

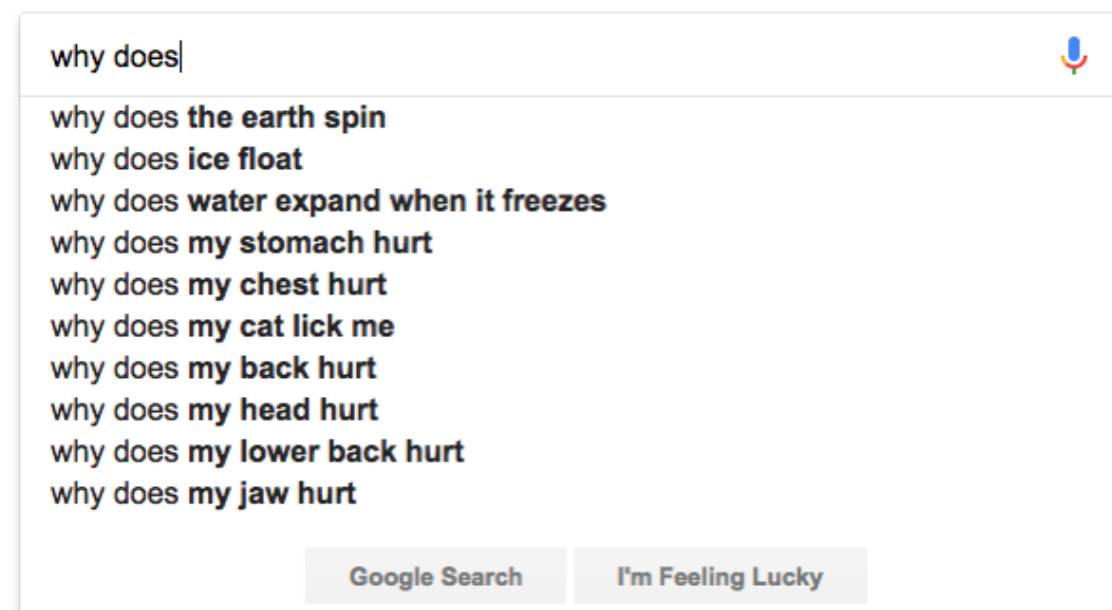
# Personalized Language Model for Query Auto-Completion

Aaron Jaech and Mari Ostendorf  
University of Washington



# Query Auto-Completion

- Search engine suggests queries as the user types
- Idea from Park & Chiba (2017): Use an LSTM to generate completions
  - Memory savings over most popular completion
  - Handles previously unseen prefixes
- Can we do better by adapting the LM to provide personalized suggestions?

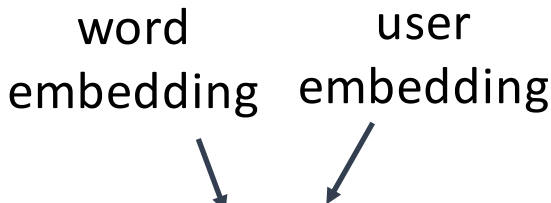


# RNN Language Model Adaptation

- Learn an embedding,  $c$ , for each user and use it to adapt the predictions
- **Method #1:** Concatenate the user embedding with the input at each step\*
  - Same as applying a constant linear shift to the bias vector (in recurrent & output layers)
  - Leaves most of the recurrent model parameters unchanged
- **Method #2:** Low-rank adaptation of recurrent weight matrix (FactorCell model)

*Adjust  $b$  and  $W$ !*

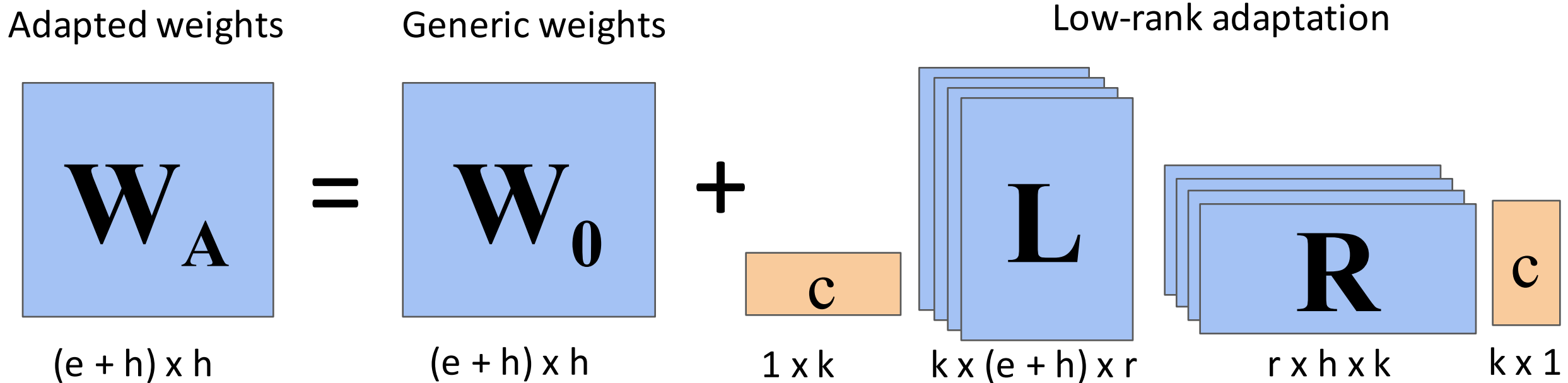
word embedding      user embedding


$$\begin{aligned}h_t &= \sigma(\widehat{W}[h_{t-1}, e_t, c] + b) \\ &= \sigma(W[h_{t-1}, e_t] + \underbrace{Vc + b}_{b_c})\end{aligned}$$
$$\widehat{W} = [W \ V]$$

*Concatenating the user embedding is the same as shifting the bias.*

\* Referred to here as ConcatCell (Mikolov & Zweig, 2012)

# FactorCell Model



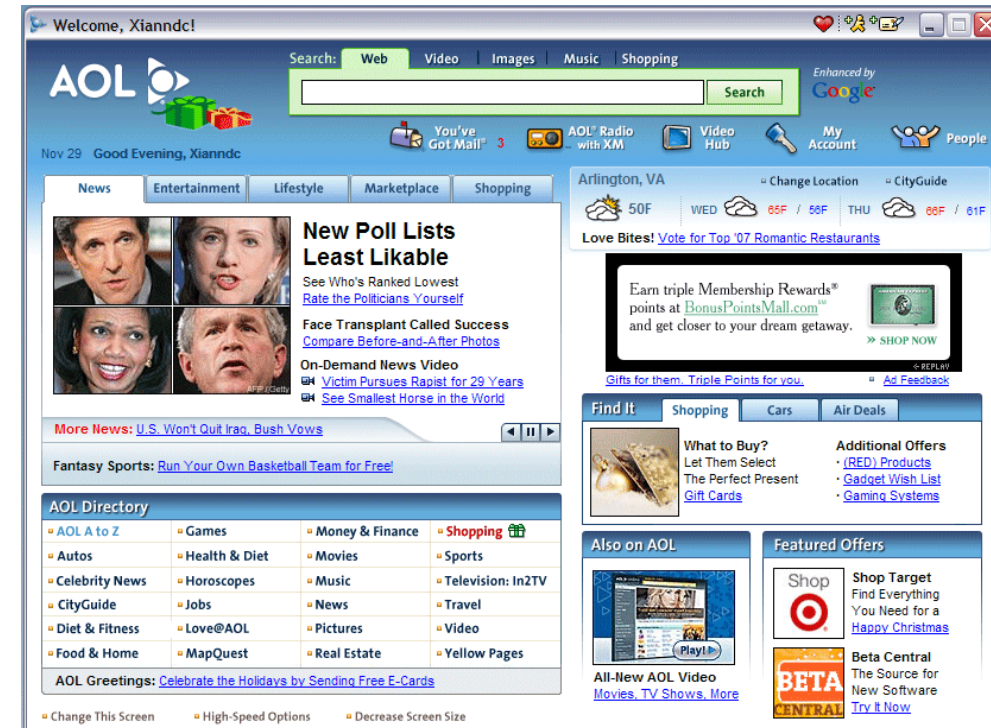
- The adaptation matrix is formed from a product of the context embedding with left and right bases.
- The two bases tensors ( $\mathbf{L}$  and  $\mathbf{R}$ ) hold  $k$  different rank  $r$  matrices, each the same size as  $\mathbf{W}$ . Context vectors give a weighted combination.

# Learning

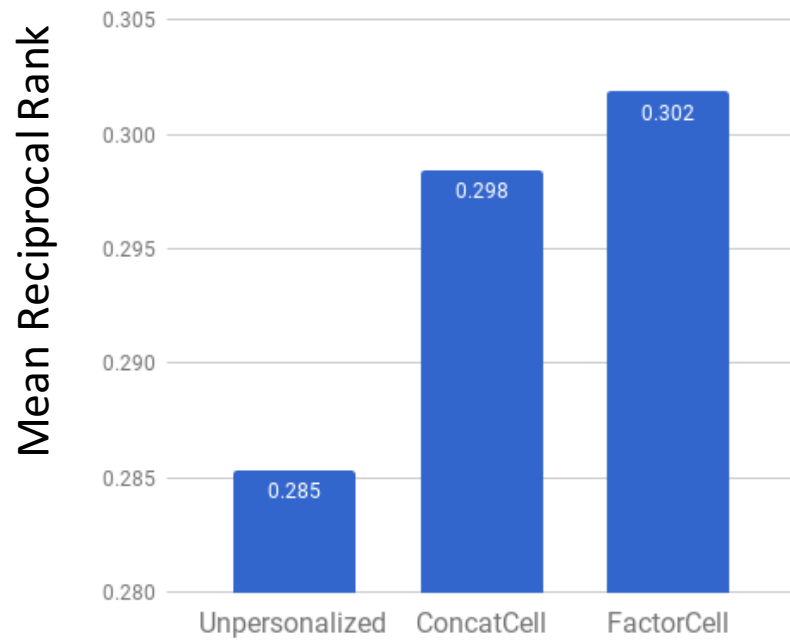
- User embeddings, recurrent layer weights and  $\{L, R\}$  tensor learned jointly
- Need online learning to adapt to users that were not previously seen
- In joint training, learn a cold-start embedding for set of infrequent users
- During evaluation
  - Initialize each users' embedding with learned cold-start vector
  - Make query suggestions
  - After user selects a query, back-propagate and **only** update the user embedding

# Data & Experiments

- Using AOL 2006 Query Log data, 173K users and 12 million queries for training
- User embedding size = 32, LSTM size = 600
- Evaluate on 500K queries with disjoint user population
- Mean reciprocal rank (MRR) as a metric

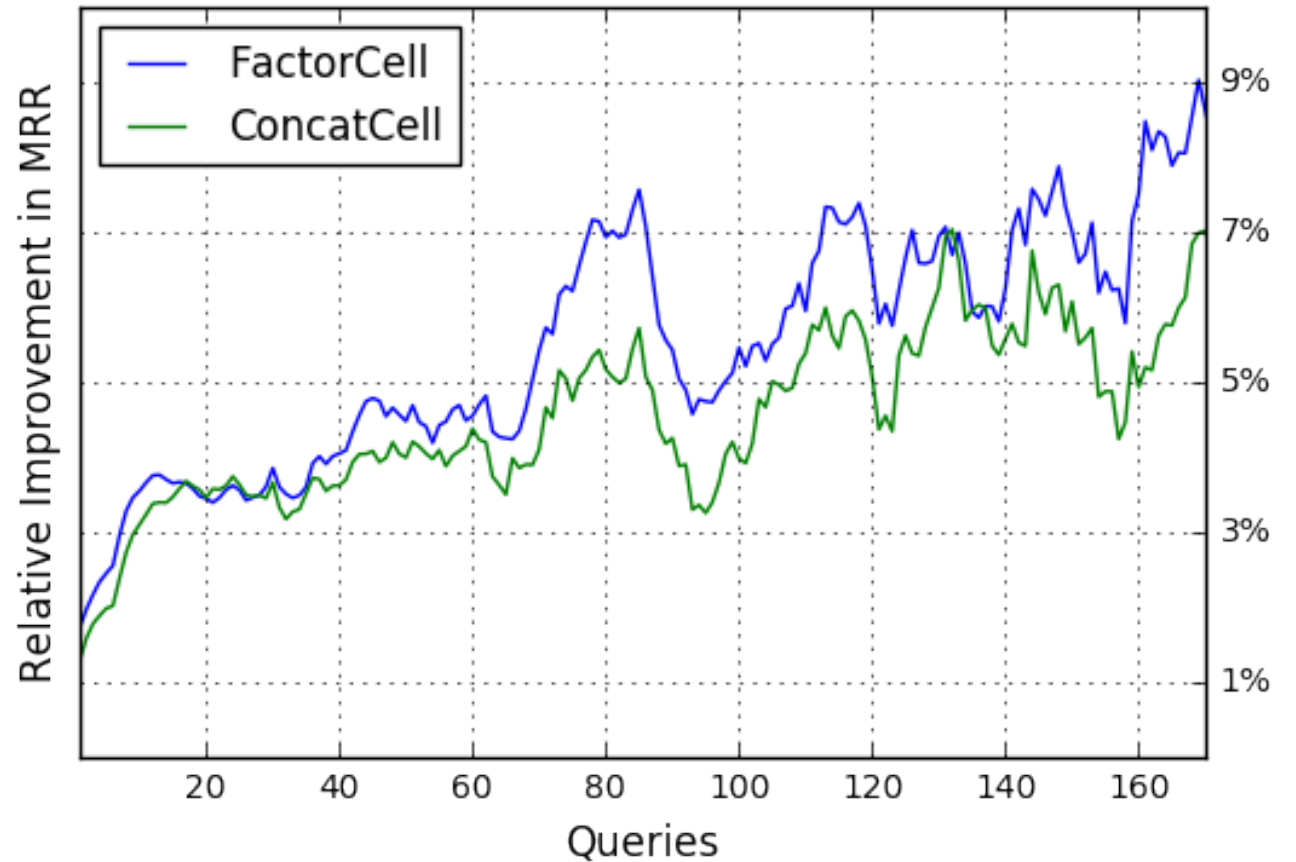


# Experimental Results



Performance for users with > 50 queries

Benefit improves over time!



# Qualitative Comparison

What queries are boosted the most after searching for “high school softball” and “math homework help”?

FactorCell	ConcatCell
high school musical	horoscope
chris brown	high school musical
funnyjunk.com	homes for sale
funbrain.com	modular homes
chat room	hair styles

Queries that most decrease in likelihood with the FactorCell include travel agencies and plane tickets.



# Recent Related Work: Florini & Lu, NAACL 2018

- Also personalized LSTM for query prediction
- ConcatCell adaptation framework
- User embedding learned separately
- No online learning
- Assessed on two datasets, but different split of AOL data
- Confirms benefit of adapted LM

# Conclusions

- Personalization helps and the benefit increases as more queries are seen
- Stronger adaptation of the recurrent layer (FactorCell) gives better results than concatenating a user vector
  - No extra latency/computation due to caching of adapted weight matrix
- Try out the FactorCell on your data
  - [http://github.com/ajaech/query\\_completion](http://github.com/ajaech/query_completion)

THANKS!

# Qualitative Comparison

What queries are boosted the most after searching for “prada handbags” and “versace eyewear”?

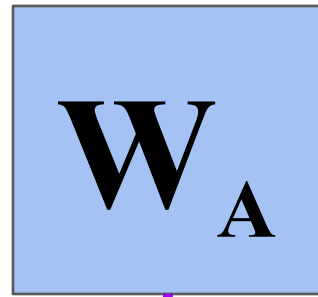
FactorCell	ConcatCell
neiman marcus	craigslist nyc
pottery barn	myspace layours
jc penny	verizon wireless
verizon wireless	jensen ackles
bed bath and beyond	webster dictionary

# Backup Slides

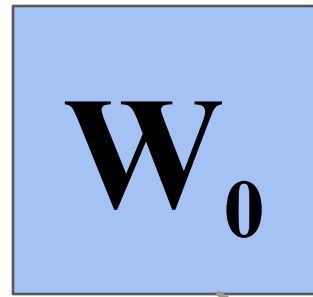
# FactorCell Model

The adapted weight matrix is a drop-in replacement for  $\mathbf{W}$

Adapted weights



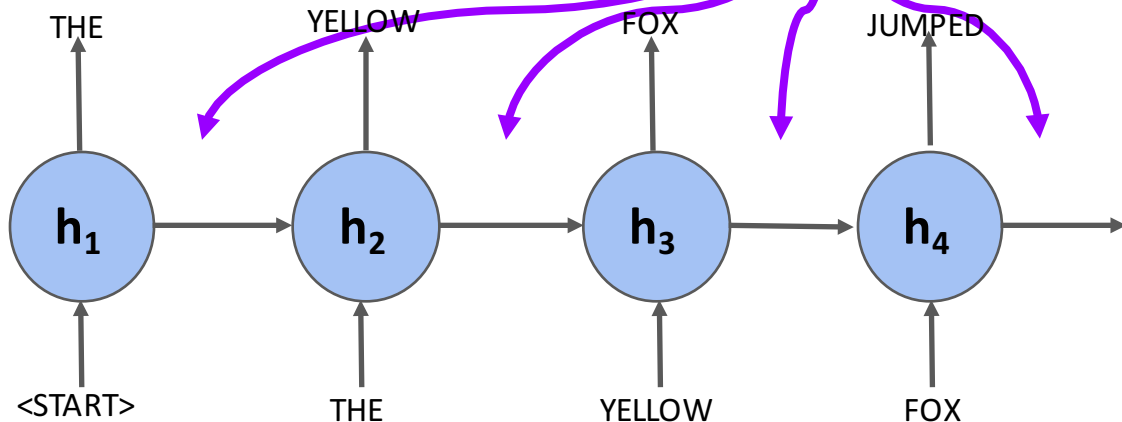
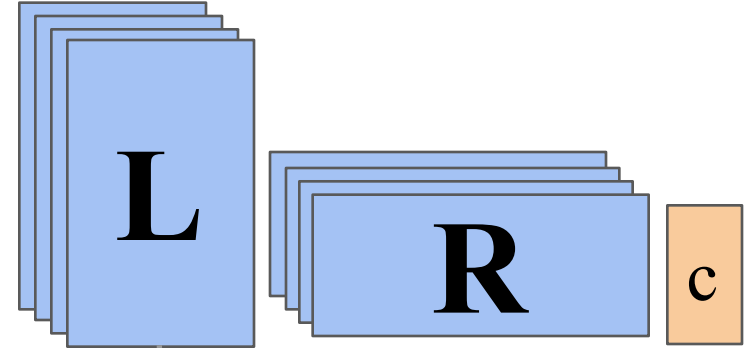
Generic weights



+



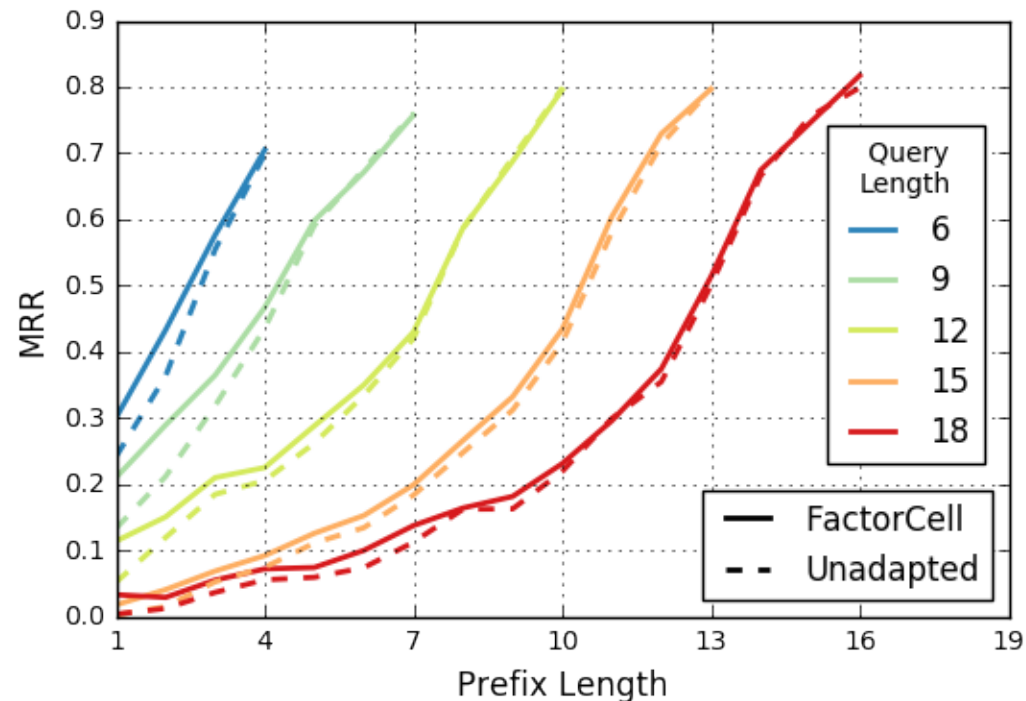
Low-rank adaptation



$$h_{t+1} = \sigma((\mathbf{W}_0 + \mathbf{W}_c)[h_t, e_t] + b)$$

Much larger change in recurrent layer than what ConcatCell does

# Prefix and query length



- Longer queries are more difficult
- Suggestion quality improves as prefix length increases