
# Semantic Product Search

Simon Hughes PhD, Principal Data Scientist (Core Search), Home Depot

# Talk Outline

- Challenges of Building a Semantic Product Search Engine

- Neural IR (Content Based Matching)
    - Metric Learning
    - Tokenization approaches
    - Neural architectures for semantic search
    - Mining for Training Data

- Extreme Multi-Label Classification (Classification Approaches)
    - Definition of the problem
    - Application to semantic product search (Amazon)

- System architecture for combining multiple retrieval models

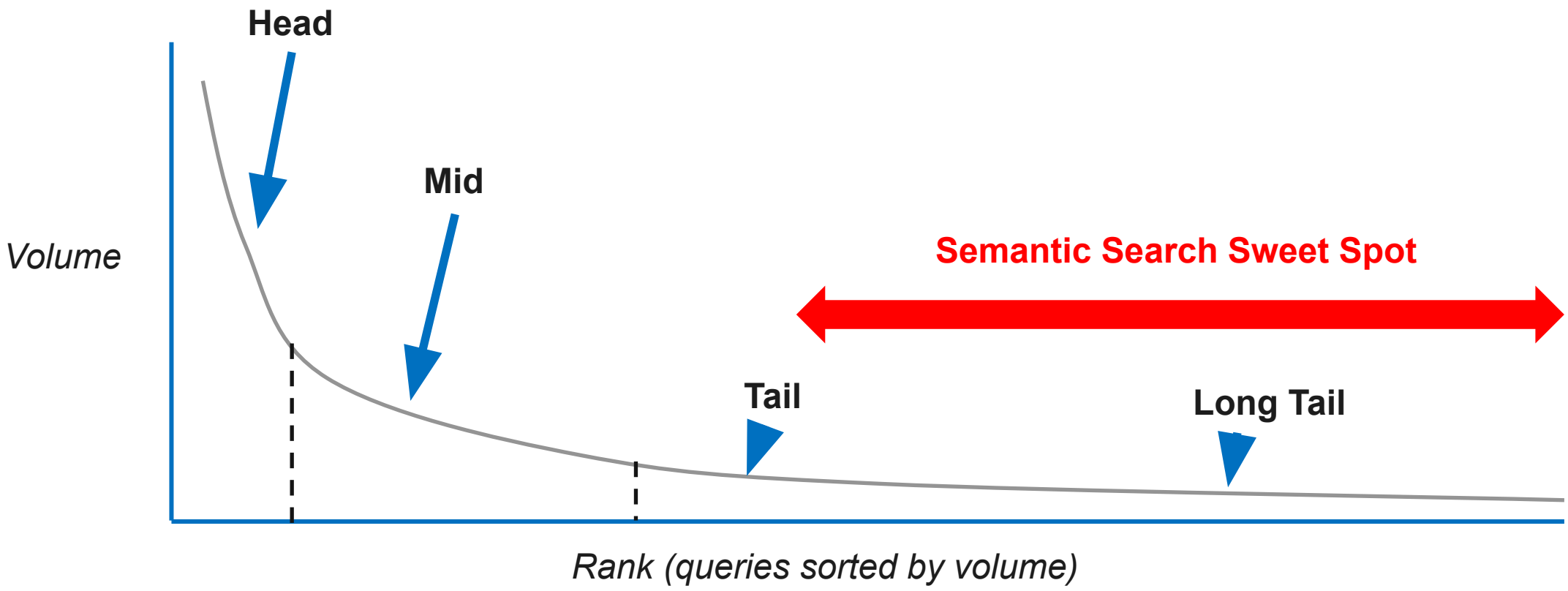
*Please note that any opinions shared in this talk are my own, and do not necessarily reflect those of Home Depot.

# Challenges of Building a Product Search Engine

- Wide variety of query types:
  - Dimension queries
    - Users specify dimensions in a variety of ways
    - E.g. 24 x 8 house vent, 24 inch by 8 inch house vent, 8 H by 24 W house vent
  - Non product queries
    - E.g. how do I install a toilet?
  - ID queries (SKU number, Model number, etc)
    - IDs have no semantic meaning
  - Typos
  - Non-English queries
  - Etc
- Synonyms
- Vocabulary miss match problem
- Colloquial Terms
  - E.g. Industry jargon
- SKU variants (size, dimension, color, etc)
  - Search engine should match variants of matching products only if they match attributes specified in the query
- Out of stock
  - Need to match alternative products when matching products unavailable

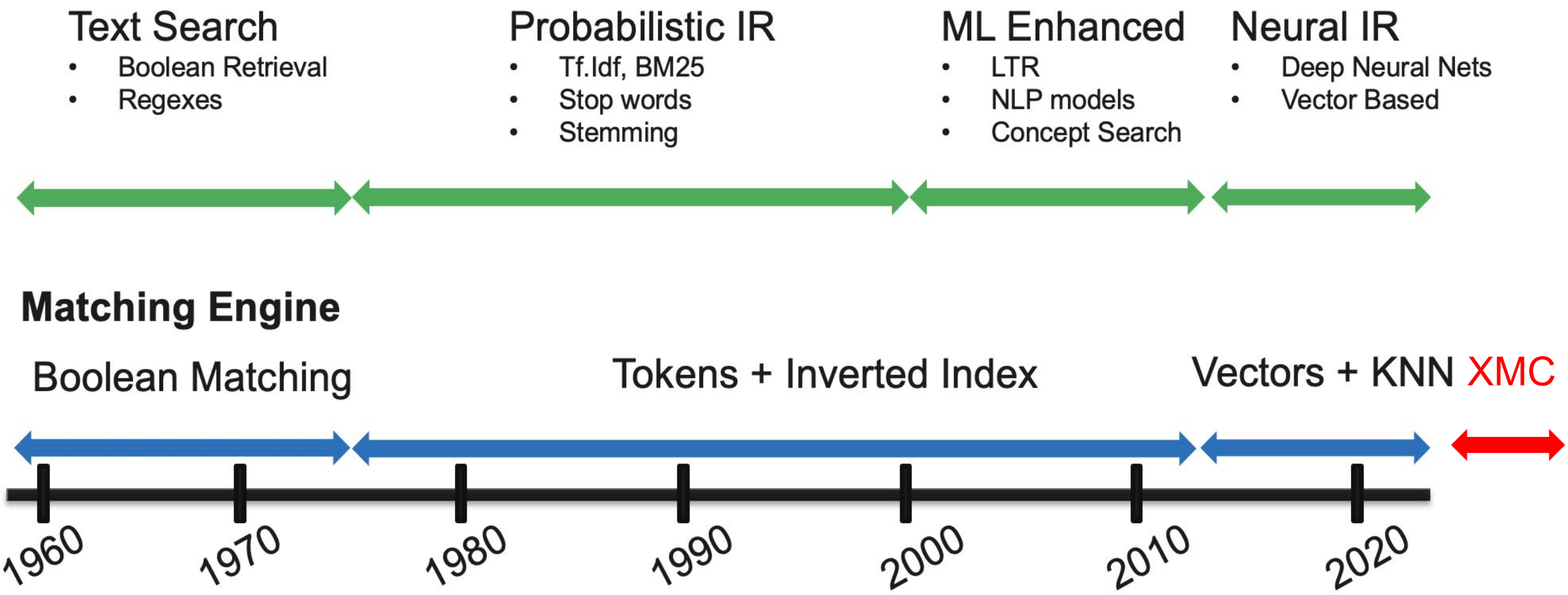
# Challenges of Building a Product Search Engine

- For most sites, queries follow Zipfian distribution (power law distribution)




*For illustration purposes only, not an actual chart based on our data

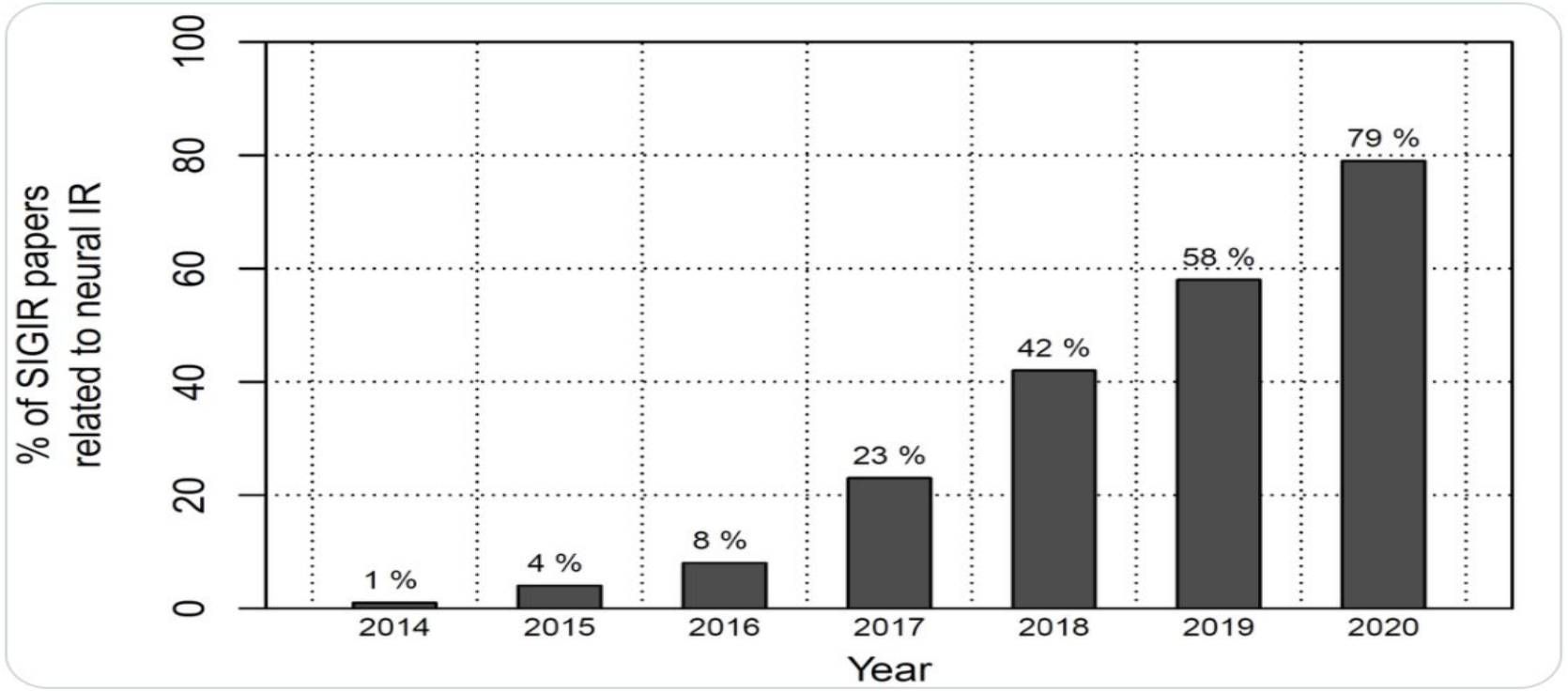
# A Brief History of Information Retrieval



# Popularity of Neural IR Over Time

**Bhaskar Mitra**
@UnderdogGeek

Not to break our annual tradition, here's the plot for the % of #NeuralIR papers at SIGIR after updating with #SIGIR2020 full papers. All usual disclaimers apply w.r.t. how you interpret this trend.




From Bhaskar Mitra on Twitter - https://twitter.com/UnderdogGeek/status/1288556021414277120

# Retrieve (Match) then Re-Rank

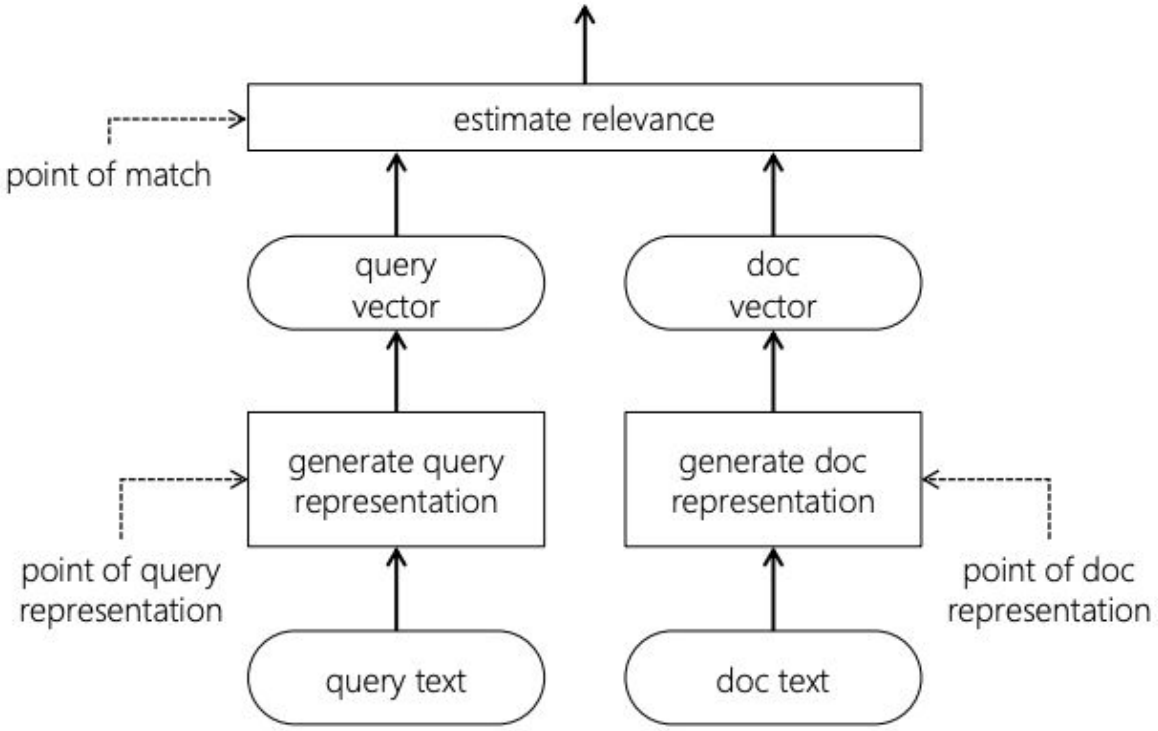
- Traditional keyword (BM25) search:
  - Boolean Retrieval
    - Which set of documents matches some or all of the query keywords?
    - Leverages inverted index and bitwise operations for speed
    - Determines which documents match, ignores relative document ordering
  - Re-Rank
    - Matching set of documents re-ranked using Okapi BM25 scoring / tf.idf
    - Different fields usually given different weighting
    - Boost queries and function queries used to implement business rules

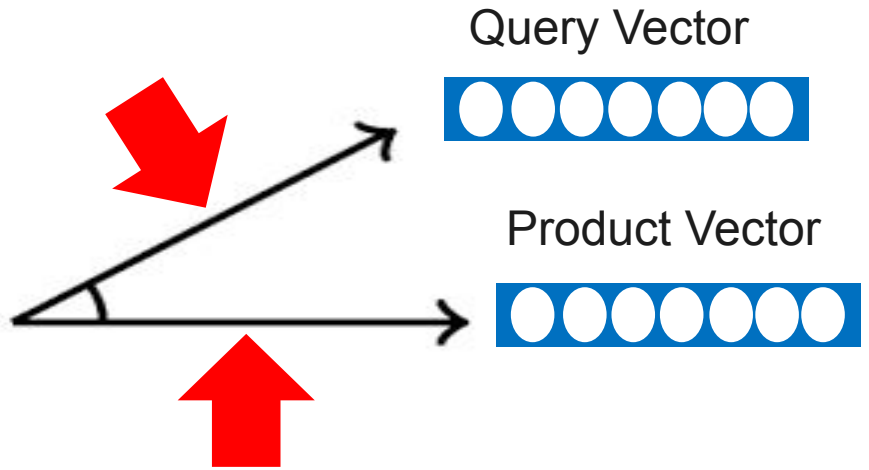
- Machine Learning Semantic Search:
  - ML Driven Retrieval
    - One or more retrieval model(s) -> initial match set
    - Models **optimize for recall** (reduce false negatives)
    - Leverages approximate KNN search algorithms
  - Re-Rank
    - Match set passed through one or more cascading LTR models
    - Models **optimize for precision** (reduce false positives)

- In this talk **I will focus on models for matching (retrieval),** which is mainly where the semantic matches occur

# Neural IR and Metric Learning

- Neural IR models learn a similarity function between the query and the product
- Usually this takes the form of 'Embedding Based Retrieval' – a vector is generated for the query and the product
- The neural model maximizes the similarity between queries and purchased products, and minimizes the similarity between queries and irrelevant products




- Retrieval is then done using approximate K-nearest neighbor search
- Specialized algorithms such as HNSW speed up KNN from linear to sub-linear time complexity at cost of accuracy

Left image taken from – 'Neural Models for Information Retrieval' - https://arxiv.org/pdf/1705.01509.pdf, p11

# Tokenization for Neural IR Models

- More traditional approaches still common:
  - Unigram, bigram, character trigram
  - More used in simpler models such as average embeddings models or feed forward models (e.g. DSSM)

- Trainable tokenizers – Subword Tokenization
  - Used in language models (e.g. BERT) and machine translation models
  - Typically used with more complex models such as transformers (and not models using average pooling or similar)
  - Uses a compression algorithm to break words into character sequences so as to optimize data compression
  - This results in more common character sequences producing longer sub-word pieces, while rarer sequences broken into shorter sequences
  - Example algorithms
    - BPE (Byte pair encoding)
    - Language modeling variants – WordPiece (BERT) and SentencePiece (ALBERT, XLNet)

- Typo correction?
  - Sub word tokenization and character ngrams can make the model more robust to typos and noise in the query terms
  - A spell checker may also be used at training and inference time

# Handling Rare and 'Out of Vocabulary' Tokens

- Machine learning models can only typically learn to represent tokens seen in the training data ('in vocabulary')
- We need to determine how to handle unseen words – **'Out of Vocabulary**' tokens, which will otherwise be ignored
- This limitation does not exist in traditional keyword retrieval, which can match any token occurring in any document

**Sources of OOV Tokens:**

- Typos in query terms
- Very rare (long tail) queries

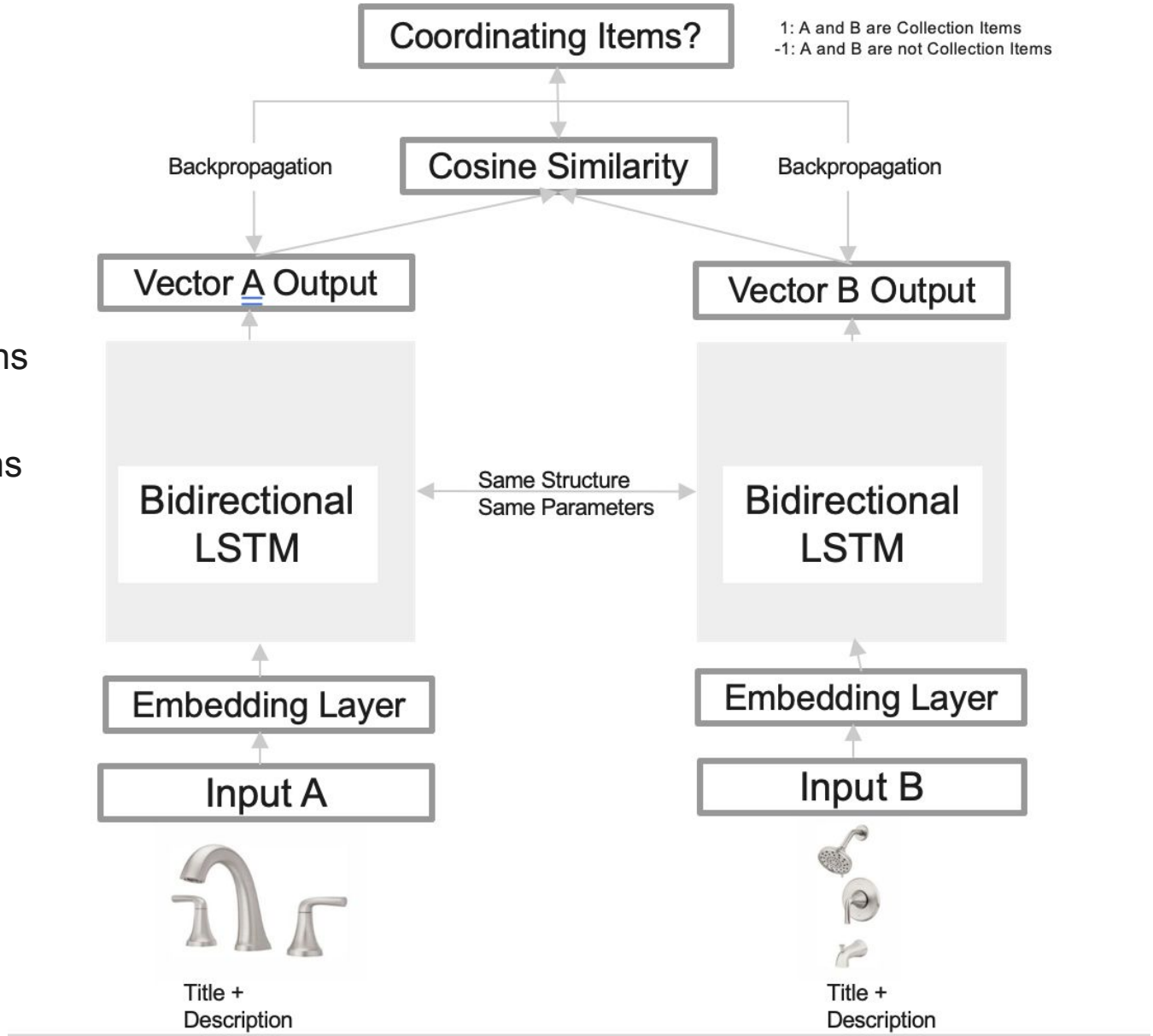
**Solutions**

- Sub-Word Tokens
  - Using either a sub-word tokens or character ngrams
  - This requires the sub-word units to carry some semantic meaning, similar idea to stemming or lemmatization
  - This does not work well for proper nouns and other terms with meaningless sub-word tokens
  - E.g. see DSSM paper (Microsoft 2013) - https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/cikm2013_DSSM_fullversion.pdf

- Hashing
  - Each OOV token is hashed into a bucket, the bucket is assigned an embedding and treated like a special token
  - Works with any type of token, however, Hash collisions can result in false positives
  - E.g. see 'Semantic Product Search' paper (Amazon 2019) - https://arxiv.org/pdf/1907.00937.pdf for more details

# Siamese Neural Networks

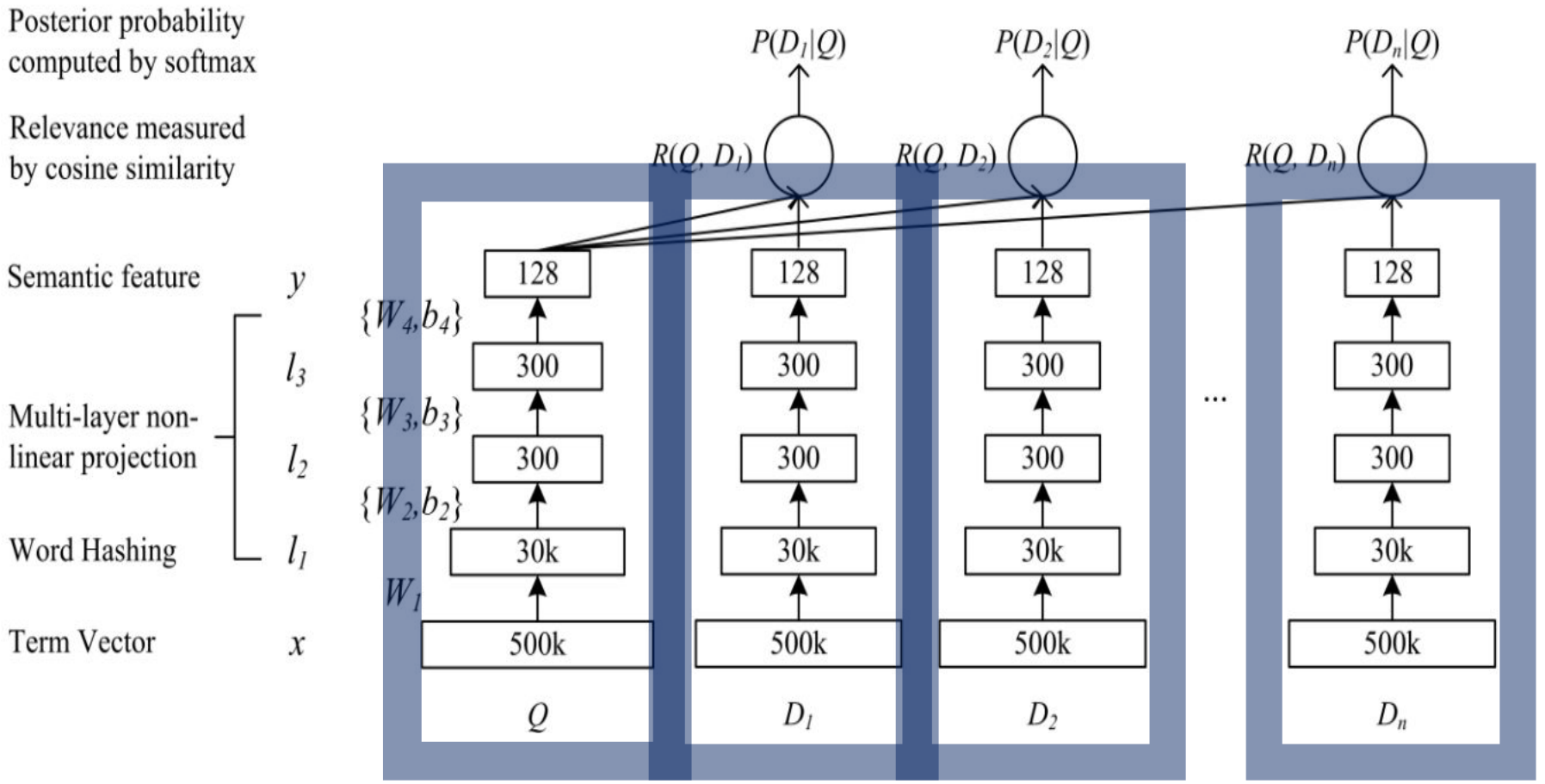
- Early neural network approaches for learning similarity functions
- Example shown on the right is used in Home Depot to learn substitutable items for alternative recommendations
- The model consists of a single encoder that takes tokens from a product's title and description and learns an embedding vector over the content
- The model is trained to maximize the cosine similarity between similar items (positive pairs), and minimize the cosine similarity between dissimilar items (randomly sampled 'negative' pairs)
- The model is called a 'Siamese' network as each encoder is identical (Siamese twin), and has the same learned parameters




RecSys '19 – 'Product collection recommendation in online retail' https://dl.acm.org/doi/10.1145/3298689.3347003 (HomeDepot)

# Deep Structured Semantic Model (DSSM) – Microsoft (2013)

- DSSM model from Microsoft
- An early Siamese network
- One of the first neural IR models
- Model is a feed forward neural network
- Original model uses character ngrams as the input layer ('word hashing')




DSSM paper - https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/cikm2013_DSSM_fullversion.pdf

# Two Tower Neural Networks

- Has a **separate** query and document encoder
- This allows the model to use different features for the product and the query
- The token embedding layer may or may not be shared between encoders
- See also JD.Com paper for an e-commerce example - https://arxiv.org/pdf/2006.02282.pdf

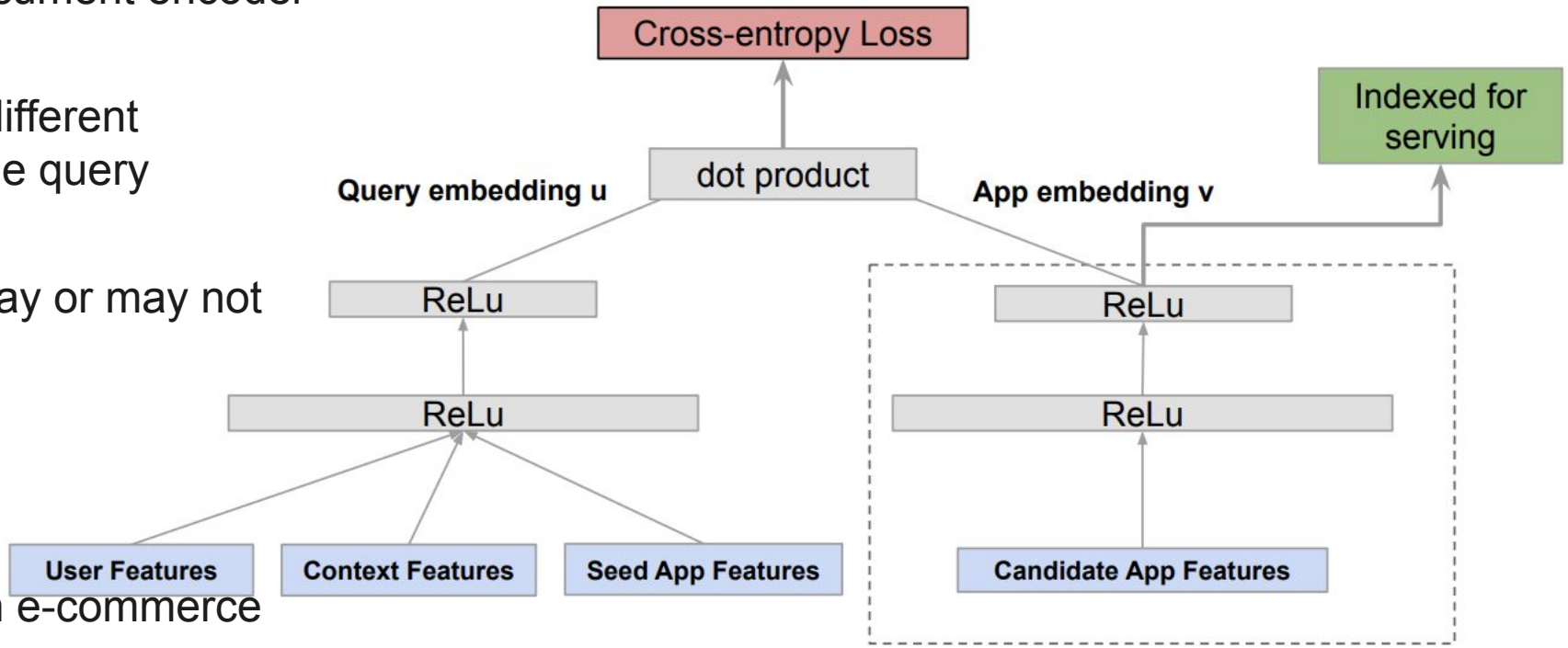
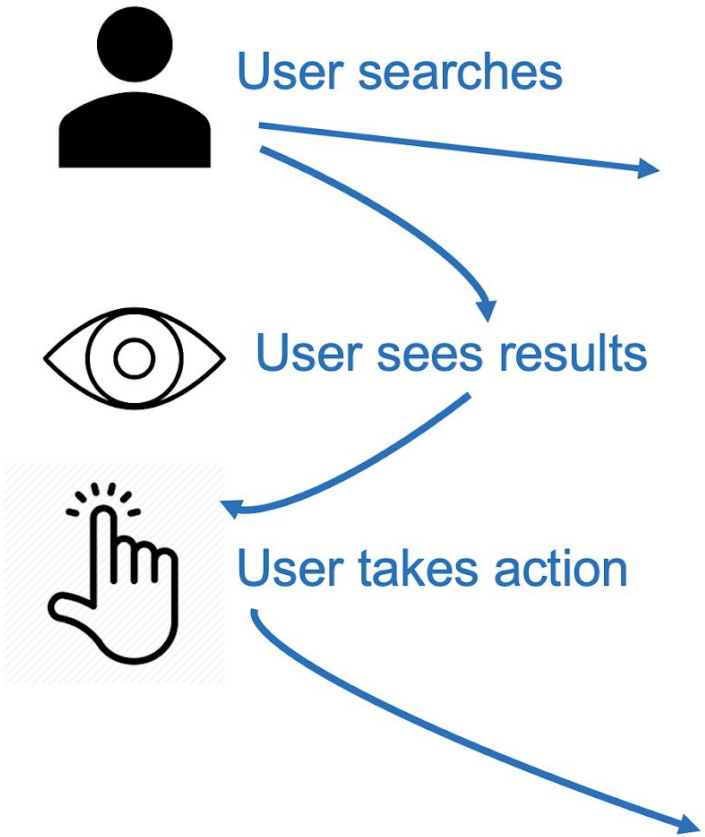



Figure 5: Two-tower model architecture for Google Play app recommendation.

Figure taken from 'Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations:' - https://dl.acm.org/doi/fullHtml/10.1145/3366424.3386195 (Google – Play App Recommendations System)

# Generating Training Data from the Click Stream




| User | Query |
|---|---|
| Simon | Hammer |
| Todd | Ryobi Drill |
| Jane | Freezer |

Relevancy labels are inferred using historical data of customer interactions

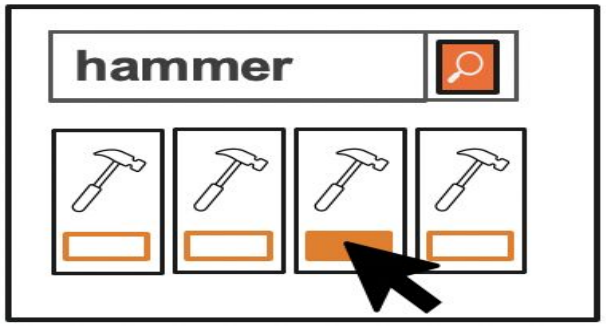
| User | Query | Action | Document | Inferred Label |
|---|---|---|---|---|
| Simon | Hammer | Purchase | doc22 | 2 |
| Simon | Hammer | Click | doc17 | 1 |
| Simon | Hammer | Skipped | doc11 | -1 |
| Todd | Ryobi Drill | Click | doc10 | 1 |
| Todd | Ryobi Drill | Purchase | doc24 | 2 |
| Todd | Ryobi Drill | Click | doc23 | 1 |
| Jane | Freezer | Click | doc19 | 1 |
| Jane | Freezer | Skipped | doc39 | -1 |

# Mining Training Data Signals

## Positive Signals?

### Purchases

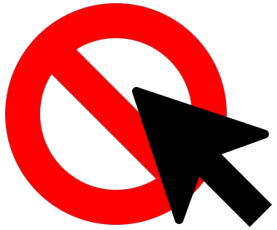
### Clicks




## Negative Signals?

### Random Docs

### Skipped Docs

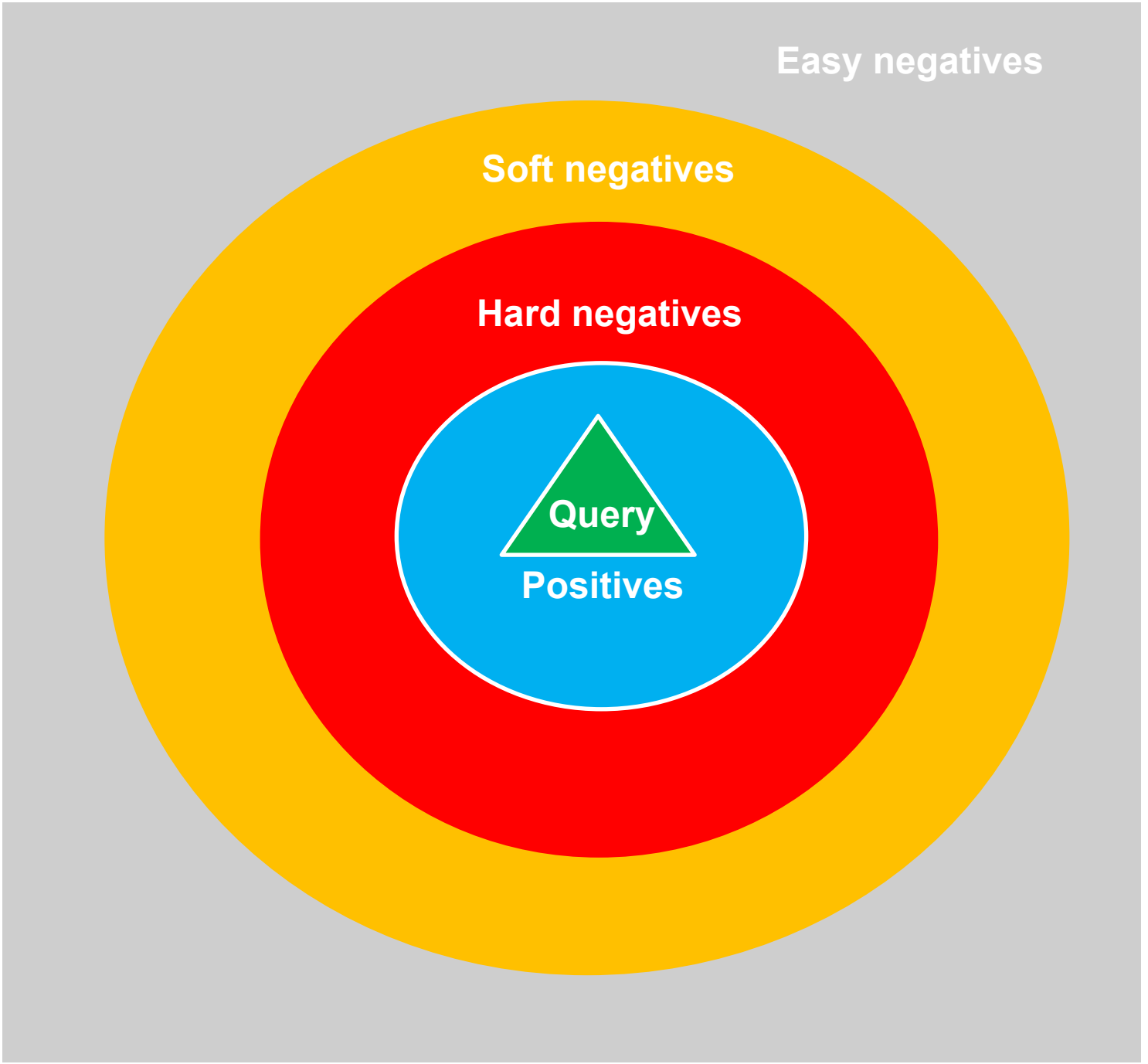


- The research argues that some random negatives are needed to differentiate the **hard negative** cases that are close to the query (e.g. skipped above) from the majority of documents that are not a match at all – **easy negatives**

- See also 'Embedding-based Retrieval in Facebook Search' - https://arxiv.org/abs/2006.11632

# Hard and Soft Negative Mining

- The neural model itself can be used to **mine** for hard negatives from the unclicked documents (similar to active learning)
- Two key approaches:
  - Local - mini-batch
  - Global hard negative mining
- It is important that hard negatives are chosen carefully:
  - Q = cordless circular saw
  - HN = circular saw without cord
  - HN = corded circular saw



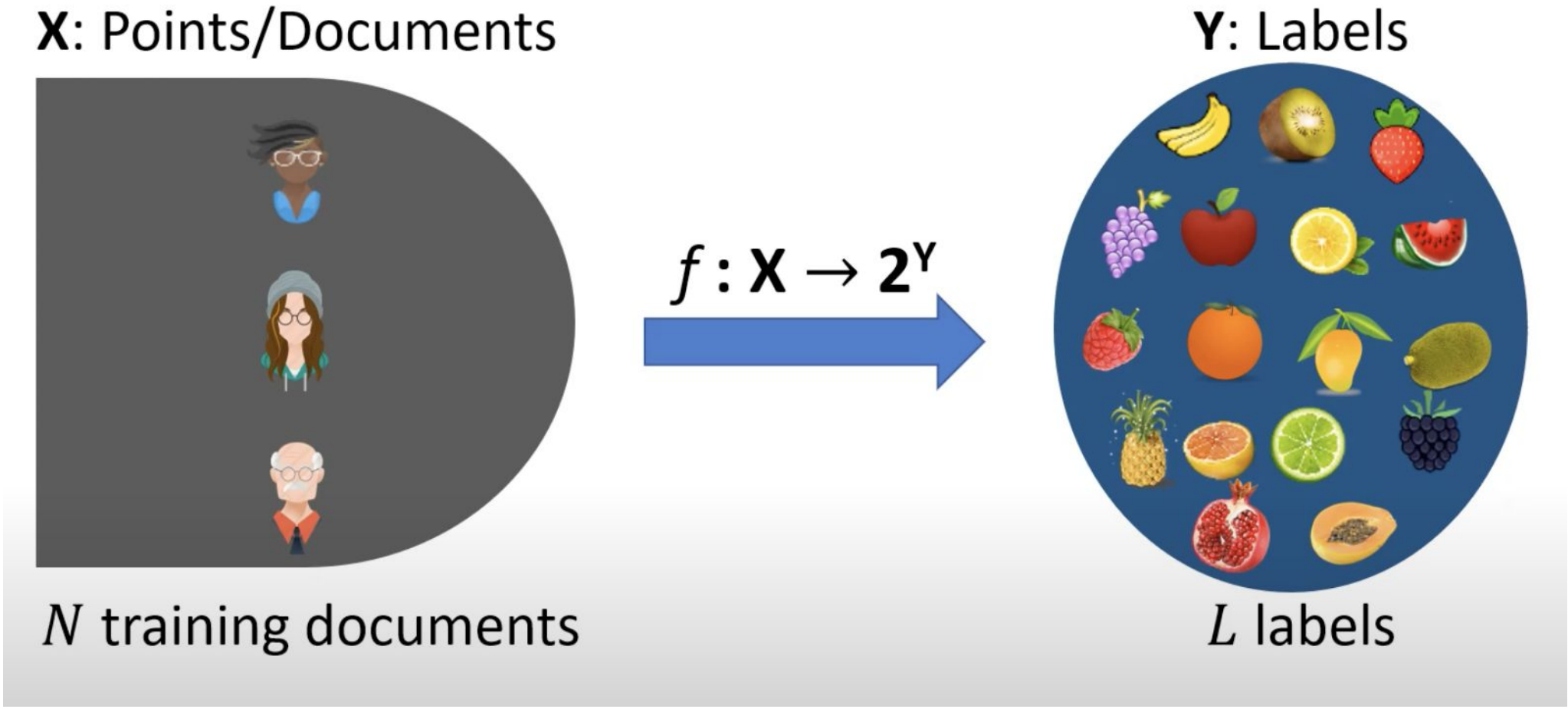
# Content Based Vs Behavioral Models

- Content Based Models
  - The token based neural IR models learn semantic representations of documents from their content for use in document retrieval
  - Any product can be matched, including those with little or no user behavior signals (‘cold start’ documents)

- Behavioral Models
  - A different formulation treats the problem as a classification problem - given a query, predict the products purchased
  - These models rely on user behavior data rather than content

- Note – both model types are typically trained on implicit signals derived from the click stream

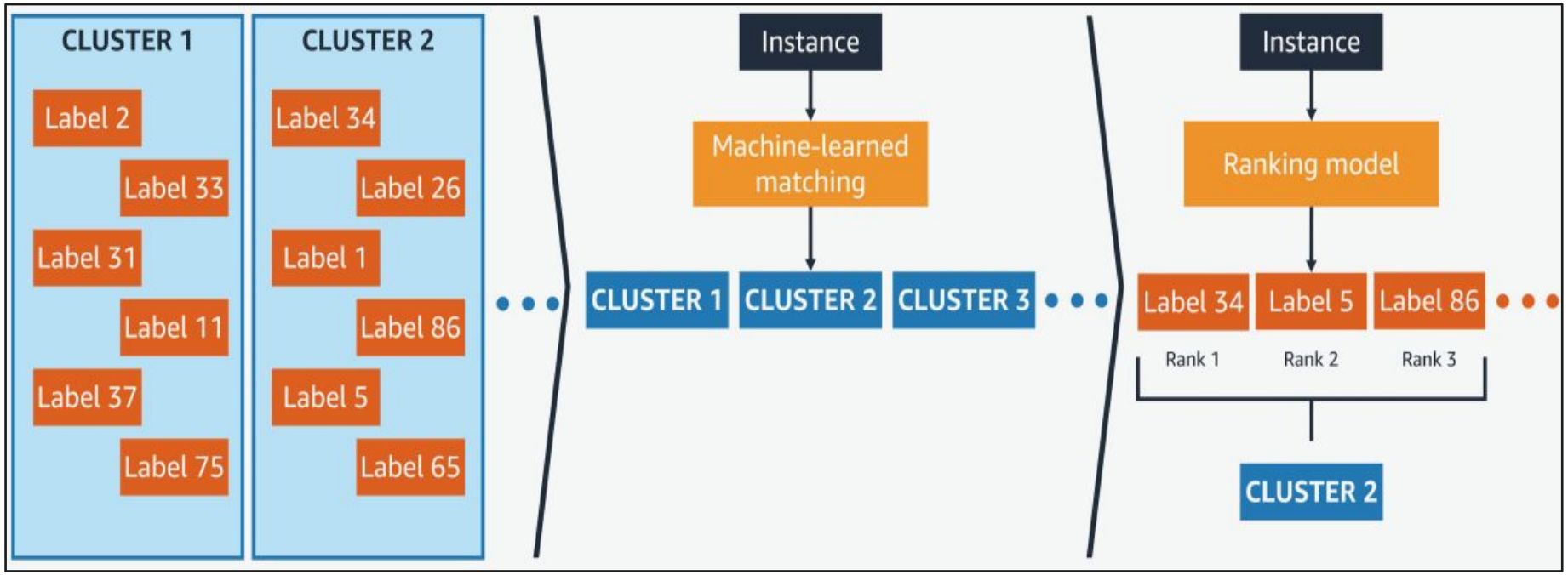
# Extreme Multi-Label Classification (XMC)




- **Extreme -** The model is predicting a very large label space (100k to Millions)
- **Multi Label Classification –** Multiple classes (products) predicted per input (query)
- Image taken from 'GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification' https://www.youtube.com/watch?v=4-EtvWlUk9Q

# PECOS Framework (Prediction in Correlated Output Spaces) – Amazon Research

## 3 Step PECOS Framework




- **1. Indexing –** PECOS groups labels (products) according to topic
- **2. Matching –** Matches the instance to the topic (shrinking the search space)
- **3. Ranking –** Re-ranks labels in the matched topic based on the input features

- KDD '21 Paper from Amazon - https://arxiv.org/pdf/2106.12657.pdf. Note: many other XMC models exist, e.g. DECAF, SLICE, GalaxC.etc
- Image taken from Amazon blog - https://www.amazon.science/blog/applying-pecos-to-product-retrieval-and-text-autocompletion

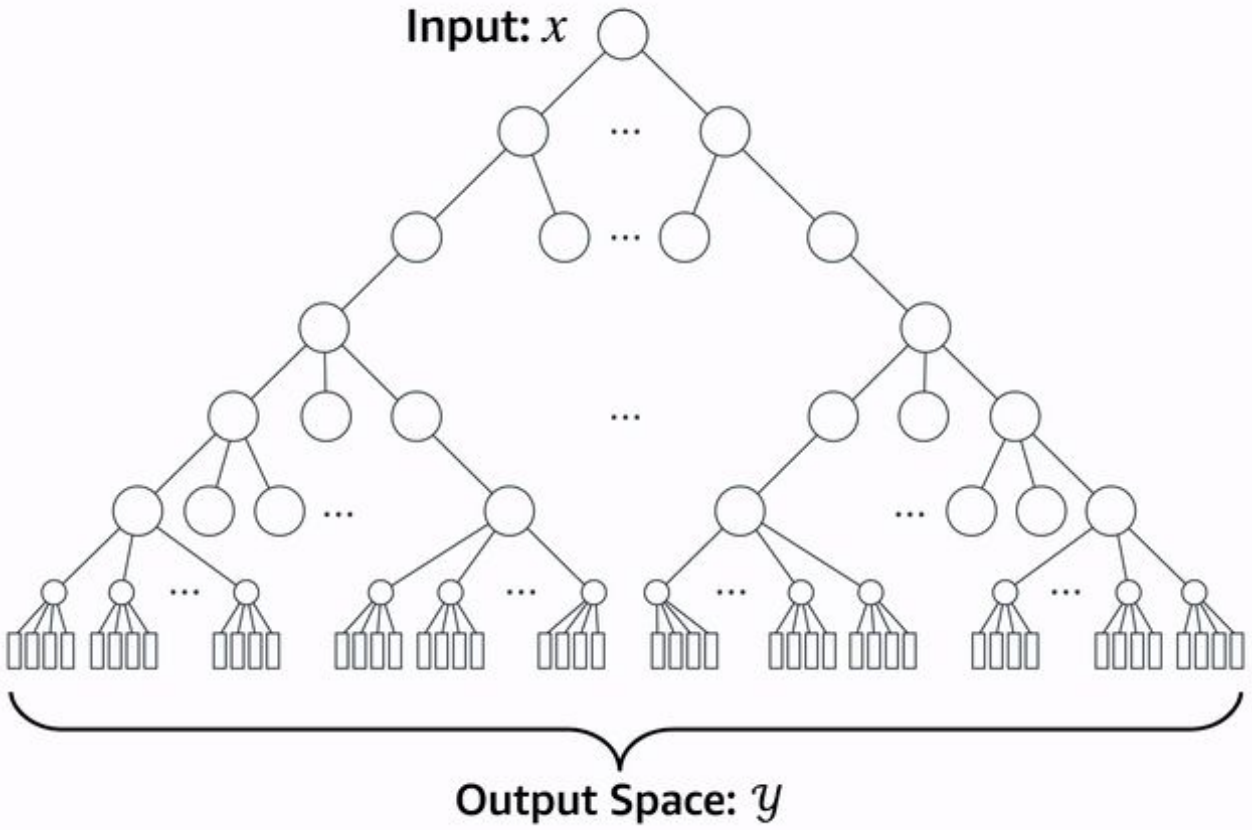
# Extreme Multi-Label Learning for Semantic Matching in Product Search

- Amazon's XR Linear Model (PECOS framework):

- **1. Indexing**
  - Products clustered based on query terms and document content using **hierarchical k-means clustering** to create a **Label Tree** (see image on right)

- **2. Matching**
  - A binary classifier is trained for each node in the tree (including leaf nodes)
  - Classifiers take query features as input and predict matching tree nodes

- **3. Ranking**
  - At inference time, a beam search is used to navigate the label tree, to determine the matching leaf nodes (products)
  - Matched products are then sorted by the classifier confidence scores (e.g. probabilities) for the matched labels




- Image taken from Amazon blog - https://www.amazon.science/blog/applying-pecos-to-product-retrieval-and-text-autocompletion
- KDD '21 Paper from Amazon - https://arxiv.org/pdf/2106.12657.pdf.

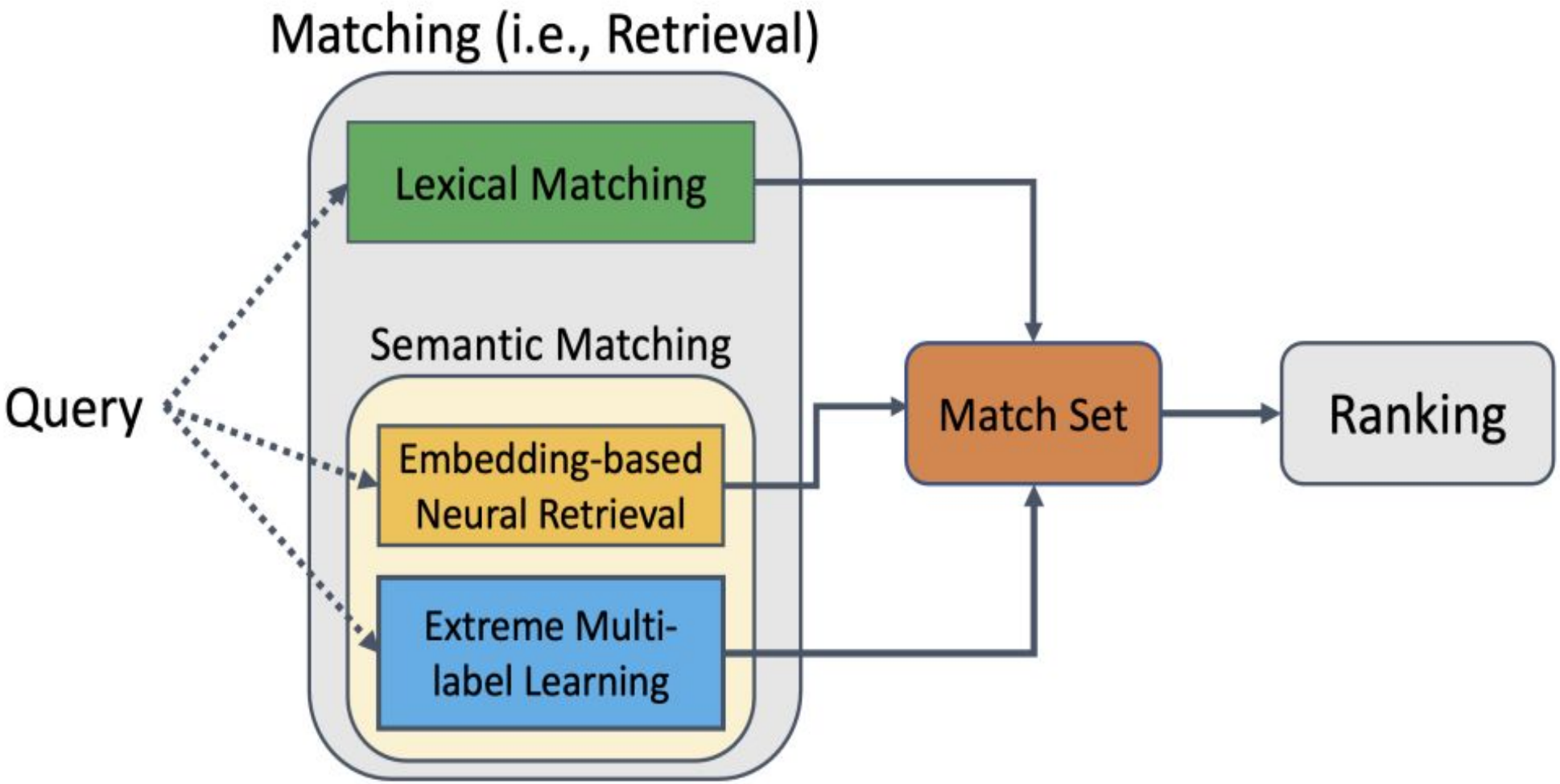
## Extreme Multi-Label Learning for Semantic Matching in Product Search

- The model learns an indexing structure using hierarchical k-means, similar to that used to solve some approximate KNN search problems
  - See my 2019 HayStack talk on Vector Search - https://haystackconf.com/2019/vectors
- Training time was **6 X faster** than their best neural model (CDSSM)
- **Recall@10** was **2 X** their best neural model
- Inference time - **1.25 ms**

## Other Extreme Multi Label Learning Methods

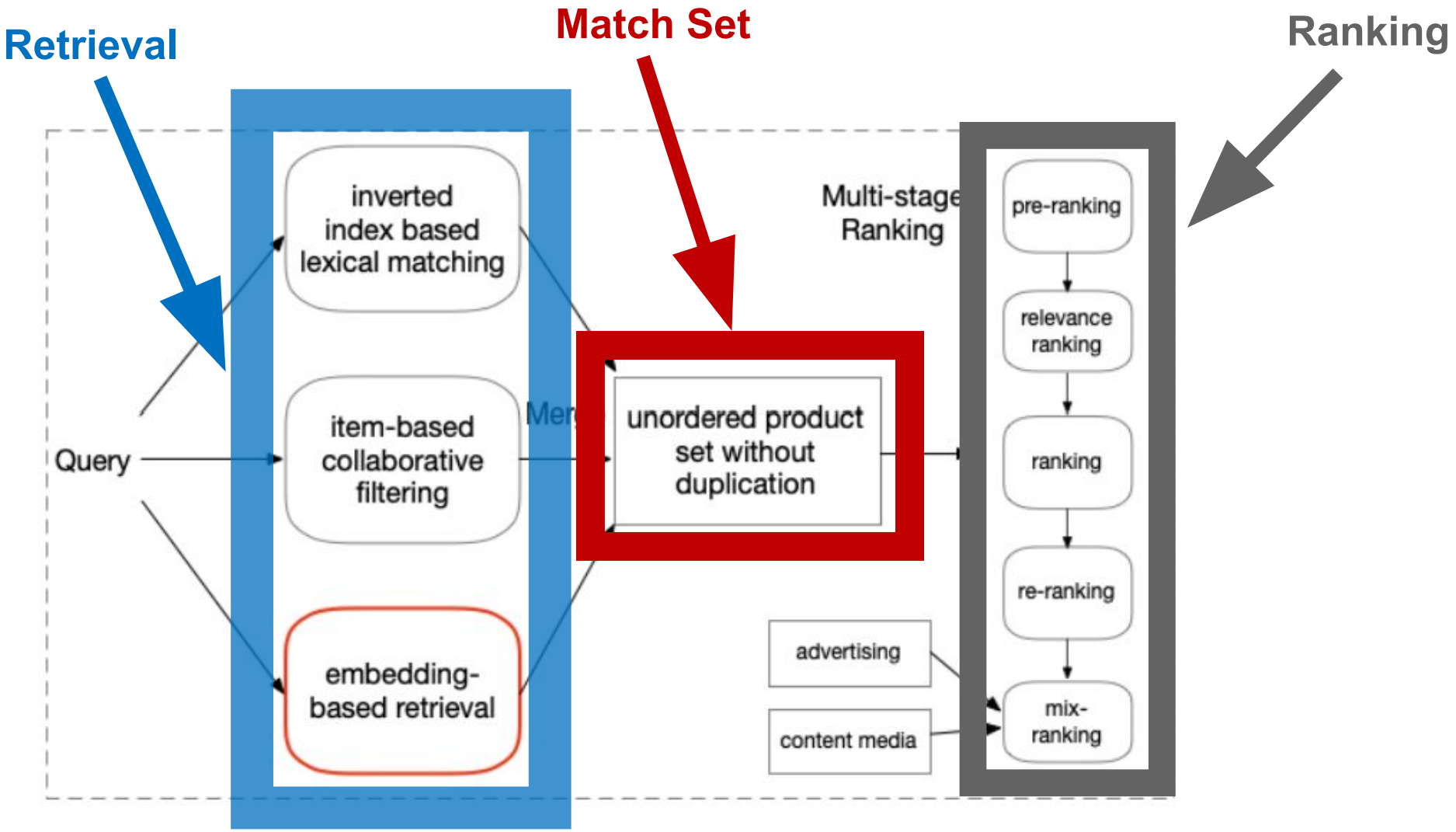
- The amazon paper discussed is the first application of XMC to search
- However there is a large body of work dedicated to this topic – see http://manikvarma.org/downloads/XC/XMLRepository.html
- Many other models exist such as SLICE and Parabel (Microsoft), DECAF, Bonsai, GalaxC, etc.
- Not all of these models are tree based, some use approximate KNN search algorithms (e.g. SLICE) to do the initial matching, followed by a classification layer

# Machine Learning Semantic Search Architecture – Amazon Example




From Amazon Paper: **"Extreme Multi-label Learning for Semantic Matching in Product Search"**
**-**https://arxiv.org/pdf/2106.12657.pdf

# Machine Learning Semantic Search Architecture – Alibaba Example




From Taobao (Alibaba) Paper: **"Embedding-based Product Retrieval in Taobao Search"** - https://arxiv.org/pdf/2106.09297.pdf

# The End

- Any Questions?