Scalable Semantic Matching of Queries to Ads in Sponsored Search

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, Ricardo Baeza-Yates, Andrew Feng, Erik Ordentlich, Lee Yang, Gavin Owens

Yahoo Research, Advertising Sciences
Sponsored Search

- **Advertisers**
  - Provide ad creative (title, description, url)
  - Provide bidterms (queries they want their ad to show for)

- **Search Engine**
  - Matches queries to bidterms (exact match + variant match)
  - Implements: **broad match**
**Sponsored Search**

**Broad Match:** advanced matching to non-provided keywords by:

- **Query rewriting:**
  - Given a user query, find $K$ semantically similar queries

- **Query-ad matching:**
  - Need to place queries and ads in same feature space
How to represent queries and ads?

1) Traditional – Bag of words

Query: cheap flights

Ad: Get the best air ticket deals

ISSUE – No way we can find that this query and this ad are related
Query and Ad Representations

2) New – move from sparse to dense vectors

- Represent **queries** and **ads** as **numeric vectors**
- Vectors need to be learned using training data (**search sessions**)  
- We want queries/ads with **similar contexts** to have **similar vectors**

<table>
<thead>
<tr>
<th>Session 1</th>
<th>Session 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>trip_ideas</td>
<td>trip_ideas</td>
</tr>
<tr>
<td>cheap_flights</td>
<td>air_tickets</td>
</tr>
<tr>
<td>holiday_travel_deals</td>
<td>holiday_travel_deals</td>
</tr>
</tbody>
</table>

Query → 0.2 1.1 7.2 0.8 3.1 vector

Ad → 0.9 2.6 3.1 0.1 2.2 vector
search2vec = word2vec [1] where:

- words = \{queries, search ads, search links\}
- documents = search sessions (uninterrupted sequences of user actions on the search engine)

Search Sessions Dataset

S1  hoka_running_shoe_reviews adid_2283077190 hoka_shoes_for_bad_feet hoka_shoes amazon zappos_shoes slc_231234142

S2  king_tut king_tut_exibit king_tut_exibit_seattle adid_3858375378 gas_caps gas_capReplacement_for_cars adid_1066604760 gas_doorReplacement_for_cars slc_81285142 fuel_door_covers autozone_auto_parts adid_253157233

S3  hoka_one_one run_florida hoka_shoes shoes_with_sl_2_last shoes_with_a_bigger_toe_box stans_shoes clarks_shoes slc_1567342

S4
Example search session:

Query8, Ad1, Query2, Query6

\[ v_{i}^{new} = v_{i} + \eta \left( 1 - \sigma(v_{i}^T u_{neigh}) \right) u_{neigh} \]

\[ v_{i}^{new} = v_{i} - \eta \sigma(v_{i}^T u_{neg}) u_{neg} \]

current query/ad

Q8 A1 Q2 Q6

neighborhood

embedding space

Search2Vec
Search2Vec – after training

Query-to-Query similarity

cheap_flights

0.2 1.1 7.2 0.8 3.1

air_ticket_deals

0.21 1.2 6.8 0.74 3.2

cosine similarity=0.9
Search2Vec – after training

Query Rewriting

Try our state-of-the-art system for query rewriting.

<table>
<thead>
<tr>
<th>scuba</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Relevance</td>
</tr>
<tr>
<td>No results</td>
<td>+</td>
</tr>
</tbody>
</table>

Input Type:

- Search Query

Output Type:

- Search Query

Search
Query-to-Ad matching

ad_243609_341454

| 0.2 | 1.1 | 7.2 | 0.8 | 3.1 |

similarity=0.871

mystery_games

| 0.2 | 1.2 | 6.8 | 0.7 | 3.2 |

ad id: 243609
bidterm id: 341454

ad title: Host a Fun Murder Mystery Party

ad description: Huge selection of fun murder mystery games for all ages, groups. #1 site for instant downloads and boxed sets of exciting murder mystery party games

mystery
mystery games
mystery games
murder mystery party
mystery party
generated keywords
free murder mystery games for parties
detective games
murder mystery game
how to host a mystery party for kids
murder mystery dinner
friends game night
murder mystery parties at home
murder mystery dinner party
...
How can we leverage additional **search-specific context**?

What is context for a query in web search?

1. **Other queries** – in user search sessions
2. **Ads** - clicked (positives), **dwell time**, skipped ads (negatives)
3. **Search results** - clicked (represented with url)
Search2Vec – additional context

Dwell-time sensitive updates

- gradient multiplier: \( n_i = \log(1+t) \)
  \( t = \) dwell time in minutes

- ad clicks with longer dwell time
  - \( \rightarrow \) larger learning rate

- ad clicks with short dwell time
  - \( \rightarrow \) small learning rate
Ad skips as implicit negative signal

- Skipped ads = ads at high positions skipped in favor of ad click at lower position

**Session:**
- Q1
  - Ad click 5
  - Ad view 3
  - Ad view 8
- Q2
  - Ad view 1
  - Ad click 4
- Q1
  - Ad view 2
  - Ad view 5
  - Ad view 6
Search2Vec – final model

- \( a = \text{action (q, ad, slc)} \)
- \( D = \text{immediate context as positives} \)
- \( Dr = \text{random negatives (5 per session)} \)
- \( Dn = \text{implicit negatives (skipped ads)} \)

- \( v = 300 \text{ dim vector} \)
- \( c = 5 \) (context window size)
- 10B sessions -> 80M vectors

\[
\begin{align*}
\arg\max_{\theta} \sum_{(a,c) \in D} \log \frac{1}{1 + e^{\langle v_c, v_a \rangle}} + \sum_{(a,c) \in Dr} \log \frac{1}{1 + e^{\langle v_c, v_a \rangle}} + \sum_{(q,ad) \in Dn} \log \frac{1}{1 + e^{\langle v_q, v_{ad} \rangle}}
\end{align*}
\]
20K judgments <query, ad, score, grade>:
- grade = {Bad, Fair, Good, Excellent, Perfect}
- <cheep tickets, travelocity ad, 0.831, Perfect>
Search2Vec – A/B test

- 2 ways to increase Revenue Per Search:

1) Increase Depth: find more ads for queries that have ads

   - Query 1: Ad 1, Ad 2, Ad 3, Ad 4, Ad 5
   - Query 2: Ad 6, Ad 7

2) Increase Coverage: find ads for queries that do not have ads

   - Query 3: Ad 8, Ad 9
   - Query 4: Ad 5

   - Ads: existing ads
   - Ads: new ads
Search2Vec – A/B test

- For each query find closest 30 ads in embedding space above 0.7 similarity and store in a `<query, ad list>` table

- **Control**: does not include this table:

- **Bucket**: includes this table:

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Query Coverage</th>
<th>Auction Depth</th>
<th>CTR</th>
<th>Click Yield</th>
<th>Revenue per Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-machine</td>
<td>+1.14%</td>
<td>+2.13%</td>
<td>+0.5%</td>
<td>+1.7%</td>
<td>+7.07%</td>
</tr>
</tbody>
</table>

- Low overlap with other match types: 90% pairs are unique
Search2Vec – Limitations

1. **size of vocabulary**
   - **problem**: single 256GB machine can train up to 80M vectors
   - **solution**: distributed training

2. **cold ads**
   - **problem**: new ads added daily (no clicks to train ad vectors)
   - **solution**: content vectors (create ad vectors from ad text)

3. **tail queries**
   - **problem**: not enough observations to train a query vector
   - **solution**: index head query vector-based expansions
Distributed Search2Vec

Training

- Initialize pair of vectors $\mathbf{v}$ (input) and $\mathbf{u}$ (output) for each word in vocab
- Update $\mathbf{v}$ of center word and $\mathbf{u}$'s of neighbors and random negatives

```
  w1 w2  w3 w4 w5 w6 w7 w8 w9 ...
  w3 w4  w6 w7
  w3 w4  w6 w7
```

- Updates involve vector multiply-accumulates ($\mathbf{v}+=\alpha \mathbf{u}$, $\mathbf{u}+=\alpha \mathbf{v}$, $\mathbf{v}+=\beta \mathbf{u}$, $\mathbf{u}+=\beta \mathbf{v}$), with $\alpha$, $\beta$ determined by ($\mathbf{u}\cdot\mathbf{v}$, $\mathbf{u}\cdot\mathbf{v}$).
Parameter Server (PS) - distributed in-memory store for model parameters (vectors), supports: GET, PUT

1 Client:
- Take a mini-batch of data (e.g. 200 sessions)
- PS GET: v vectors for each word from mini-batch and u vectors for neighbors and random negatives
- Client calculates gradient updates for all v and u
- PS PUT: updates v and u vectors in key-value store (no locks)
Distributed Search2Vec

Our solution:

Negative sampling, compute $u \cdot v$

PS Shards:

Send word indices and seeds

Clients:

Update vectors ($v += au$, $\ldots$)

Each shard stores a part of every vector

Aggregate results & compute “$a$, $\beta$”

We can train large vocab model in 1 day
Distributed Search2Vec

1. send word indices and seed
   
   waits ....

3. aggregate partial dot products and compute $\alpha$, $\beta$ weights

   $$\alpha_{\text{full}} = \alpha_{1-30} + \alpha_{31-60} + \ldots + \alpha_{371-300}$$

   $$\beta_{\text{full}} = \beta_{1-30} + \beta_{31-60} + \ldots + \beta_{371-300}$$

4. sends weights and seeds (again)

   next mini-batch ...

2. negative sampling + calculate partial dot products $u \cdot v$

   $$\alpha_{1-30} = \sum_{i=1}^{30} v_i u_i^{\text{pos}}$$

   $$\beta_{1-30} = \sum_{i=1}^{30} v_i u_i^{\text{neg}}$$

5. update partial vectors $v += \alpha u$, ...

   $$v_{\text{new}}^{\text{pos}} = v_{1-30}^{\text{pos}} + \eta (1 - \sigma(\alpha_{\text{full}})) u_{1-30}^{\text{pos}}$$

   $$v_{\text{new}}^{\text{neg}} = v_{1-30}^{\text{neg}} - \eta \sigma(\beta_{\text{full}}) u_{1-30}^{\text{neg}}$$

   $$u_{\text{pos}}^{\text{new}} = u_{1-30}^{\text{pos}} + \eta (1 - \sigma(\alpha_{\text{full}})) v_{1-30}^{\text{pos}}$$

   $$u_{\text{neg}}^{\text{new}} = u_{1-30}^{\text{neg}} - \eta \sigma(\beta_{\text{full}}) v_{1-30}^{\text{neg}}$$
Distributed S2V – A/B test

More vectors: ~300M query & ad vectors

- **Control**: prod + 1 machine s2v
- **Bucket**: prod + 1 machine s2v + distributed s2v

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Query Coverage</th>
<th>Auction Depth</th>
<th>CTR</th>
<th>Click Yield</th>
<th>Revenue per Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>distributed search2vec</td>
<td>+2.44%</td>
<td>+2.39%</td>
<td>+0.2%</td>
<td>+1.81%</td>
<td>+9.39%</td>
</tr>
</tbody>
</table>
Cold-start ad vectors

2. cold ads

How to generate vectors for new ads?

<table>
<thead>
<tr>
<th>Title</th>
<th>Ancestry DNA Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Learn More About Yourself &amp; Your Family History.</td>
</tr>
<tr>
<td>Display URL</td>
<td>23andme.com/AncestryDNATesting</td>
</tr>
<tr>
<td>Bid Term</td>
<td>dna_testing</td>
</tr>
</tbody>
</table>
Cold-start ad vectors

<table>
<thead>
<tr>
<th>source</th>
<th>n-gram</th>
<th>has vector</th>
<th>similarity to bid term</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>ancestry</td>
<td>YES</td>
<td>0.66</td>
</tr>
<tr>
<td>title</td>
<td>dna</td>
<td>YES</td>
<td>0.76</td>
</tr>
<tr>
<td>title</td>
<td>testing</td>
<td>YES</td>
<td>0.24</td>
</tr>
<tr>
<td>title</td>
<td>ancestry_dna</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>dna_testing</td>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>title</td>
<td>ancestry_dna_testing</td>
<td>YES</td>
<td>0.87</td>
</tr>
<tr>
<td>description</td>
<td>learn</td>
<td>YES</td>
<td>0.11</td>
</tr>
<tr>
<td>description</td>
<td>more</td>
<td>YES</td>
<td>0.03</td>
</tr>
<tr>
<td>description</td>
<td>learn_more</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>about</td>
<td>YES</td>
<td>0.08</td>
</tr>
<tr>
<td>description</td>
<td>family_history</td>
<td>YES</td>
<td>0.62</td>
</tr>
<tr>
<td>description</td>
<td>your_family</td>
<td>YES</td>
<td>0.37</td>
</tr>
</tbody>
</table>

\[ v_{ad\_content} = \prod_{i} v_p(i) \]
Cold-start ad vectors

Offline Evaluation

- Ad data
  - ad meta-data
  - content vectors
    - 30M ads
    - context vectors
      - cosine similarity
      - \( V_{\text{ad-context}} \)

- Train data
  - search sessions
  - 30M ads
  - 260M queries

- Ad data
  - ad meta-data
  - content vectors
    - 30M ads
    - context vectors
      - cosine similarity
      - \( V_{\text{ad-context}} \)
Cold-start ad vectors

- **Offline Evaluation**
  - $V_{\text{ad-context}}$: ad vectors learned from sessions
  - $V_{\text{ad-content}}$: ad vectors formed from content
  - $\text{sim}$: average cosine sim. between $V_{\text{ad-context}}$ and $V_{\text{ad-content}}$
  - High $\text{sim}$ tells us we came close to the “ground truth”

<table>
<thead>
<tr>
<th>method</th>
<th>average</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>0.574</td>
<td>0.059</td>
</tr>
<tr>
<td>phrases</td>
<td>0.665</td>
<td>0.067</td>
</tr>
<tr>
<td>CRF phrases</td>
<td>0.604</td>
<td>0.075</td>
</tr>
<tr>
<td>bid term only</td>
<td>0.731</td>
<td>0.128</td>
</tr>
<tr>
<td>anchor phrases</td>
<td><strong>0.792</strong></td>
<td><strong>0.077</strong></td>
</tr>
</tbody>
</table>
Cold-start ad vectors – A/B tests

More ad vectors: additional 50M ad vectors

- **Control**: prod + distributed s2v
- **Bucket**: prod + distributed s2v + cold ad vectors

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Query Coverage</th>
<th>Auction Depth</th>
<th>CTR</th>
<th>Click Yield</th>
<th>Revenue per Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold Start Ad Vectors</td>
<td>+7.05%</td>
<td>+4.36%</td>
<td>-0.6%</td>
<td>+3.96%</td>
<td>+9.83%</td>
</tr>
</tbody>
</table>
Our system today

- Tail queries A/B tests – still to come

- search2vec today:
  - top BROAD match algorithm
  - 30%+ of all BROAD match impressions

- Read more about it at: yahooresearch.tumblr.com

Vectors for Research Purposes

- 8M query vectors + 4K <query, query, grade> data available
- Webscope program:
- Comparison to word2vec on query rewriting task:

<table>
<thead>
<tr>
<th>Method</th>
<th>oAUC</th>
<th>Macro NDCG@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>word2vec</td>
<td>0.817</td>
<td>0.929</td>
</tr>
<tr>
<td>search2vec</td>
<td>0.880</td>
<td>0.959</td>
</tr>
</tbody>
</table>
Thank You!

Questions?
3. **tail queries**

How to generate vectors for tail queries?

How to do online matching and leverage search2vec?

- Build an index for online matching
- Leverage head queries and form documents from their search2vec rewrites (gives us semantic expansions)
- For a new query: textual match against document, retrieve vector of the top result
3. **tail queries**

**Step 1:** find top \( K = 10 \) queries for each head query from the vocabulary

<table>
<thead>
<tr>
<th>query</th>
<th>expansions</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>scuba diving</td>
<td>equipment</td>
<td>0.792</td>
</tr>
<tr>
<td>scuba diving</td>
<td>gear</td>
<td>0.766</td>
</tr>
<tr>
<td>scuba diving</td>
<td>gear</td>
<td>0.765</td>
</tr>
<tr>
<td>scuba diving</td>
<td>equipment</td>
<td>0.764</td>
</tr>
<tr>
<td>scuba shop</td>
<td></td>
<td>0.763</td>
</tr>
<tr>
<td>bread maker</td>
<td></td>
<td>0.728</td>
</tr>
<tr>
<td>bread machines</td>
<td></td>
<td>0.722</td>
</tr>
<tr>
<td>bread machine</td>
<td>reviews</td>
<td>0.621</td>
</tr>
<tr>
<td>bread machine</td>
<td>recipes</td>
<td>0.605</td>
</tr>
<tr>
<td>met opera</td>
<td>address</td>
<td>0.824</td>
</tr>
<tr>
<td>met opera</td>
<td>nyc</td>
<td>0.819</td>
</tr>
<tr>
<td></td>
<td>metropolitan opera</td>
<td>0.805</td>
</tr>
<tr>
<td>met opera</td>
<td>nyc</td>
<td>0.793</td>
</tr>
<tr>
<td>met</td>
<td>opera</td>
<td>0.790</td>
</tr>
<tr>
<td>free stock</td>
<td>tickers</td>
<td>0.763</td>
</tr>
<tr>
<td>stock ticker</td>
<td>app</td>
<td>0.760</td>
</tr>
<tr>
<td>stock pro</td>
<td>best real time stock apps</td>
<td>0.757</td>
</tr>
<tr>
<td>stock pro</td>
<td>best stock tracker app</td>
<td>0.741</td>
</tr>
<tr>
<td>free stock</td>
<td>apps</td>
<td>0.732</td>
</tr>
</tbody>
</table>
3. **tail queries**

**Step 2: form query documents (flatten)**

<table>
<thead>
<tr>
<th>id</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>scuba_diving_gear</strong></td>
<td>scuba diving equipment diving gear scuba equipment scuba gear scuba shop</td>
</tr>
<tr>
<td><strong>bread_machines</strong></td>
<td>bread maker bread machines cusinart bread maker bread machine reviews bread machine recipes</td>
</tr>
<tr>
<td><strong>met_opera_ny</strong></td>
<td>met opera address met opera nyc metropolitan opera house new york city met opra metropolitan opera in nyc</td>
</tr>
<tr>
<td><strong>stock_pro</strong></td>
<td>free stock tickers stock ticker app best real time stock apps best stock tracker app free stock apps</td>
</tr>
</tbody>
</table>
### Tail query vectors

#### Step 3: invert index for fast matching

<table>
<thead>
<tr>
<th>input query</th>
<th>top result</th>
<th>matched document</th>
</tr>
</thead>
<tbody>
<tr>
<td>metropolitan opera house that is in new york city</td>
<td>new_york_city_opera</td>
<td>metropolitan opera tosca ny city opera metropolitan opera promo code new york city opera new york city opera company metropolitan opera website metropolitan opera house lincoln center met opera new york metropolitan opera dress code metropolitan opera discount tickets</td>
</tr>
<tr>
<td>malware bytes free edition software download</td>
<td>free_antimalware_software</td>
<td>free malware software download malware bytes download free malwarebytes downloads anti malware malware anti malware antivirus free download norman malware free anti malwarebytes free edition free antimalware software</td>
</tr>
<tr>
<td>what is the best stock ticker trading app in appstore?</td>
<td>stock_pro</td>
<td>free stock tickers stock ticker app best stock chart best real time stock charts best stock tracker app free stock apps stock tracker software good stocks to day trade free stock market ticker stock pro</td>
</tr>
</tbody>
</table>
Evaluation

Tail query vectors

- Evaluation

40M

50M

ordered by frequency

use for creating index

use for testing the matching

index

V_{q-context}

V_{q-index (top result)}

cosine similarity

0.3 1.3 6.2 0.5 3.1

0.2 1.2 6.8 0.7 3.2
## Tail query vectors - Evaluation

### Offline Evaluation

- $V_{q\text{-context}}$: query vectors learned from sessions (50M)
- $V_{q\text{-index}}$: query vectors formed by leveraging index (50M)
- $\text{sim}$: average cosine sim. between $V_{q\text{-context}}$ and $V_{q\text{-index}}$
- High $\text{sim}$ tells us we came close to the “ground truth”

<table>
<thead>
<tr>
<th>method</th>
<th>average</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>words</td>
<td>0.452</td>
<td>0.101</td>
</tr>
<tr>
<td>phrases</td>
<td>0.574</td>
<td>0.120</td>
</tr>
<tr>
<td>CRF phrases</td>
<td>0.514</td>
<td>0.119</td>
</tr>
<tr>
<td>elastic co-occurred queries $K=10$</td>
<td>0.621</td>
<td>0.084</td>
</tr>
<tr>
<td><strong>elastic s2v $K=10$</strong></td>
<td><strong>0.717</strong></td>
<td><strong>0.091</strong></td>
</tr>
</tbody>
</table>