Big Data for Product Search in eCommerce

Presentation - TTU 2016 Symposium on Big Data

Alan Mi

URL for slides: (link good for 3 days from 4/29)

https://goo.gl/50XKwa

2016-04-29
My Background

• Assistant Prof in CS Tsinghua U - Us vs Enterprises
• Investing and trading technology consulting / quant / NN modeling
• Aircraft Presence Prediction
• Ecommerce Product Search Alg’s, Data Science, NN
  alannmi@gmail.com
• Data Scientist
• Working for a Huge Retailer
• To talk from online retailers’ angle
• NN Slides (click here)
Outline

50 mins to consume -> Q&A
Fly over some slides
Worth your while

1. The Product Search Problem
2. Big Data in Search
3. Big Data for Optimization for Search Results
4. Big Data for Mining Click Log for Search
5. Appendix: Google’s word2vec
Keywords

• **Search Engine**: core software: Endeca, Solr, Google, Bing, …

• **Search**: Solution / Offering wrapping Search Engine

• **Product Search** =
  - Catalog Search =
  - eCommerce Search =~
  - eRetailer Site Search

• **Web Search** = Document Search
Outline

1. The **Product Search Problem**
2. Big Data in Search
3. Big Data for Optimization for Search Results
4. Big Data for Mining Click Log for Search
5. Appendix: word2vec
Product Search

Ecommerce Search – a comparison with Web Search:
• As profound as Google Search – how much math is behind it?!
• As lucrative to the retailer as Google Search is to Google
• As difficult (almost intractable) as Web Search problem Google faces
• Improvement never ending because profit keeps coming
So... The Search Problem

Google: document

Amazon: product record

Search phrase set

- washer
- handtruck
- dolly
- d3200

Document set

- Whirlpool WTW5000DW
- Gasket for strainer nut
- Handtruck A440
- Nikon D3200 DSLR Camera

Saudagor Shirt [washer friendly]
Product Search vs Web Search

• **Search is Search is Search**

• Product description **records** vs Web **documents** from crawling

• **Types:** Text vs. varieties (PDF, video,…) "neural networks" filetype:ppt OR filetype:pptx

• **Profit vs. commission – Money is Money is Money**

• **Scale:** Google much larger than Amazon

• **Sorting:** relevance/profitability/… vs relevance

• Fundamentally the **same challenges**

• **Overlap non-product Search** – CS questions – site search be dexterous

• **Overlap from Web Search Engines** – Google Shopping
Product Search – Example **Difficulties**

- **Customer Intent** - different types of queries
  - “a **good solid cheap bike on sale**” – for product
  - “where is my **bike order**?” – for CS document
  - “which store is close to me?” - Ditto

- **Context-sensitivity**
  - Browse to find “**washer**”, to mean “gasket”, unsuccessful
  - Switching to Search, assuming in **browsing context**
  - Result: **washers and dryers** – disastrous Search results!
Math Definition - an Optimization Problem

• Given
  Search phrase set $T$ and
  Document set $D$

• Find a list of relevance-descending associations where result relevance is maximized
  Relevance: short for true relevance
  Result relevance: Human reads first result page only – brute force cut-off, weighted by position, first in position more important

• Full associations: each $t$ in $T$ -> every $d$ in $D$, except most irrelevant (0.0)

• Not a math function
  If considering only relevant associations: $1-M$ $N-1$ $1-1$ (model #, SKU#)

• Ground truth for the objective function (customer feeling) – not easy
So... The Search Problem

Search phrase set:
- washer
- handtruck
- dolly
- d3200

Relevance:
- 100%
- 90%
- 100%

Document set:
- Google: document
  - Whirlpool WTW5000DW
  - Gasket for strainer nut
  - Handtruck A440
  - Nikon D3200 DSLR Camera

- Amazon: product record
  - Saudagor Shirt [washer friendly]
Evaluation of Search Result Accuracy

• Kaggle Data Science - Active Competitions: gesture recognition, chess ratings, traffic forecasting
• The Search relevance problem on Kaggle
• Now by human, expensive
• Search Engine Evaluation Problem
  Predict the relevance of search results
Outline

1. The \textbf{Product Search Problem}

2. \textbf{Big Data in Search}

3. Big Data \textit{for Optimization for Search Results}

4. Big Data \textit{for Mining Click Log for Search}

5. Appendix: word2vec
Google Search Engine Components

Major parts of **Hummingbird**:

- PageRank
- [Panda, Penguin, Payday] (spam)
- Pigeon (local)
- Top Heavy (ads)
- Mobile Friendly
- Pirate
- **Knowledge Graph (2012)** - semantic network on DBs
  - Obama=>the President, Apple=>brand:fruit,
  - “When did the first man reach the moon?” => moon landing wiki page
- **RankBrain (2015)**
Google’s RankBrain

- **Problem:**
  Search phrase does not appear in Web document
  long phrases – extract gist
  never seen queries: **15%**

- **E.g.:** What’s the title of the consumer at the highest level of a food
  chain -- linked to pages that contains “Predator”

- **Manual** synonyms & Knowledge Graph approach reached limit
Google’s RankBrain (Con’t)

• Google is reticent about how it works, info sketchy

• **Machine Learning**
  
  • Training offline, using *historical search data*
  
  • 3rd most important signal of 200, after links and content
  
  • Connects original and complicated queries into shorter answers

• Why embedding: plain words don’t show distance

• **word2vec**: word's relation to other words in context, encoded in 2-layer NN's hidden layer’s input weight matrix n-th row

• Vector: a point in N-dim space, *distance can be calc’ed*: boy-girl, boy-man
Atlas of the Brain –

• “nature” 28 April 2016

• Jack Gallant (neurosci. Berkeley): Our goal was to build a giant atlas that shows how meaning of words is represented in the brain

• A single brain spot is associated with a number of related words

• A single word lights up many spots
Brain Map

- Words and related terms exercise the same regions of the brain

  Thesaurus map
  Colocated: “wife”, “husband”, “children”, “parents”
  Colocated: “killed”, “convicted”, “murdered” and “confessed”

- Brain Map: [click here](#)
- The Nature Video: [click here](#)
- The Nature Article: [click here](#)
- The Guardian’s article for laymen [click here](#)
Cerebral Cortex -
Semantics Embedded, not Words
Cerebral Cortex Unfolded
Stories Are Read to Subjects with MRI On
Word “top” with Clothing
Word “top” with Numbers
Word “top” with Buildings (other hemisph.)
Different People – Semantics Encoding Diff?
Shedding Light, yes, but not Bionics!

• Confirmative of human efforts
• Barely scratched the surface – not enough principles learned for applying
Google’s **word2vec** 2-Layer Neural network
word2vec – Trained (Courtesy: Chris Moody)
word2vec – Trained (Courtesy: Chris Moody)
word2vec – Products Clustered by Similarity
word2vec Works Wonders!

• If you can remember one thing from my talk, watch this video:

Text By the Bay 2015: Chris Moody, A Word is Worth a Thousand Vectors (click here)
Outline

1. The **Product Search Problem**
2. Big Data in Search
3. Big Data for **Optimization**
4. Big Data for **Mining Click Log for Search**
5. Appendix: word2vec
Search Result Order

eCommerce: complex Optimization Problem

**User-controlled:**

- Relevance
- Popularity

**System-controlled:**

- Profitability
- **Low Profitability** – liquidation mode for defense
- ...

Search
**Search Result Order Optimization** – an Example

Metrics – bounded search in 7-D space, Time needed: $9^7$

<table>
<thead>
<tr>
<th>Metric</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td>0.00</td>
</tr>
<tr>
<td>Profit</td>
<td>0.40</td>
</tr>
<tr>
<td>Social Media</td>
<td>0.05</td>
</tr>
<tr>
<td>Ads</td>
<td></td>
</tr>
<tr>
<td>Hotness</td>
<td></td>
</tr>
<tr>
<td>Competition</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td></td>
</tr>
</tbody>
</table>
Problem: Combinatorial Explosion

• It takes too long – Doesn’t fit LP model - needs Big Data
• No disk space for historical input (log) data – needs Big Data

• Benign evolution of error surface day by day
• An error surface that can be taken advantage of
• Before throwing Big Data at the prob., you have to do due diligence
Objective Function: 2-D Illustration vs 7-D Space
Low Land Areas in **Moving Window** of Charts

- High chance: similar shape every day
- Daily lowland chart based sampling calculation
- Weekly full landscape calculation
- Weekly lowland chart tuning
- With full results, if a point in 7-D space is lower than 90% of other points, it’s a low land point
- Overlap 3 consecutive full results’ recurring low land points
Objective Function: 2-D illustration vs 7-D Space
Improve the Speed

Fast algorithm based on similar Error Surfaces

• Identify the recurring low area (X,Y) to search in
• Time needed: $9^7 \rightarrow 3^6$
• Still too long
Avoid Weights of Similar Proportions

Metrics – bounded search in 7-D space, Time needed: 9^7

- Sales
- Profit
- Social Media
- Ads
- Hotness
- Competition
- Returns
Fit for Big Data?

• 5000 optimization units parallel-able
• CPU bound
• Data is not too big
• Daily – overlapping problem
• **Big alg’s calling for Big Data, just for the sake of scalability**
• Yes, Spark!
Outline

1. The Product Search Problem
2. Big Data in Search
3. Big Data for Optimization
4. Big Data for Mining Click Log
5. Appendix: word2vec
Big Data for Mining Click Log for Search

• Gigantic amount of data
• But for extracting and storing true relevance / semantic association
History Repeats Itself

Three approaches for (eCommerce & web document) Search:

1. **Word match** - Classical “term-to-property”
2. **Concept match** - Semantic Network (Knowledge Graph, RankBrain) and
3. **Phantom match** - Did they put money where their month (or finger) is?
Mining the Historical Click Log

The basic idea

• Customer A searched for
t=“small red mountain bike for girls”
and bought product
p=”black electric skateboard”
(No keyword overlap!)

• Customer B searches for the same phrase, we should show p among other things

• Evidences accumulate – induction– synthesizing the evidences

• Not related to the “Also bought” feature on Amazon, transaction mining
Using GRNN for Search

- Tolerates missing properties
- Tolerates bad or missing spelling correction (misspelled words)
- RankBrain is using page view info? I think so
Relevance

• Nominal relevance = literal relevance = intuitive relevance
  “red bike” <-> “black bike for girls”

• Phantom relevance - whimsical
  “small red mountain bike for girls”
  <->
  “black electric skateboard”
  “pampers” <-> “cigar”

• True relevance: both

• Factual, evidence-based, behavioral, money-backed
General Regression Neural Networks (GRNN)

\[ Y(x) = \frac{\sum Y_i e^{-\frac{(d_i^2)}{2\sigma^2}}}{\sum e^{-\frac{(d_i^2)}{2\sigma^2}}} \]

Where,
\[ d_i^2 = (x - x_i)^T(x - x_i) \]
Term-Product Semantic Network Algorithm

Relevance search and anomaly detection in bipartite graphs
by Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, et al
New Relationships Created from Algorithm

Note: initial graph setup has Term and Product Nodes connect by PV relationships. The PV relationships have Pa and Pat properties on them. These remain constant.

We then create Uprod and Uterm relationships which we update each iteration.
GRNN Calls for Big Data

• **No training needed**
• Stores all “training” observations, the whole sample
• **Huge** disk space demand
• Innate parallel processing
• Feels like **Map-Reduce** computation structure
A Fundamental Challenge

• Crude queries too many to store whole and raw
• Google: significant percentage new queries
• Query decomposition - NLP could help
• How to solve the problem?
  Virgin land, or so I thought if we have a Googler in the audience 😊

On that note, in closing: Let me bid
  A challenge to the students:
  Go cultivating that virgin land
Thanks!
Outline

1. The Product Search Problem
2. Big Data in Search
3. Big Data for Optimization for Search Results
4. Big Data for Mining Click Log for Search
5. Appendix: word2vec
Google’s word2vec – an example

• Suppose our corpus has 3 sentences:
  • “the dog saw a cat”
  • “the dog chased the cat”
  • “the cat climbed a tree”

• Generate dictionary and encode words as 1-hot vectors, V=8
  • a       [1 0 0 0 0 0 0 0]
  • cat     [0 1 0 0 0 0 0 0]
  • chased  [0 0 0 1 0 0 0 0]
  • climbed [0 0 0 1 0 0 0 0]
  • dog
  • saw
  • the
  • tree    [0 0 0 0 0 0 0 1]
Google’s word2vec 2-Layer Neural network
Neural Network

• Input/output layers: 8 (size of vocabulary), V=8
• Hidden layer: 3 neurons, N=3
• WI: VxN=8x3, WO: NxV=3x8, WI & WO initialized randomly:

\[ WI = \]
\[
\begin{pmatrix}
-0.094491 & -0.443977 & 0.313917 \\
-0.490796 & -0.229903 & 0.065460 \\
0.072921 & 0.172246 & -0.357751 \\
0.104514 & -0.463000 & 0.079367 \\
-0.226080 & -0.154659 & -0.038422 \\
0.406115 & -0.192794 & -0.441992 \\
0.181755 & 0.088268 & 0.277574 \\
-0.055334 & 0.491792 & 0.263102
\end{pmatrix}
\]

\[ WO = \]
\[
\begin{pmatrix}
0.023074 & 0.479901 & 0.432148 & 0.375480 & -0.364732 & -0.119840 & 0.266070 & -0.351000 \\
-0.368008 & 0.424778 & -0.257104 & -0.148817 & 0.033922 & 0.353874 & -0.144942 & 0.130904 \\
0.422434 & 0.364503 & 0.467865 & -0.020302 & -0.423890 & -0.438777 & 0.268529 & -0.446787
\end{pmatrix}
\]
word2Vec Training

• Train it for relationship “cat” -> “climbed”:
  feed context word “cat”,
  target word “climbed” shows high probability

• Training data:
  Input X: [0 1 0 0 0 0 0 0] (Selector, selecting 2\textsuperscript{nd} row)
  Output Y: [0 0 0 1 0 0 0 0] (100% probability that “climbed”)

• Hidden layer vector doing dot product with WO produces similarity probabilities
word2Vec Training (Con’t)

• Initially with random weights:
  \[ H = X*W_I = [-0.490796 -0.229903 0.065460] \] (2\textsuperscript{nd} row of WI matrix)
  This vector is the embedding for “cat”, for the moment of training
  \[ Y=H*W_O = [0.100934 -0.309331 -0.122361 -0.151399 0.143463 -0.051262 -0.079686 0.112928] \]
  Where 0.100934= (-0.490796)*0.023074 +(-0.229903)*(-0.368008) + 0.065460*0.422434

• Normalize the output vector to [0, 1] for probability
• Back propagate error to train it …
• Actually we’ve been training it for relationship “climbed” -> “cat” as well as “cat” <-> “dog” … using search phrases
Result Interpretation

At the end of the training:
• Rows of WI are the vectors encoding the vocab words
• Let's say that at the end of the training, the second row is [-0.1 -0.44, +0.2], this is the vector for “cat”
• When search phrase is "cat", use 1-hot to locate this vector
  [-0.1 -0.44, +0.2]
• We look for words similar to cat - we find vectors similar to it
Methods: Euclidean distance, cosine similarity, or algorithms that avoids linear scan, e.g., LSH
• Or perform dot product with WO to get 8 probabilities and search for the largest few to deem them semantically close?