Search @ Nextdoor

What are our Neighbors Searching For?

Haystack 4/28/22 bojan@nextdoor.com Bojan Babic

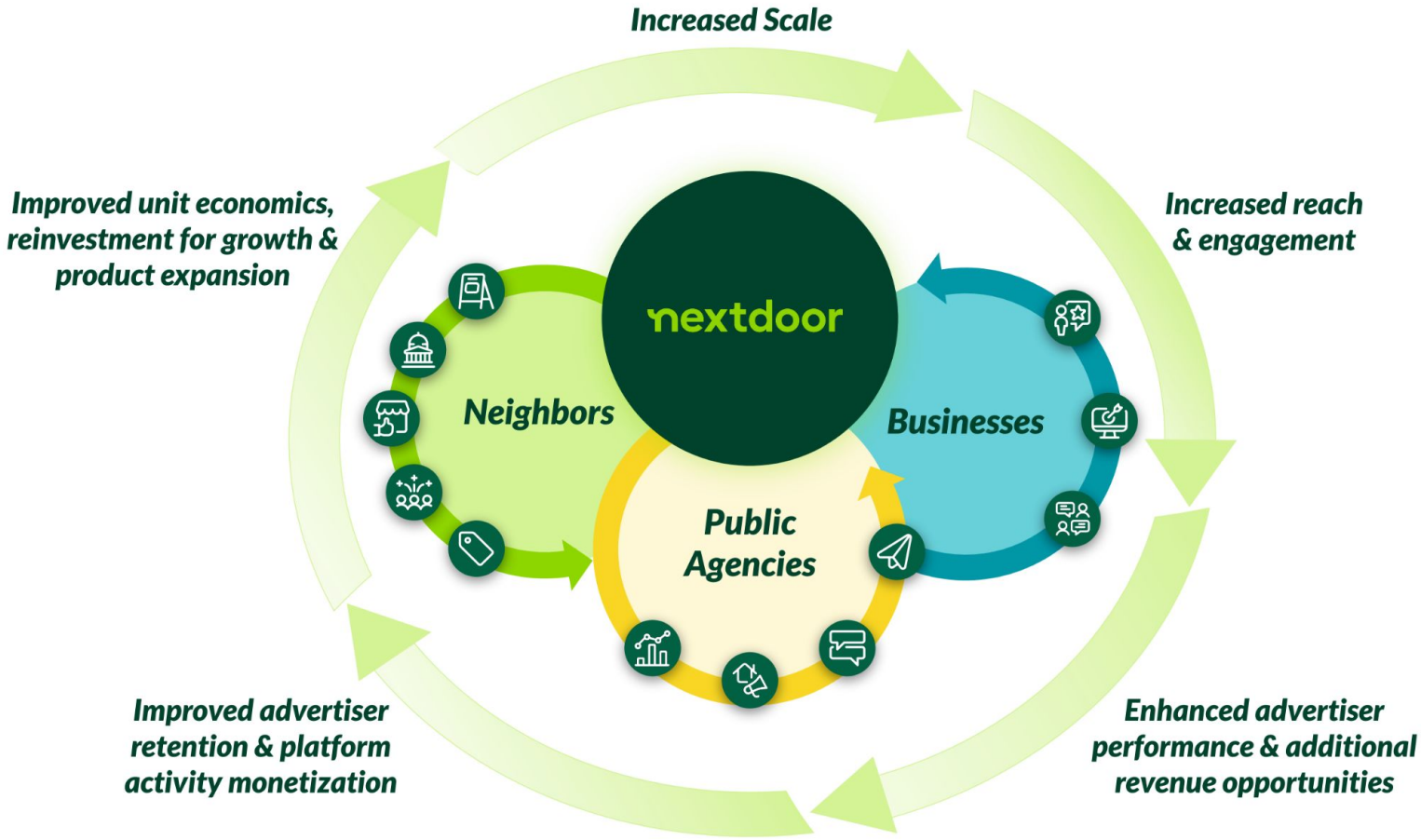
Bojan Babic
@bojanbabic
bojan@nextdoor.com

Relevance @ Groupon

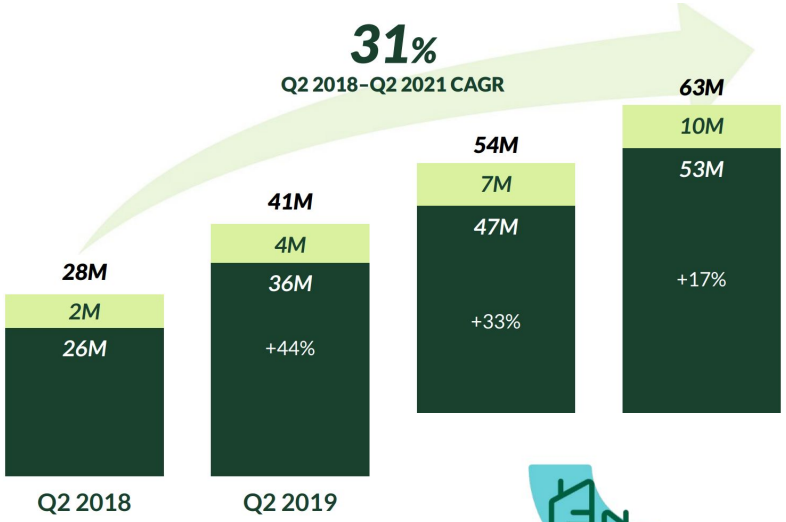
ML Platform @ Hipcamp

Search @ Nextdoor

What is Nextdoor?



Nextdoor Scale



~ 1 in3
U.S. Households

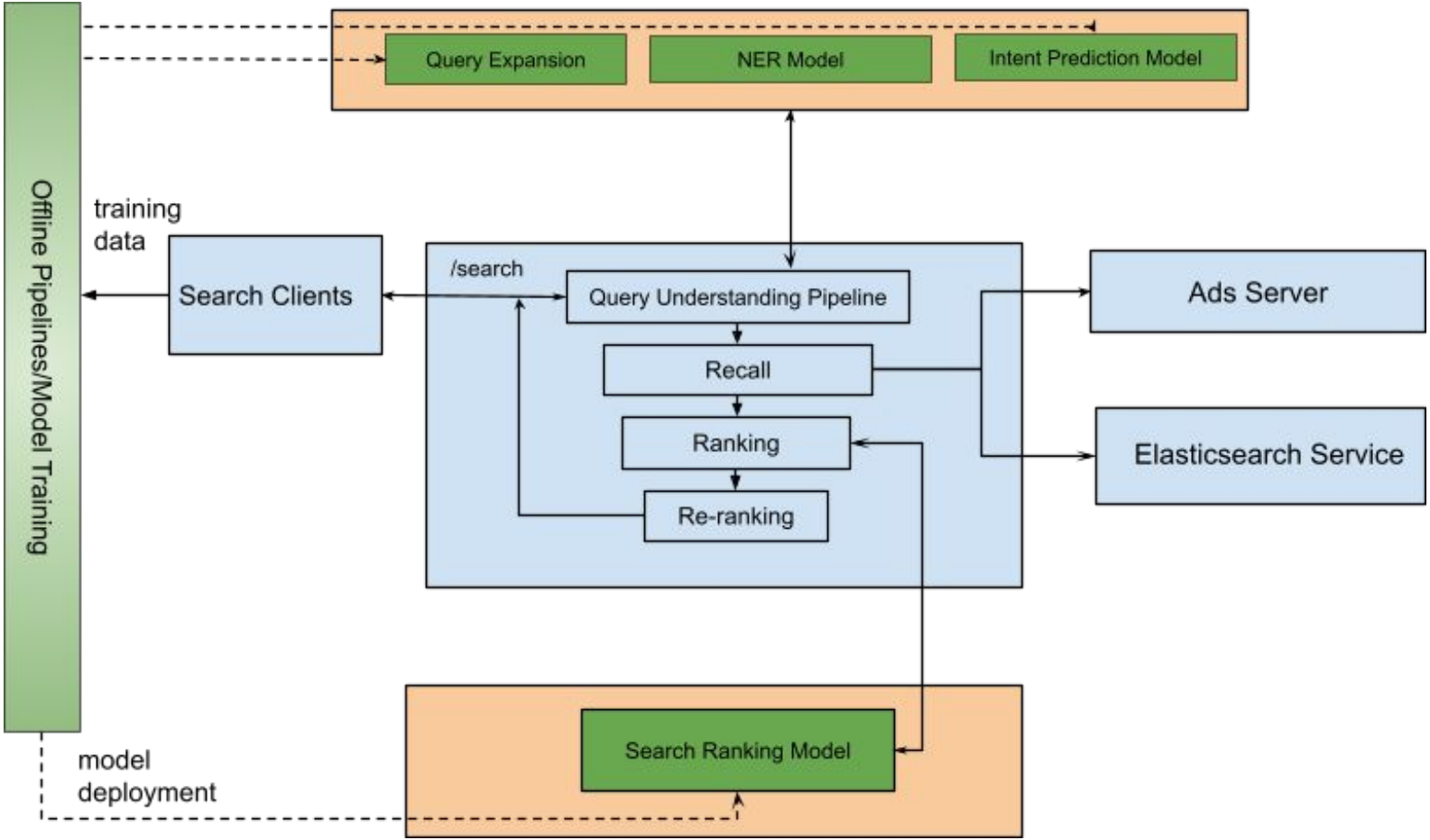
70M+
Verified Neighbors

35M+
Weekly Active Neighbors (WAUs)

Modern Search - Guiding Principles

- Semantic Understanding
  - Search needs to go beyond TFIDF and BM25. It is required that search engines understand the entities, have a robust taxonomy and have a rich set of attributes of each vertical that is supported
  - Support a corpus of documents that is ever evolving.
- Robustness to
  - rare inputs
  - corpus invariance
  - variable input lengths
  - errors in input
- Sensitivity to context
  - leverage explicit and implicit information
    - “concerts during memorial day”
      - upcoming events
      - consider location and date
- Efficiency
  - Multi-tier telescoping architecture
    - retrieval/recall
    - ranking
    - co-ranking
- Monitoring
  - NDCG, MAP, MRR

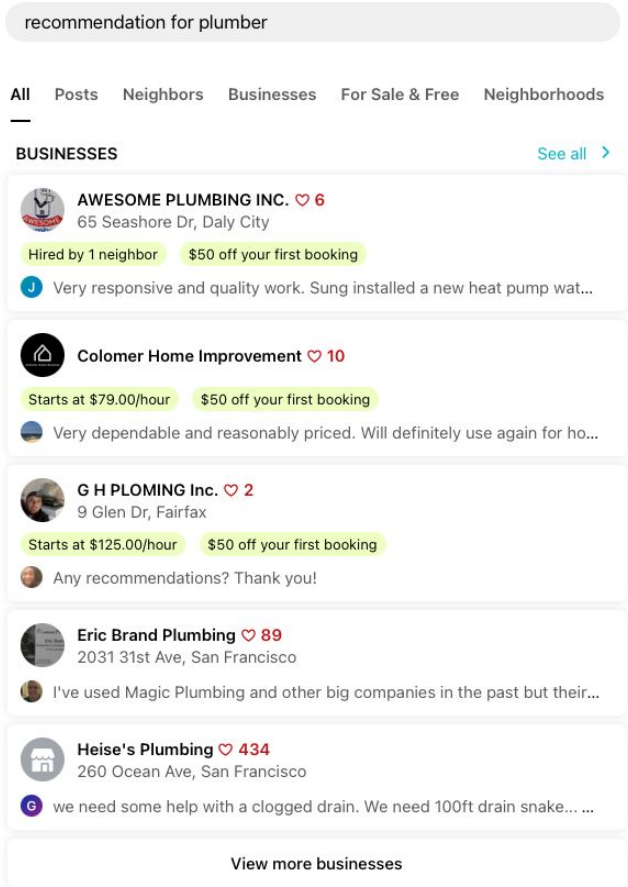
Nextdoor High-level Architecture



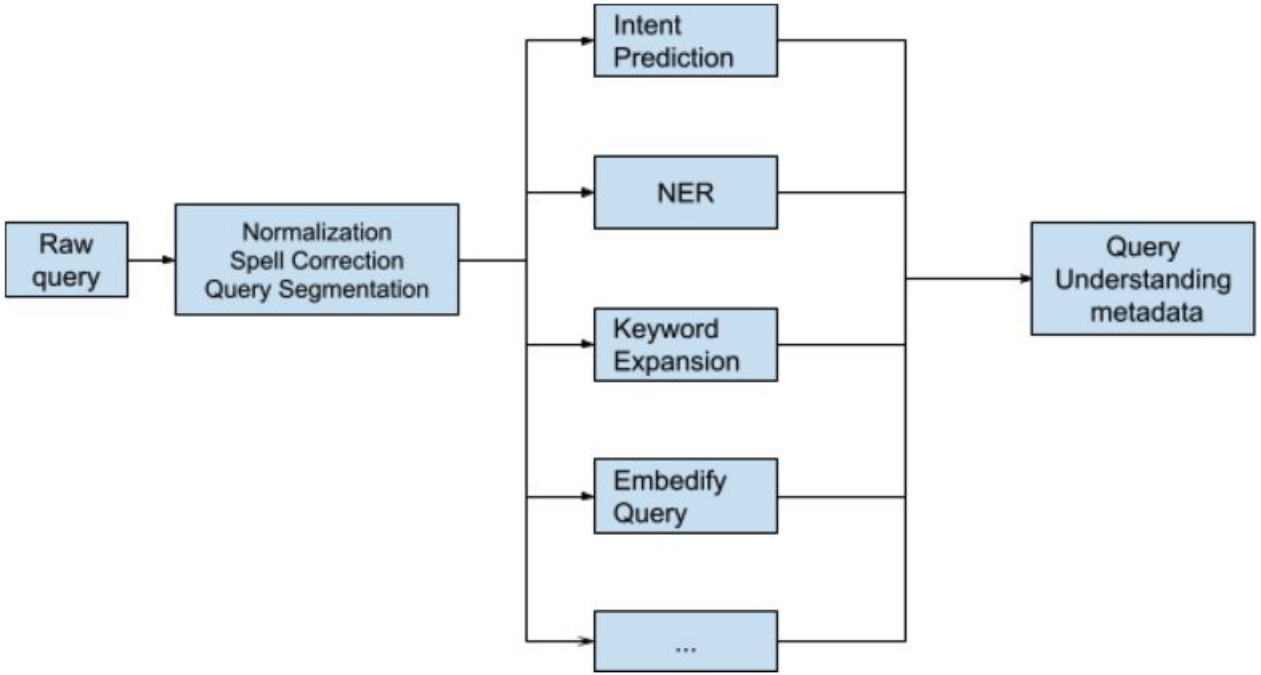
Query Understanding

Query Understanding - Overview

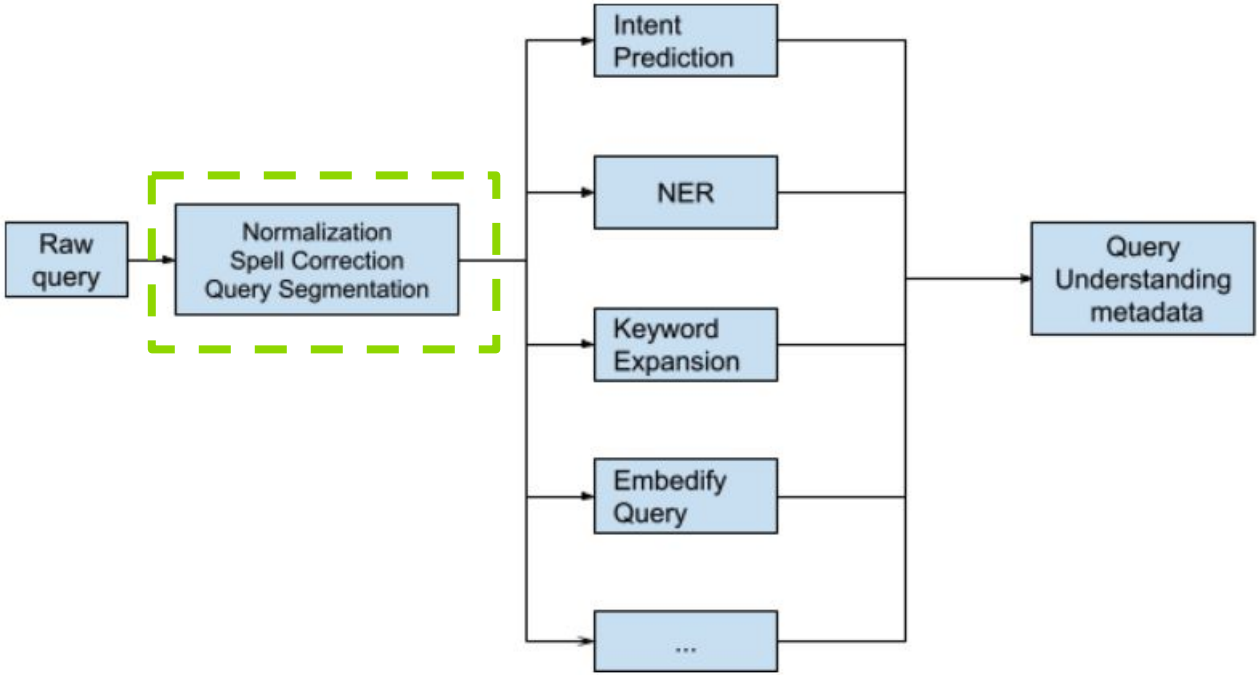
- Query Understanding is the stage in the search flow where given the input query, we extract high level features on the customer intent.
- Some documents in the search result page have ecommerce elements (i.e. classifieds, local deals), some fall under classical information retrieval systems (post and business search), while other have the elements of the social graph search (neighbors).
- Streamlined our search products so our customers have more clarity on what to expect after clicking on the search result page.
- There are many dimensions that involve interpreting user intent and subsequently returning the content matching that intent.



Query Understanding - Stages



Query Understanding - Stages

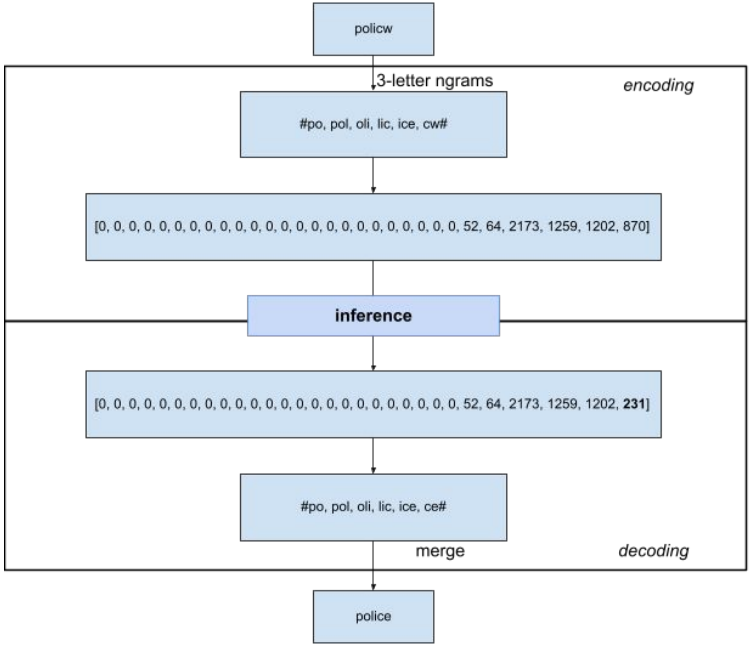


Query Understanding - Spell Correction

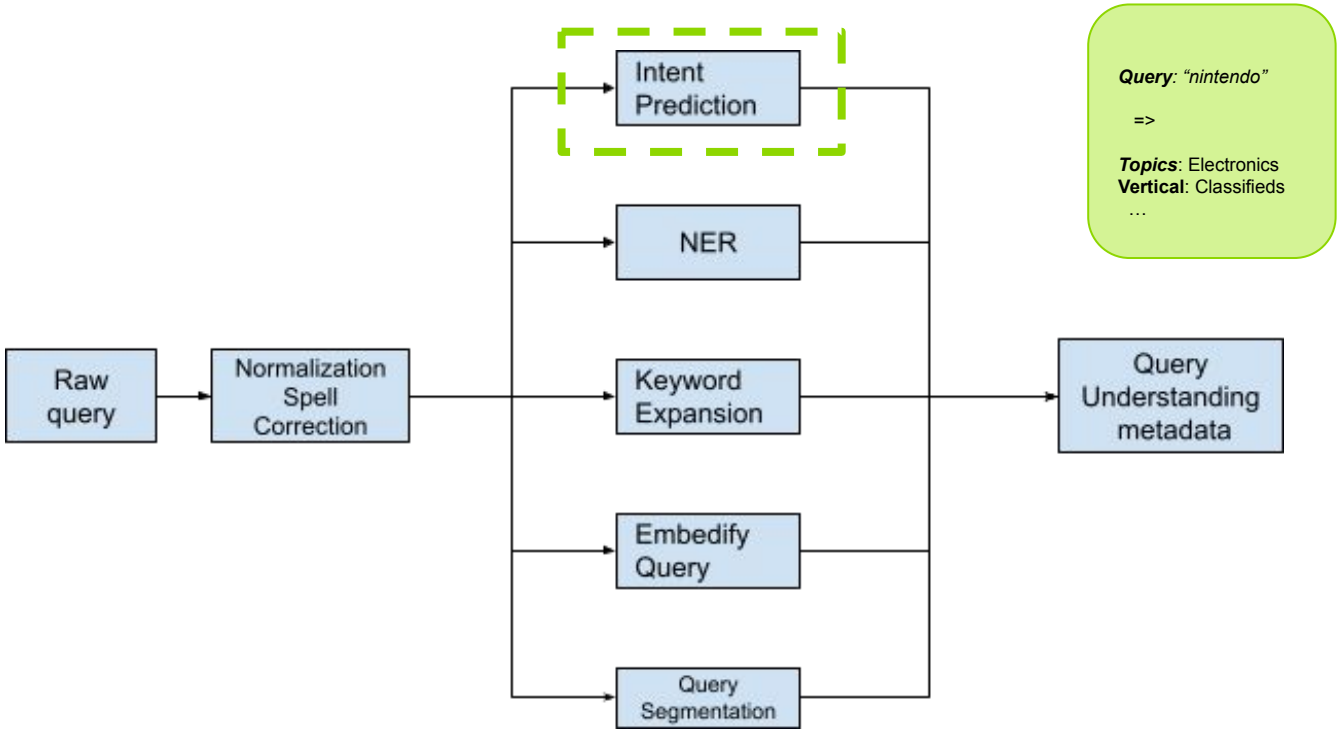
Previous work

- At Nextdoor there have been many attempts to solve spell correction. These approaches range from simple word edit distance and statistical approach or using HMM.

After lots of experiments, we decided to go with the Deep Learning Approach utilizing sequence to sequence architecture with Attention

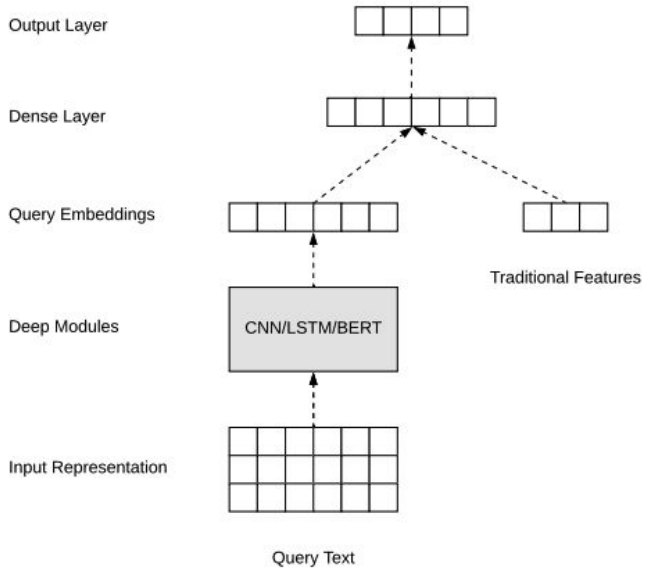


Query Understanding - Intent Prediction

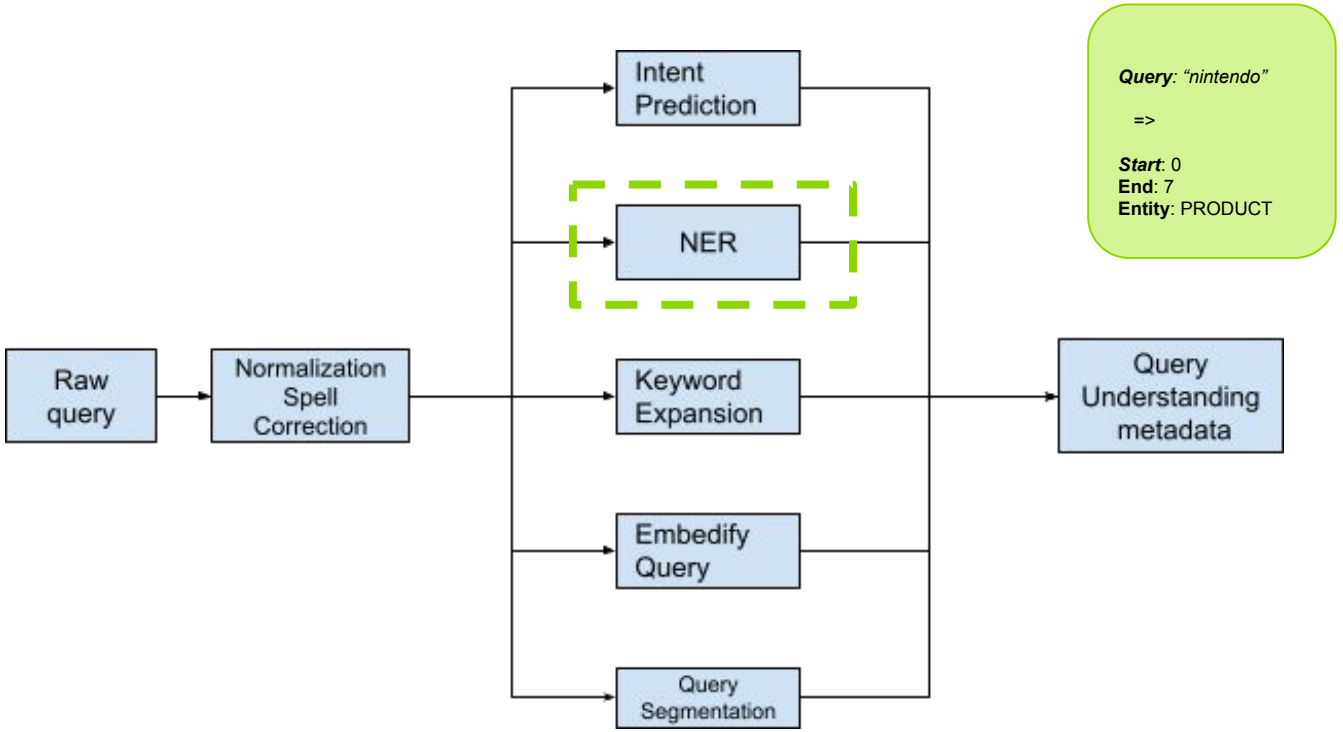


Query Understanding - Intent Prediction

- Basic idea was to rely on the bi-partite graph between queries and documents the customer clicked on.
- For example, if someone searches for *'recommendation for plumber"* followed by the click on the search result from the Plumber topic, we would consider these as the positive sample for associating the query with the topic at hand.
- Problem:
  - Lots of bad labels
  - Incomplete taxonomy / ever changing taxonomy
- Labeling task at scale
  - prodi.gy
  - Appen.com
- Applicable in many areas
  - Autocomplete
  - Reranking of the verticals that we support



Query Understanding - Intent Prediction

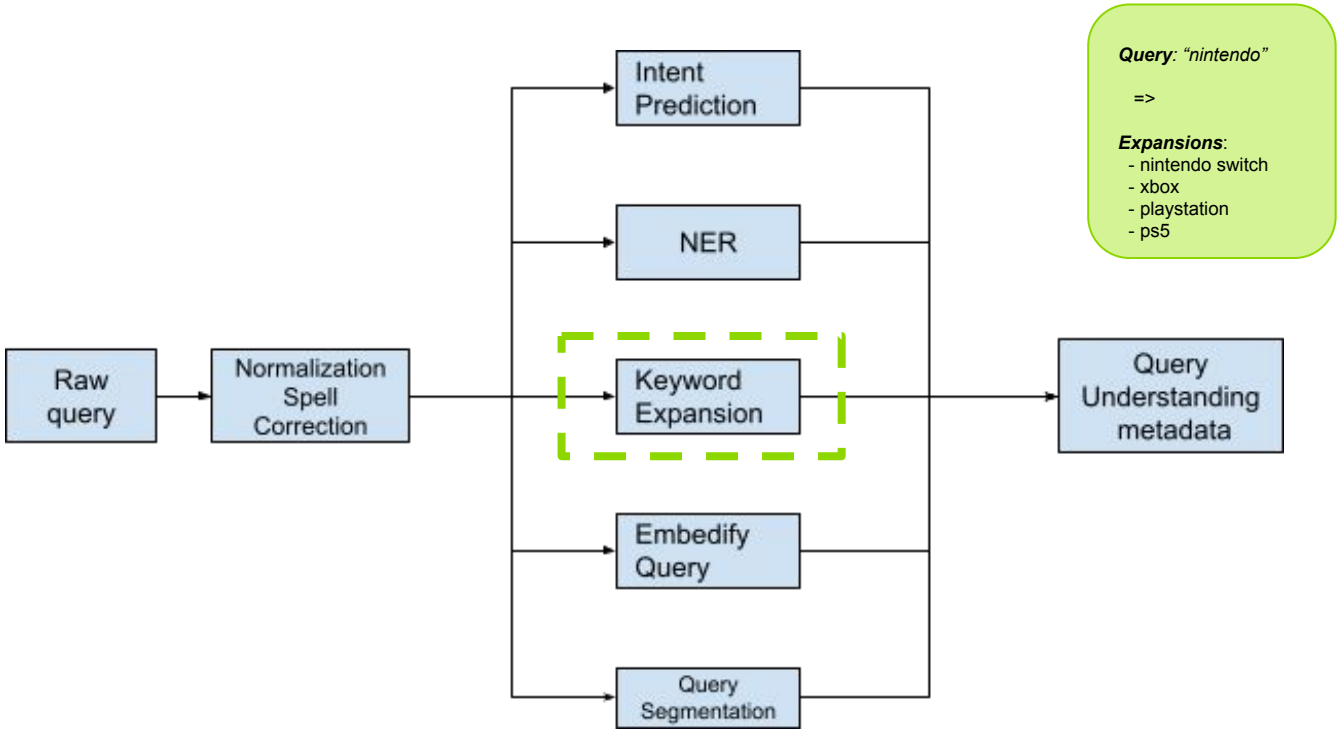


Query Understanding - Natural Language Recognition

- Helps identifying the different types of entities, such as people, places, or businesses, that can be present in queries
- Available open-source solutions for NER did not work for us
    - The primary reason is that most existing NER models were trained on sentence-level content, and did not perform well when we evaluated them on search queries, which are much shorter.
    - Another reason is that we wanted to introduce some Nextdoor-specific entity types, such as SERVICE (e.g. “handyman” or “plumber”)
- Transformer-based architecture with a token classification added on top of it.

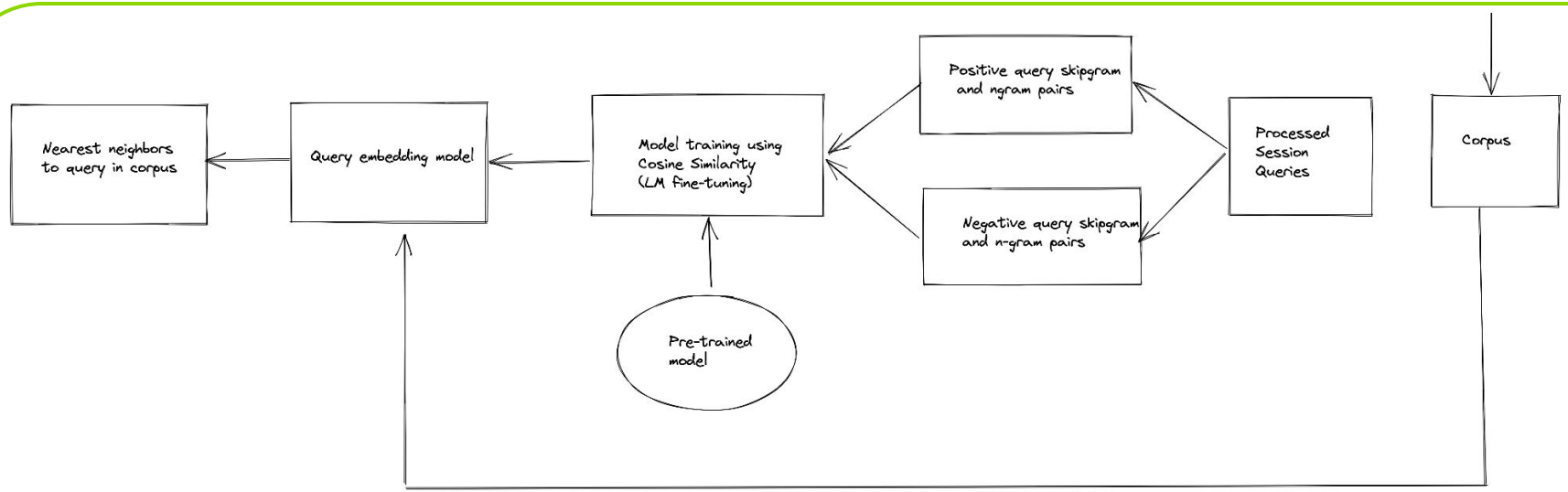
Base Model	PRODUCT Recall	PERSON Recall	LOC Recall	Harmonic Mean
roberta-base	0.842281	0.876099	0.8337	0.850304
distilbert-base-cased	0.856841	0.881495	0.8238	0.853388
distilbert-base-uncased	0.844893	0.921463	0.8242	0.861546

Query Understanding - Keyword Expansion



Query Understanding - Keyword Expansion

- **Token pair generation**
  - **Tokens** created from query n-grams / across-query skipgrams
  - Processed tokens should exist in the corpus
  - Limited to 10M total pos & neg token pairs
    - Ex: pos: (*"att"*, *"at&t wifi"*) neg: (*"apple mac"*, *"pool repair"*)
- **Model**
  - MiniLM L6 Transformer
  - Dense(384, 100) head to reduce dimensionality
- **Objective**
  - Contrastive learning w/ cosine similarity loss
- **Output**
  - Top K nearest neighbors to a query



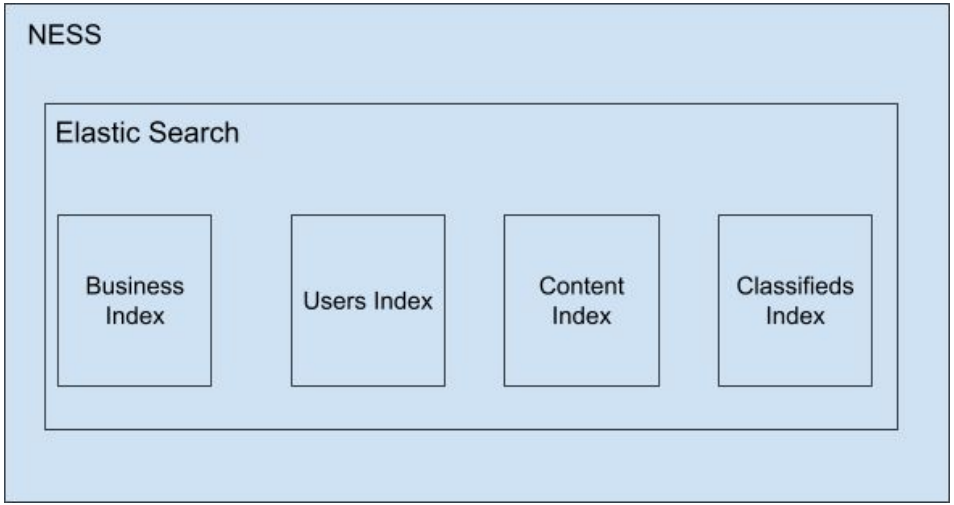
Query Understanding - Metadata

- Before we construct ES query, Query Understanding stage provide us the following Meta data:
  - Location
  - Predicted Vertical
  - Predicted Topic for Vertical
  - Expended Keywords
  - Embedded Query
  - Connections
  - User context

Recall

Recall - Overview

- TFIDF retrieval
- Multiple Verticals
  - User
  - Business
  - Neighborhoods
  - Posts
  - Classifieds/For Sale and Free
- Expansions from the embeddings space
  - Xbox -> nintendo switch, ps5
  - Sofa -> couch, ottoman



Ranking

Ranking - Approach

- Learning to Rank
  - Aim of the LTR is to find optimal scoring function F such that loss over the objective function is minimal.

$$\hat{R}(F) = \frac{1}{N} \sum_{i=1}^{N} L(F(X_i), Y_i)$$

  - Learn to discriminate space of positive and negative labels using feedback from out customers
- Feature engineering
  - Traditional and deep learning features

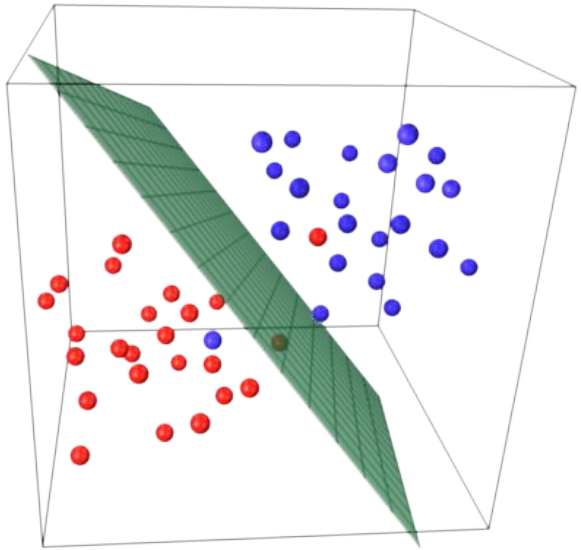


image taken from https://appliedmachinelearning.blog/2017/03/09/understanding-support-vector-machines-a-primer/

Ranking - LambdaRank

- Bridging the gap between evaluation metrics and loss functions has been studied actively in the past.
- LambdaRank or its tree-based variant LambdaMART has been one of the most effective algorithms to incorporate ranking metrics in the learning procedure
- The basic idea is to dynamically adjust the loss during the training based on ranking metrics

$$l(\mathbf{y}, \mathbf{s}) = \sum_{y_i > y_j} \Delta\text{NDCG}(i, j) \log_2(1 + e^{-\sigma(s_i - s_j)})$$

- We use LGBMRanker implementation from LightGBM
  - We rank 200 documents with latencies ~30ms

Ranking Features - Embeddified Query

- Three letter ngrams as input
  - cat -> #cat# -> #-c-a, c-a-t, a-t-#
  - only 50K vocab size, no OOV issue, almost no collision
- Capture subword semantics (prefix & suffix)
- Tried performance of model with transfer learning from fasttext and query2vec
- Word with small typos have similar representation

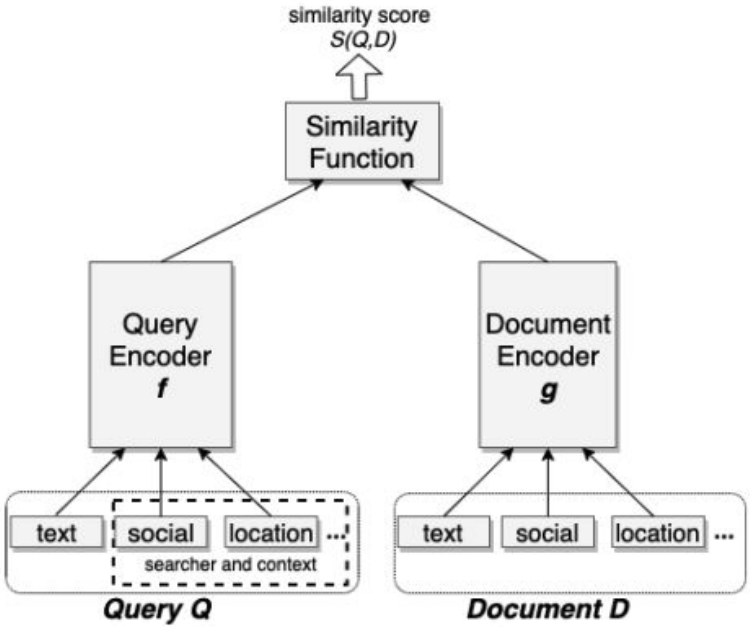
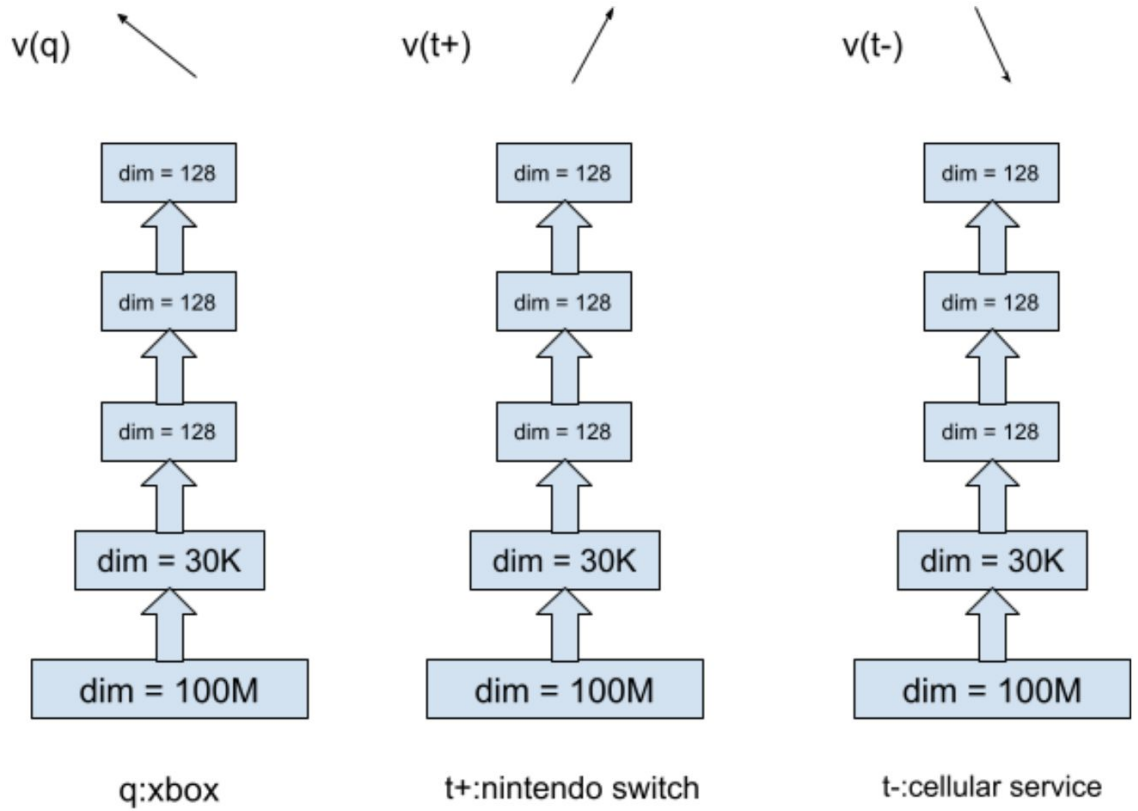


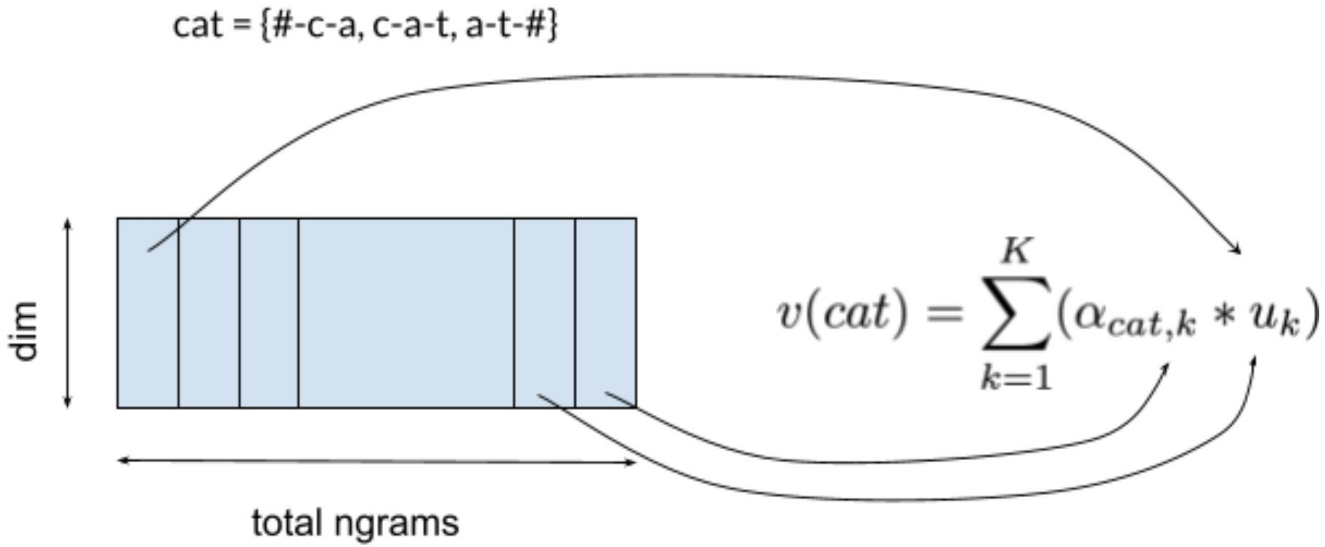
Figure 2: Unified Embedding Model Architecture

Ranking - Embeddified Query (Training - cnt)



Ranking - Embeddified Query (Inference)

- Query embeddings real time computation



- Document embedding lookup

Wrap up

Also, we are hiring about.nextdoor.com/careers

engblog.nextdoor.com

Please reach out to bojan@nextdoor.com

References

- Introduction to Neural Information Retrieval - https://www.microsoft.com/en-us/research/uploads/prod/2017/06/fntir2018-neuralir-mitra.pdf
- DSSM - https://www.microsoft.com/en-us/research/project/dssm/
- DistilBERT, a distilled version of BERT, smaller, faster, cheaper and lighter - https://arxiv.org/pdf/1910.01108.pdf
- Deep Search Query Intent Understanding - https://arxiv.org/pdf/2008.06759.pdf
- Fine-tuning of pretrained language models - https://ruder.io/recent-advances-lm-fine-tuning/
- Embeddings based retrieval - https://arxiv.org/pdf/2006.11632.pdf
- LambdaRank - https://storage.googleapis.com/pub-tools-public-publication-data/pdf/1e34e05e5e4bf2d12f41eb9ff29ac3da9fdb4de3.pdf
- AI Powered Searhc - https://www.manning.com/books/ai-powered-search