Deep Semantic Matching for Amazon Product Search

Yiwei Song

Amazon Product Search
Deep Semantic Matching for Amazon Product Search

Yiwei Song
Priyanka Nigam
Vijai Mohan
Vihan Lakshman
Weitian Ding
Ankit Shingavi
Choon Hui Teo
Hao Gu
Bing Yin
Amazon Product Search

• Amazon is 4th most popular site in US [1]
• Majority of Amazon retail revenue is attributed to search
• Nearly half of US internet users start product search on Amazon [2]

[2] https://retail.emarketer.com/article/more-product-searches-start-on-amazon/5b92c0e0ebd40005bc4dc7ae
Semantic Matching in Product Search

• Goal of Semantic Matching is to reduce customers’ effort to shop
  Reduced query reformulations
  Bridge the vocabulary gap between customers’ queries and product description
What is a match for a query?

Lexical Match

Zion Health Adama Clay Minerals Shampoo 16 Fluid Ounce by Zion Health

“health shampoo”
What is a match for a query?

Lexical Match

"health shampoo"

Semantic Match

Zion Health Adama Clay Minerals Shampoo - 16 Fluid Ounce
by Zion Health

ArtNaturals Organic Moroccan Argan Oil Shampoo and Conditioner Set - (2 x 16 Fl Oz / 473ml) - Sulfate Free - Volumizing & Moisturizing - Gentle on Curly & Color Treated Hair - Infused with Keratin
by ArtNaturals
What is a match for a query?

Lexical Match

“countertop wine fridge”

Antarctic Star 17 Bottle Wine Cooler/Cabinet Refrigerator Small Wine Cellar Beer Counter Top Fridge Quiet Operation Compressor Freestanding Black by Antarctic Star
What is a match for a query?

**Lexical Match**

“countertop wine fridge”

Antarctic Star 17 Bottle Wine Cooler/Cabinet Refrigerator Small Wine Cellar Beer Counter Top Fridge Quiet Operation Compressor Freestanding Black by Antarctic Star

**Semantic Match**

DELLA 048-GM-48197 Beverage Center Cool Built-in Cooler Mini Refrigerator w/ Lock- Black/Stainless Steel by DELLA
Semantic Matching augments Lexical Matching

Query

Neural Network

Query Embedding

KNN Search

Product Embeddings

P1: 0.12598, 0.058533, 0.010078, 0.045166, -0.09845, 0.014076, …
P2: 0.051819, 0.0054588, 0.0047226, 0.045959, -0.015015, …
P3: 0.010887, -0.015808, -0.098145, 0.039215, -0.058655, -0.085388, …
P4: 0.042053, 0.087402, 0.070129, 0.082397, -0.051056, -0.089478, …

Semantic Matches

Merge

Ranking

Lexical Matches
Neural Network Representation Model

Widely adopted in web search engines: Google, Microsoft Bing, Baidu, Yahoo

Product search?
- Shorter queries with patterns
- Structured product attributes
- Smaller search space and non purchased are equally important
Data

“artistic iphone 6s case”

purchased
Data

“artistic iphone 6s case”

purchased

Impressed but not purchased
Data

“artistic iphone 6s case”

purchased

Impressed but not purchased

Random
Loss Function

“artistic iphone 6s case”

- **purchased**
  - High
- **Impressed but not purchased**
  - Medium
- **Random**
  - Low

Similarity between Query Embed and Product Embed
Loss Function

- For purchases:
  \[
  \text{loss}(y, \hat{y}) = \begin{cases} 
  0, & \hat{y} \geq 0.9 \\
  (\hat{y} - 0.9)^2, & \hat{y} < 0.9 
  \end{cases}
  \]

- For impressed but not purchased:
  \[
  \text{loss}(y, \hat{y}) = \begin{cases} 
  0, & \hat{y} \leq 0.55 \\
  (\hat{y} - 0.55)^2, & \hat{y} > 0.55 
  \end{cases}
  \]

- For randomly-sampled:
  \[
  \text{loss}(y, \hat{y}) = \begin{cases} 
  0, & \hat{y} \leq 0.2 \\
  (\hat{y} - 0.2)^2, & \hat{y} > 0.2 
  \end{cases}
  \]

- Point-wise hinge loss for 3 data types
- Margins are hyper-parameters tuned on our dataset
- Loss of each data point is weighted by purchase and impression count of query-product pair
Loss Function
N-gram Average Neural Network

Cosine Similarity

Query Embedding

Average, Normalize, Activation

Dense Layer

Product Embedding

Average, Normalize, Activation

Shared Embedding Layer

Query Ngrams

N-gram Parser

Title Ngrams

Query

Product Title

Product Attributes
N-gram Average Neural Network

“artistic iphone 6s case”

“artistic", "iphone", "6s", "case",
"artistic#iphone", "iphone#6s", "6s#case",
"artistic#iphone#6s", "iphone#6s#case",
"#ar", "art", "rtl", ..., "#ca", "cas", "ase", "se#"

- **Word-ngrams capture phrase level information.**
  “chocolate milk” vs “milk chocolate”
  “for iphone” indicates iphone accessories

- **Character-ngrams capture subword information**
  Morphism: “chocolates” vs “chocolate”
  Compound words: “amazontv”, “firetvstick”
  Parts number: “36x14” vs “36 x 14”
  Typos: “iphone”

- **OOV ngrams?**
  Vocab size grows exponentially with n!
N-gram Average Neural Network

Build N-gram vocab by frequency
Hash OOV N-gram to a bin to group low count tokens

"artistic iphone 6s case"

"artistic", "iphone", "6s", "case",
"artistic#iphone", "iphone#6s", "6s#case",
"artistic#iphone#6s", "iphone#6s#case",
"#ar", "art", "rtl", ..., "#ca", "cas", "ase", "se#"

Out of Vocab?

No

Yes

Hash()

"artistic"
"iphone#6s"
"se#"

"artistic#iphone#6s"
"artistic#iphone"
N-gram Average Neural Network

- Word Unigram baseline on small dataset
N-gram Average Neural Network

- Word Unigram baseline on small dataset
- Use more data
N-gram Average Neural Network

- Word Unigram baseline on small dataset
- Use more data
- Add Word Bigram
N-gram Average Neural Network

- Word Unigram baseline on small dataset
- Use more data
- Add Word Bigram
- Add Character Trigram
N-gram Average Neural Network

- Word Unigram baseline on small dataset
- Use more data
- Add Word Bigram
- Add Character Trigram
- Add OOV hashing for ngrams
N-gram Average Neural Network

- Word Unigram baseline on small dataset
- Use more data
- Add Word Bigram
- Add Character Trigram
- Add OOV hashing for ngrams
- More tokens/parameters overfits on small dataset
Increase Vocab Size by Model Parallelism

Performance increases with more parameters in model

- 3000 MM
- 500 MM
- 180 MM

MAP

Epoch

Time to Train 1 Billion Params (Hours)

Number of GPUs

- 64
- 128
- 256
- 512
- 1024
- 2048
- 4096
- 8192
Structured Product Features

- Merge Structured Product Features with a dense layer after batch_norm and dropout
- Model learns to put weights on product attribute features
Still Day 1

We are building an engineering component to bring semantic matching from offline to online:

- KNN search over one hundred millions documents within milli-seconds latency
Still Day 1

- Better Matching Models
  - Richer text features: customer reviews and product description
  - Image feature
  - Better negative sampling
  - Query reformulate
- Ranking Improvements
  - Sparse features such as product id
Thank you

Questions? Want to join us?

ywsong@amazon.com

https://www.amazon.jobs/en/teams/search.html