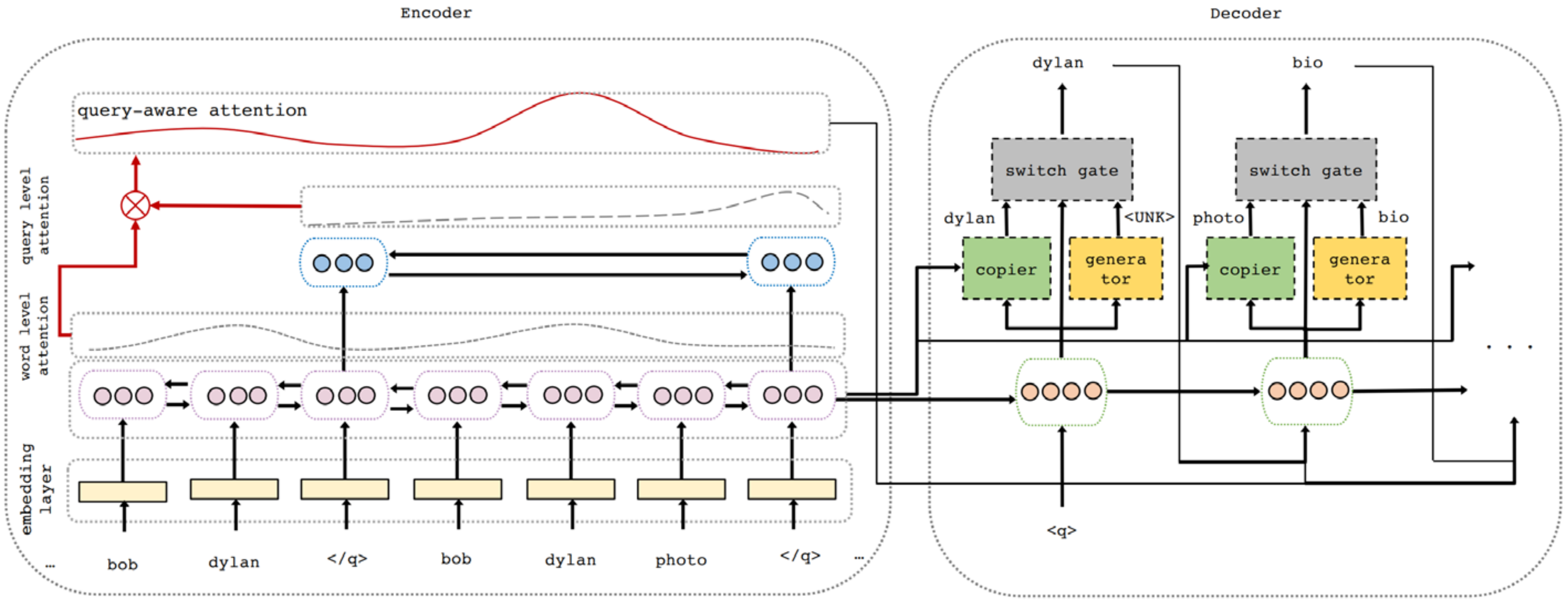
- Co-occurrence based models
  - Suffer from data sparsity and lack of coverage for rare or unseen queries.
  - Dealing with these highly diverse sessions makes using co-occurrence based model almost impossible.
- ACG
  - Query-aware attention mechanism to capture the structure of the session context.
  - 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.

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

Attend, Copy, Generate (ACG)

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.

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.



- Word level hidden layer output: h_i
- Decoder hidden layer output: s_{t-1}
- Word-level attention:
  - $l_{t,i} = \eta(s_{t-1}, h_i)$
  - $a_{t,i} = \frac{\exp(l_{t,i})}{\sum_j^n \exp(l_{t,j})}$

- Query level hidden layer output: g_j
- Query-level attention:

$$l^q_{t,j} = \eta(s_{t-1}, g_j, y_{t-1})$$

$$a^q_{t,j} = \frac{\exp(l^q_{t,j})}{\sum_i^n \exp(l^q_{t,i})}$$

- Overall attention:

$$a_{t,i} = \frac{a^w_{t,i} a^q_{t,j}}{\sum_{i'}^n a^w_{t,i'} a^q_{t,j'}}$$

$$l^p_{t,i} = \eta(s_t, h_i)$$

$$p(y_t = x_i | y_{<t}, X, \text{copy}) = \frac{\exp(l^p_{t,i})}{\sum_{j=0}^n \exp(l^p_{t,j})}$$

$$p(\text{copy}) = \sigma(w^T s_t)$$

$$p(\text{generate}) = 1 - p(\text{copy})$$

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

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 $\langle UNK \rangle$ and target generator is not $\langle OOV \rangle$: the switch gate shall choose generation ($t_{\text{switch}} = 0$).

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

(3) target copier is $\langle UNK \rangle$ and target generator is $\langle OOV \rangle$: the switch gate shall choose generation ($t_{\text{switch}} = 0$).

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

$$\text{loss}_{\text{switch}} = (p(\text{copy}) - t_{\text{switch}})^2$$

$$p(q|X) = \prod_{t=1}^{n} \Big(p(\text{generate}|y_{<t},X)\, p(y_t|y_{<t},X,\text{generate}) + p(\text{copy}|y_{<t},X)\, p(y_t|y_{<t},X,\text{copy}) \Big)$$

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

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.

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

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

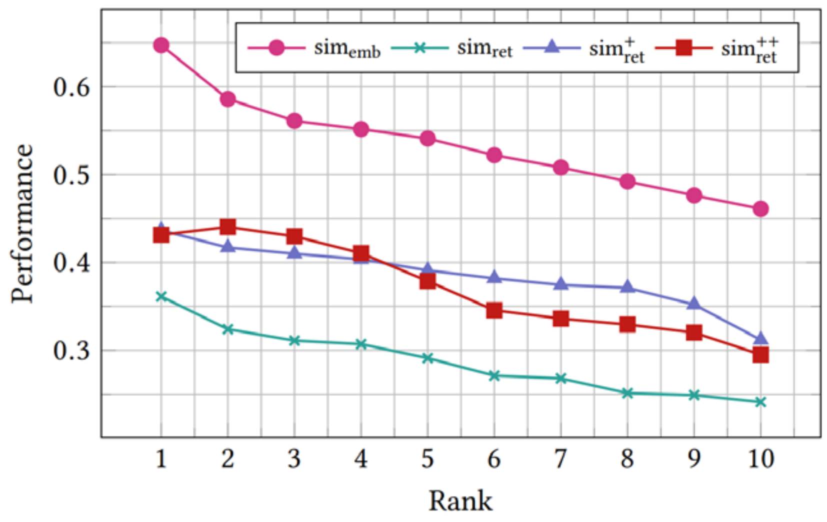
Table 1: Performance of the different methods as discriminative models. $^{(x)}$ indicates that the improvements with respect to the method in row x is statistically significant, at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

#	Model	*MRR*
1	**MPS**	0.5216
2	**BaseRanker**	$0.5530^{(1)}$
3	**BaseRanker + Seq2Seq**	$0.5679^{(1,2)}$
4	**BaseRanker + HRED [42]**	$0.5727^{(1,2)}$
5	**BaseRanker + (Seq2Seq + QaA)**	$0.5744^{(1,2)}$
6	**BaseRanker + (Seq2Seq + CM)**	$0.5851^{(1,2,3,4,5)}$
7	**BaseRanker + ACG**	$0.5941^{(1,2,3,4,5,6)}$

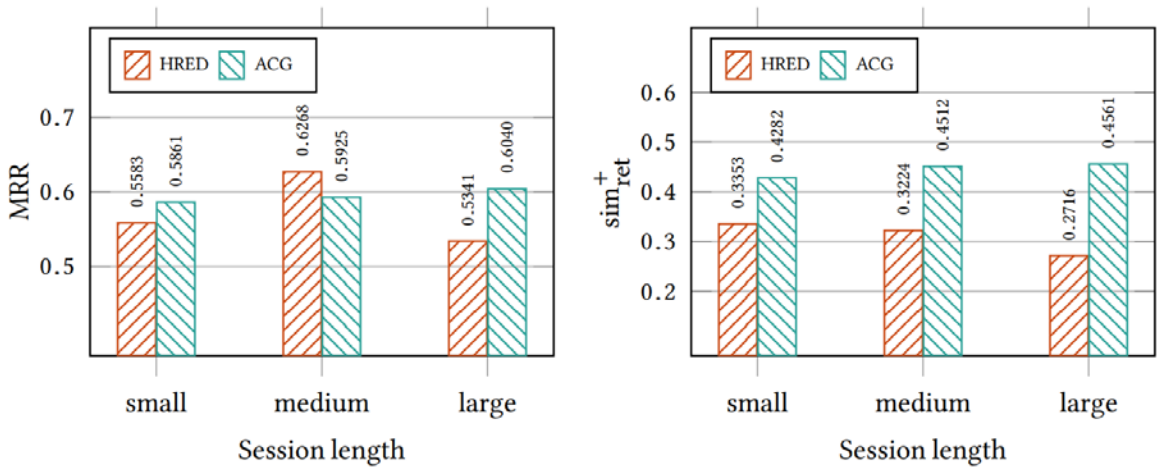
#	Method	Overlap Based	Embedding Based	Retrieval Based		
		PER (%)	sim_{emb}	sim_{ret}	sim^{+}_{ret}	sim^{++}_{ret}
1	**seq2seq**	84.11 (± 6.3)	0.5170 (± 0.003)	0.1630 (± 0.008)	0.2424 (± 0.009)	0.1955 (± 0.008)
2	**BaseRanker + seq2seq (top-1)**	72.23 (± 8.1)	0.5019 (± 0.006)	0.4375 (± 0.009)	0.3751 (± 0.008)	0.3916 (± 0.008)
3	**seqsSeq + QaA**	80.90 (± 5.0)	0.5517 (± 0.004)	0.2012 (± 0.009)	0.2916 (± 0.008)	0.2330 (± 0.008)
4	**seq2seq + CM**	71.16 (± 3.5)	0.6119 (± 0.003)	0.3563 (± 0.009)	0.4173 (± 0.009)	0.3950 (± 0.008)
5	**HRED [42]**	81.50 (± 4.9)	0.5455 (± 0.004)	0.2667 (± 0.008)	0.3250 (± 0.009)	0.3443 (± 0.007)
6	**BaseRanker + HRED [42] (top-1)**	72.36 (± 7.3)	0.5200 (± 0.004)	0.4504 (± 0.009)	0.3812 (± 0.009)	0.4091 (± 0.007)
7	**ACG**	**68.03** (± 3.6)	**0.6473** (± 0.004)	0.3612 (± 0.008)	**0.4366** (± 0.009)	**0.4315** (± 0.008)
8	**BaseRanker + ACG (top-1)**	70.66 (± 7.1)	0.5196 (± 0.004)	**0.4594** (± 0.008)	0.3927 (± 0.009)	0.4111 (± 0.007)

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

Detailed results



Performance of the generated queries at different ranks.



(a) Evaluation based on Discrimination **(b) Evaluation based on Generative**

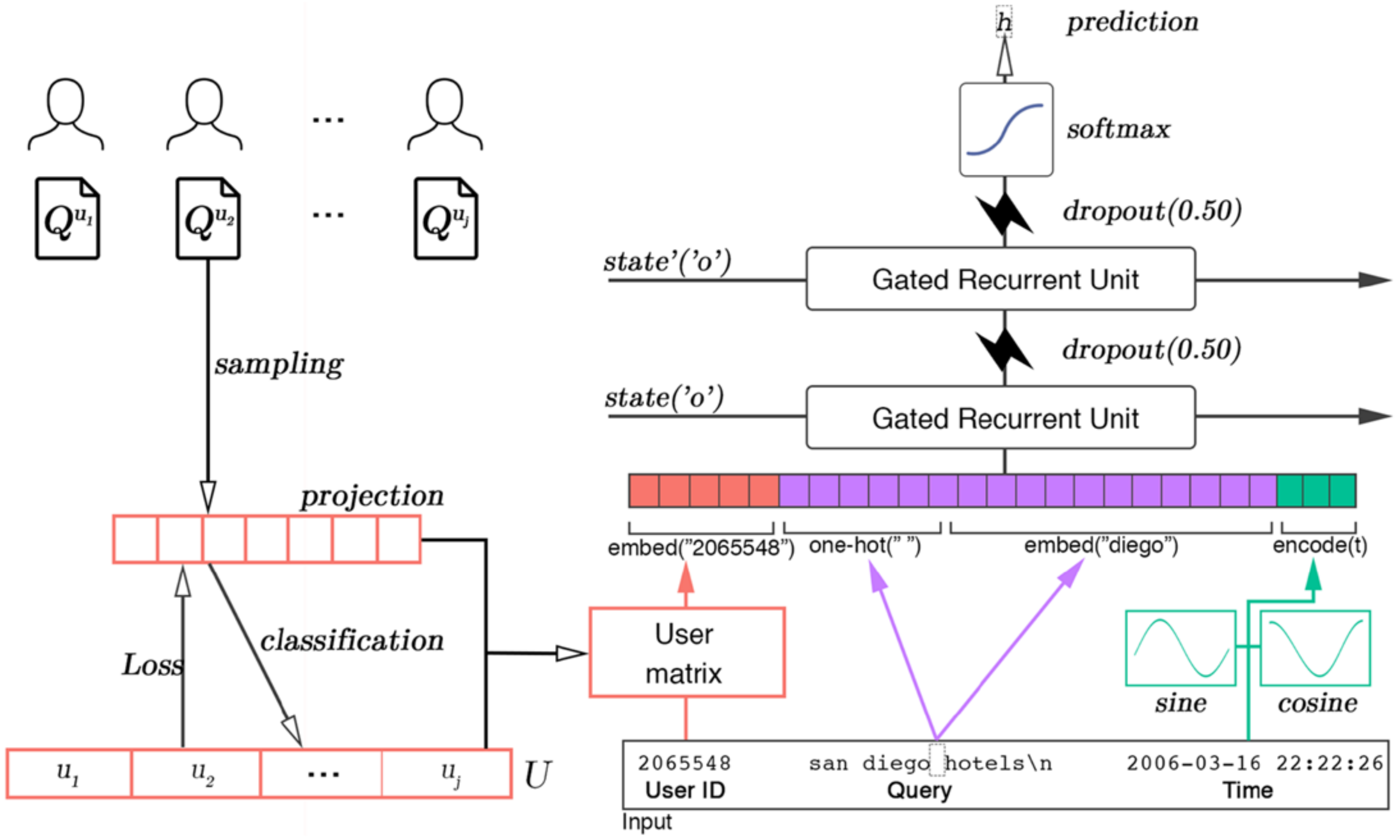
Figure 4: Performance of ACG compared to HRED on sessions with different lengths.

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. "Learning to attend, copy, and generate for session-based query suggestion." In CIKM, pp. 1747-1756. 2017.

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

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

Fiorini, Nicolas, and Zhiyong Lu. "Personalized neural language models for real-world query auto completion." In NAACL-HLT, pp. 208-215. 2018.

Personalized neural Language Model

- User representation
  - Each query q_i is a set of words $\{w_1, ..., w_n\}$.
  - User u∈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.

$$sin\left(\frac{2\pi(3600x+60y+z)}{86400}\right)$$

$$cos\left(\frac{2\pi(3600x+60y+z)}{86400}\right)$$

Fiorini, Nicolas, and Zhiyong Lu. "Personalized neural language models for real-world query auto completion." In NAACL-HLT, pp. 208-215. 2018.

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.

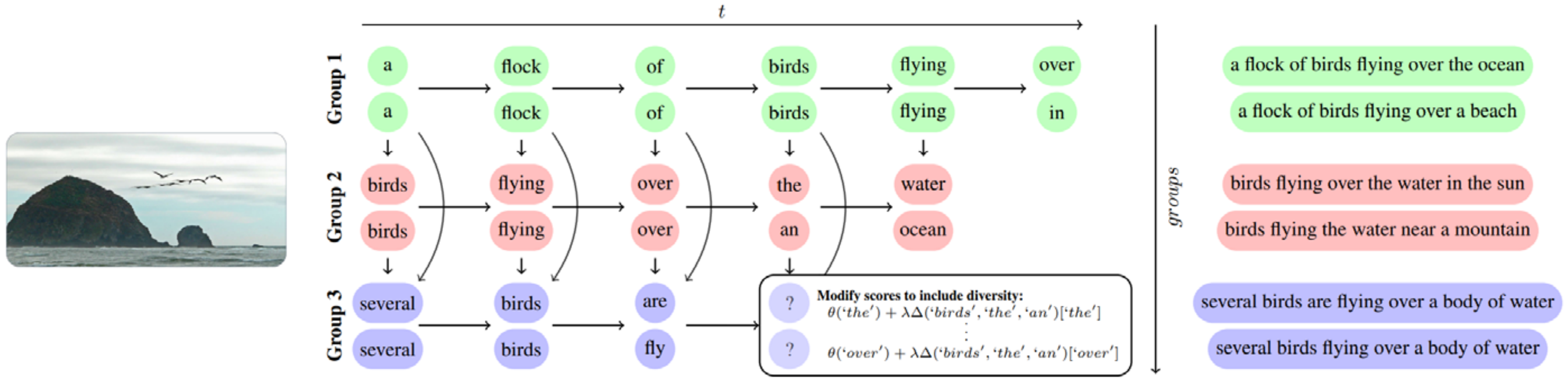


Figure 2: Diverse beam search operates left-to-right through time and top to bottom through groups. Diversity between groups is combined with joint log-probabilities, allowing continuations to be found efficiently. The resulting outputs are more diverse than for standard approaches.

Fiorini, Nicolas, and Zhiyong Lu. "Personalized neural language models for real-world query auto completion." In NAACL-HLT, pp. 208-215. 2018.
Vijayakumar, Ashwin K., Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. "Diverse beam search: Decoding diverse solutions from neural sequence models." *arXiv preprint arXiv:1610.02424* (2016).

Table 1: MRR results for all tested models on the AOL and biomedical datasets with their average prediction time in seconds.

Model	AOL dataset MRR Seen	AOL dataset MRR Unseen	AOL dataset MRR All	AOL dataset Time	Biomedical dataset MRR Seen	Biomedical dataset MRR Unseen	Biomedical dataset MRR All	Biomedical dataset Time
MPC (Bar-Yossef and Kraus, 2011)	**0.461**	0.000	0.184	**0.24**	0.165	0.000	0.046	**0.29**
NQLM(L)+WE+MPC+λMART (Park and Chiba, 2017)	0.430	0.306	0.356	1.33	0.159	0.152	0.154	2.35
Our models in this paper								
NQAC	0.406	0.319	0.354	0.94	0.155	0.139	0.143	1.73
NQAC$_U$	0.417	0.325	0.361	0.98	**0.191**	0.161	0.169	1.77
NQAC$_{UT}$	0.424	0.326	0.365	0.95	0.101	**0.195**	0.157	1.81
NQAC$_{UT}$+D	0.427	0.326	0.366	1.32	0.186	0.185	0.185	2.04
NQAC$_{UT}$+MPC	**0.461**	0.326	0.380	0.68	0.165	**0.195**	**0.187**	1.20
NQAC$_{UT}$+MPC+λMART	0.459	**0.330**	**0.382**	1.09	0.154	0.179	0.172	2.01

- 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.

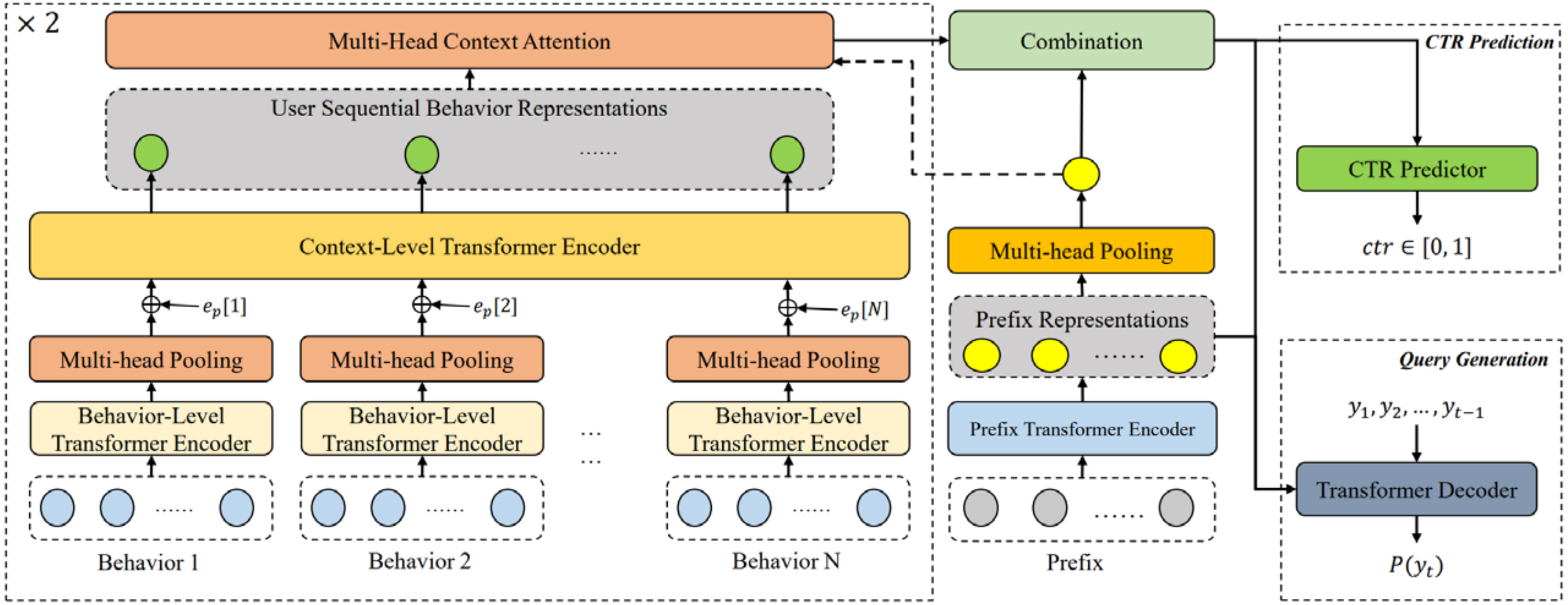
Fiorini, Nicolas, and Zhiyong Lu. "Personalized neural language models for real-world query auto completion." In NAACL-HLT, pp. 208-215. 2018.

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.



Yin, Di, Jiwei Tan, Zhe Zhang, Hongbo Deng, Shujian Huang, and Jiajun Chen. "Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach." In KDD, pp. 2998-3007. 2020.

Context_q	休闲裤女 (casual pants for girls) 九分休闲裤 (nine-cent casual pants) 超短牛仔裤 (short jeans) 秋季套装女 (autumn suit for girls) 牛仔外套 (denim jacket)
Context_i	阿迪达斯情侣运动裤 (adidas couple sports pants) 嘻哈宽松街头休闲裤 (hip-hop loose street leisure pants) 彪马运动休闲裤女 (puma sports leisure pants for girls) 新款高腰显瘦女裤 (new high waist slim pants for women) 韩版宽松运动裤(korean version loose slacks)
Prefix	运动 (sports)
Golden	运动裤女 (sports pants for girls)
MPC	运动鞋2019新款(sports shoes 2019 new style) 运动套装女 (sports suit for girls) 运动外套女 (sports coat for girls)
Transformer	运动鞋2019新款(sports shoes 2019 new style) 运动套装女 (sports suit for girls) 运动套装 (sports suit)
$M^2A(QG)$	**运动裤女(sports pants for girls)** **运动休闲裤女(sports leisure pants for girls)** **运动裤女宽松(sports pants for girls loose)**

Examples of query candidates recommended by different models for the same prefix and history behavior.

Model	Seen			Unseen		
	BLEU	MRR	UMRR	BLEU	MRR	UMRR
V_q	57.69	0.562	0.777	36.07	0.221	0.646
V_i	57.17	0.554	0.773	32.72	0.186	0.642
$V_q + V_i$	61.49	0.573	0.783	37.41	0.229	0.645
$V_q + V_i + L_c$	**62.97**	**0.575**	**0.788**	37.29	0.229	0.644
$V_q + V_i + L_u$	60.22	0.569	0.776	58.88	0.545	0.750
$V_q + V_i + L_c + L_u$	62.23	0.573	0.785	**59.17**	**0.548**	**0.758**

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.)

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

Yin, Di, Jiwei Tan, Zhe Zhang, Hongbo Deng, Shujian Huang, and Jiajun Chen. "Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach." In KDD, pp. 2998-3007. 2020.

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

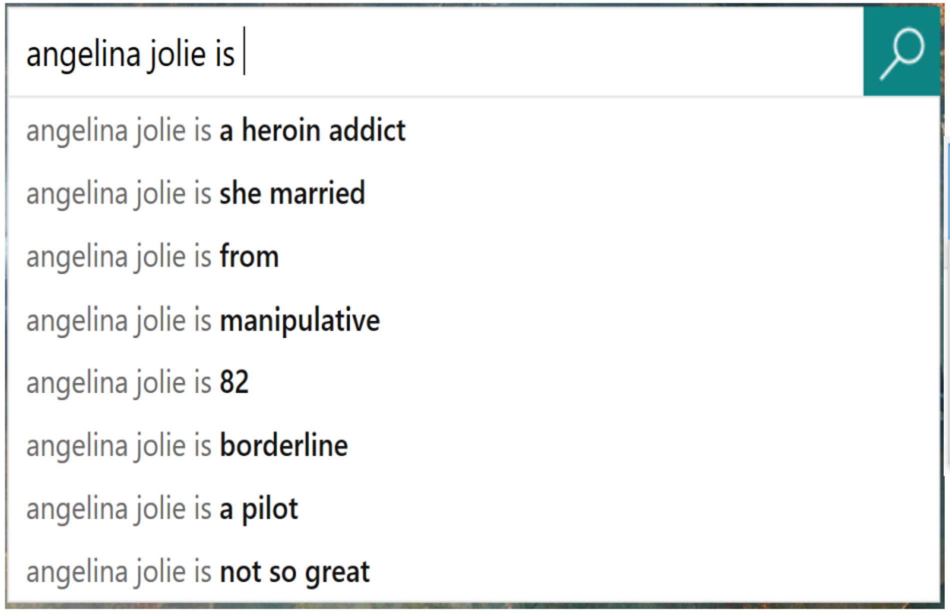
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
- Personalization [20 min]
- Handling defective suggestions and prefixes [20 min]
  - LSTMs for inappropriate query suggestion detection
  - Offline Spell Correction
  - Online Spell Correction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

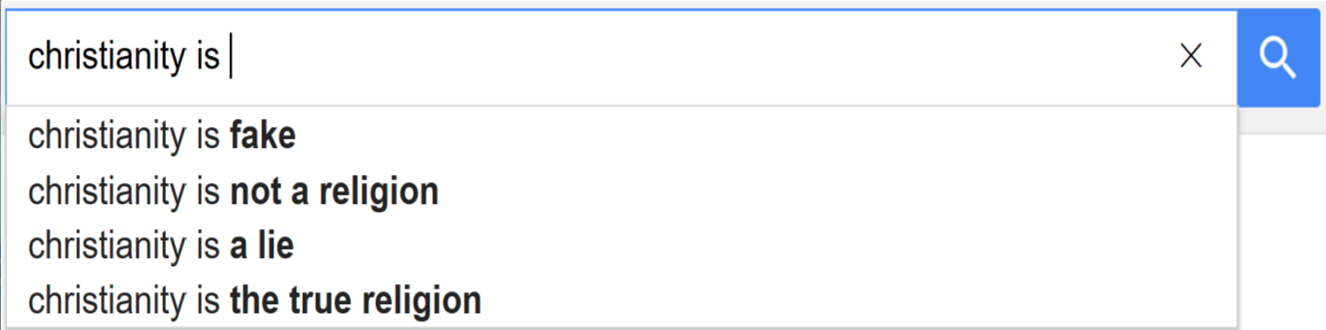
Agenda

- 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]
  - **LSTMs for inappropriate query suggestion detection**
  - Offline Spell Correction
  - Online Spell Correction
- Summary and Future Trends [5 min]

Inappropriate query suggestion detection

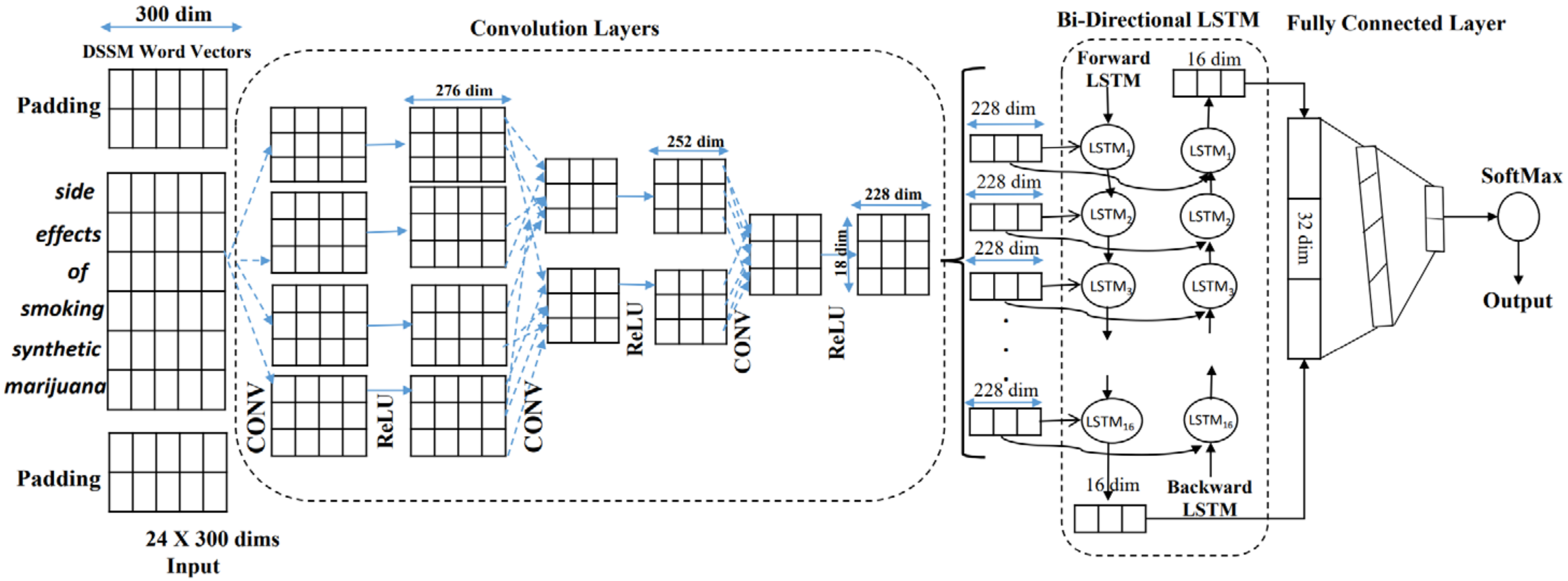
- Inappropriate: if it may cause anger, annoyance to certain users or exhibits lack of respect, rudeness, discourteousness towards certain individuals/communities or may be capable of inflicting harm to oneself or others.





Yenala, Harish, Manoj Chinnakotla, and Jay Goyal. "Convolutional Bi-directional LSTM for detecting inappropriate query suggestions in web search." In *PAKDD*, pp. 3-16. Springer, Cham, 2017.

CONV+BiLSTMs for Inappropriate query suggestion detection



- Randomly initialize the DSSM word vectors for these padded unknown words from the uniform distribution [-0.25, 0.25].
- Use three 3 × 25 filters.

Yenala, Harish, Manoj Chinnakotla, and Jay Goyal. "Convolutional Bi-directional LSTM for detecting inappropriate query suggestions in web search." In *PAKDD*, pp. 3-16. Springer, Cham, 2017.

Results

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

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

Label	Training	Validation	Test	Total
Inappropriate	4594	212	178	4984
Clean	65447	4788	3822	74057
Total	**70041**	**5000**	**4000**	**79041**

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

Model	Precision	Recall	F1 Score
PKF	0.625	0.2142	0.3190
BDT	0.7926	0.2784	0.4120
BDT-DSSM	0.9474	0.3051	0.4615
SVM	0.8322	0.3593	0.5019
SVM-DSSM	0.9241	0.4101	0.5680
CNN	0.7148	0.8952	0.7949
LSTM	0.8862	0.7047	0.7850
BLSTM	0.8018	0.8285	0.8149
C-BiLSTM	0.9246	0.8251	**0.8720**

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

Yenala, Harish, Manoj Chinnakotla, and Jay Goyal. "Convolutional Bi-directional LSTM for detecting inappropriate query suggestions in web search." In *PAKDD*, pp. 3-16. Springer, Cham, 2017.

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 = $(x_1, x_2, \cdots, x_n)$ ➔ Y = $(y_1, y_2, \cdots, y_n)$
- 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.

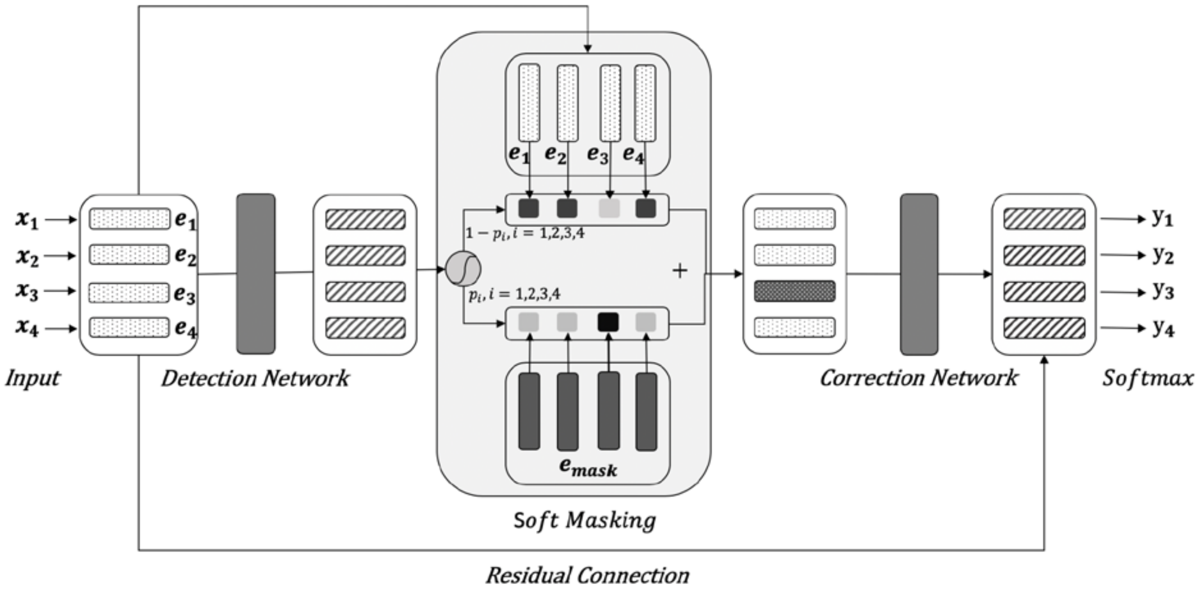
Table 1: Examples of Chinese spelling errors

Wrong: 埃及有金子塔。Egypt has golden towers.
Correct: 埃及有金字塔。Egypt has pyramids.
Wrong: 他的求胜欲很强，为了越狱在挖洞。He has a strong desire to win and is digging for prison breaks
Correct: 他的求生欲很强，为了越狱在挖洞。He has a strong desire to survive and is digging for prison breaks.

Zhang, Shaohua, Haoran Huang, Jicong Liu, and Hang Li. "Spelling Error Correction with Soft-Masked BERT." In *ACL*, pp. 882-890. 2020.

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 = $(g_1, g_2, \cdots, g_n)$, where g_i =1 means the character is incorrect and 0 means it is correct.
- For i-th character, soft-masked embedding is $e_i' = p_i . e_{mask} + (1 - p_i)e_i$
- 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.



$$\mathcal{L}_d = -\sum_{i=1}^{n} \log P_d(g_i|X)$$ Detection loss

$$\mathcal{L}_c = -\sum_{i=1}^{n} \log P_c(y_i|X)$$ Correction loss

$$\mathcal{L} = \lambda \cdot \mathcal{L}_c + (1 - \lambda) \cdot \mathcal{L}_d$$

Zhang, Shaohua, Haoran Huang, Jicong Liu, and Hang Li. "Spelling Error Correction with Soft-Masked BERT." In *ACL*, pp. 882-890. 2020.

Agenda

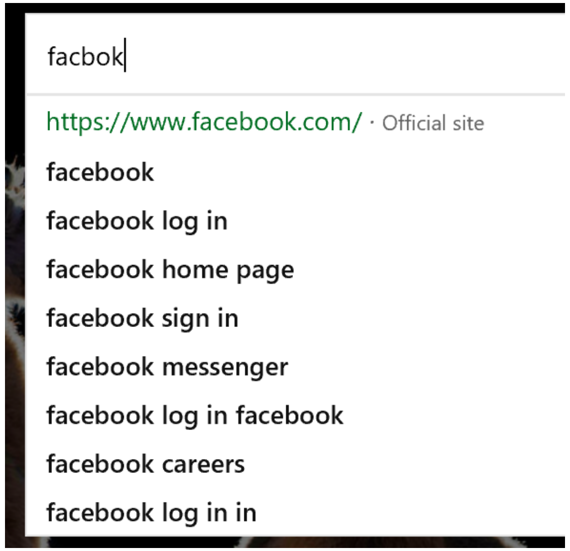
- Components in Query Auto Completion systems [20 min]
- Ranking [20 min]
- Natural Language Generation [20 min]
- Personalization [20 min]
- Handling defective suggestions and prefixes [20 min]
  - LSTMs for inappropriate query suggestion detection
  - Offline Spell Correction
  - **Online Spell Correction**
- 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

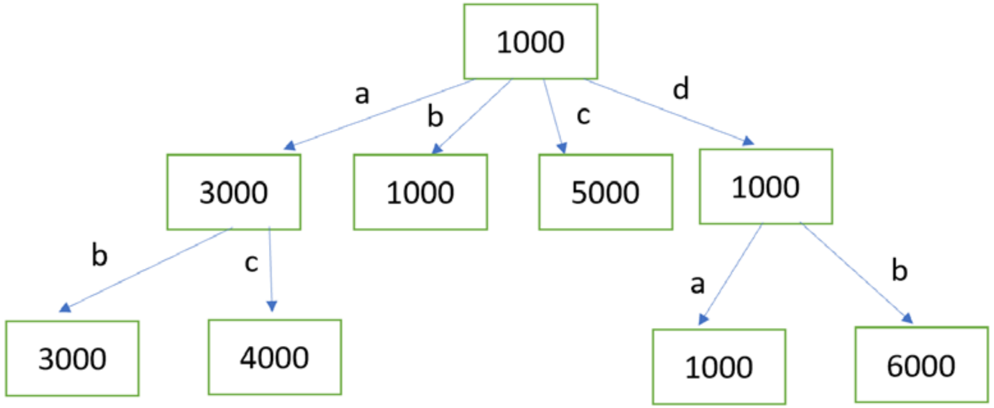
Table 1. Types of misspellings

Cause	Misspelling	Correction
Typing quickly	exxit mispell	exit misspell
Keyboard adjacency	importamt	important
Inconsistent rules	concieve conceirge	conceive concierge
Ambiguous word breaking	silver light	silverlight
New words	kinnect	kinect



Duan, Huizhong, and Bo-June Hsu. "Online spelling correction for query completion." In *Proceedings of the 20th international conference on World wide web*, pp. 117-126. 2011.

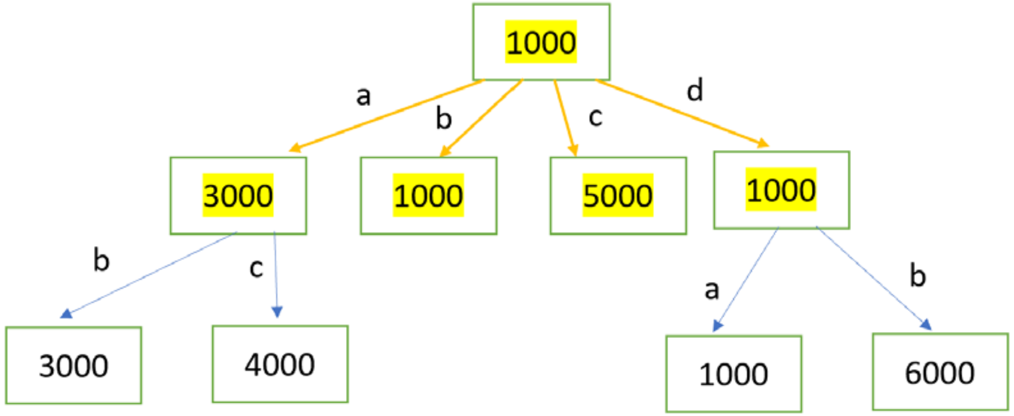
Trie with online spell correction



From	To	Cost
a	b	5000
a	c	5000
a	d	6000

Conversion 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:

a 3000	b 1000+5000	d 1000 + 6000	c 5000 + 5000

Duan, Huizhong, and Bo-June Hsu. "Online spelling correction for query completion." In *Proceedings of the 20th international conference on World wide web*, pp. 117-126. 2011.

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

s c h	w a r	z e	n e g g	e r
s h	w a r	t z e	n e g	e r

Figure 1: Example segmentation into transfemes

Duan, Huizhong, and Bo-June Hsu. "Online spelling correction for query completion." In *Proceedings of the 20th international conference on World wide web*, pp. 117-126. 2011.

Learning conversion rules: Noisy channel method

- Given a sequence of transfemes $s = t_1, t_2, \ldots, t_{l^s}$, 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→q) as the sum of all possible segmentations.
- S(c→q) is the set of all possible joint segmentations of c and q.
- Solution using EM.

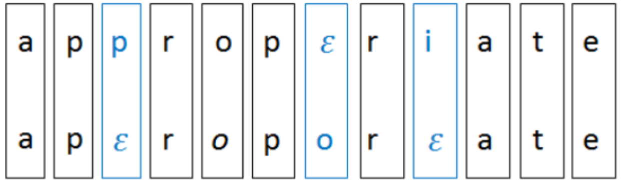


Figure 2: Example transformation with $L = 1$

Figure 3: Comparing transformations with $L = 1$ and $L = 2$

$$p(c \to q) = \sum_{s \in S(c \to q)} p(s)$$
$$= \sum_{s \in S(c \to q)} \prod_{i \in [1, l^s]} p(t_i | t_1, \ldots, t_{i-1})$$

Duan, Huizhong, and Bo-June Hsu. "Online spelling correction for query completion." In *Proceedings of the 20th international conference on World wide web*, pp. 117-126. 2011.

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.

Completion vs Error Correction

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

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

$$\text{cand}_{\text{new}} := \left\{ \begin{array}{l} s_{1:t+1} : \log P(s_{1:t+1} \mid s_{1:m}) \\ \quad \text{for all } s_{t+1} \in C, \text{ for all } s_{1:t} \in \text{cand} \end{array} \right\};$$

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

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

General Beam Search

$$\arg\max_{\hat{s}_{1:n}} P(\hat{s}_{1:n} \mid s_{1:m})$$

$$\arg\max_{\hat{s}_{1:n}} \frac{P(s_{1:m} \mid \hat{s}_{1:n}) P(\hat{s}_{1:n})}{P(s_{1:m})}.$$

$$\arg\max_{\hat{s}_{1:n}} \log P(s_{1:m} \mid \hat{s}_{1:n}) + \log P(\hat{s}_{1:n}).$$

$\log P(s_{1:m} \mid \hat{s}_{1:n})$ 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.

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

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.

Beam Search with Edit Distance

- Dynamic programming algorithm of edit (completion) distance costs O(m·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.

cand := $\{s_{1:m} : 0\}$, result := { }

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

$$\text{cand}_{\text{new}} := \left\{ \begin{array}{l} s_{1:t+1} : \log P(s_{1:t+1} \mid s_{1:m}) \\ \quad \text{for all } s_{t+1} \in C, \text{ for all } s_{1:t} \in \text{cand} \end{array} \right\};$$

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

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

cand := {empty string " ": 0}, result := { }

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

$$\text{cand}_{\text{new}} := \left\{ \begin{array}{l} \hat{s}_{1:t+1} : \log P(s_{1:m} \mid \hat{s}_{1:t+1}) + \log P(\hat{s}_{1:t+1}) \\ \quad \text{for all } \hat{s}_{t+1} \in C, \text{ for all } \hat{s}_{1:t} \in \text{cand} \end{array} \right\};$$

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

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

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

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.

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. $\mathcal{O}(\log L)$ 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.

Yadav, Nishant, Rajat Sen, Daniel N. Hill, Arya Mazumdar, and Inderjit S. Dhillon. "Session-aware query auto-completion using extreme multi-label ranking." In PKDD, pp. 3835-3844. 2021.

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

[1] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. Learning to attend, copy, and generate for session-based query suggestion. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pages 1747–1756, 2017.

[2] Giovanni Di Santo, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Comparing approaches for query autocompletion. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 775–778, 2015.

[3] Huizhong Duan and Bo-June Hsu. Online spelling correction for query completion. In Proceedings of the 20th international conference on World wide web, pages 117–126, 2011.

[4] Nicolas Fiorini and Zhiyong Lu. Personalized neural language models for real-world query auto completion. In Proceedings of NAACL-HLT, pages 208–215, 2018.

[5] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. Learning user reformulation behavior for query auto-completion. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pages 445–454, 2014.

[6] Jyun-Yu Jiang and Wei Wang. Rin: Reformulation inference network for context-aware query suggestion. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 197–206, 2018.

[7] Gyuwan Kim. Subword language model for query autocompletion. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5022– 5032, 2019.

[8] Dong-Ho Lee, Zhiqiang Hu, and Roy Ka-Wei Lee. Improving text auto-completion with next phrase prediction. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 4434–4438, 2021.

[9] Bhaskar Mitra and Nick Craswell. Query autocompletion for rare prefixes. In Proceedings of the 24th ACM international on conference on information and knowledge management, pages 1755–1758, 2015.

[10] Agnes Mustar, Sylvain Lamprier, and Benjamin Piwowarski. Using bert and bart for query suggestion. In Joint Conference of the Information Retrieval Communities in Europe, volume 2621. CEUR-WS. org, 2020.

References

[11] Alexandra Olteanu, Fernando Diaz, and Gabriella Kazai. When are search completion suggestions problematic? Proceedings of the ACM on Human-Computer Interaction, 4(CSCW2):1–25, 2020.

[12] Dae Hoon Park and Rikio Chiba. A neural language model for query auto-completion. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 1189– 1192, 2017.

[13] Milad Shokouhi. Learning to personalize query autocompletion. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pages 103–112, 2013.

[14] Jun Song, Jun Xiao, Fei Wu, Haishan Wu, Tong Zhang, Zhongfei Mark Zhang, and Wenwu Zhu. Hierarchical contextual attention recurrent neural 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 management, pages 553–562, 2015.

[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.

[17] Sida Wang, Weiwei Guo, Huiji Gao, and Bo Long. Efficient neural query auto completion. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pages 2797–2804, 2020.

[18] Harish Yenala, Manoj Chinnakotla, and Jay Goyal. Convolutional bi-directional lstm for detecting inappropriate query suggestions in web search. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 3–16. Springer, 2017.

[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 2998–3007, 2020.

[20] Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. Spelling error correction with soft-masked bert. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 882–890, 2020.

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/