



Deep Learning for Search and Recommender Systems in Practice

Zhoutong Fu, Huiji Gao, Weiwei Guo,
Sandeep Jha, Jun Jia*, Xiaowei Liu, Jun Shi,
Sida Wang, Mingzhou Zhou, Bo Long

Aug 26, 2020



Agenda

- Introduction
- Query & User Understanding
- Candidate Retrieval
- Learning to Rank



Deep Learning for Search and Recommender Systems in Practice

Introduction



Bo Long



Jun Jia

Introduction

1 Search and
Recommender Systems

2 Applications at
LinkedIn

3 Tutorial Overview

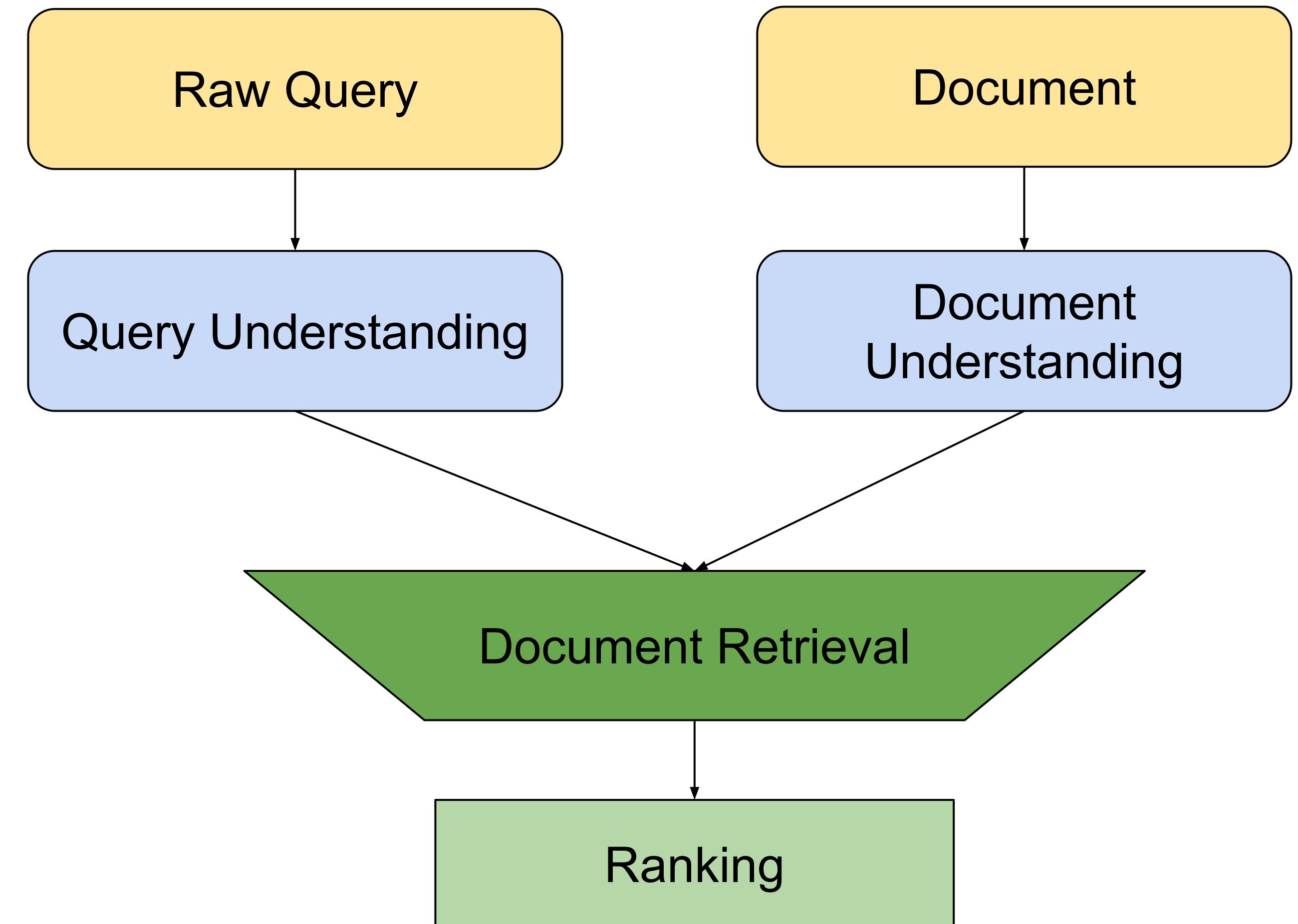
Introduction - Search and Recommender Systems



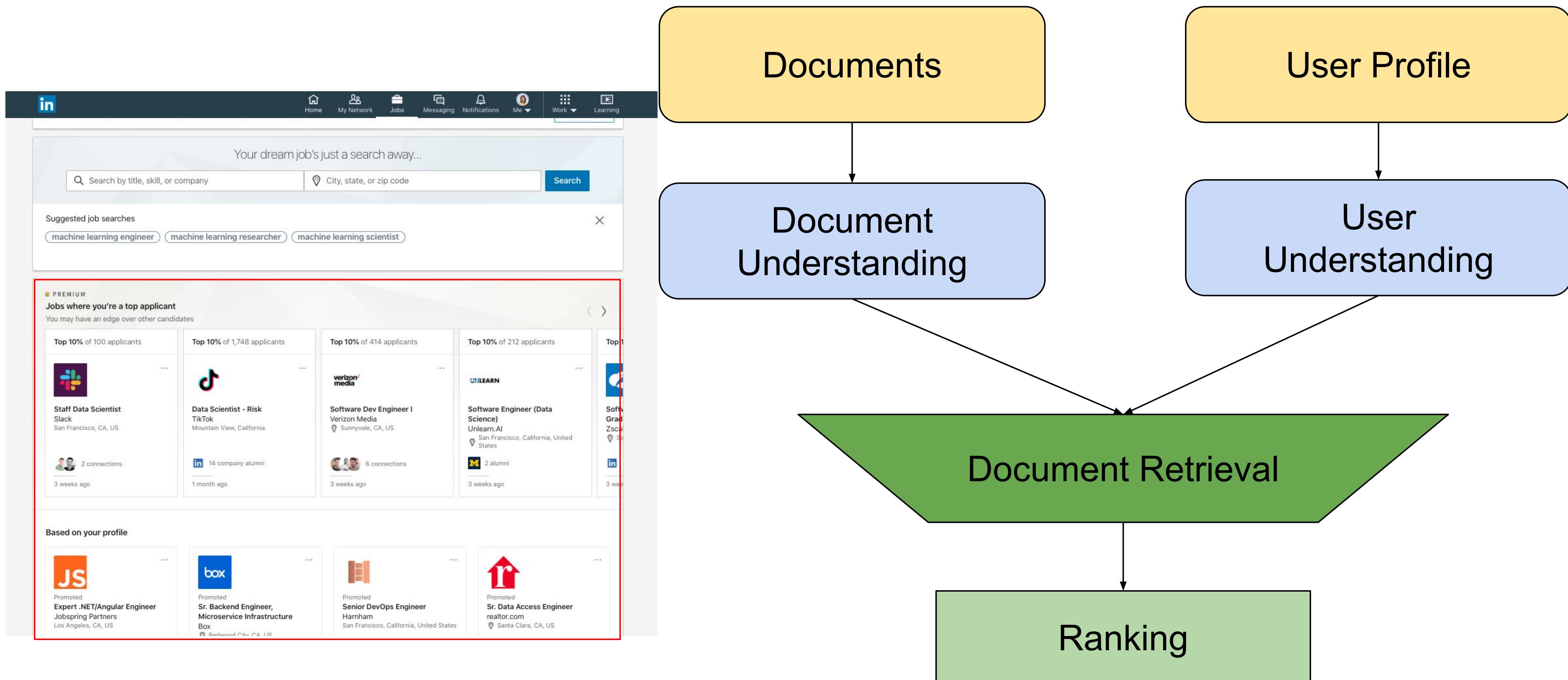
**Search and
Recommendation
are Everywhere**

Search Systems

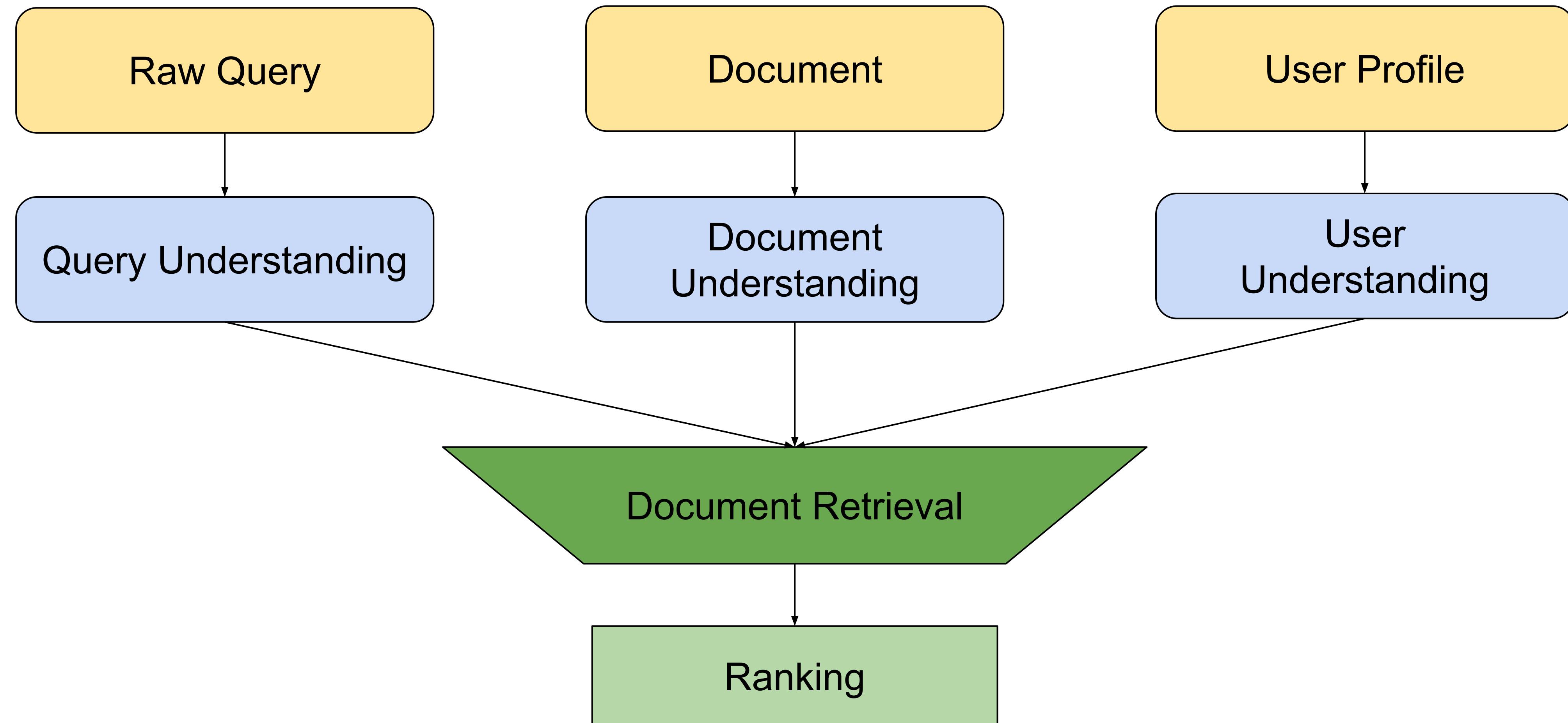
The screenshot shows the LinkedIn search interface. The search bar at the top contains 'machine learning engineer' and 'United States'. Below the search bar are filters for 'Jobs', 'Date Posted', 'LinkedIn Features', 'Company', 'Benefits', and 'All filters'. The results are sorted by relevance, showing 1,454 results. The first result is a promoted job post from Twitter for a Sr. Machine Learning Engineer. The second result is a job post from Cloudera for a Machine Learning Field Engineer. The third result is a job post from Apple for a Machine Learning Engineer. The fourth result is a job post from Amazon Web Services (AWS) for a Senior Machine Learning Device Software Engineer. The fifth result is a job post from PayPal for a Machine Learning Engineer.



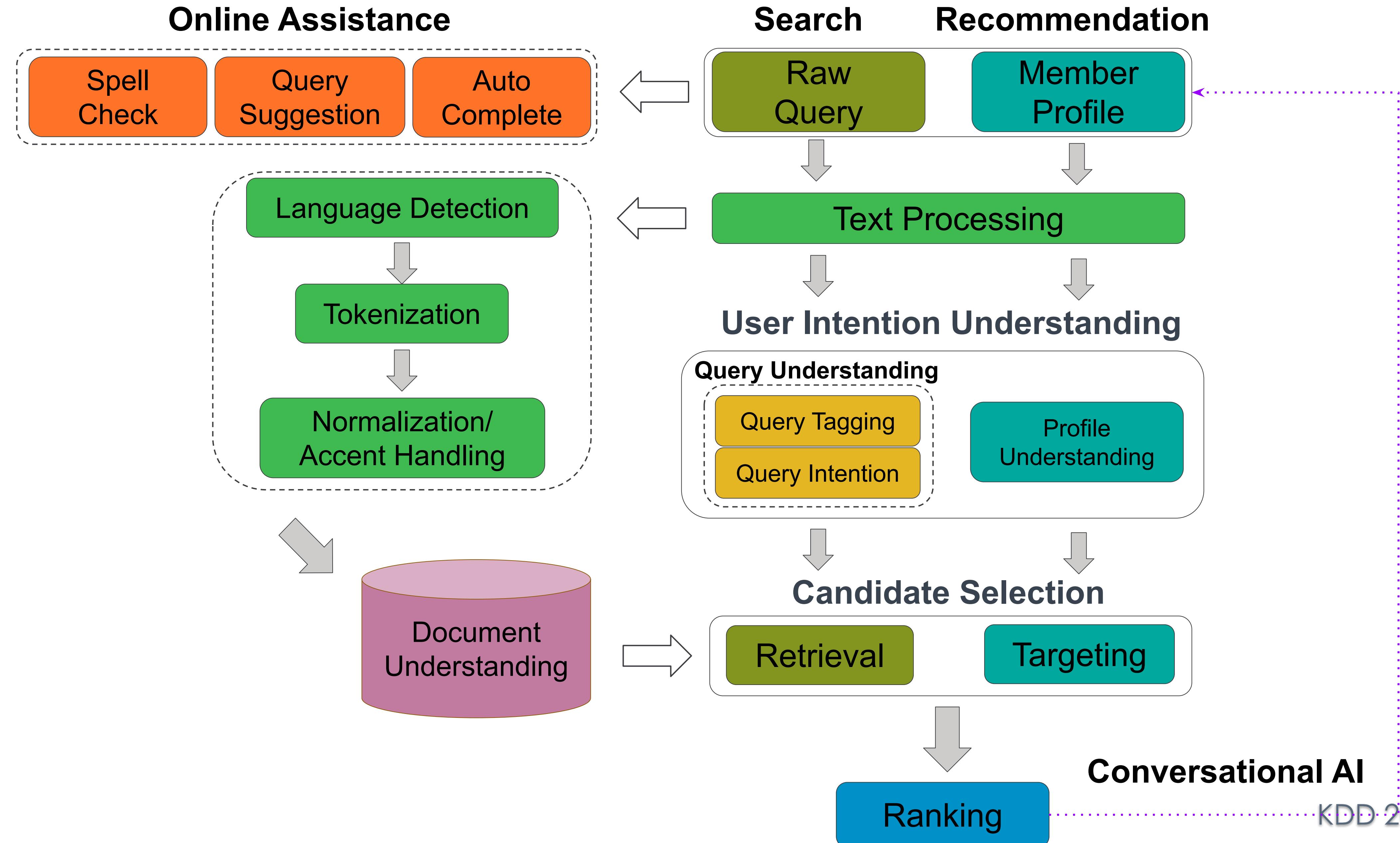
Recommender Systems



Search and Recommender Systems



LinkedIn Search & Recommendation Ecosystem



Introduction

1

Search and
Recommender Systems

2

Applications at
LinkedIn

3

Tutorial Overview

LinkedIn Search and Recommender Systems

Job Search

The screenshot shows the LinkedIn job search interface. At the top, there's a search bar with 'machine learning engineer', a location filter for 'United States', and a 'Search' button. Below the search bar are filters for 'Jobs', 'Date Posted', 'LinkedIn Features', 'Company', 'Benefits', and 'All filters'. A dropdown menu 'Sort by: Relevance' is open. The main results area shows a list of job posts:

- Sr. Machine Learning Engineer - Home Timeline** (Twitter) - Promoted, San Francisco, CA, US, \$144K – \$314K, 3 days ago, 2 applicants.
- Machine Learning Field Engineer** (Cloudera) - Promoted, Palo Alto, CA, US, 5 days ago.
- Machine Learning Engineer** (Apple) - Cupertino, CA, US, \$147K – \$400K, 1 day ago, 7 applicants.
- Senior Machine Learning Device Software Engineer** (Amazon Web Services (AWS)) - East Palo Alto, CA, US, 5 days ago, 2 applicants.
- Machine Learning Engineer** (PayPal) - San Jose, CA, US, 6 days ago.

On the right side, there's a detailed view of the first job post from Twitter, showing the job title 'Sr. Machine Learning Engineer - Home Timeline', the company 'Twitter', the location 'San Francisco, CA, US', and the posting date 'Posted 3 days ago · 19 views'. There are 'Save' and 'Apply' buttons. Below this, there's a table with columns 'Job', 'Company', and 'Connections'.

Job	Company	Connections
• 2 applicants	• 1,001-5,000 employees	• 4 connections
• Associate	• San Francisco, CA	• 42 company alumni

Job description: Twitter's Consumer Product Teams are responsible for core features of twitter.com, which includes Timelines, Tweets, Search, Trends, Recommendations, Notifications, Tweet details/permalink, and more! Our code operates at massive scale and speed, serving billions of requests per day, connecting hundreds of millions of active Twitter users to real-time information about their lives and the world we live in.

Who We Are: At Twitter, our mission is to instantly connect users to the information most meaningful to them. Realizing this involves work in areas such as machine learning, applied data science, recommendation systems, information retrieval systems, natural language

Job Recommendation

The screenshot shows the LinkedIn job recommendation feed. At the top, it says 'Jobs you may be interested in'. Below are several job posts:

- Senior Manager - Machine Learning Infrastructure** (Tinder, Inc.) - San Francisco Bay Area, 1 week ago, Easy Apply.
- Director of Applied Machine Learning** (Arm) - San Jose, CA, US, 2 weeks ago.
- Head of Data Science** (Govzilla) - Pleasanton, CA, US, 1 day ago, Easy Apply.
- AI Research Engineering Manager** (Facebook) - Menlo Park, CA, US, 1 week ago.
- Vice President, AI** (Intuit) - Mountain View, CA, US, 6 days ago.
- Director of Machine Learning** (Tubi) - San Francisco, CA, US.

On the right, there's a detailed view of the first job post from Tinder, showing the job title 'Senior Manager - Machine Learning Infrastructure', the company 'Tinder, Inc.', the location 'San Francisco Bay Area', the posting date 'Posted 1 week ago · 803 views', and 'Save' and 'Easy Apply' buttons. Below this, there's a summary table:

Job	Company	Connections
• 6/10 skills match	• 201-500 employees	• 1 connection
• 50 applicants	• Los Angeles, California	• 4 company alumni

Job description: Tinder brings people together. With tens of millions of users, hundreds of millions of downloads, 2+ billion swipes per day, 20+ million matches per day and a presence in 190+ countries, our reach is expansive—and rapidly growing. Machine learning plays a critical role at Tinder, we have dozens of Machine learning models in production that power product features like user recommendations, photo moderation, anti-spam, etc. with a variety of algorithms - from linear models to decision trees to deep neural networks and these models operate at a large scale. We are looking for a Senior Manager that will build and manage our machine learning infra team to enable and empower data science and product teams on ML projects at Tinder.

In this Senior Manager role, you will:

Rank job posts for LinkedIn member to help the user to find jobs that he/she would like to apply for (then, click the “apply” button)

LinkedIn Search and Recommender Systems

People Search

The screenshot shows the LinkedIn search results for the query "bo long artificial intelligence". At the top, there is a search bar with the query and navigation links for Home, My Network, Jobs, and Messaging. Below the search bar, there are filters for People, Connections, Locations, Current companies, and All Filters. A banner at the top of the results page reads "The Berkeley MBA - Same degree, different schedules. Evening. Weekend. Executive". Below the banner, there is a link to "Continue Search in Recruiter" and a button for "8+ additional advanced filters". The main results section shows "Showing 5,164 results". One result is highlighted for "Bo Long • 1st", who is a Machine Learning/Artificial Intelligent, Recommendation and Search professional based in San Francisco Bay Area, with a skill in Artificial Intelligence. There is a "Message" button next to the profile. Other results include "Tim Converse, Daniel Tunkelang, and 341 other shared connections".

Lead Search

The screenshot shows the LinkedIn Sales Navigator search results for the keyword "vp of marketing". The top header includes links for HOME, DEALS, LISTS, and DISCOVER, along with a search bar and "Advanced search" dropdown. The main area displays search statistics: 3,242,889 Total results, 115,241 Changed jobs in past 90 days, 69,508 Leads with TeamLink intro, 9,505 Mentioned in the news in past 30 days, and 514,2 Posted or past 30 days. On the left, there are filters for Keywords, Also try, and Past Lead and Account Activity, Geography, and Relationship. The results list two leads: "VP Marketing at Wix.com" (3rd connection) and "VP Marketing at Apple" (2nd connection). Both profiles show their job titles, company names, tenure, and location.

Recruiter Search

The screenshot shows the LinkedIn Recruiter search interface for the job title "machine learning engineer". The top navigation bar includes PROJECTS, CLIPBOARD, JOBS, REPORTS, and MORE. The main search bar contains the query "machine learning engineer". Below the search bar, there are filters for Search history and alerts, Showing results for, Custom filters, Job titles, and Locations. The results summary shows 619,529 total candidates, 187,915 are more likely to respond, 109,083 open to new opportunities, and 79,232 have company connections. One candidate profile is highlighted for "Machine Learning Engineer at Nano Web Group" with a "Save to a project" button. The past experience section lists "Co-op at IBM 2018 – 2018" and "Research Assistant at Florida Institute for Cybersecurity Research 2016 – 2018".

People Recommendation

The screenshot shows the LinkedIn People Recommendation section titled "People you may know with similar roles". It features four cards with circular profile pictures and role descriptions. The first card is for "Engineering Leader @Uber (We are hiring!)" with 44 mutual connections and a "Connect" button. The second card is for "Sr Manager, Software Engineering" with 36 mutual connections and a "Connect" button. The third card is for "Manager Machine Learning and Data" with 49 mutual connections and a "Connect" button. The fourth card is for "Engineering Manager at Change.org" with 19 mutual connections and a "Connect" button. A "See all" link is located on the right side.

LinkedIn Search and Recommender Systems

Content Search

The screenshot shows a LinkedIn search result for "KDD 2018 LINKEDIN". The top navigation bar includes Home, My Network (with 2 notifications), Jobs, and Messaging. The search bar shows the query "KDD 2018 LINKEDIN". The main content area displays a post from Steve Gebrezgier, 1st, about attending KDD 2018. It includes a profile picture, a bio, and a message encouraging attendees to stop by the LinkedIn booth. Below the post is a LinkedIn company page snippet for "LinkedIn" with 8,487,084 followers. At the bottom, there's a large image of the LinkedIn lobby and a caption: "LinkedIn @ KDD 2018" and "engineering.linkedin.com".

Content Recommendation

The screenshot shows a LinkedIn recommendation feed. The top navigation bar includes Home, My Network (with 14 notifications), Jobs, and Messaging. The search bar shows the query "Search". The main content area displays a post from Nadiya Hayes, 1st, about attending KDD 2019. It includes a profile picture, a bio, and a message encouraging attendees to stop by the LinkedIn booth. Below the post is a LinkedIn company page snippet for "Microsoft" with 7,954,475 followers. A large advertisement for Microsoft 365 Training Day: Desktop Deployment is displayed, featuring a man and a woman working on a laptop. To the right, there's a sidebar with job recommendations for "Technical Program Manager" roles at Google, Microsoft, and others. At the bottom, there's a comment from Brendan McWeeney.

KDD 2020

LinkedIn Search and Recommender Systems

The screenshot shows a LinkedIn search interface with a red box highlighting the search bar containing 'machine learning' and the word 'Query'. A blue arrow points from the 'Query' text in the search bar to the 'Huiji Gao' profile on the right. Another blue arrow points from the 'Member Profiles' heading to the list of profiles on the right.

LinkedIn Search Bar: machine learning **Query** United States

Filter Options: Jobs ▾ Date Posted ▾ LinkedIn Features ▾ Company ▾ Experience Level ▾ All filters

Sort by: Relevance

Course Details: Learning Courses (6h 32m · Beginner + Intermediate · Released: April 10, 2017 · 11 chapter quizzes)

Exercise Files: See all

By using Python to glean value from your raw data, you can simplify the often complex journey from data to value. In this practical, hands-on course, learn how to use Python for data preparation, data munging, data visualization, and predictive analytics. Instructor Lillian Pierson, P.E. covers the essential Python methods for preparing, cleaning, reformatting, and visualizing your data for use in analytics and data science. She helps to provide you with a working understanding of machine learning, as well as outlier analysis, cluster analysis, and network analysis. Plus, Lillian explains how to create web-based data visualizations with Plot.ly, and how to use Python to scrape the web and capture your own data sets.

Learning Objectives:

- Getting started with Jupyter Notebooks
- Visualizing data: basic charts, time series, and statistical plots
- Preparing for analysis: treating missing values and data transformation
- Data analysis basics: arithmetic, summary statistics, and correlation analysis
- Outlier analysis: univariate, multivariate, and linear projection methods
- Introduction to machine learning
- Basic machine learning methods: linear and logistic regression, Naïve Bayes

Huiji Gao
Engineering Manager - Machine Learning and AI at LinkedIn

Experience

- LinkedIn 4 yrs 3 mos
Engineering Manager - Machine Learning and AI Aug 2018 – Present · 9 mos San Francisco Bay Area
Lead LinkedIn Personalization and Search AI Foundation team - Provide LinkedIn users with intelligent experience through natural language understanding (across multiple languages) and personalization powered by Machine Learning and Artificial Intelligence.
- Staff Machine Learning Engineer Mar 2018 – Jul 2018 · 5 mos San Francisco Bay Area
Promote LinkedIn's search relevance foundation with high-quality search results and satisfactory searcher experience powered by Machine Learning and Artificial Intelligence
- Senior Machine Learning Engineer - Computational Advertising and Information Retrieval Jun 2016 – Feb 2018 · 1 yr 9 mos San Francisco Bay Area
Ads Relevance:
Worked on a variety of ads relevance products, including audience expansion behavior modeling, campaign performance optimization, and CTR prediction. Developed several important classes of machine learning models that have generated double-digit an... See more
- Applied Research Engineer Feb 2015 – May 2016 · 1 yr 4 mos San Francisco Bay Area
Computational Advertising

Role

- Research Assistant Arizona State University Aug 2009 – Dec 2014 · 5 yrs 5 mos
Design and implement a disaster relief system ACT (ASU Coordination Tracker) to enhance the coordination among relief organizations.
- Mining large-scale location-based social network data to study human mobile behavior

Member Profiles

Actions: Save Apply

Opportunities - Deep Learning for Search and Recommender Systems

Why Deep Learning?

- **Deep Semantics** from High Dimension and Sparse Data
 - Synonymous, Disambiguation, etc.
- **Easy Feature Engineering**
 - Hand Crafted Features V.S. Auto Feature Representation
- **Model Flexibility**
 - Model end-to-end process
 - Various NN components to model and cooperate systematically
- **Multi-level Feature Representation**
 - Hierarchical Representations
character -> token -> word -> phrase -> sentence

[Young et. al. 2018]

Applying Deep Learning in LinkedIn Search & Recommendation

- **Feature Driven**
 - Representation Learning
 - Using features generated from deep learning models
e.g., word embedding
- **Model Driven**
 - Power product features directly with deep learning models
 - CNN/LSTM/Seq2seq/GAN/BERT based deep NLP models

Applying Deep Learning in LinkedIn Search & Recommendation

- Offline

People Search (NDCG@10)		Job Search (NDCG@10)	Ads Click-through Rate (AUC)	Ads Conversion (AUC)
Wide Features	-	-	-	-
CNN	+1.32%	+2.36%	+3.43%	+1.37%
LinkedIn BERT	+1.98%	+5.14%	-	-

- Online (CNN)
 - Job Search
 - Job apply clicks: +1.62%
 - Onsite paid applicants: +1.38%
 - People Search
 - Satisfied clicks: +0.78%
 - Successful sessions: +0.37%

Applying Deep Learning in LinkedIn Search & Recommendation

Query Suggestion - english market

No Personalization

Coverage: +80% Impressions, +26% CTR

+1% Total job application

With Personalization

+1.18% Session Success for passive job seekers

Introduction

1 Search and
Recommender Systems

2 Applications at
LinkedIn

3 Tutorial Overview

Major Components of a Search/Recommender System

1. Query & User Understanding

A query or a userprofile are processed so that the systems can use the processed information in the other components

2. Candidate Retrieval

Retrieve all the relevant items to the user or query with high recall

3. Learning to Rank

Rank the items by the order of the relevance to the user's intent or the query with high precision

Query and User Understanding

- Query and user understanding provides personalization features for candidate retrieval and ranking.
- Infer the intent of a user by extracting semantic meaning from the query and other information.

Candidate Retrieval

- Syntactic retrieval
 - based on string matching
 - use inverted index
 - can include different fields
- Semantic retrieval
 - based on vector space representation.
 - approximate nearest neighbor search

Learning to Rank

- Ranking strategies:
point-, pair- and
list-wise.
- Common networks:
Siamese,
Interaction-based,
Wide & Deep.

Handson Sessions

1.Query & User Understanding

- Intent Classification
 - CNN with DeText
- Query Auto Completion
 - Incomplete User Input
 - Suggest completions that save keystrokes

2. Candidate Retrieval

- ElasticSearch
- Embedding Based Retrieval
- ANN Algorithms
 - LSH
 - IVFPQ
 - HNSW

3. Learning to Rank

- GLMix & GDMix
- Global Effect
 - Linear Models
 - DeText
- Random Effect
 - Large Scale Linear Models



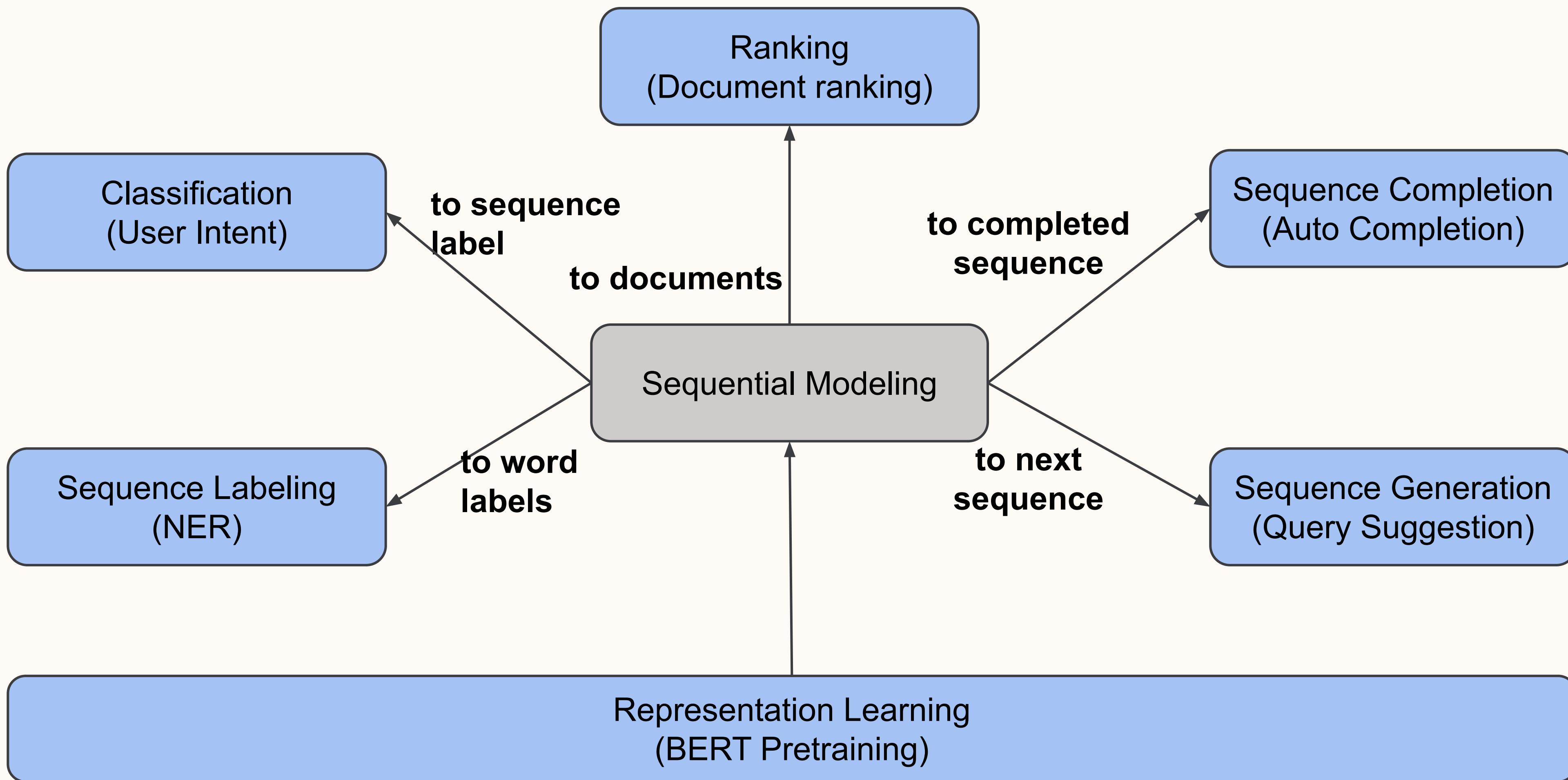
Deep Learning for Search and Recommender Systems in Practice

- Query & User Understanding

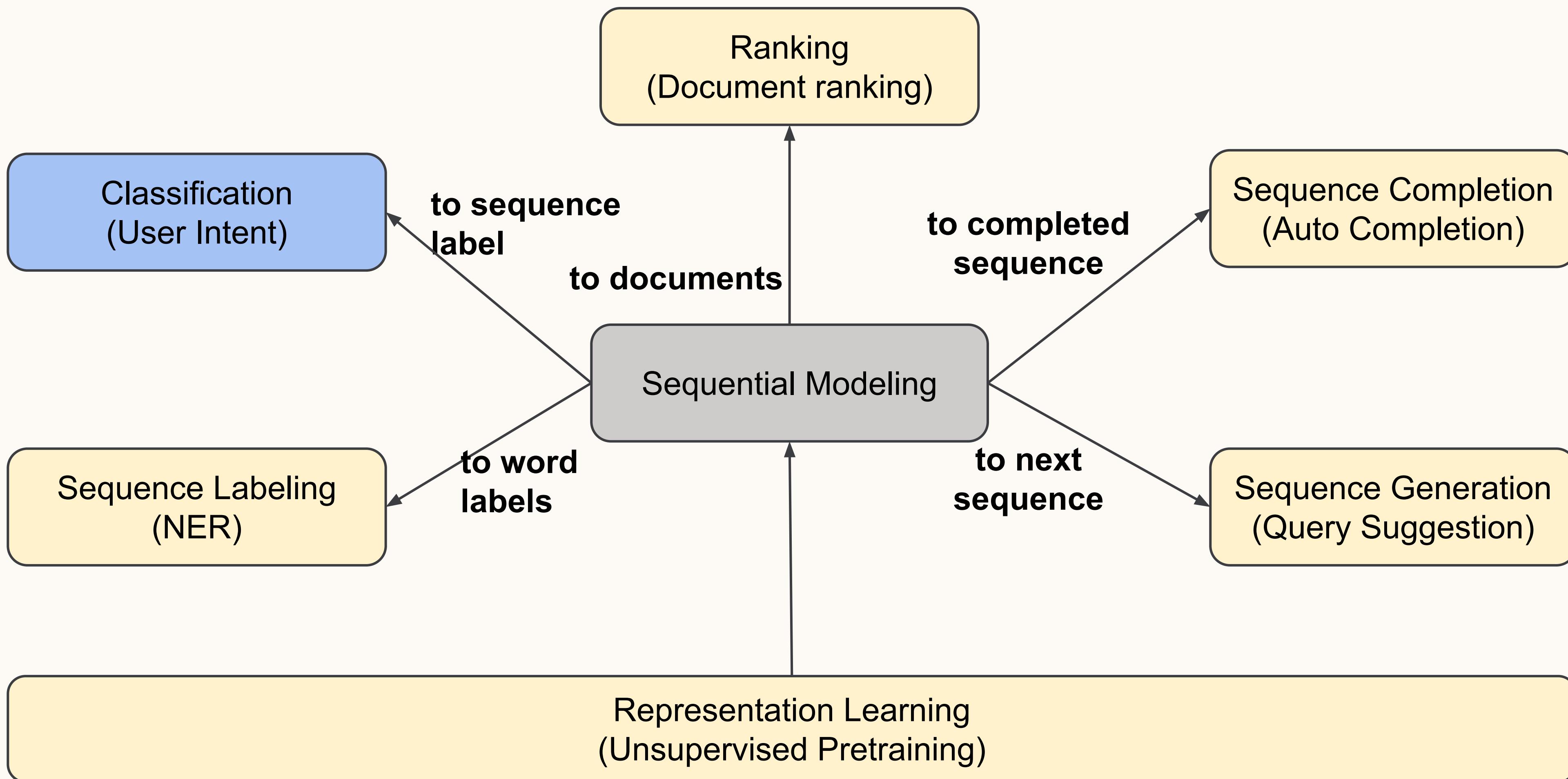


Weiwei Guo

Query and User Understanding

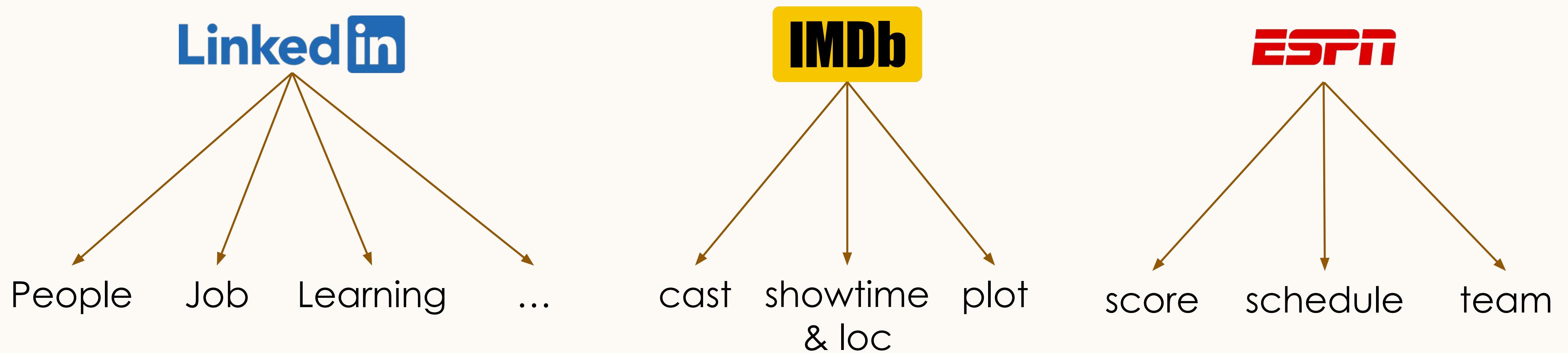


Query and User Understanding



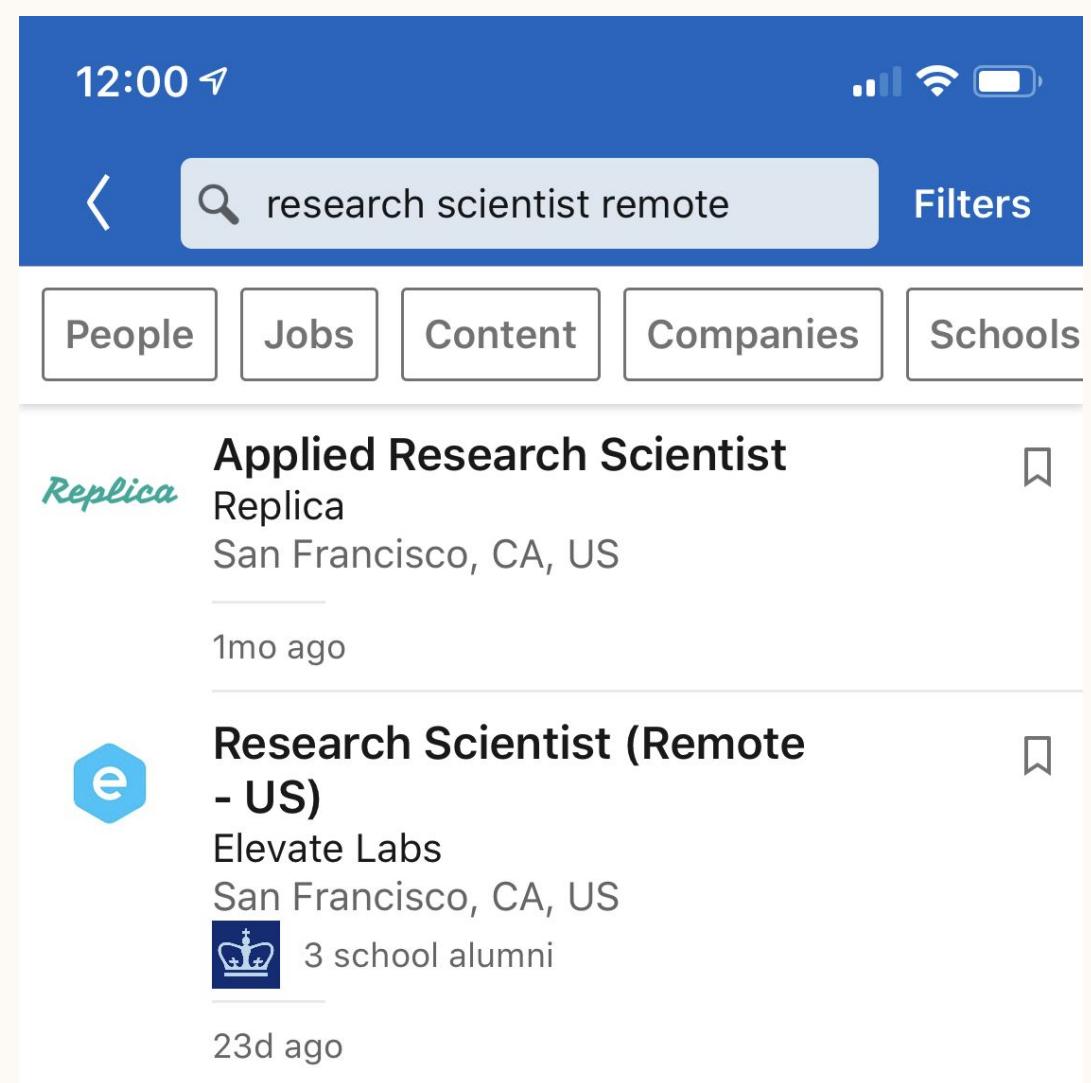
User Intent

Task: Identifying what information the user is looking for

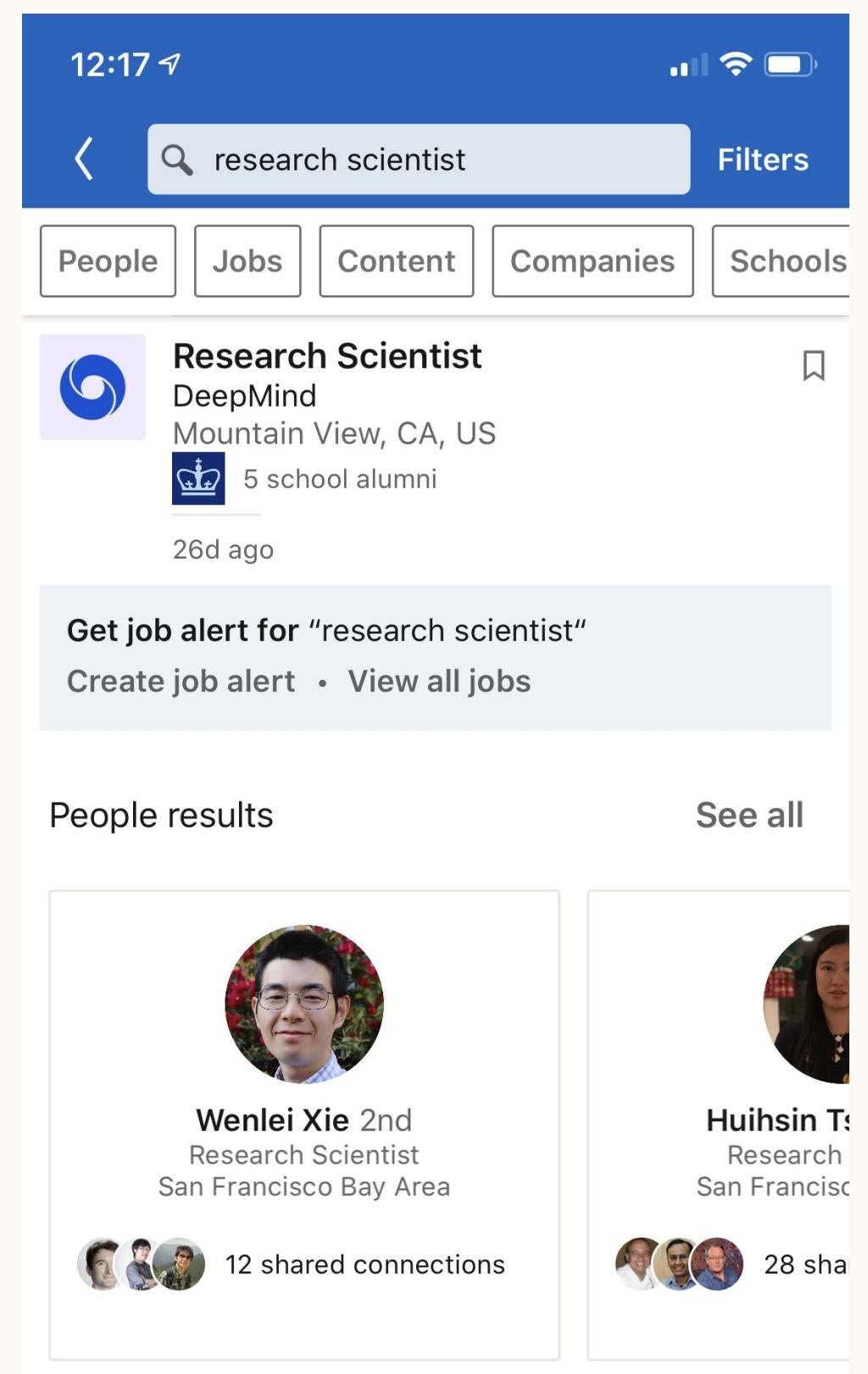


User Intent - Application

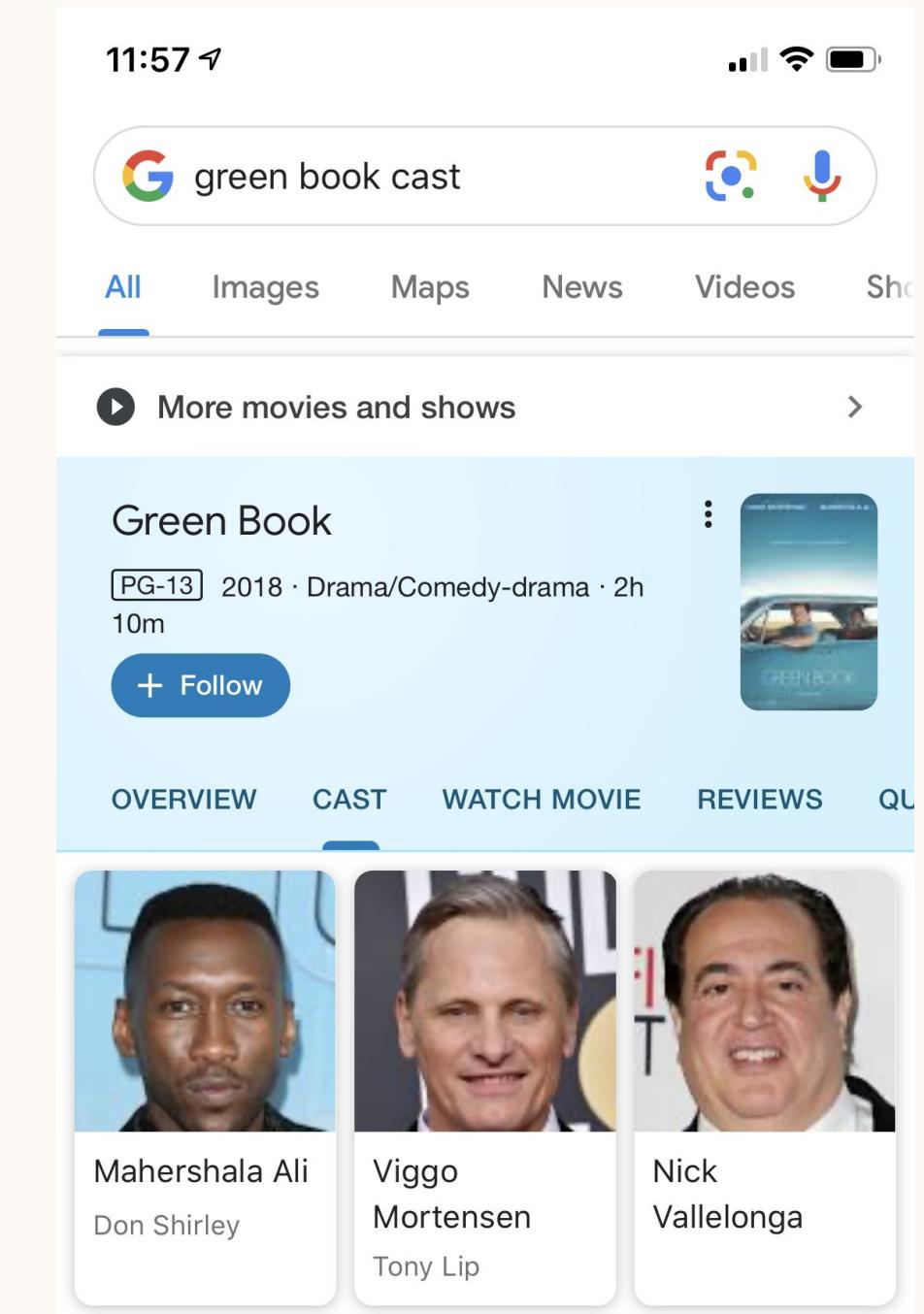
Identify Search Vertical



Search Result
Ranking/Blending



triggering QA

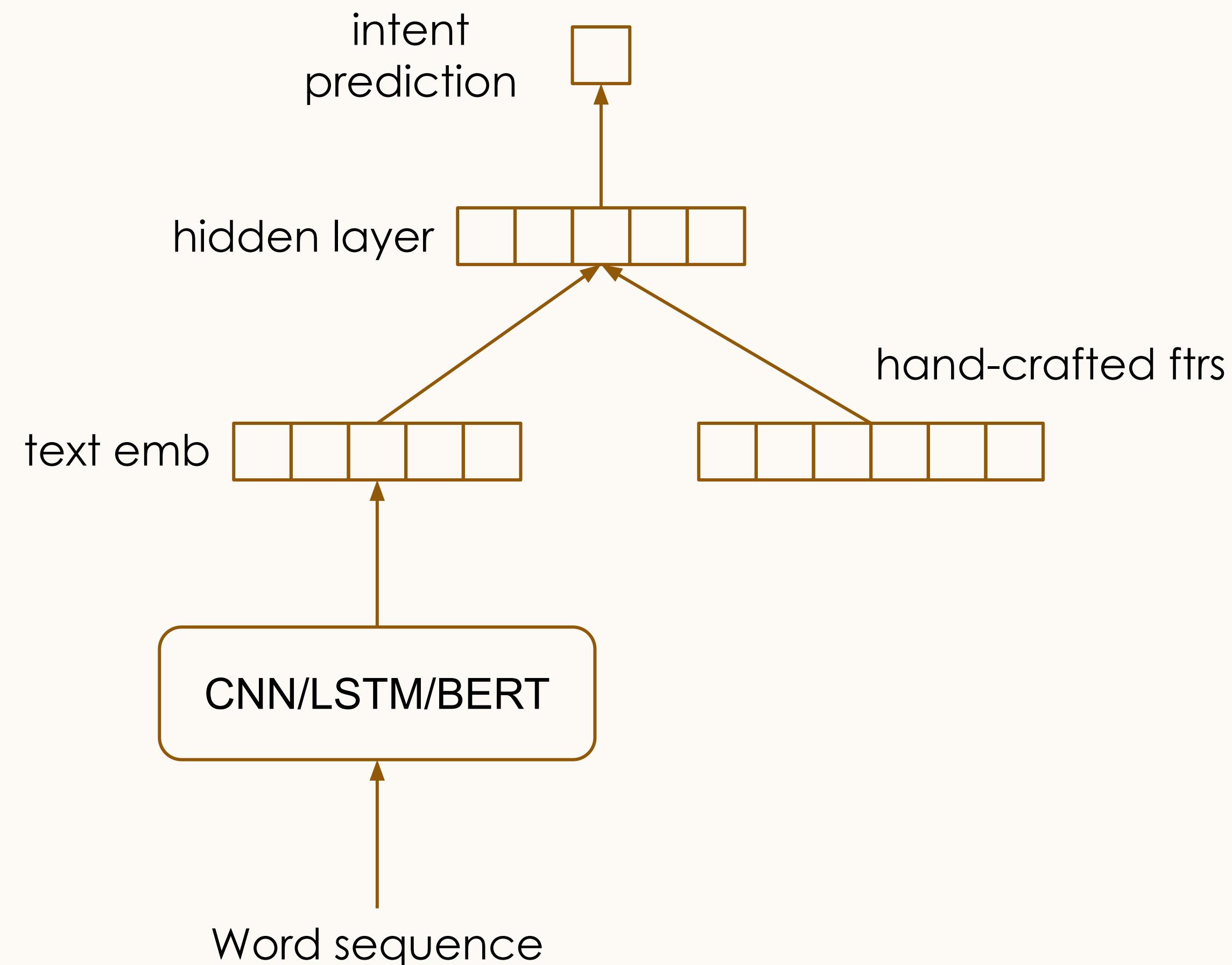


Approach: Text Classification

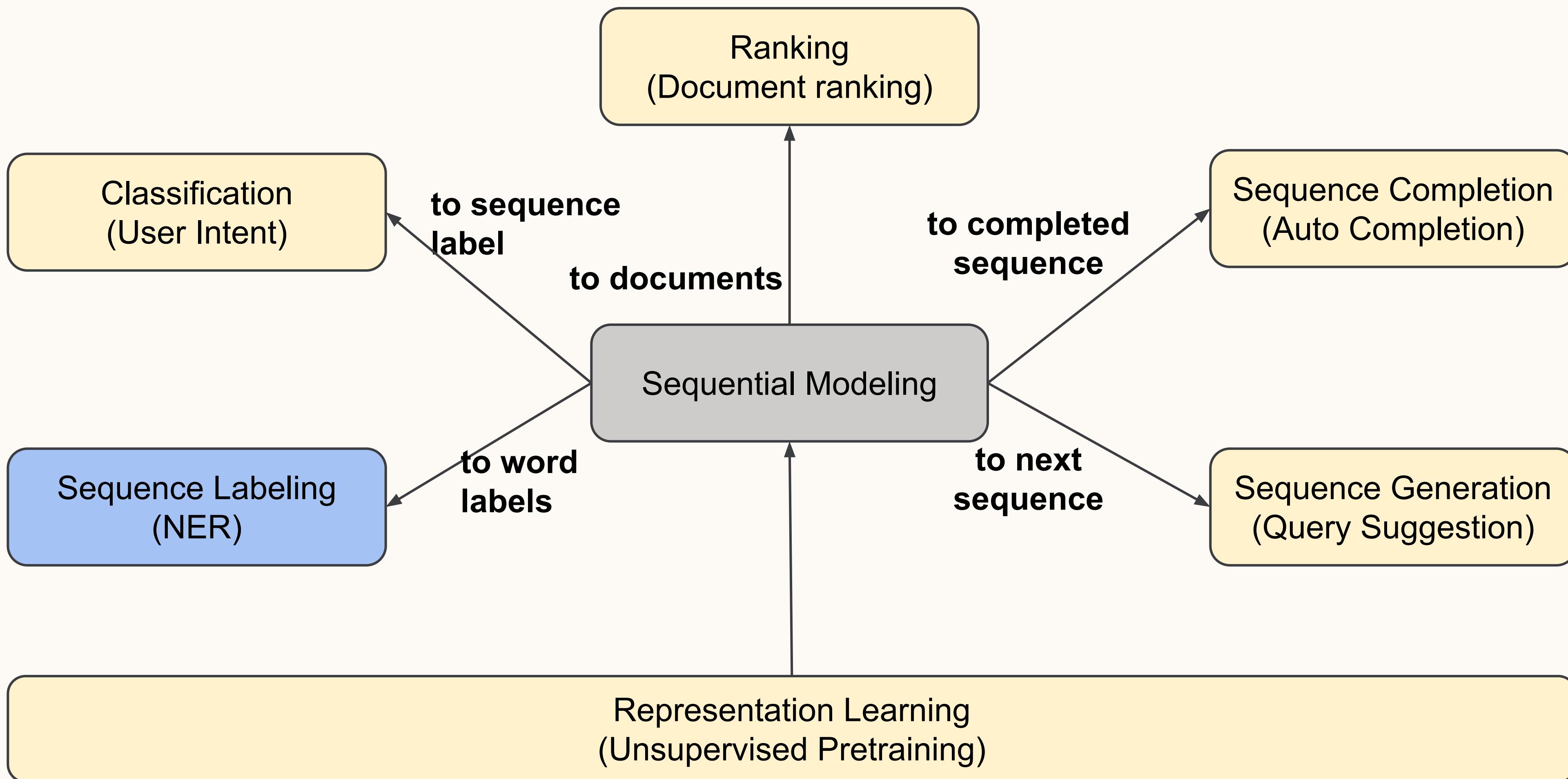
- Traditional Methods
 - Features: Bag-of-words, n-gram, etc
 - Models: Logistic Regression, naive bayes, SVM
- Deep Learning Methods
 - Word embedding + linear classifier (FastText)/CNN/LSTM/Transformer/...

Deep Learning Approaches

(Kim 2014, Hashemi et al. 2016, Liu et al. 2020)



Query and User Understanding



Sequence Labeling - Task

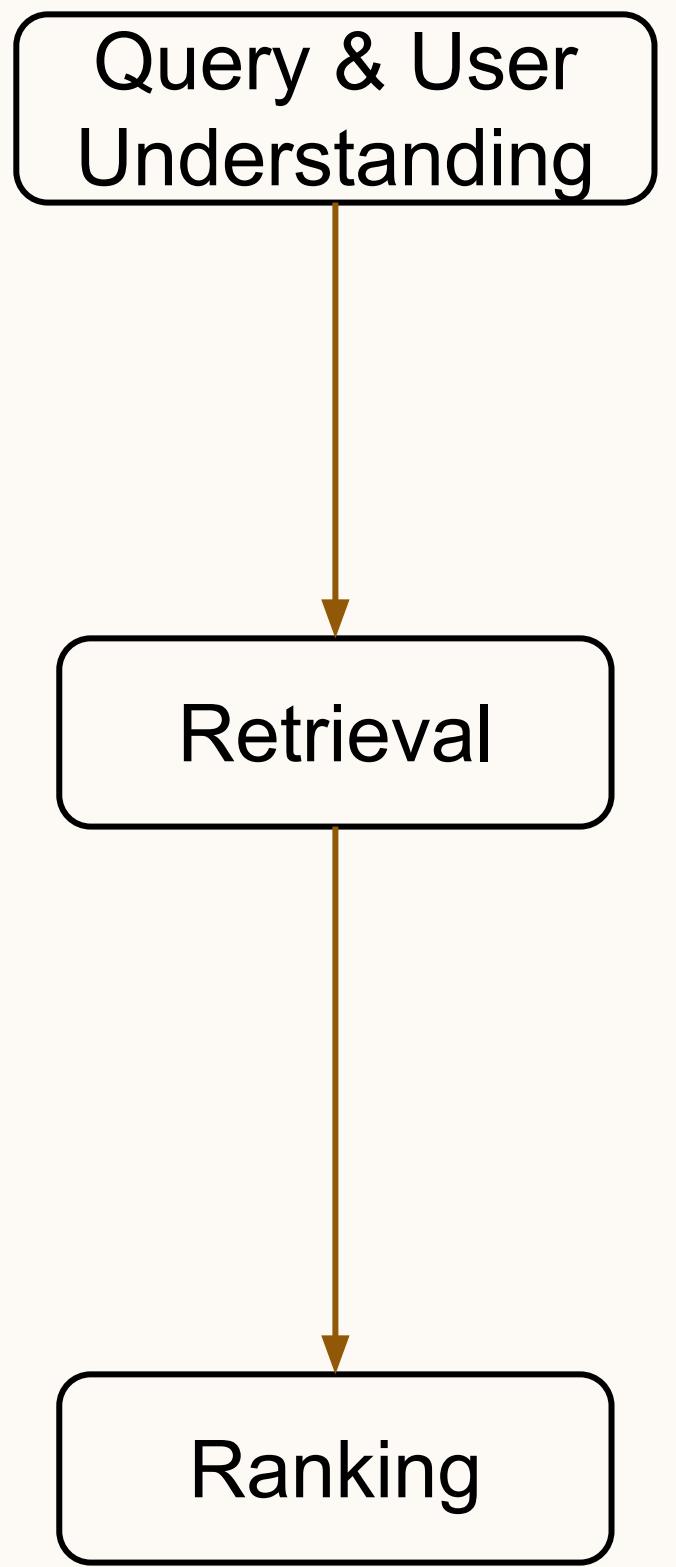
- Document understanding

Washington D.C. was named after George Washington.
LOCATION PERSON

- Query understanding

pinterest program manager
COMPANY TITLE

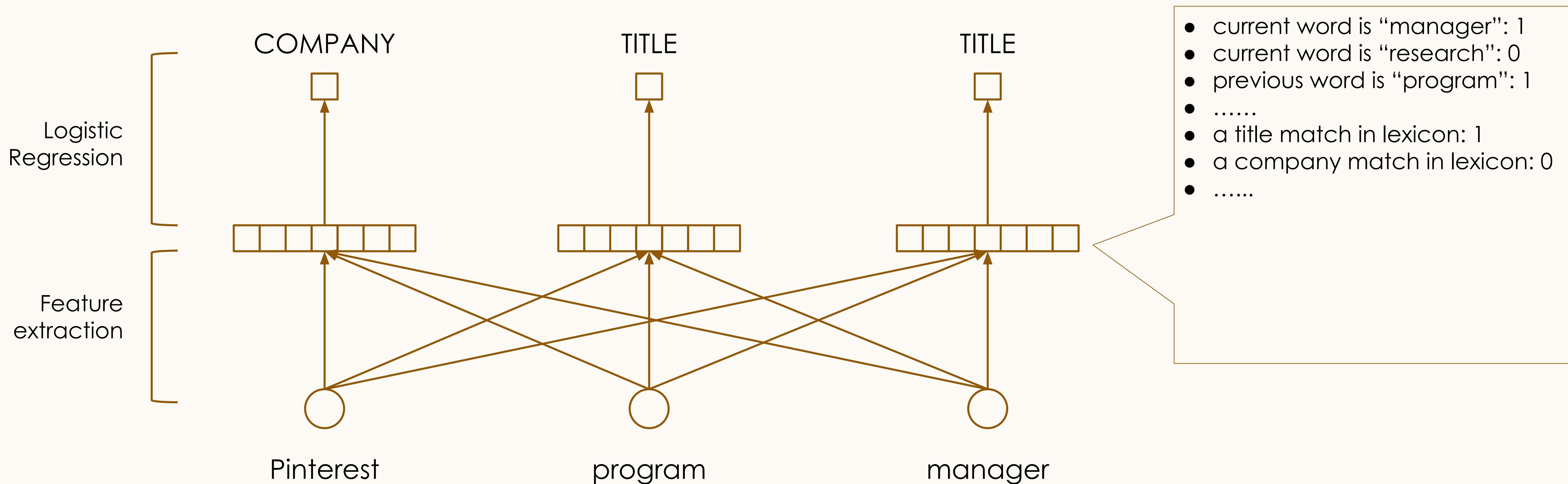
Sequence Labeling - Application



- Ignore the unimportant words
 - “looking for a **research scientist** job”
- Extracting keywords from documents
- Resolve ambiguity
 - sap (company vs skill)

Sequence Labeling - Approach

Classification on each word

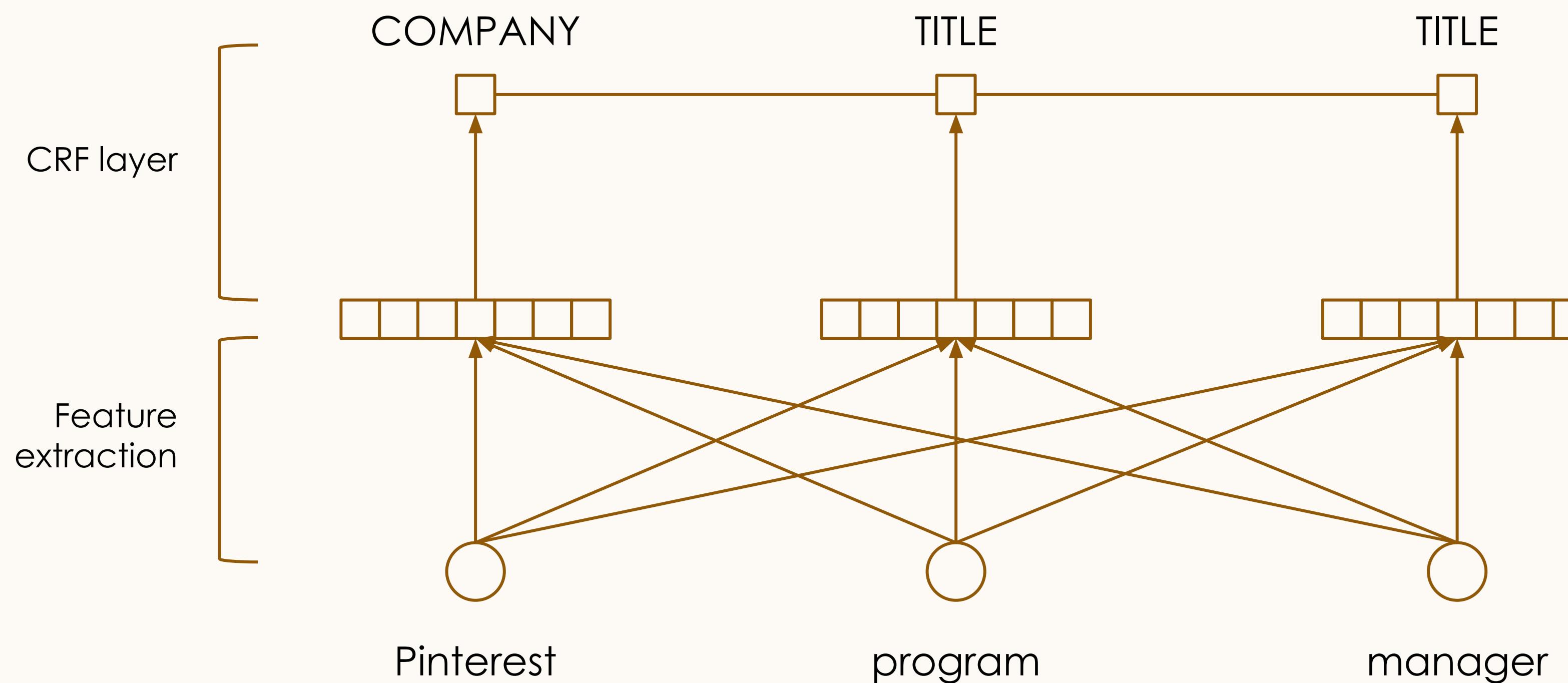


- Cons: Ignore the word label relations

Sequence Labeling - Approach

(Lafferty et al. 2001)

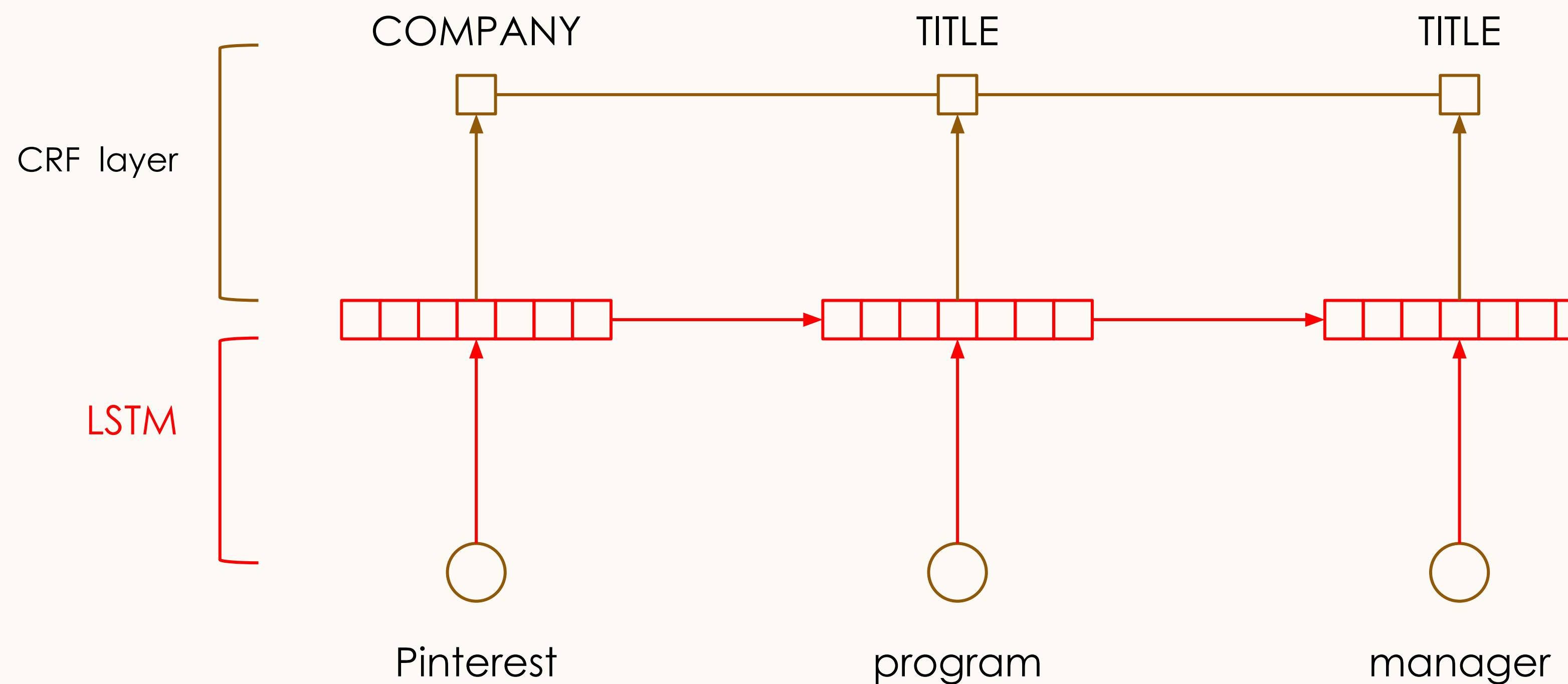
Conditional Random Field: Sequential prediction



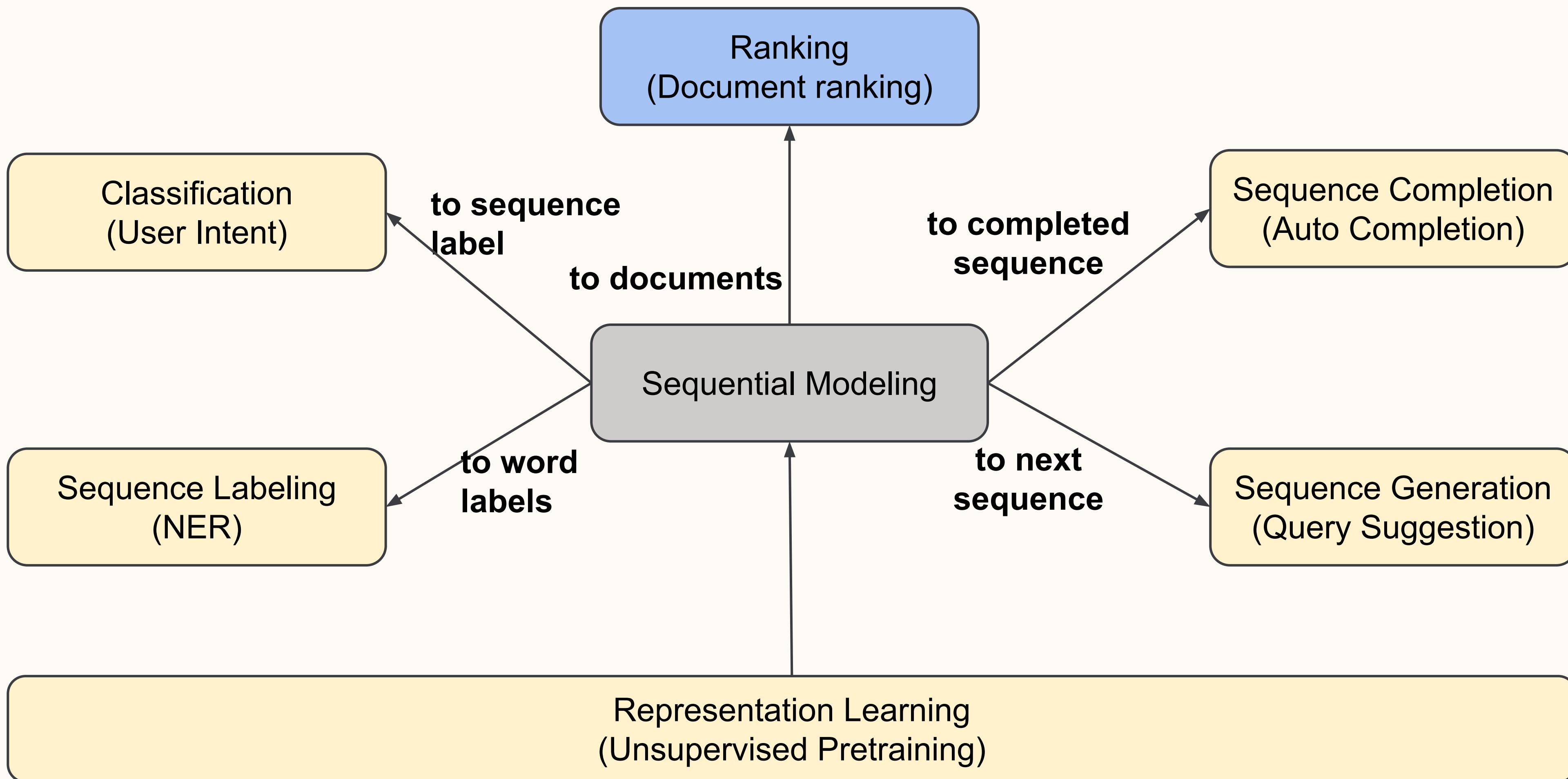
Sequence Labeling - Approach

(Huang et al. 2015, Ma & Hovy 2015)

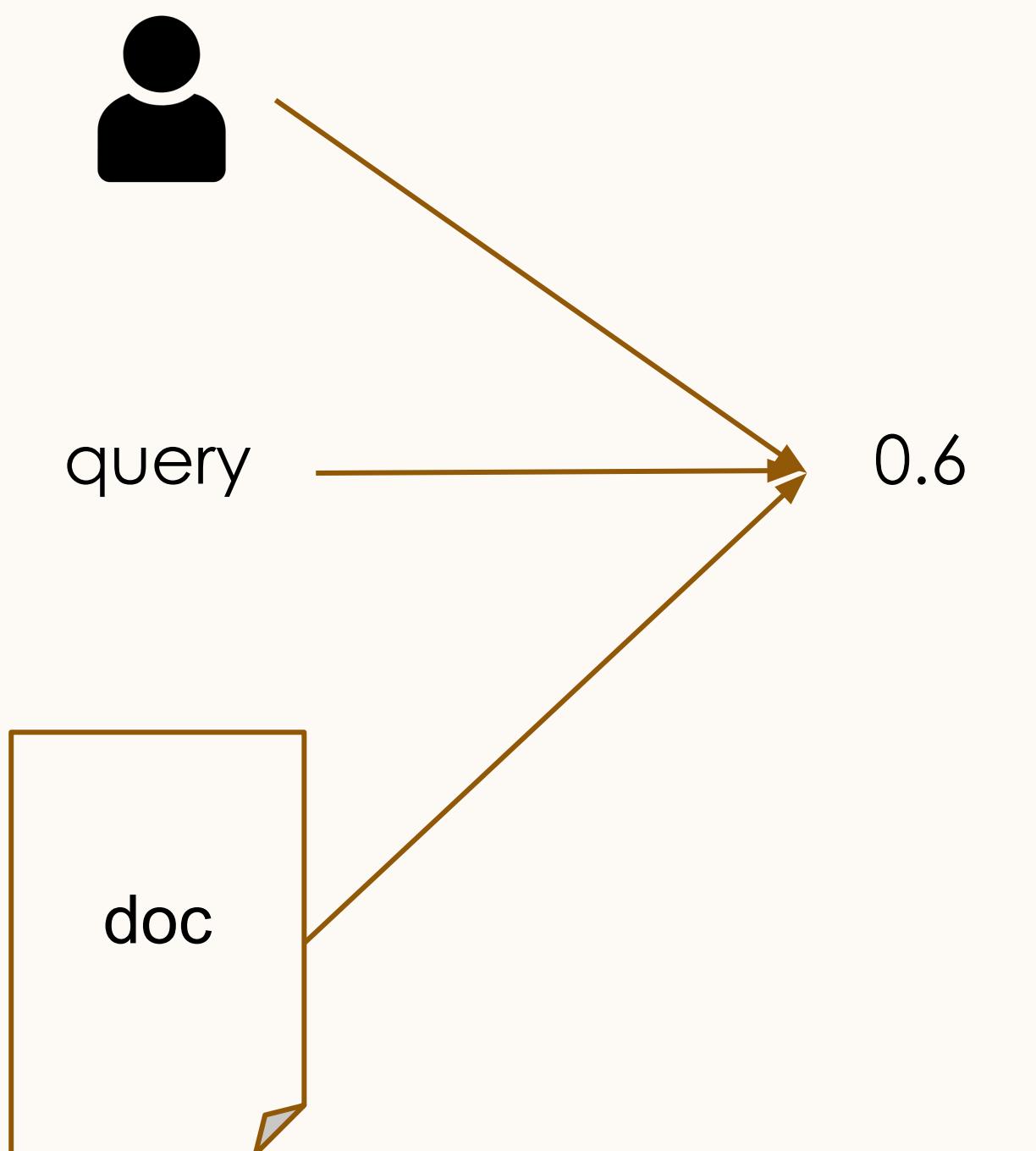
Deep learning approach: extracting powerful contextual features



Query and User Understanding



Ranking - Task

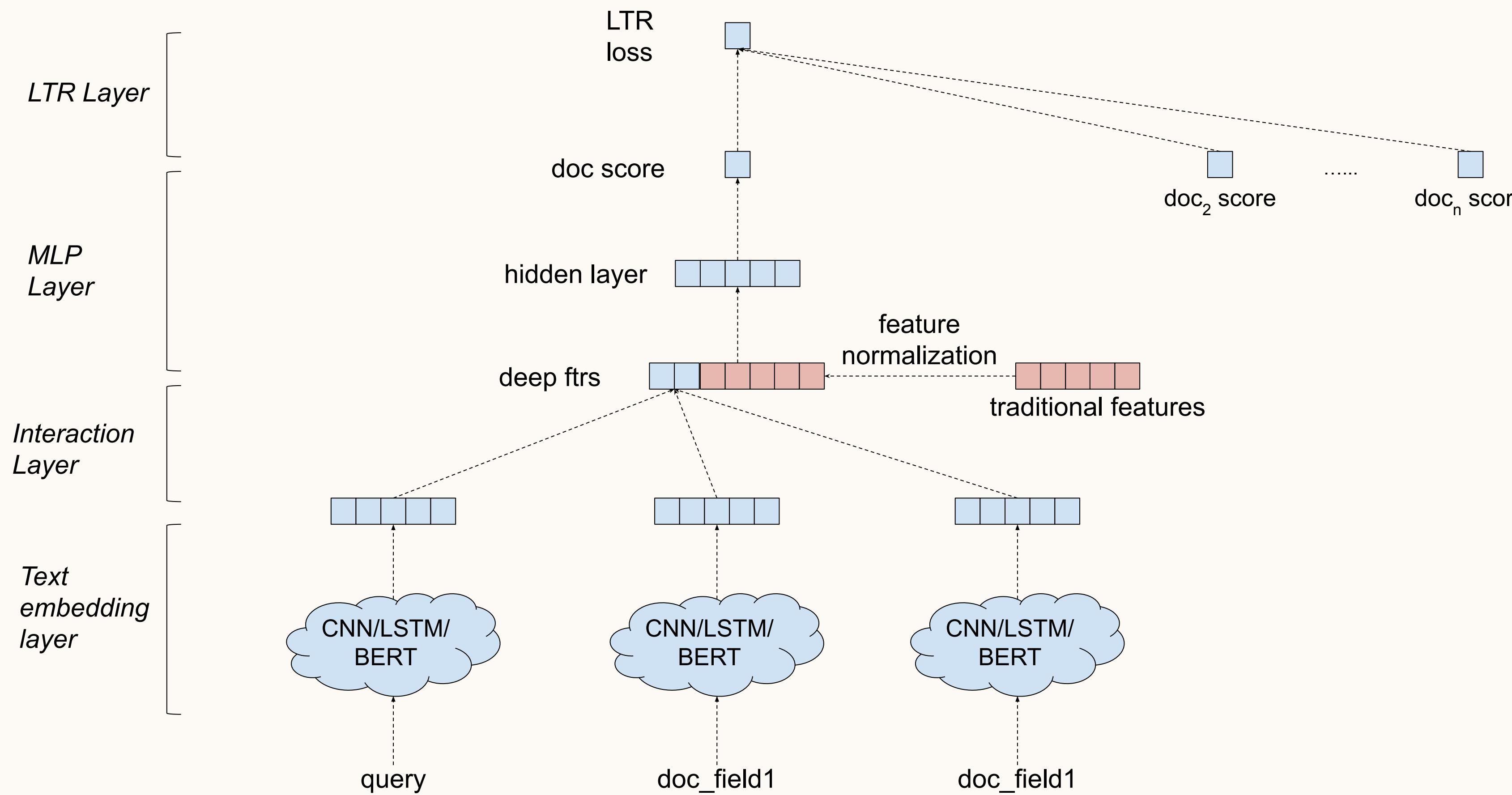


Ranking - Traditional Methods

- Hand-crafted features
 - Query/user/document matching features
 - Cosine similarity between query and doc title
 - Clickthrough rate from query to this doc based on search log
 -
 - Document alone
 - popularity
 - number of incoming links
 -

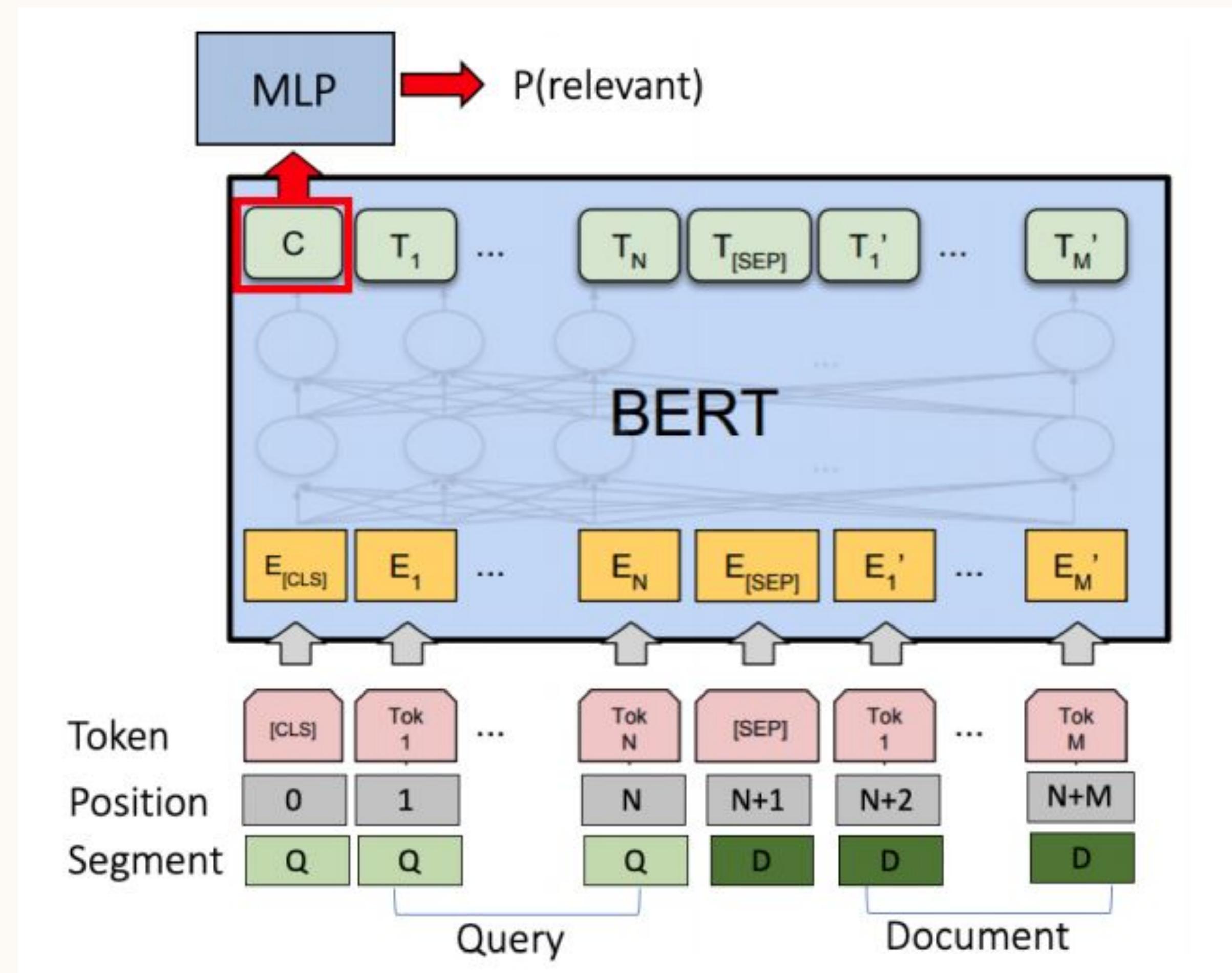
Ranking - Representation based Methods

Guo et al. 2020



Ranking - Interaction based Methods

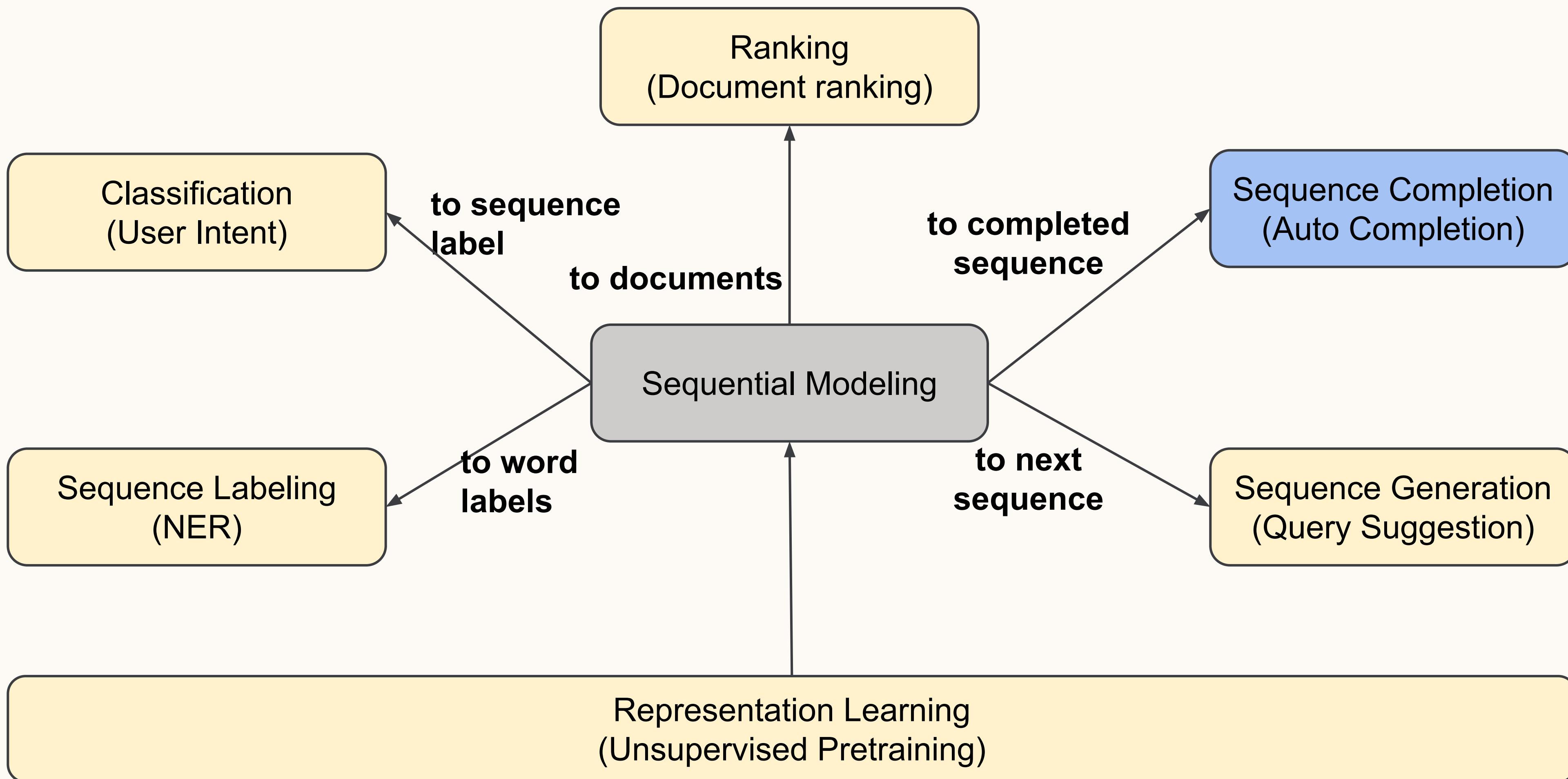
(Dai & Callan 2019)



Ranking - Comparison

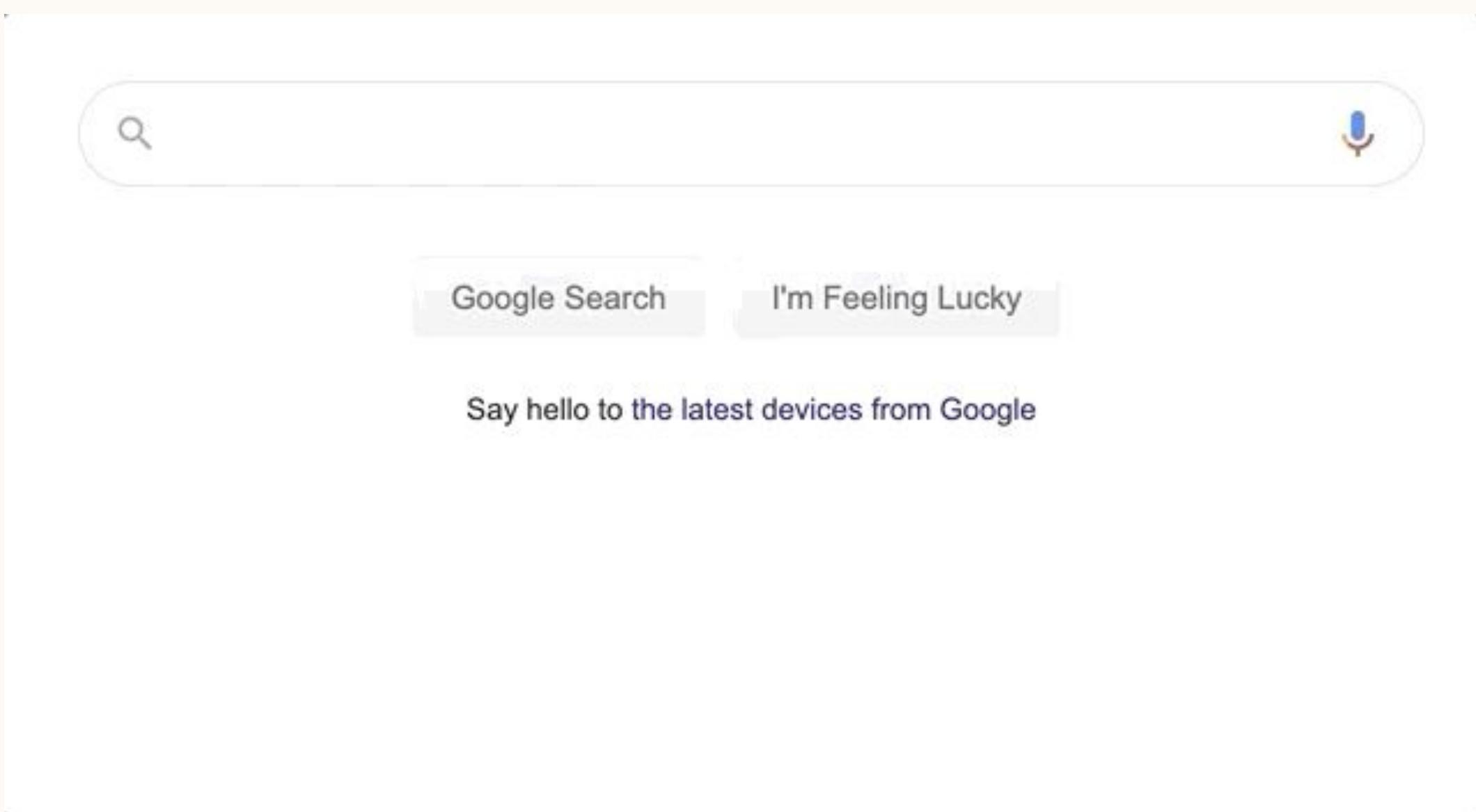
	Representation based	Interaction based
interaction between query and docs	word-level	text-level
Relevance	-	Better
Doc precomputing	✓	✗

Query and User Understanding

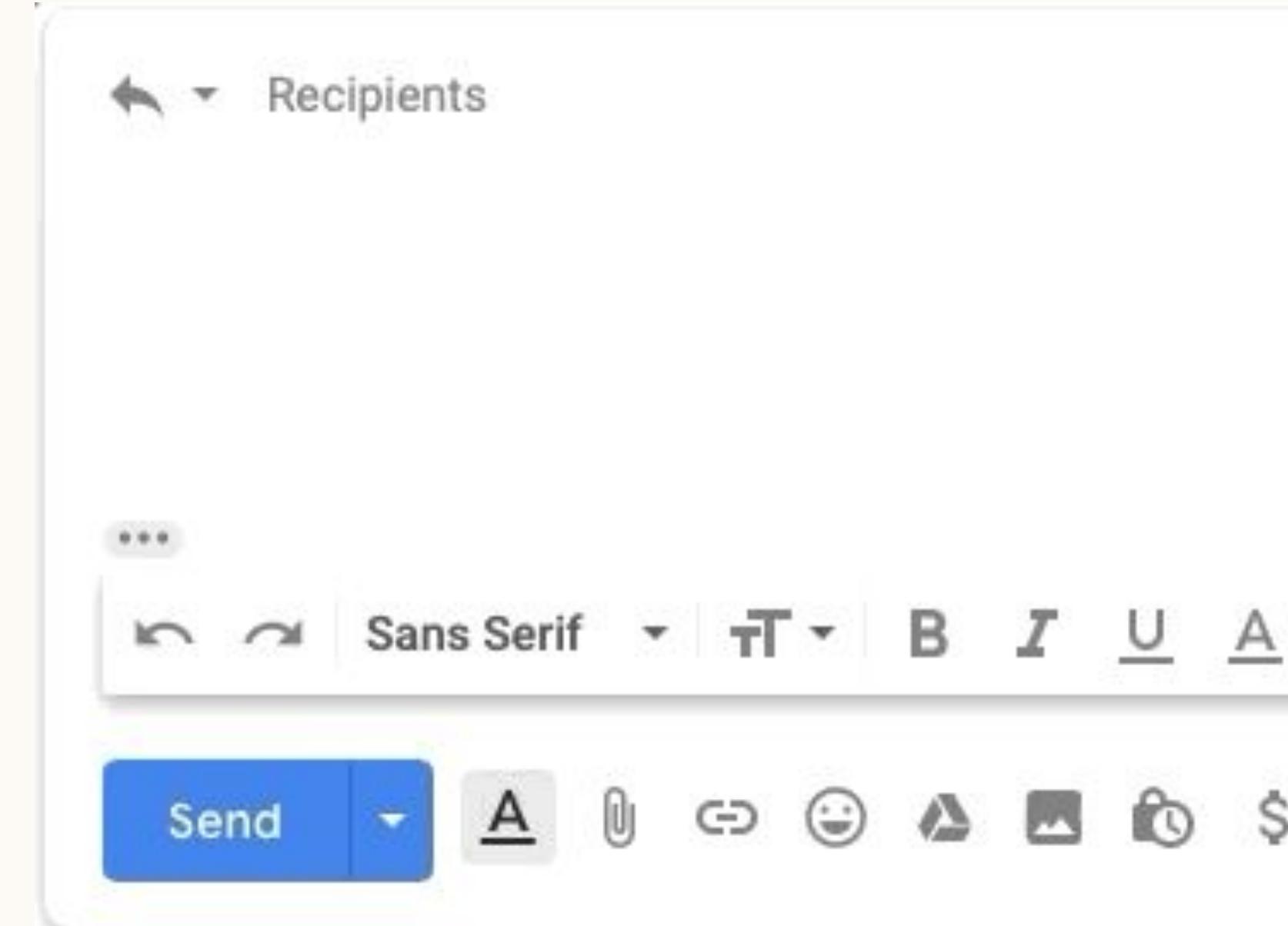


Sequence Completion - Task

On queries



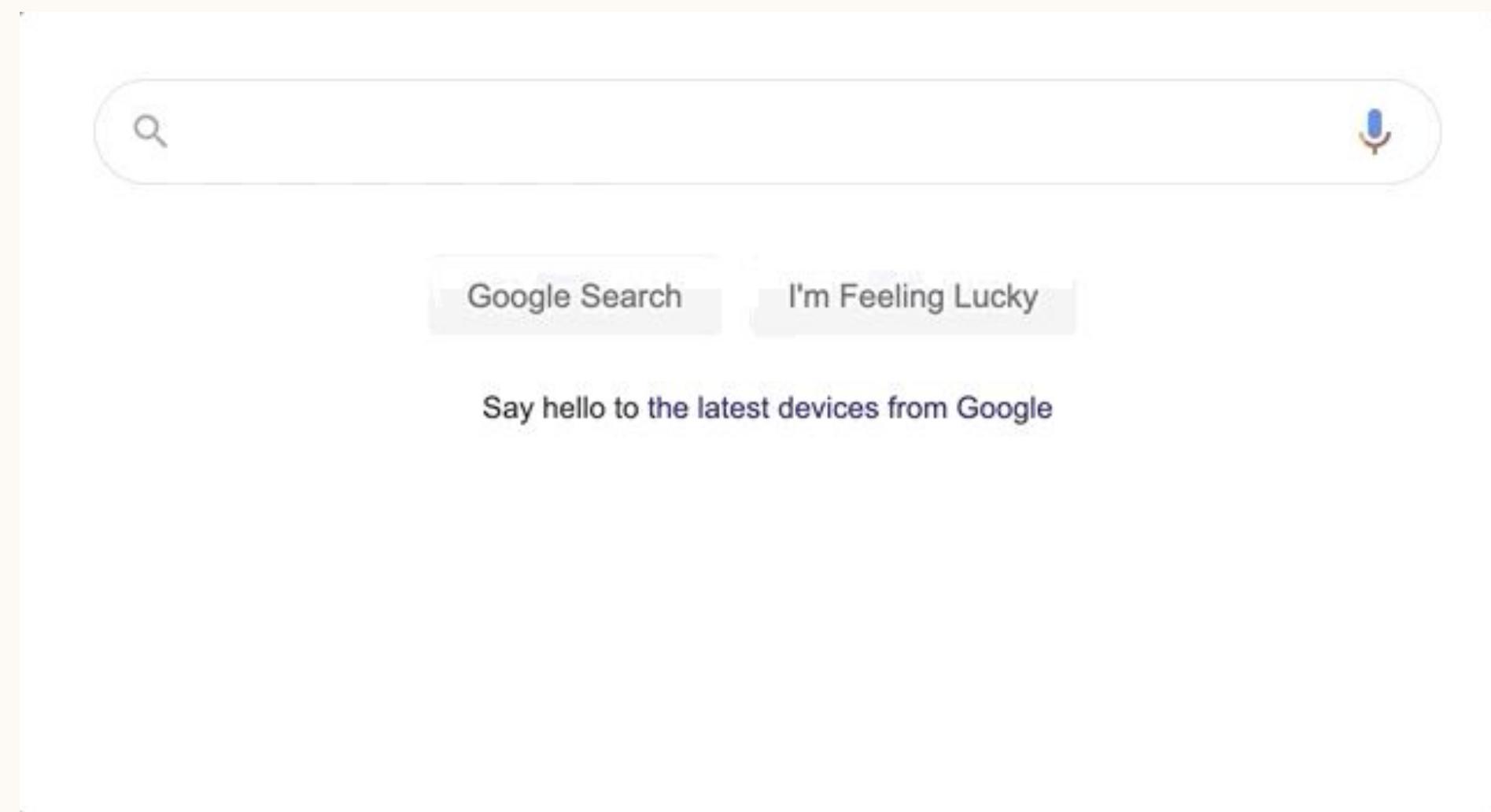
On documents



Sequence Completion - Goal

Better user experience

- Speed up human-computer interaction
 - Save $35/40 = 87.5\%$ keystrokes in the example

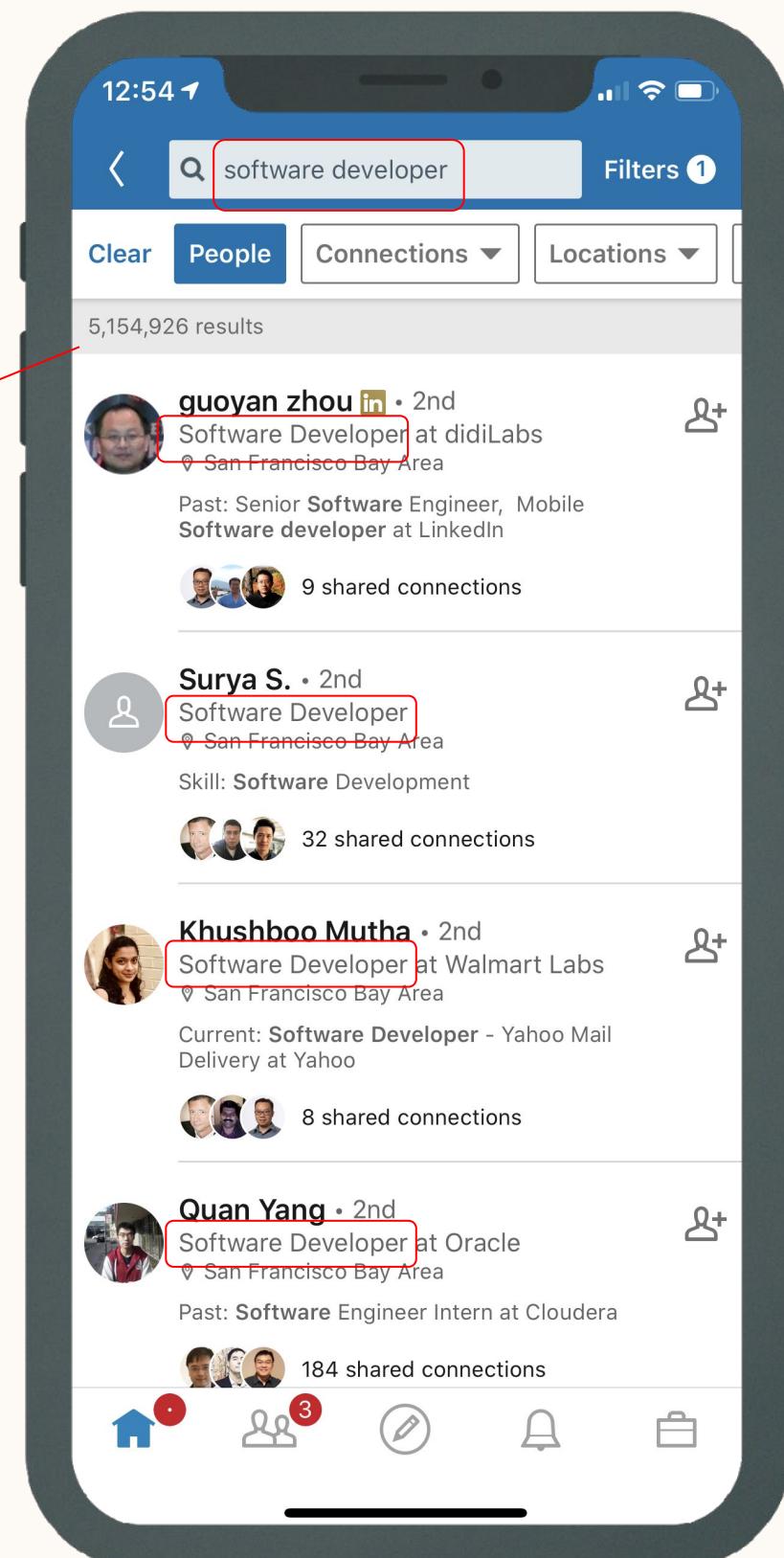


Sequence Completion - Goal

Guide Users to Better Search Results

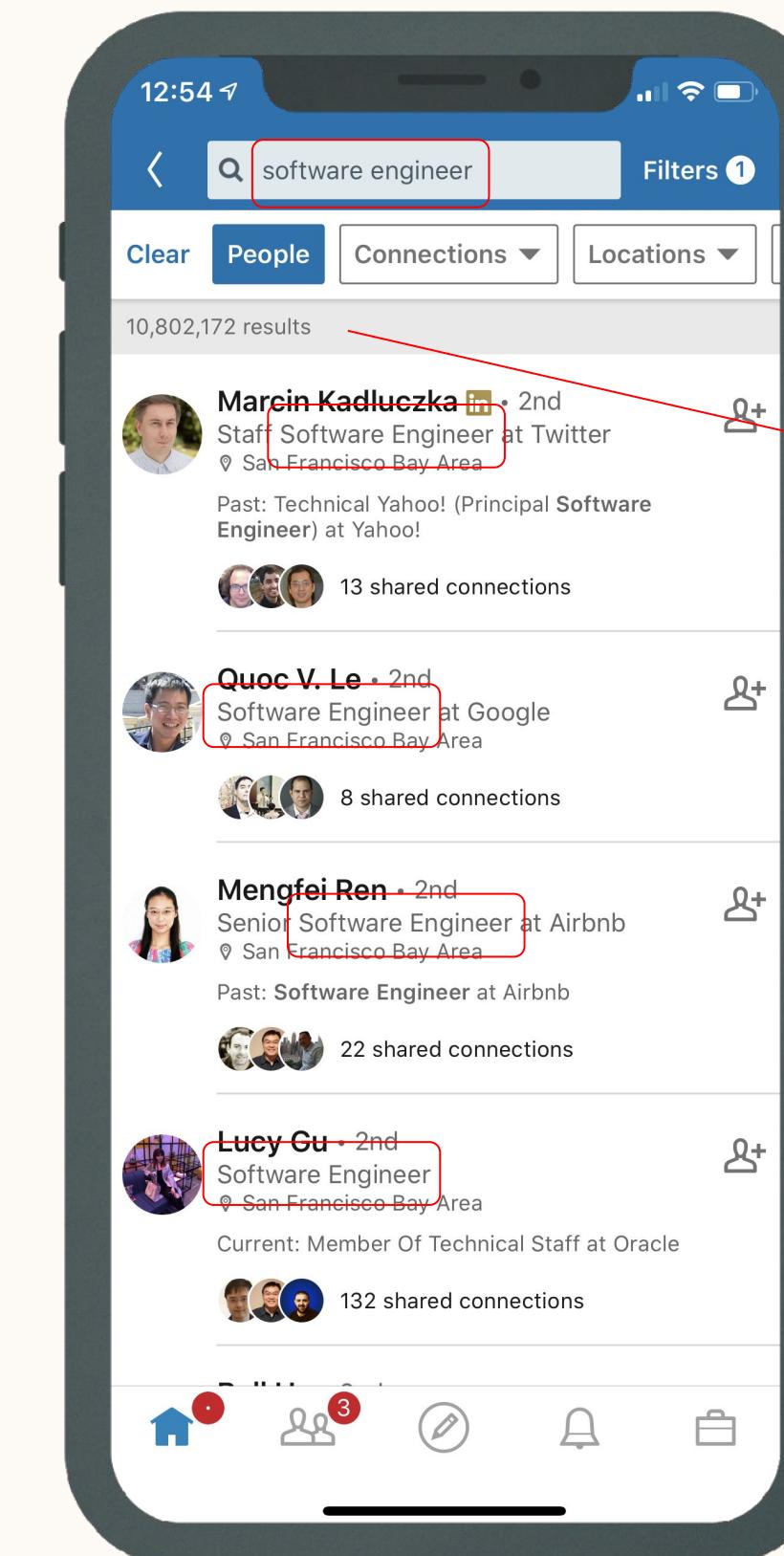
software developer

5,154,826
results



software engineer

10,802,172
results



query prefix: **sof**

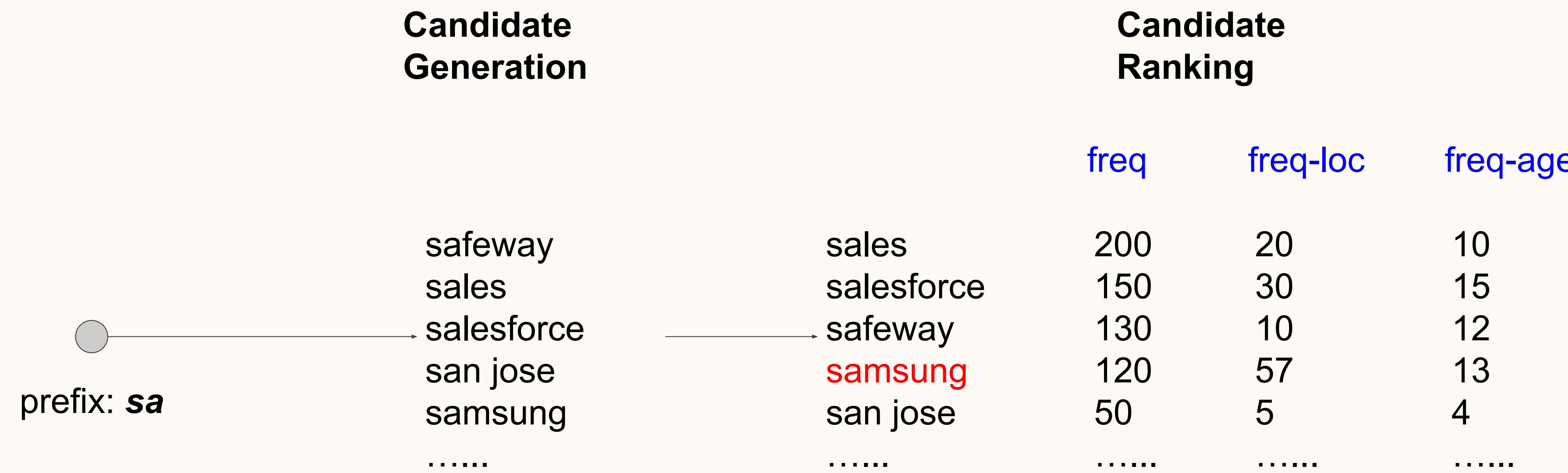
Sequence Completion - Challenge

- Limited Context
 - Little information in a query prefix
- Latency
 - For each keystroke, the results need to be returned in 50ms

Sequence Completion - Traditional Methods

Memorizing the query frequency

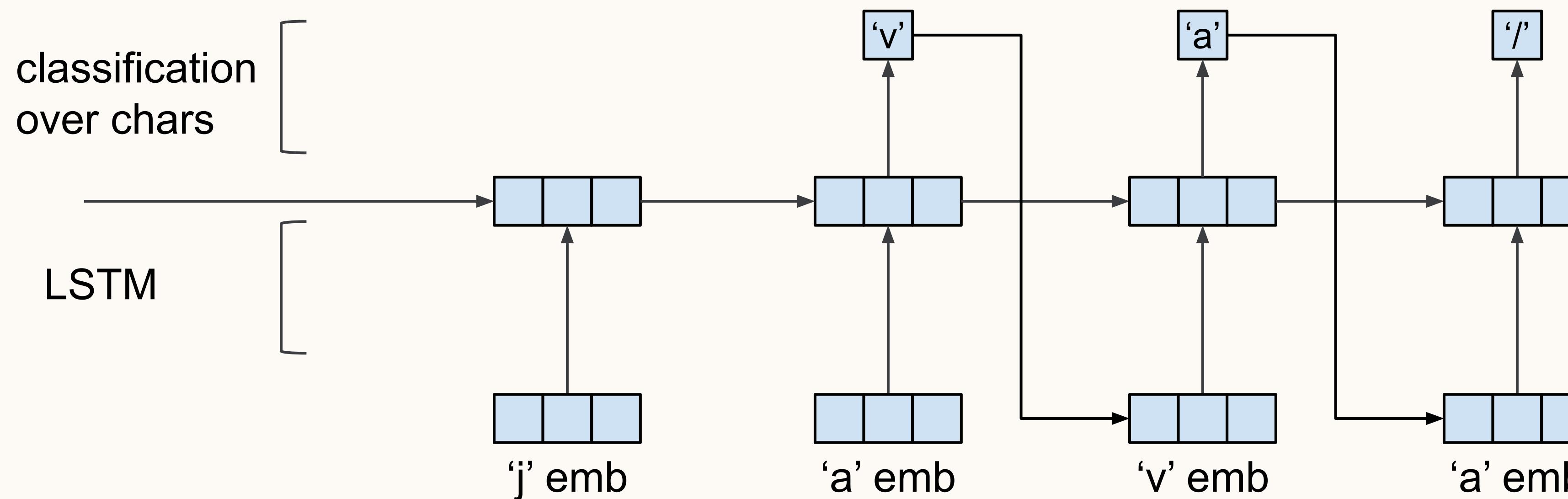
(Bar-Yossef & Kraus, 2011)



Sequence Completion: Deep Learning Approach

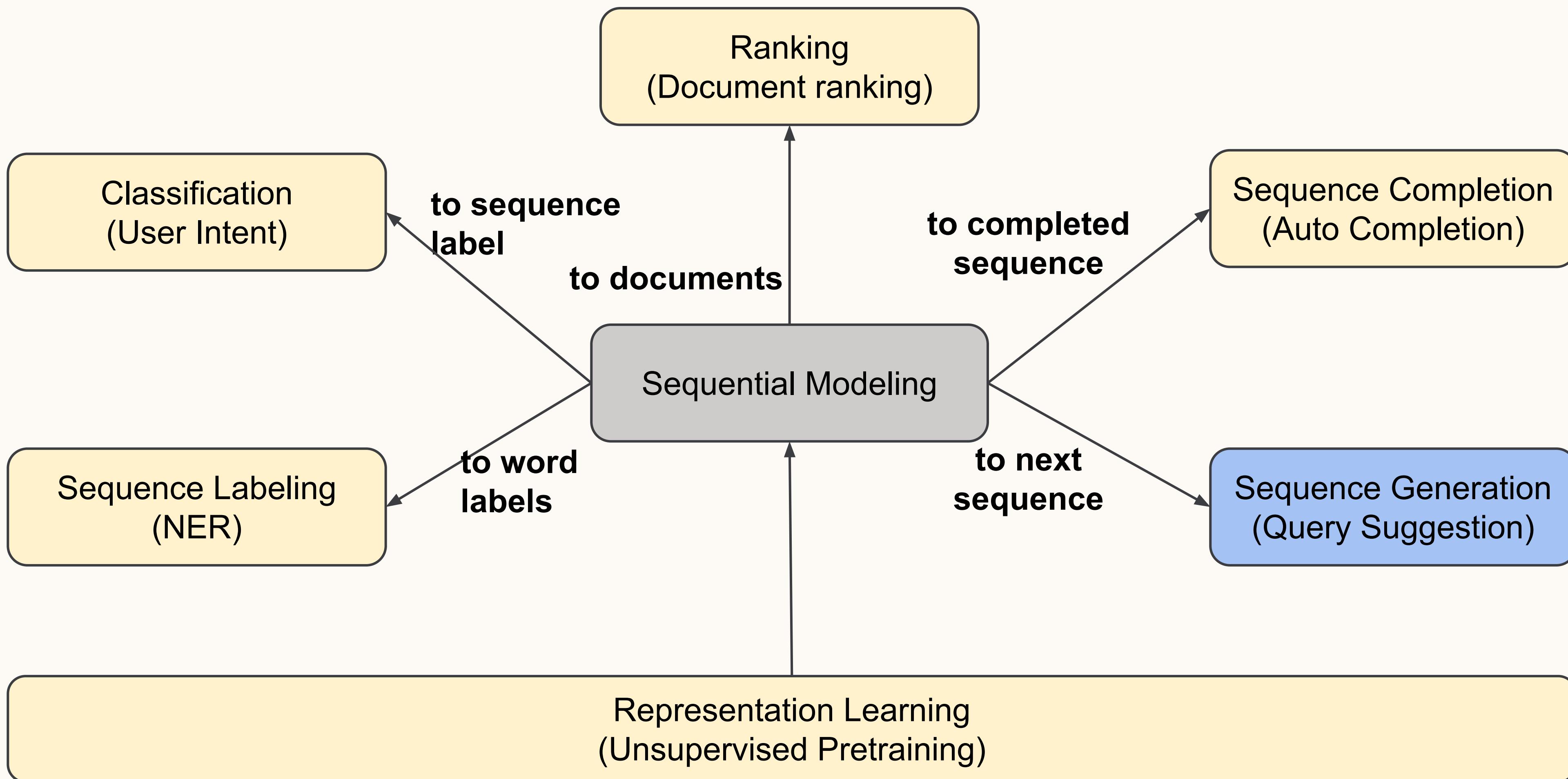
(Park 2017, Wang et al. 2020)

prefix: ja → query: java



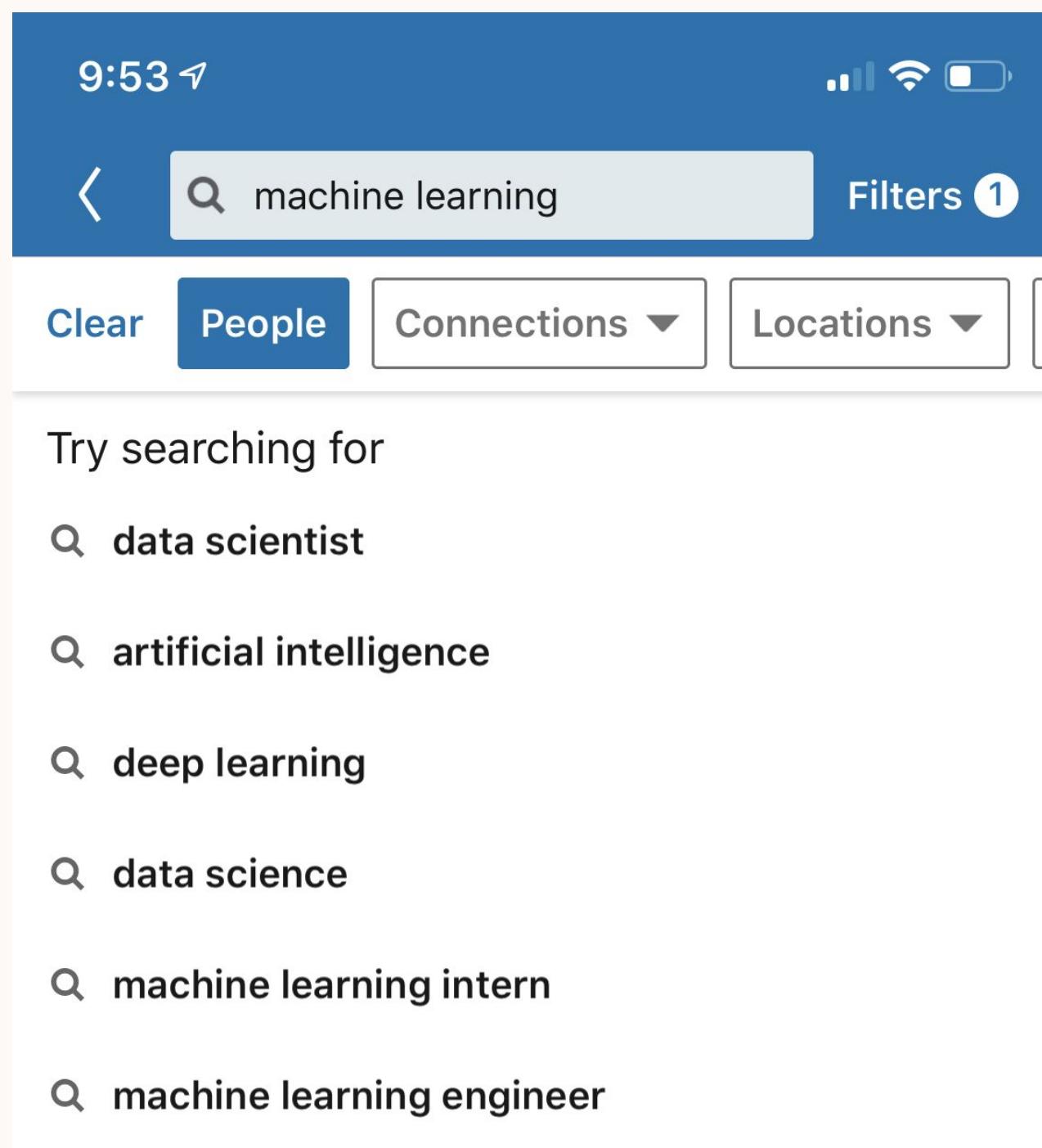
- Summarize the context by LSTM
- Generating the whole query by beam search

Query and User Understanding

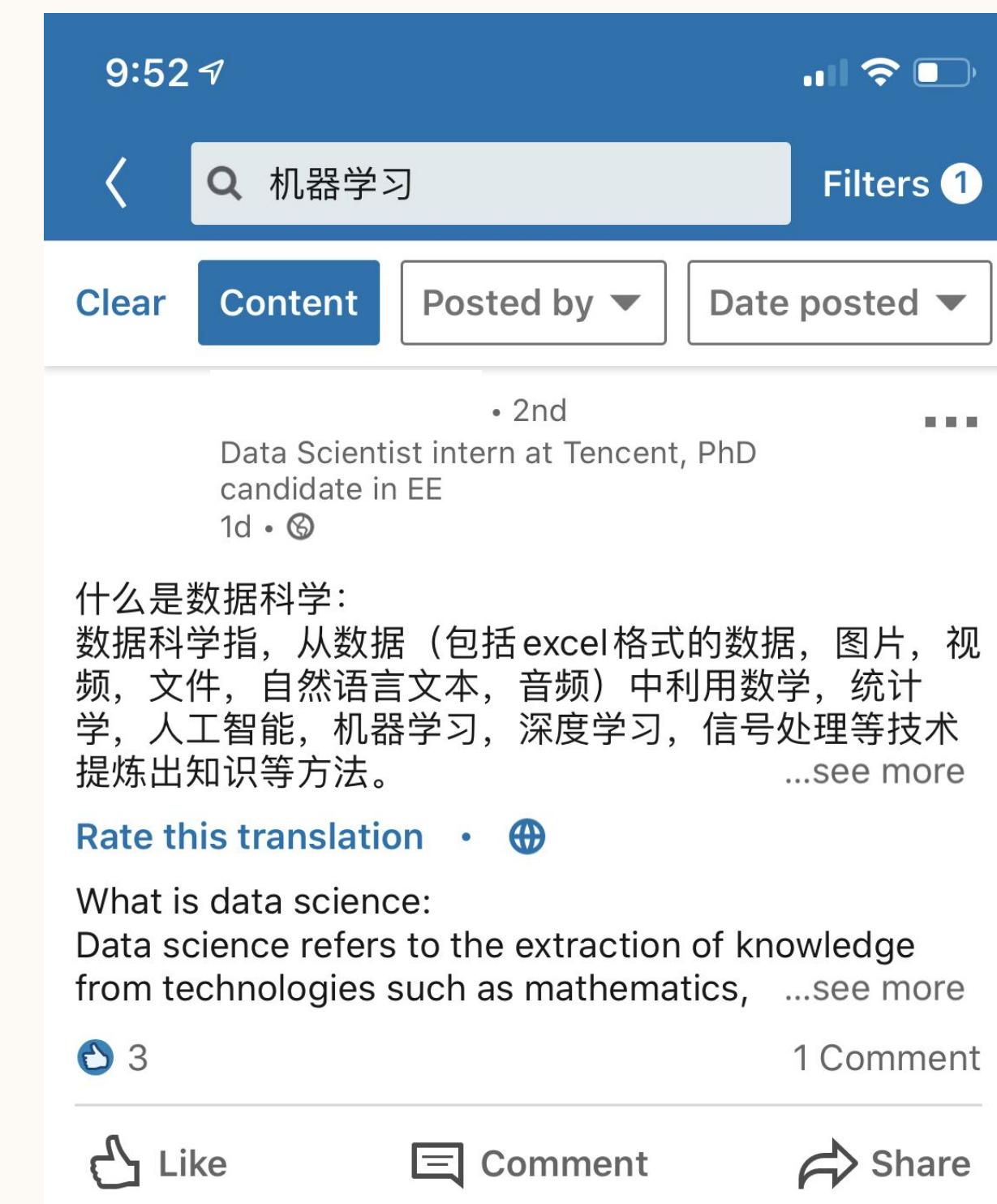


Sequence Generation - Task

Query Suggestion on queries



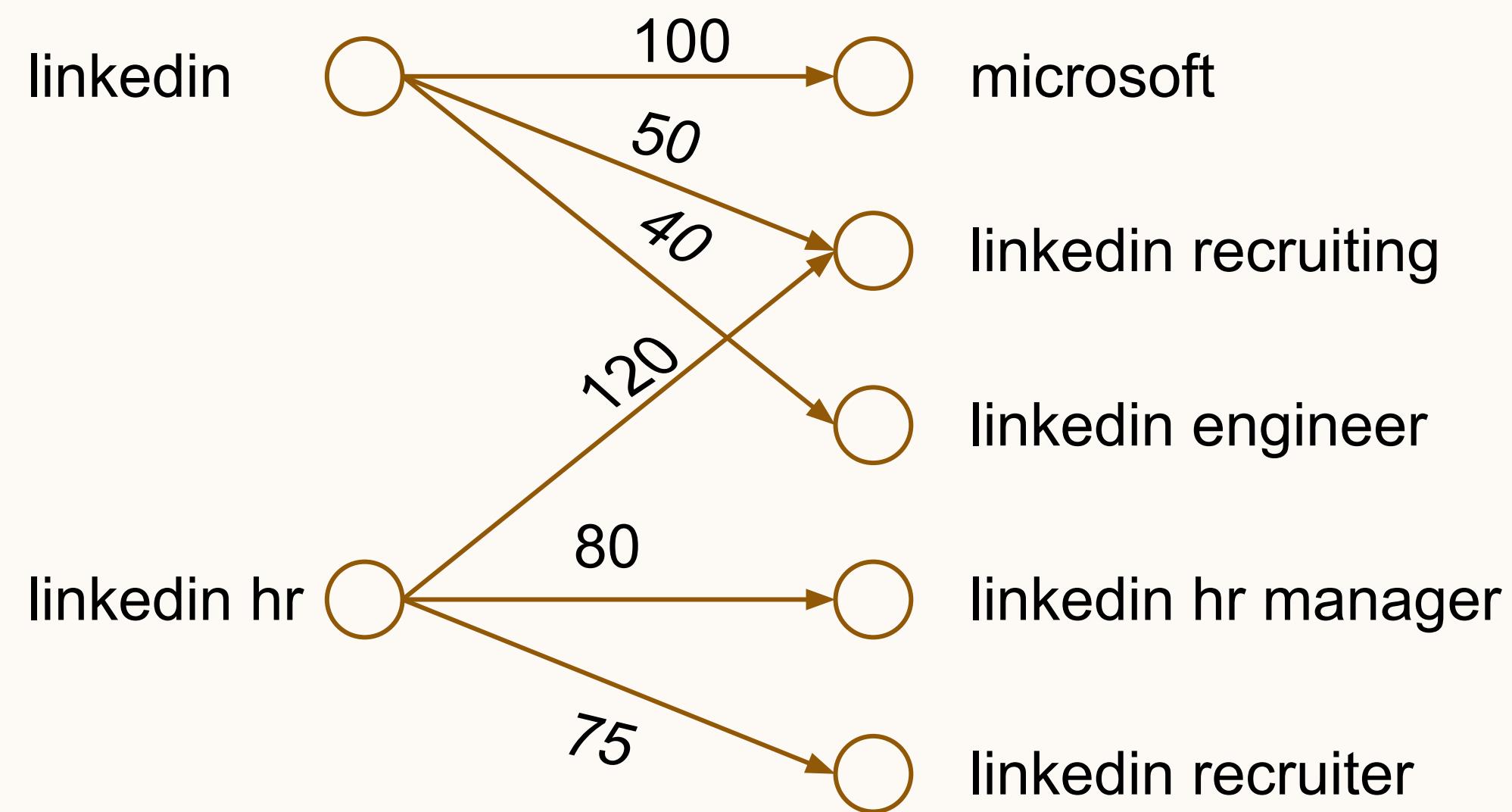
Machine Translation on documents



Sequence Generation - Traditional Methods

Based on query co-occurrence

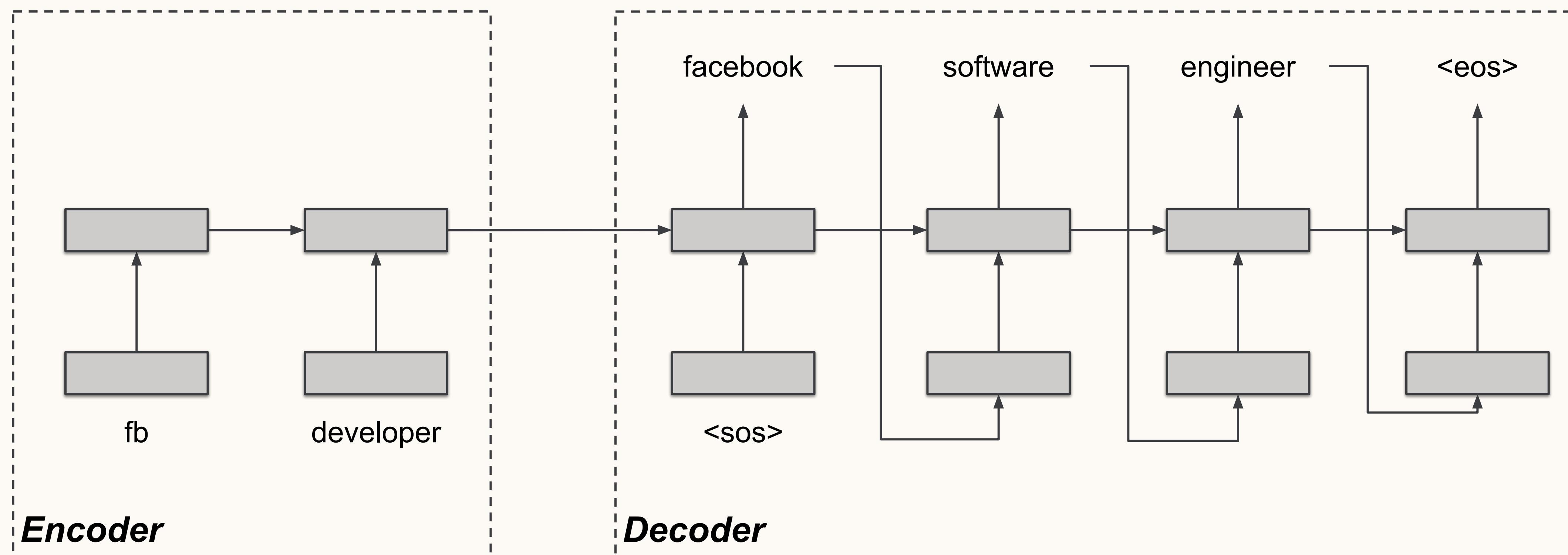
- Collaborative Filtering



- Problems
 - No word semantics understanding
 - Cannot handle new queries

Sequence Generation - Deep Learning

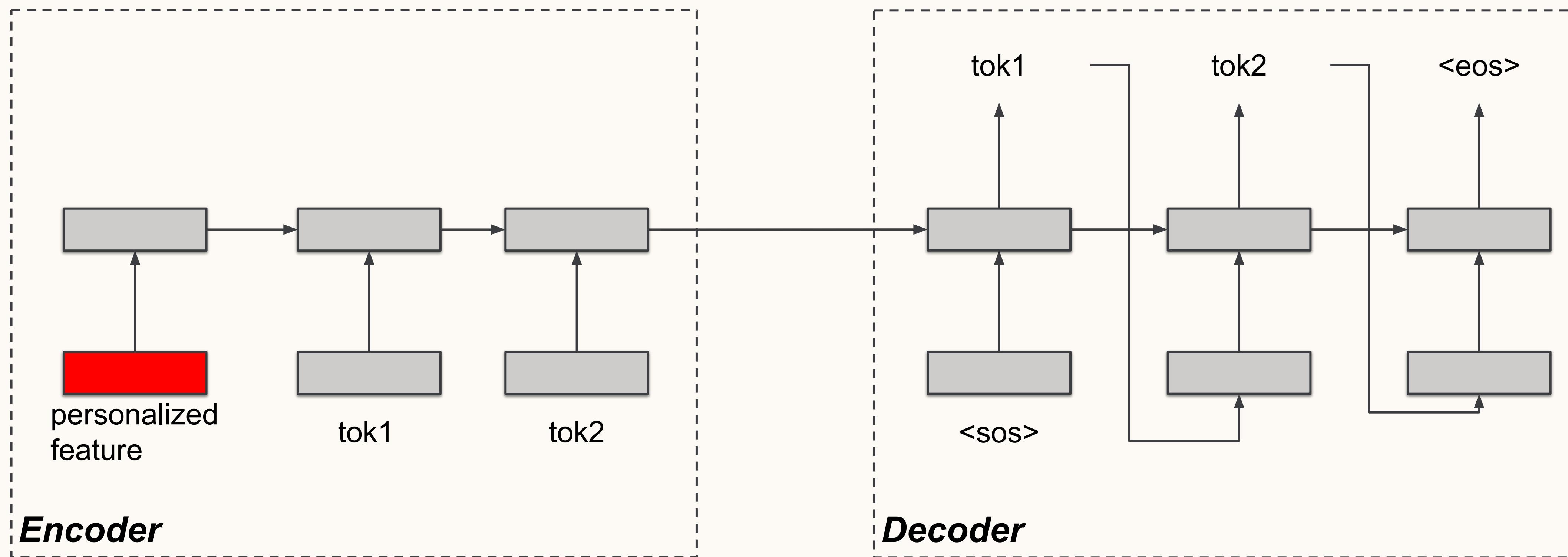
Sequence-to-sequence Framework



Sequence Generation - Deep Learning

(Zhong et al. 2020)

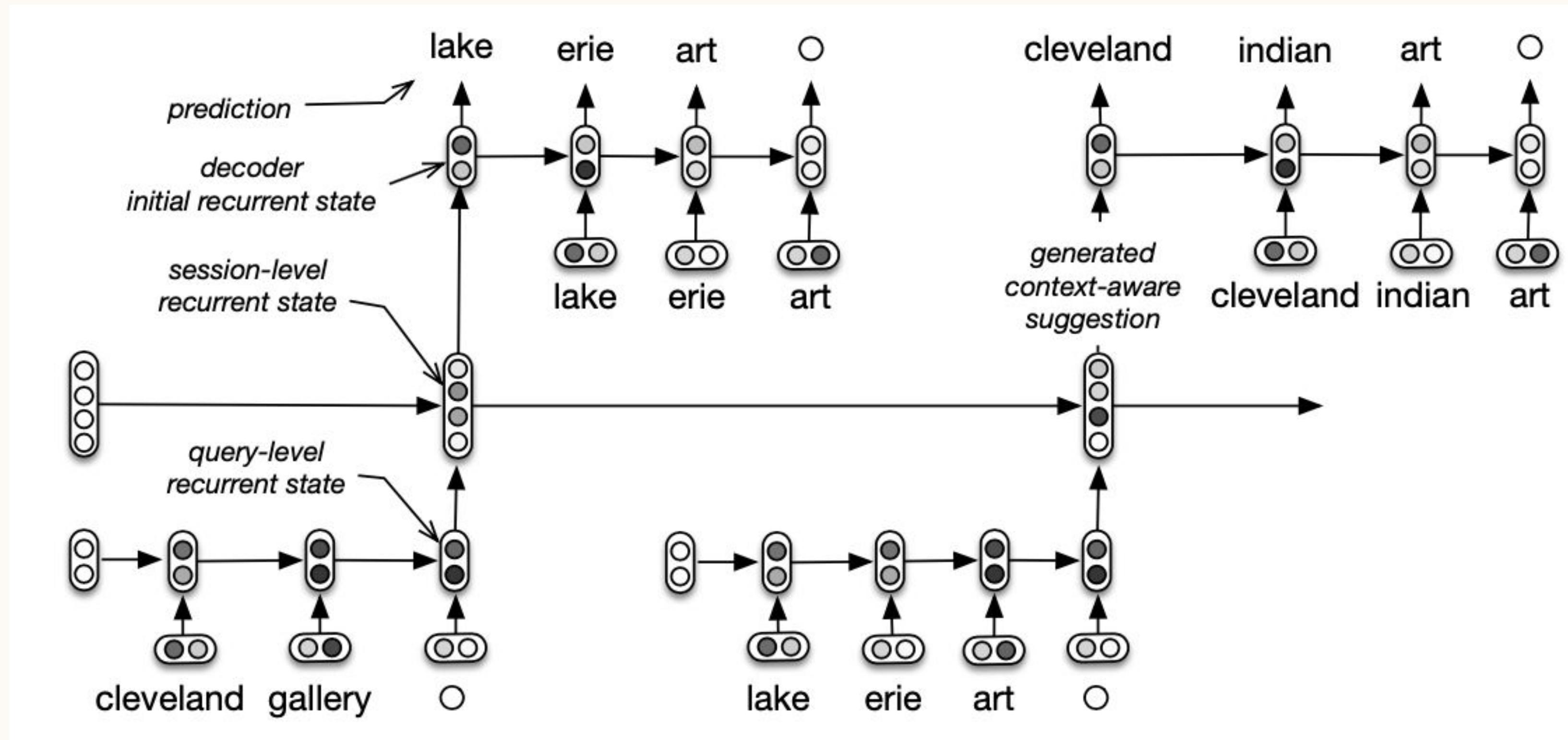
Incorporate Personalized Features



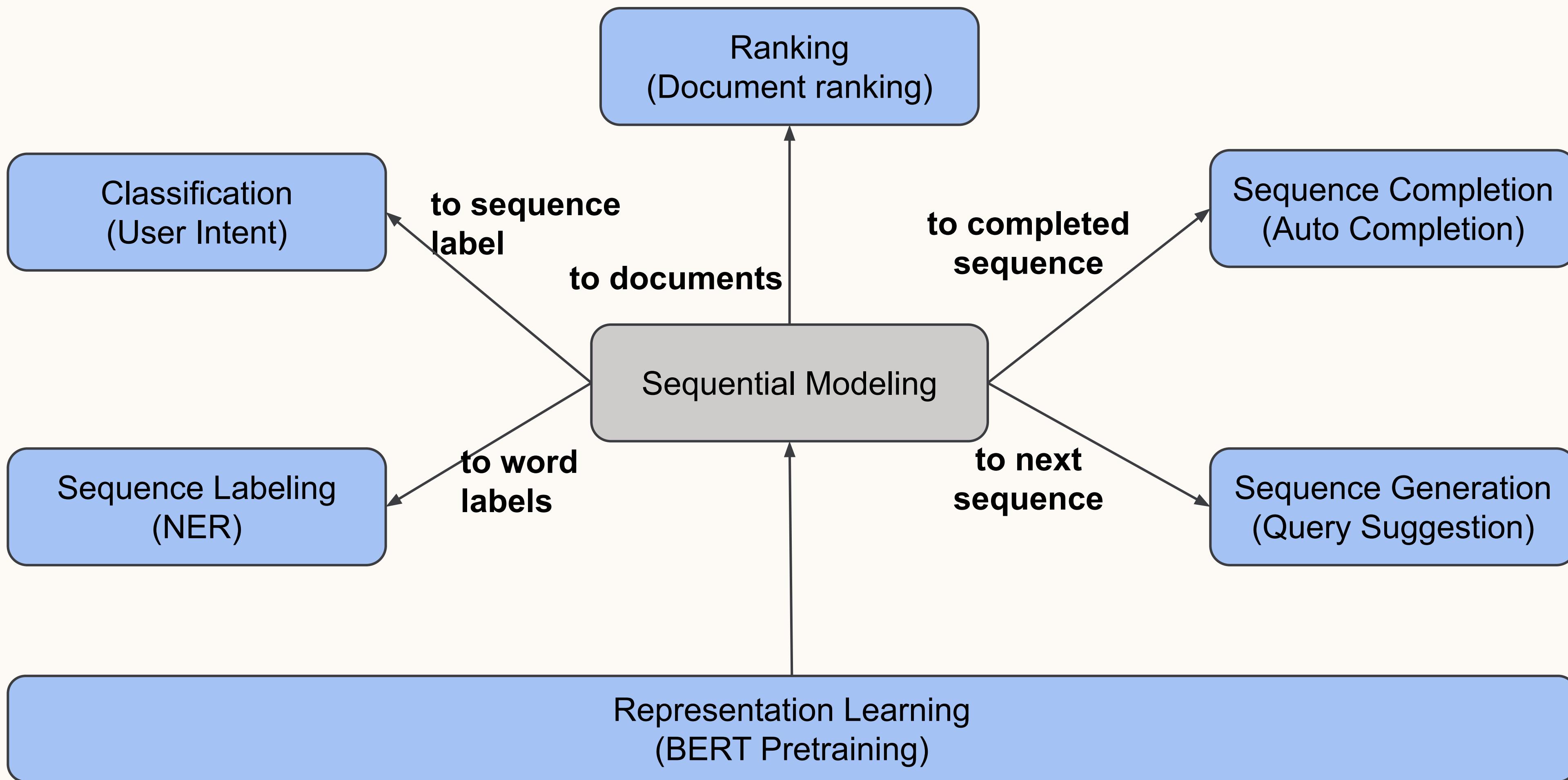
Sequence Generation - Deep Learning

Incorporate Session Data

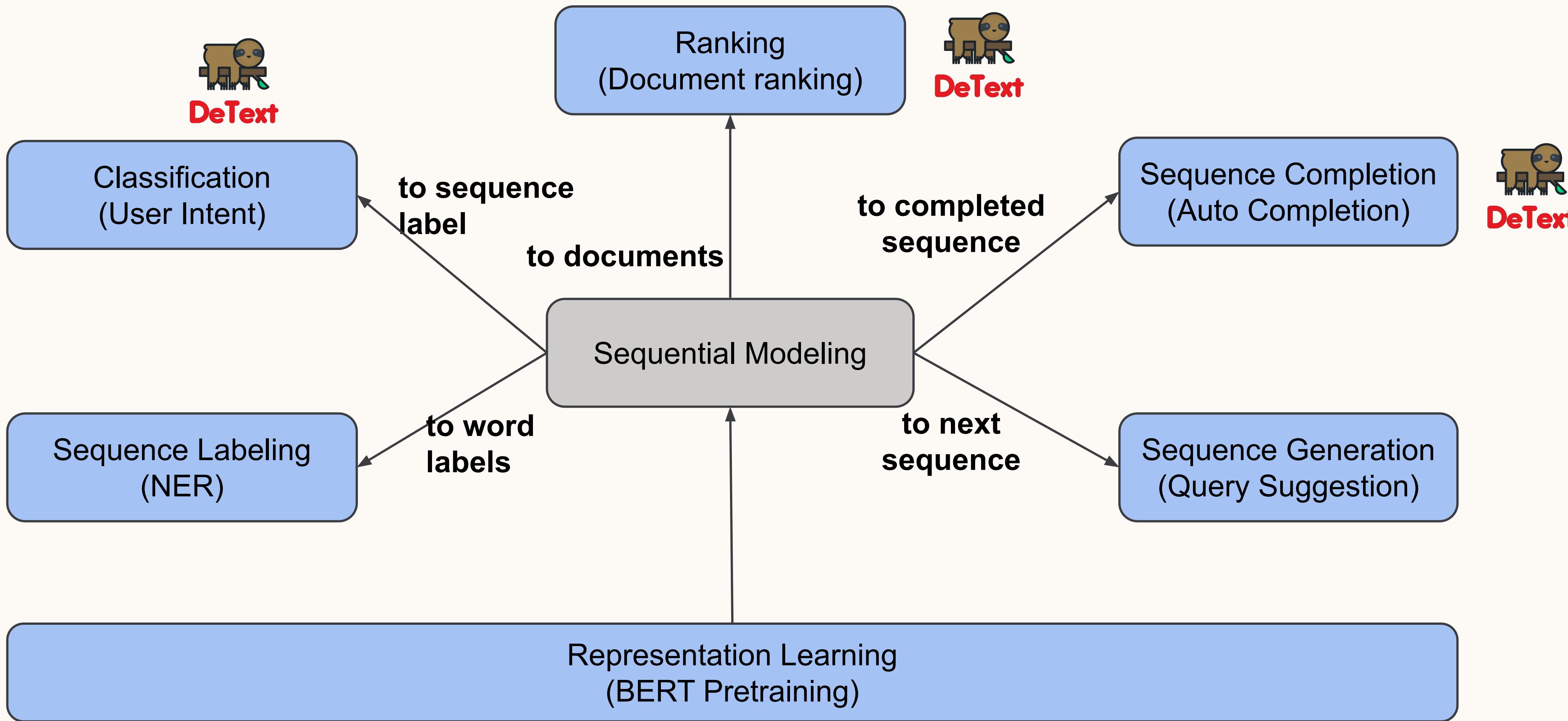
(Sordoni et al. 2015, Kazi et al. 2020)



Query and User Understanding



DeText: a deep text understanding framework



DeText: a deep text understanding framework

- Github
<https://github.com/linkedin/detext>
- LinkedIn Engineering Blog
- Publications:
 - DeText for Search Ranking
 - Query Intent
 - Query Auto-completion
 - Personalized Query Suggestion,
Seq2seq with User Feedback



Reference

- Weiwei Guo, Xiaowei Liu, Sida Wang, Huiji Gao, Ananth Sankar, Zimeng Yang, Qi Guo, Liang Zhang, Bo Long, Bee-Chung Chen, Deepak Agarwal. 2020. DeText: A Deep Text Ranking Framework with BERT. In CIKM.
- Sida Wang, Weiwei Guo, Huiji Gao, Bo Long. 2020. Efficient Neural Query Auto Completion. In CIKM.
- Liu, Xiaowei, Weiwei Guo, Huiji Gao, and Bo Long. "Deep Query Intent Understanding at Scale." arXiv preprint (2020).
- Jiangling Zhong, Weiwei Guo, Huiji Gao, Bo Long. 2020. Personalized Query Suggestion. In SIGIR.
- Michaeel Kazi, Weiwei Guo, Huiji Gao and Bo LongIncorporating User Feedback into Sequence to Sequence Model Training. In CIKM. 2020.
- Guo, Weiwei, Huiji Gao, Jun Shi, Bo Long, Liang Zhang, Bee-Chung Chen, and Deepak Agarwal. "Deep Natural Language Processing for Search and Recommender Systems." In KDD. 2019.
- Guo, Weiwei, Huiji Gao, Jun Shi, and Bo Long. "Deep Natural Language Processing for Search Systems." In SIGIR. 2019.
- Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In SIGIR.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL.
- Y. Kim. 2014. Convolutional neural networks for sentence classification. In EMNLP.
- Homa B Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In WSDM, Workshop on Query Understanding.
- Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." 2001.
- Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).
- Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354.



Query & User Understanding

Hands-on: Query Intent Classification & Query Auto Completion



Xiaowei Liu



Sida Wang



Deep Learning for Search and Recommender Systems in Practice

-

Candidate Retrieval



Huiji Gao

Candidate Retrieval

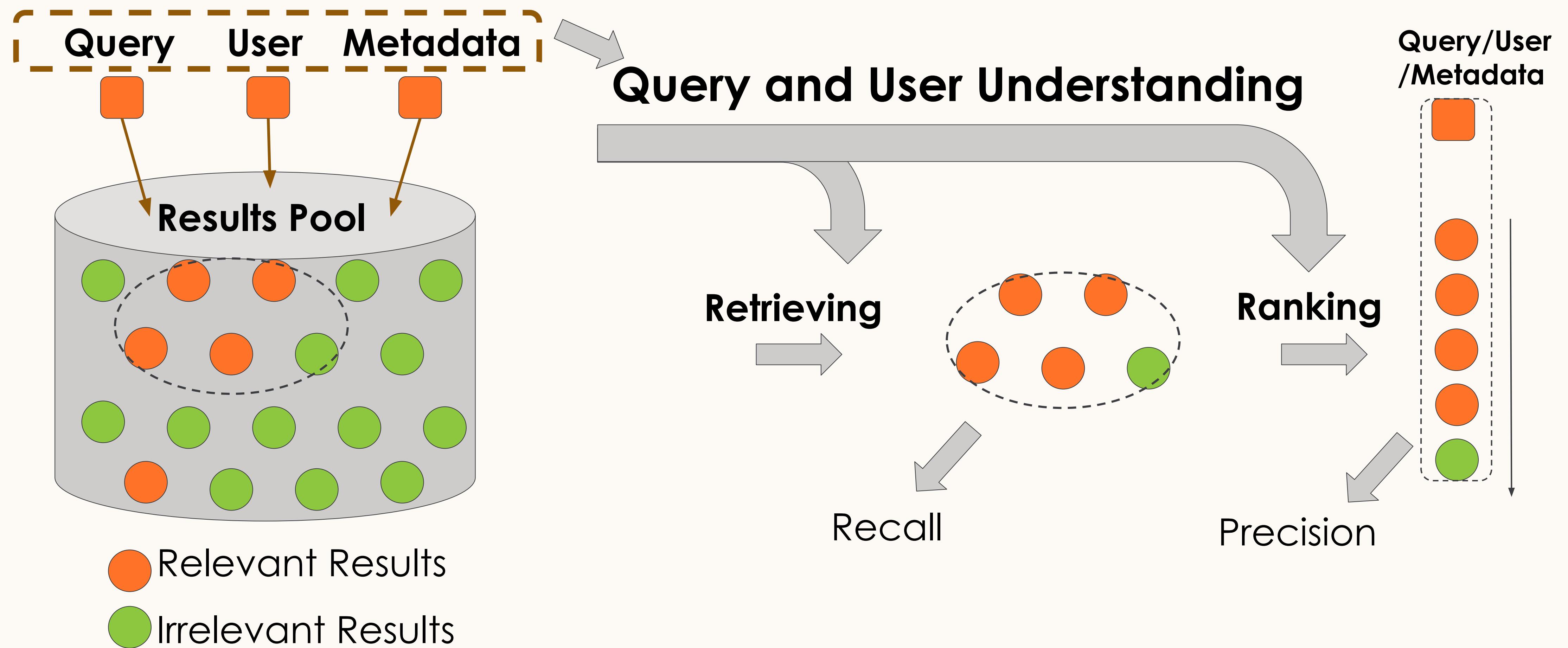
1 Candidate Retrieval in General

2 Lexical Based Retrieval

3 Embedding Based Retrieval

Candidate Retrieval

Retrieval 101



Candidate Retrieval

General “Query” in Search and Recommender Systems

- **Search Query**

Input by a user in search systems, e.g., “KDD 2020”

- **User Profile**

Key phrase from User Profiles to represent “query” in Recommender Systems, e.g., “Machine Learning Engineer” in a user’s job title

- **Metadata**

Used as filters, such as the “location” filter in search or targeting properties in ads

Candidate Retrieval

1 Candidate Retrieval in General

2 **Lexical Based Retrieval**

3 Embedding Based Retrieval

Lexical-based Retrieval

- **Keyword Based Retrieval**

Query: Mike Apple Software Engineer

Inverted Index (Boolean)

mike	Doc_1, Doc_2, Doc_4, Doc_5 , ...
apple	Doc_2, Doc_4, Doc_5 , Doc_7, ...
software	Doc_2, Doc_4, Doc_5 , Doc_11, ...
engineer	Doc_1, Doc_2, Doc_4, Doc_5 , ...
developer	Doc_2, Doc_10, Doc_11, Doc_20, ...

Return: **Doc_2, Doc_4, Doc_5**

Lexical-based Retrieval

- **Keyword Based Retrieval**

Query: Mike Apple Software Engineer

Inverted Index (Doc Weights)

mike	Doc_1:1.0, Doc_2:1.0 , Doc_4:3.0, Doc_5: 1.0, ...
apple	Doc_2:2.0 , Doc_4:3.0, Doc_5:1.0, Doc_7: 2.0, ...
software	Doc_2:1.0 , Doc_4:1.0, Doc_5:2.0, Doc_11: 1.0, ...
engineer	Doc_1:2.0, Doc_2:1.0 , Doc_4:2.0, Doc_5: 3.0, ...
developer	Doc_2:1.0, Doc_10:3.0, Doc_11:2.0, Doc_20: 1.0, ...

Return: **Doc_4:9.0 > Doc_5:7.0 > Doc_2:5.0**

Lexical-based Retrieval

- Entity Based Retrieval

- Understanding Queries with Entity Tagging

Query: Mike Apple Software Engineer

Inverted Index (Boolean)

Mike Apple Software Engineer
FN CN T T

FN:{mike}	Doc_1, Doc_2 , Doc_4 , Doc_5 , ...
CN:{apple}	Doc_2 , Doc_4 , Doc_6, Doc_7, ...
T:{software}	Doc_2 , Doc_4 , Doc_5 , Doc_11, ...
T:{engineer}	Doc_1, Doc_2 , Doc_4 , Doc_5 , ...
T:{developer}	Doc_2, Doc_10, Doc_11, ...
U: {apple}	Doc_5, Doc_10, Doc_12, ...

Return: **Doc_2**, **Doc_4**

KDD 2020

Lexical-based Retrieval

- **Keyword + Entity Based Hybrid Retrieval**

Query: Mike Apple Software Engineer KDD

Inverted Index (Boolean)

Mike Apple Software Engineer KDD
FN CN T T

FN:{mike}	Doc_1, Doc_2, Doc_4 , Doc_5, ...
CN:{apple}	Doc_2, Doc_4 , Doc_6, Doc_7, ...
T:{software}	Doc_2, Doc_4 , Doc_5, Doc_11, ...
T:{engineer}	Doc_1, Doc_2, Doc_4 , Doc_5, ...
T:{developer}	Doc_2, Doc_10, Doc_11, ...
KDD	Doc_1, Doc_2 , Doc_3, ...

Return: **Doc_2**

KDD 2020

Challenges on Lexical Matching Based Retrieval

- **Semantic Matching**

“Software Engineer” v.s. “Software Developer”

- **Language Variants**

“Baby” v.s. “Babies”, “Tooth” v.s. “Teeth”

- **Spelling Errors**

“Infarmation Retrieval”

Candidate Retrieval

- 1 Candidate Retrieval in General
- 2 Lexical Based Retrieval
- 3 Embedding Based Retrieval

Embedding Based Retrieval

• Retrieval based on Query/Document Embeddings: A Toy Example

The screenshot shows a search results page from a search engine. The query "How tall is the tower in Paris?" is entered in the search bar. Below the search bar, there are filters for "All", "Images", "Videos", "Maps", and "News". The "All" filter is selected. To the right of the search bar are icons for camera and search. Below the search bar, there is a Microsoft logo and a link to "Show results from Microsoft >". The search results section has a heading "Paris Eiffel tower height". The first result is a snippet from Reference.com stating: "The Eiffel Tower is 1,063 feet tall. It stands as France's symbol in the world and the showcase of Paris. Designed by Gustave Eiffel, the tower was built for the 1889 Exposition Universelle in celebration of the 100th anniversary of the French Revolution." Below this snippet is a small image of the Eiffel Tower and a link to "www.reference.com/geography/tall-eiffel-tower-eac8b1f1bb563ab2". There is also a "Is this answer helpful? ". Below the main result, there is a "PEOPLE ALSO ASK" section with several questions: "What is the Tower of Paris called?", "How tall is the Eiffel Tower at Kings Island?", "How tall is Paris Jackson?", and "How old is the Eiffel Tower?".

Why Embedding Based Retrieval?

Eiffel tower is **implicit** in the search query
“People also ask” provides **similar** queries / questions

- ✓ “Tower in Paris” - “Eiffel tower”
- ✓ “How tall” - “Height”

Embedding Based Retrieval

Embedding Based Retrieval

- **Why Embedding Based Retrieval**

Retrieval based on Query/Document Embeddings

- **Semantic Match**
- **Language Variants**
- **Spelling Errors**

Embedding Based Retrieval

- **What is Embedding Based Retrieval**

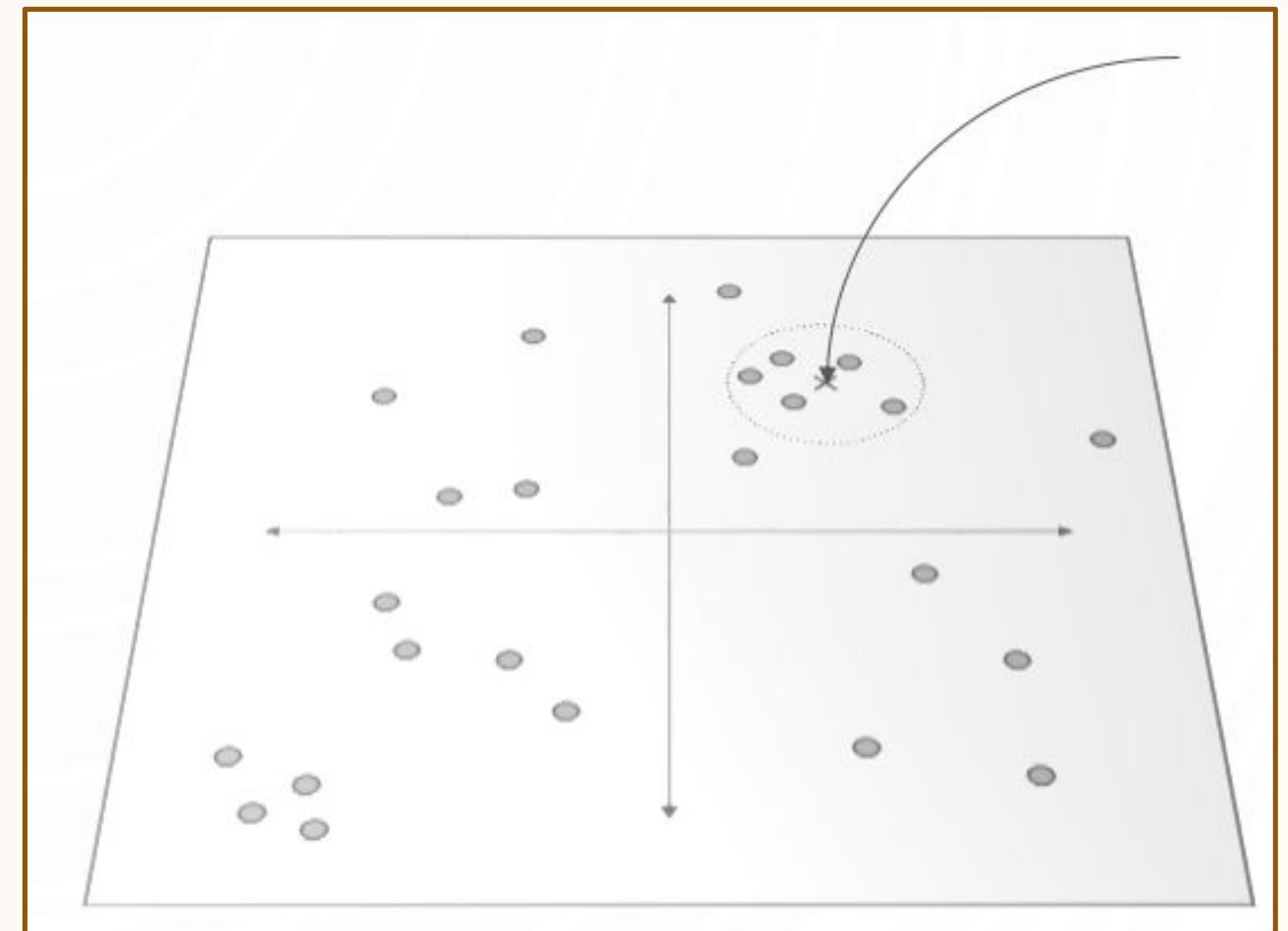
Definition:

The task to find closest points in a set to a given point.

Retrieval task: find the most relevant documents from the corpus to a query in the embedding space.

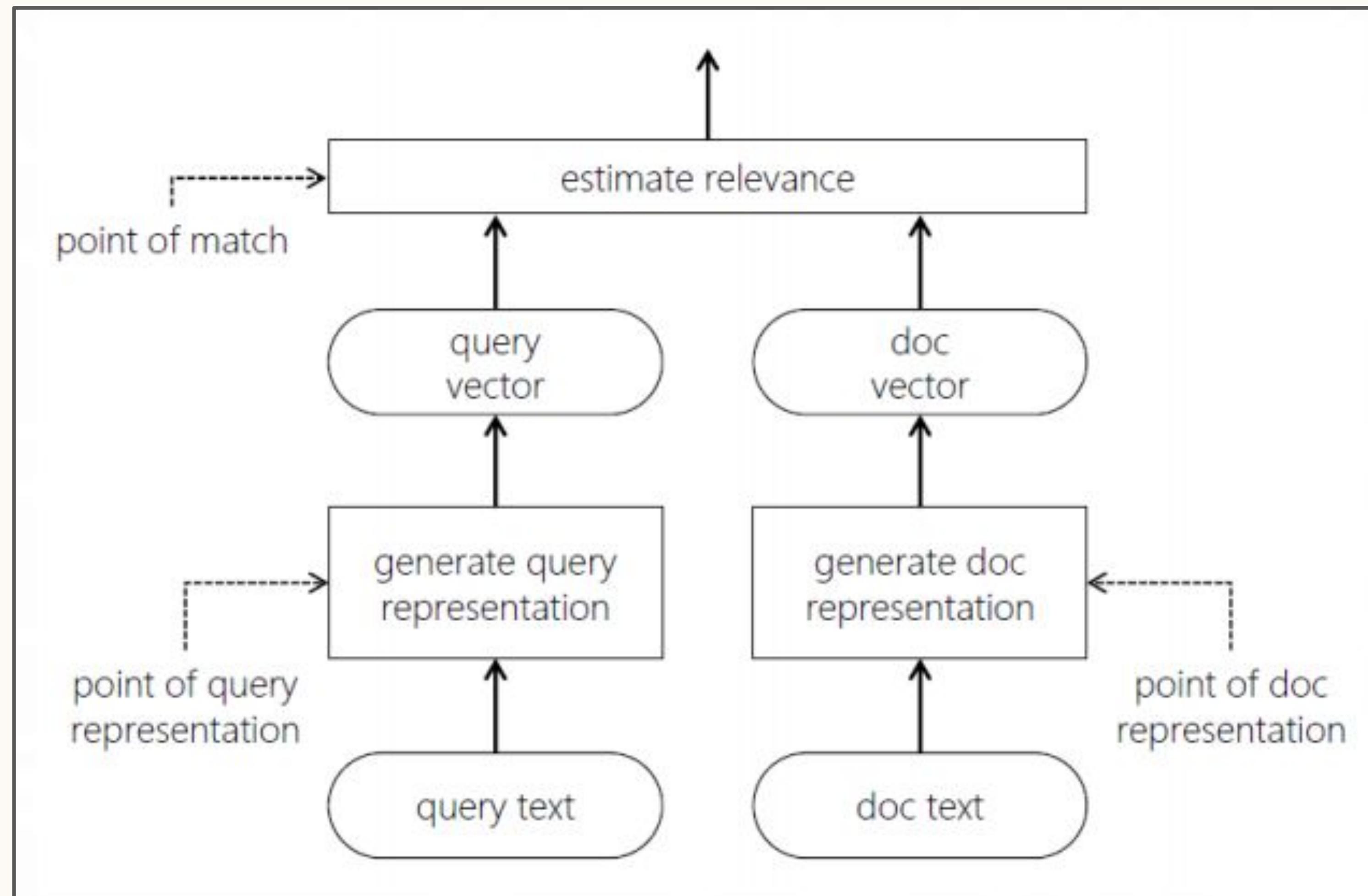
Retrieval for the concept rather than the keywords: Q&A, image, video etc.

Synonyms: vector search, semantic search, inexact search, KNN search



Embedding Based Retrieval

- Overview



Embedding Based Retrieval

- **How to**

Embedding Generation

Generate query and document embeddings under the same feature space

Index Building

Index embeddings while optimizing on effectiveness and efficiency

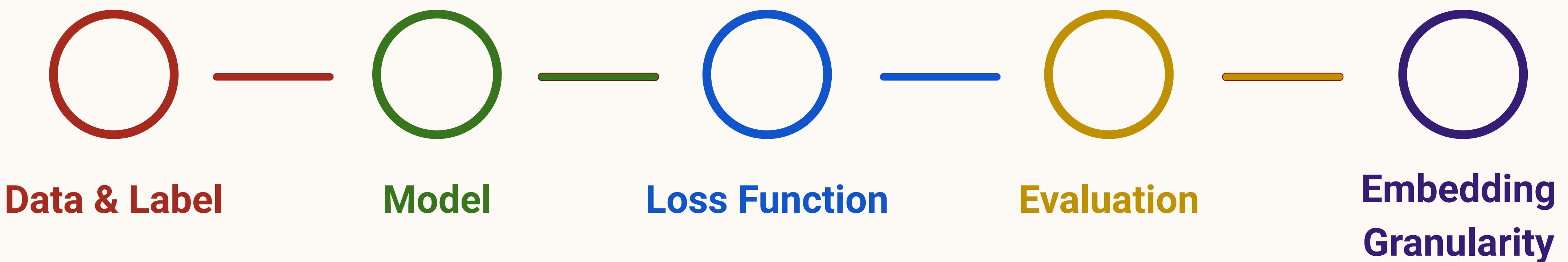
KNN Retrieval

Retrieval top K documents based on the query embedding and index

Embedding Based Retrieval

Embedding Generation

- Ensure Query and Document Embeddings are under the Same Feature Space



Embedding Based Retrieval

Embedding Generation - Data & Label

- **(Query, Doc) Pairs**
- **Labels: Positive / Negative / Hard Negative / Hard Positive**
 - Positive: Impressed and Clicked
 - Negative: Not-impressed and Not Relevant
 - Hard Negative: Impressed and Not-Clicked
 - Hard Positive: Not-impressed and Relevant

Embedding Based Retrieval

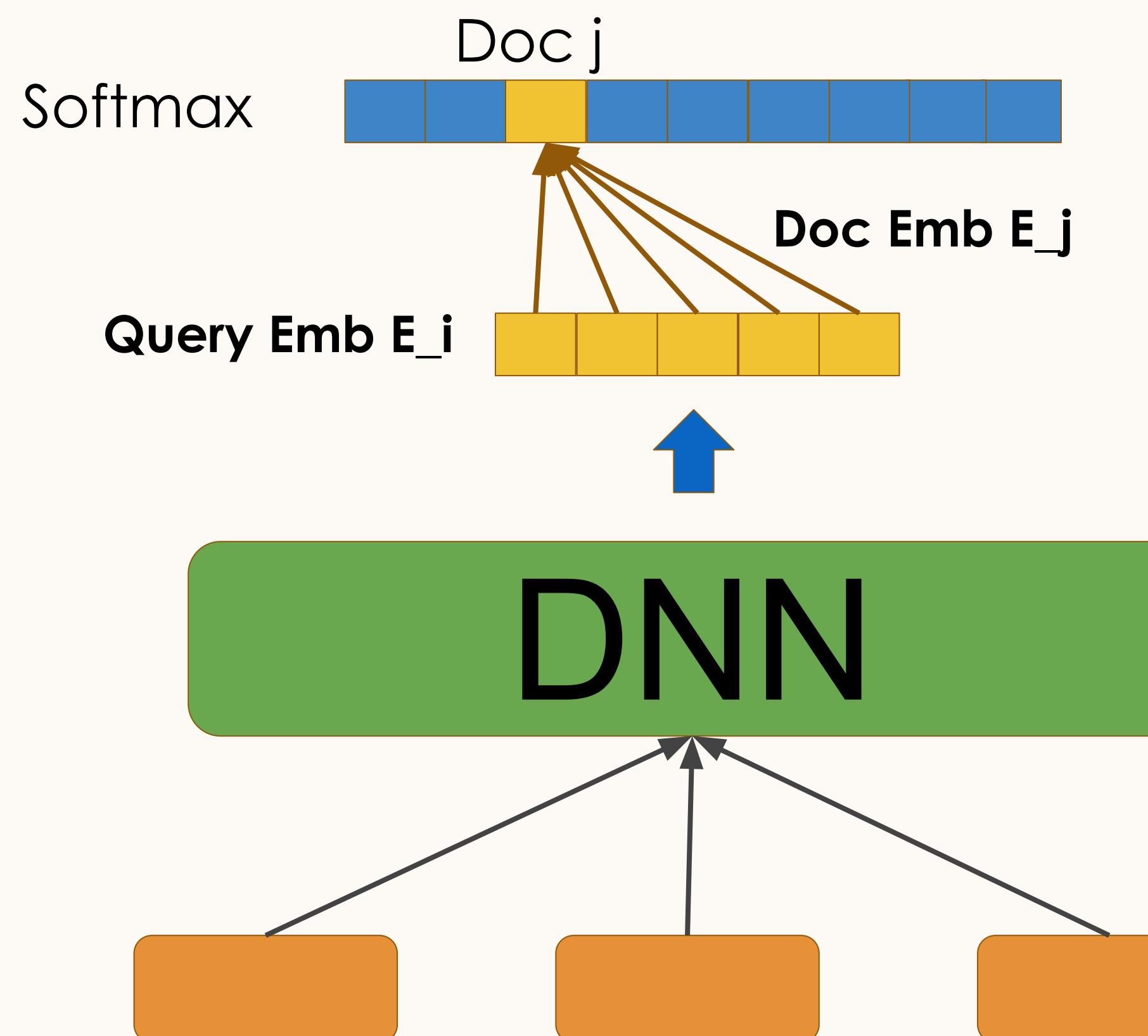
Embedding Generation - Model & Loss Function

- **Neural Networks for Embedding Generation**
 - Model the Relevant **Doc** Based on the **Query** input
 - Model Relevance Score Based on the Input from **<Query, Doc>**
- “**Query**” refers to the general concept which can be User Profiles, Historical Activities, Search Queries, etc

Embedding Based Retrieval

Embedding Generation - Model & Loss Function

• Neural Networks for Embedding Generation



Model the Relevant **Doc** based on a **Query**

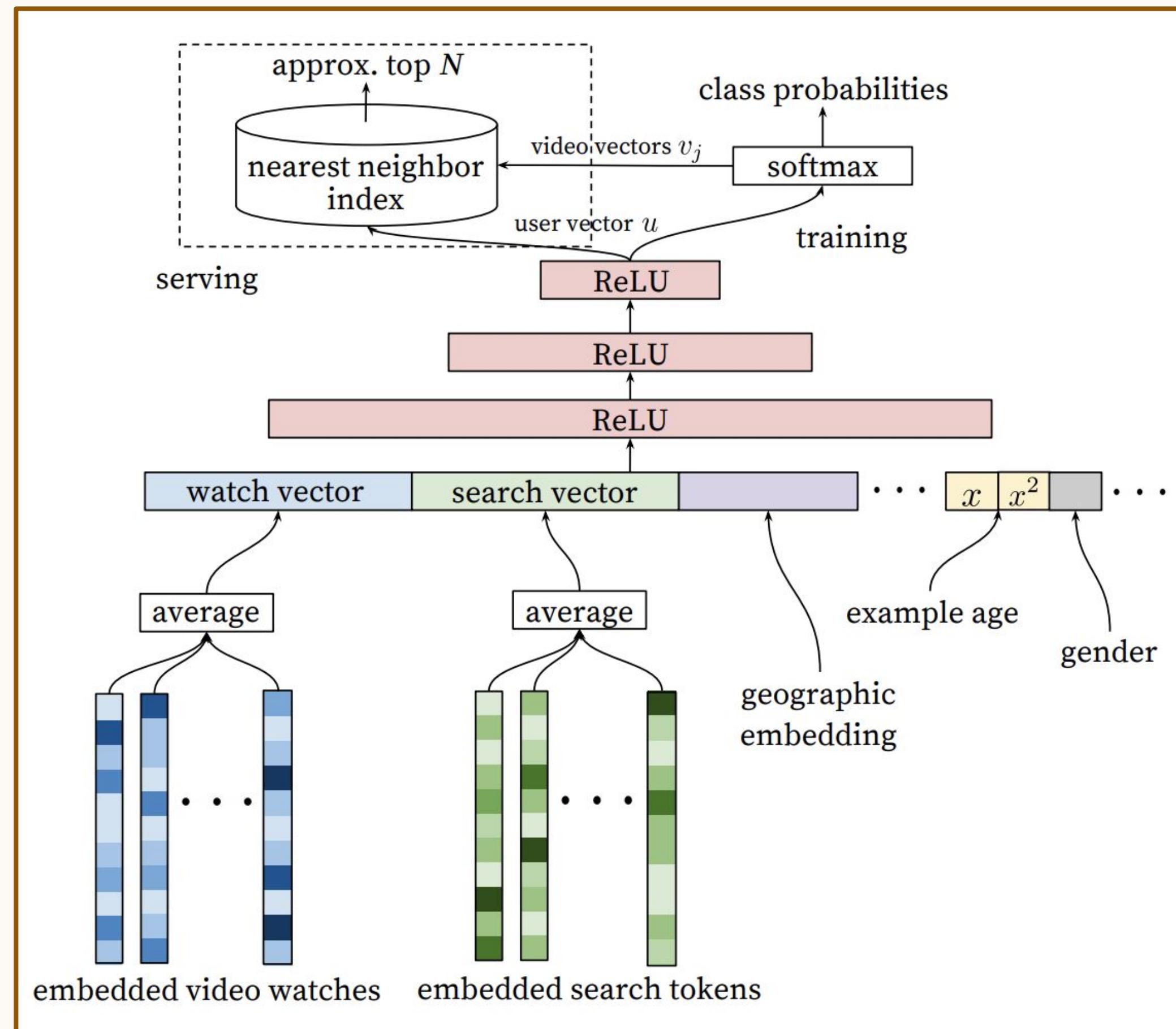
- Input: Query/User-level Data
- Target: Doc id to be clicked (or other actions) by the user given the current user status
- Generated Embeddings:
 - Query/User Embeddings & Doc Embeddings
- Loss Function:
 - Softmax on the targeted doc id

Embedding Based Retrieval

[Paul et al 2016]

Embedding Generation - Model & Loss Function

• Neural Networks for Embedding Generation



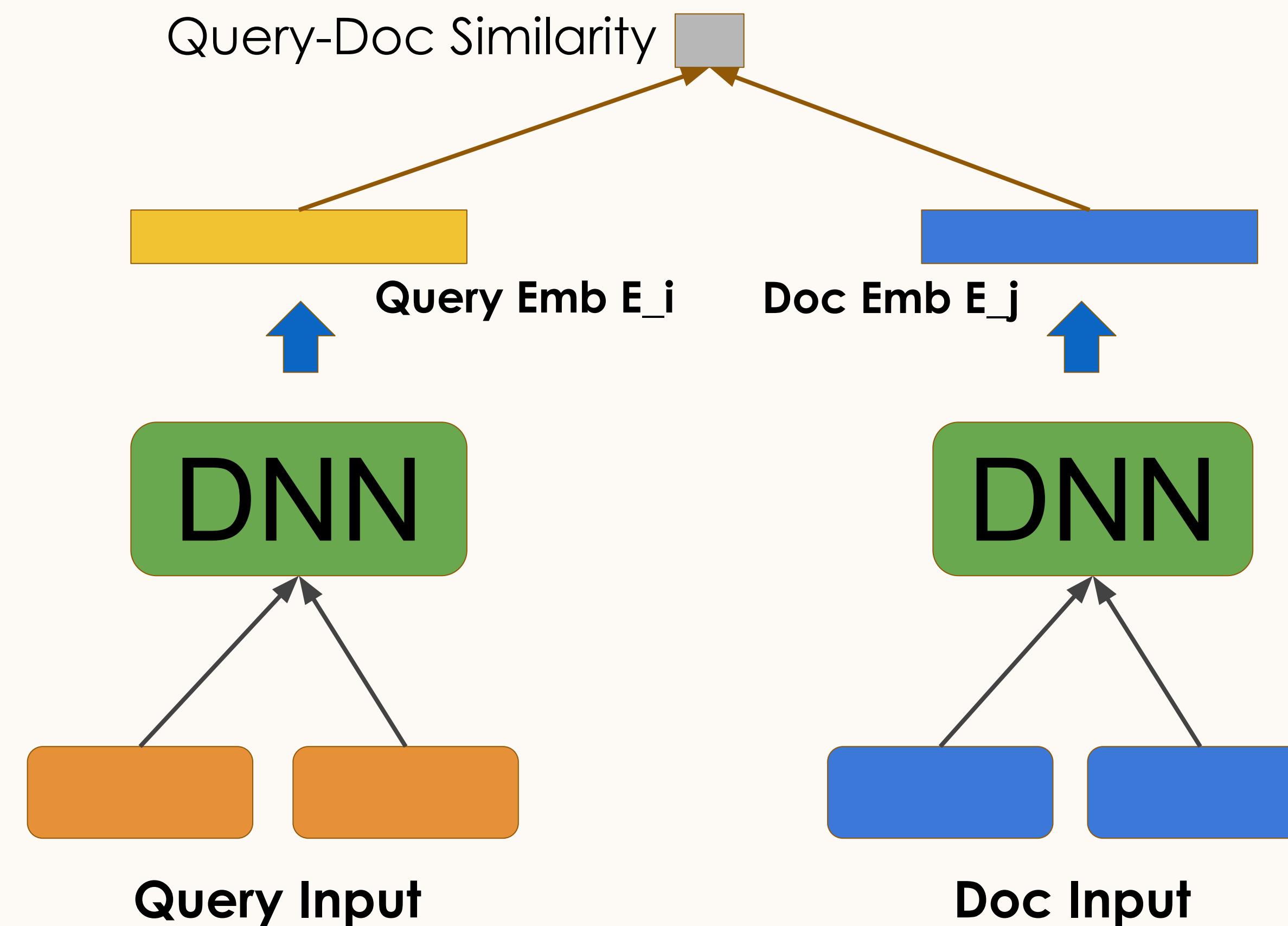
Model the Relevant **Doc** based on a **Query**

- Input: Query/User-level Data
- Target: Doc id to be clicked (or other actions) by the user given the current user status
- Generated Embeddings:
 - Query/User Embeddings & Doc Embeddings
- Loss Function:
 - Softmax on the targeted doc id

Embedding Based Retrieval

Embedding Generation - Model & Loss Function

- **Neural Networks for Embedding Generation**



Model Relevance Score based on the input **<Query, Doc>**

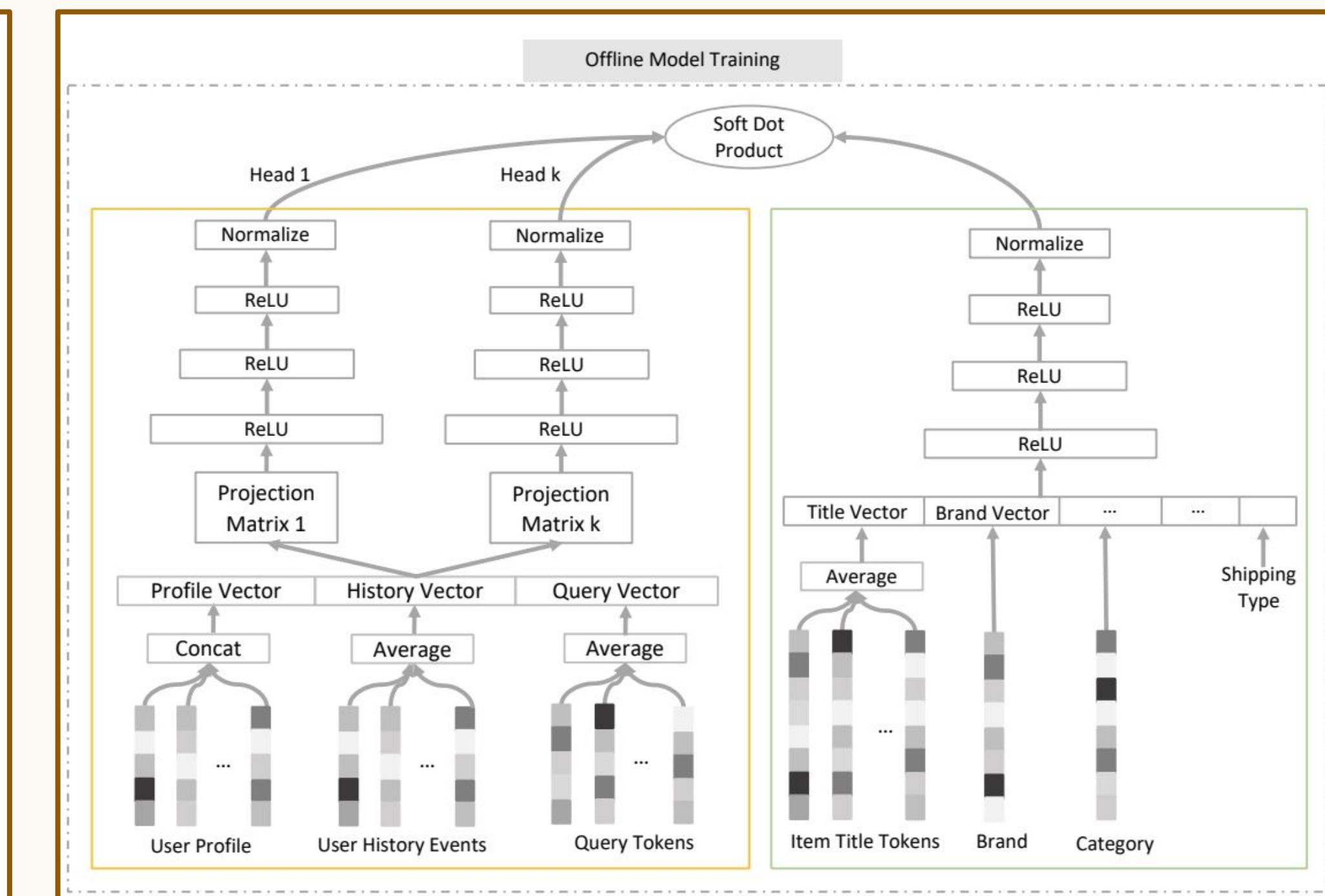
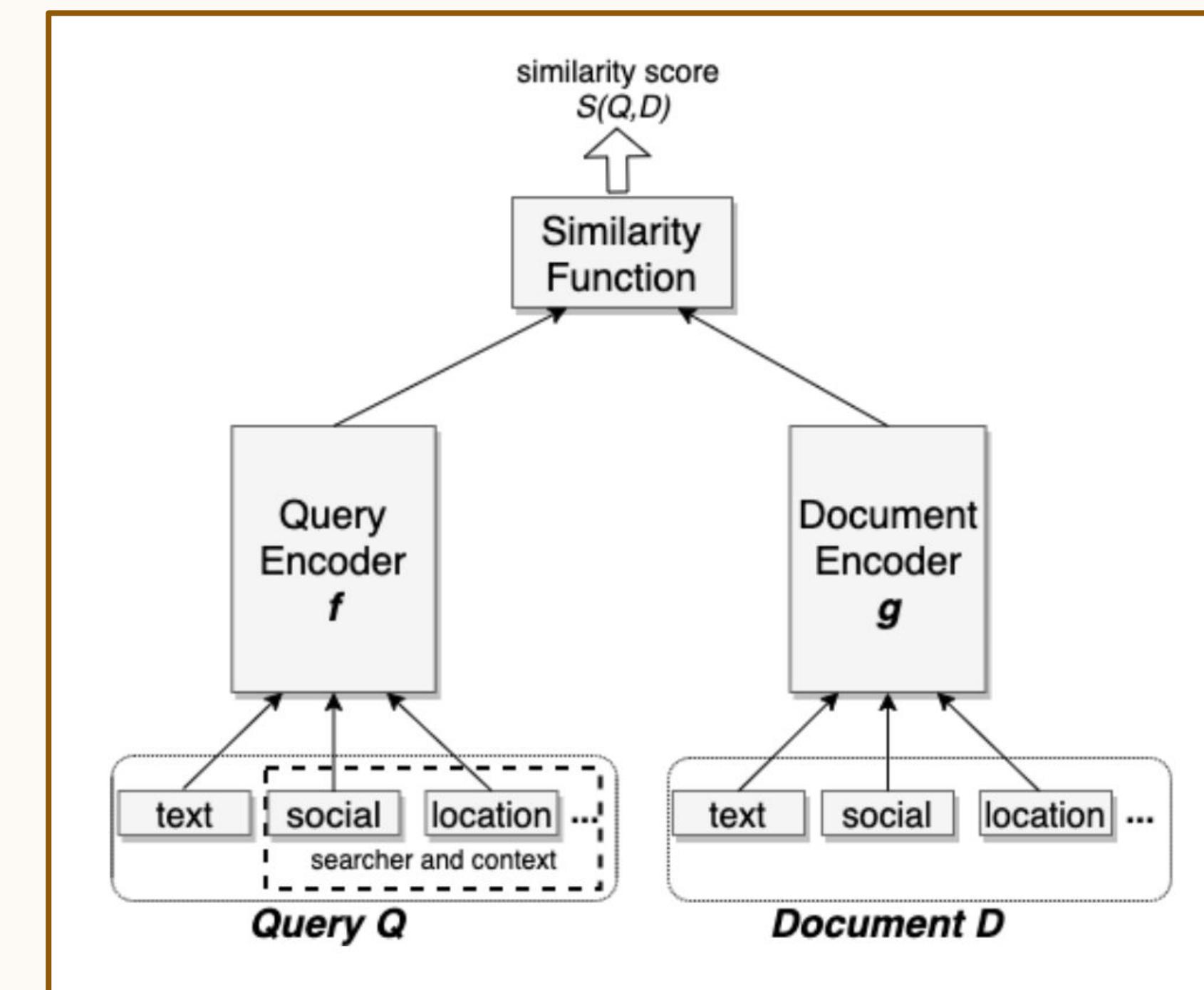
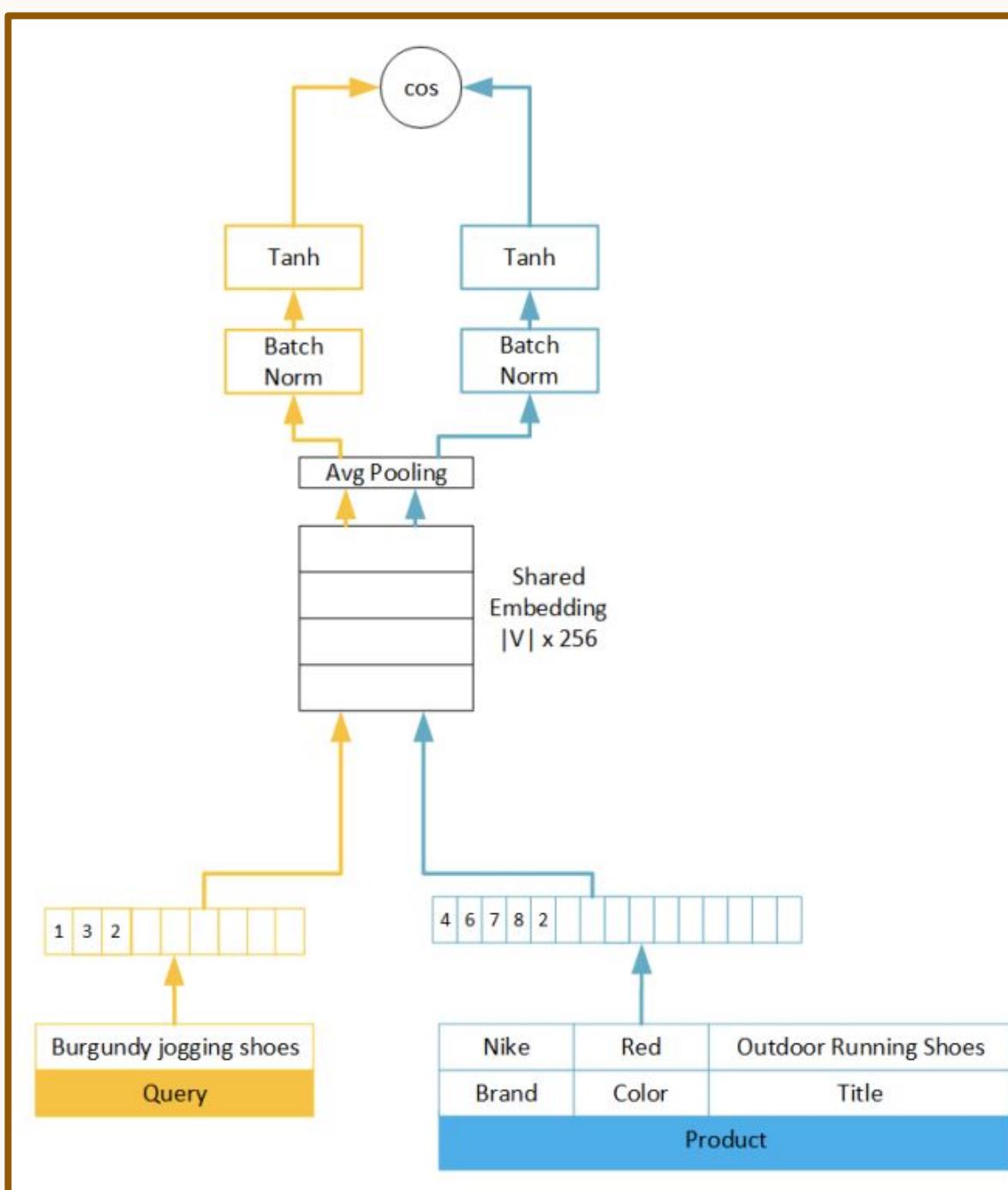
- Input: Query/User-level & Doc-level Data
- Target: Relevance Score of the **<Query, Doc>**
- Generated Embeddings:
 - Query/User Embeddings & Doc Embeddings
- Loss Function:
 - Loss: 2-part / 3-part Hinge Loss

Embedding Based Retrieval

[Priyanka et al 2019; Jui-Ting et al 2020; Han et al 2020]

Embedding Generation - Model & Loss Function

• Neural Networks for Embedding Generation



[Priyanka et al 2019]

[Jui-Ting et al 2020]

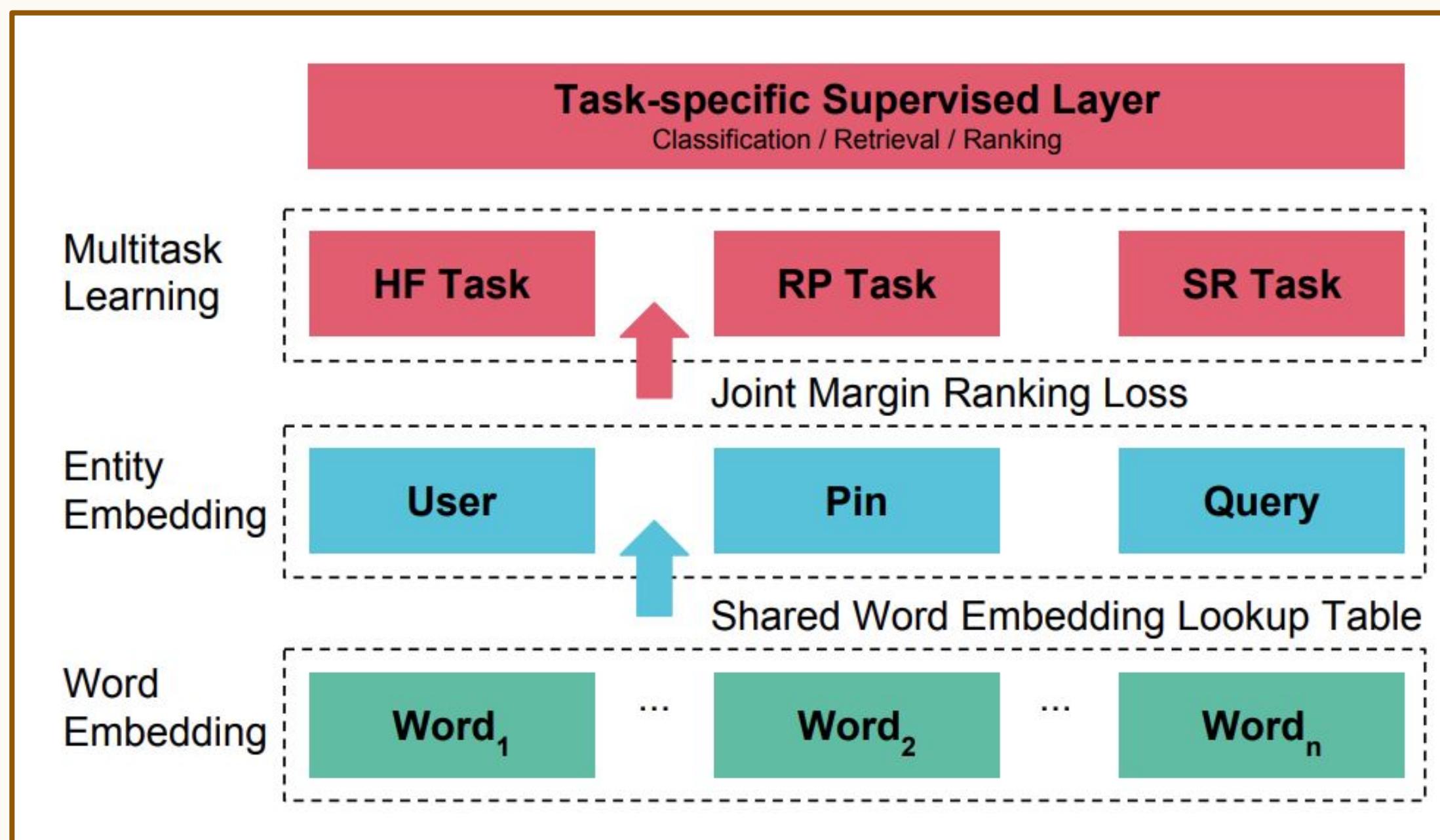
[Han et al 2020]

Embedding Based Retrieval

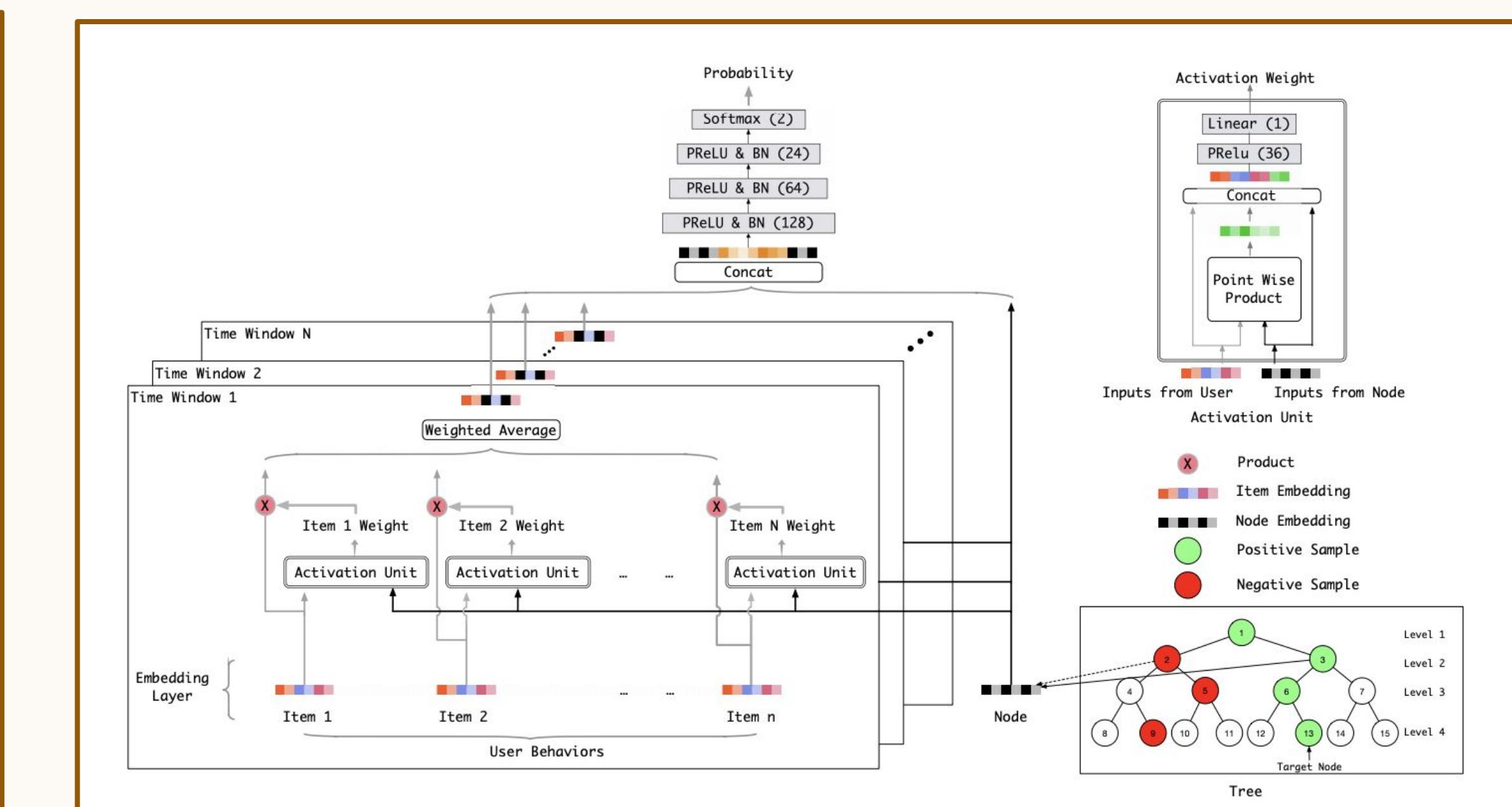
[Han et al 2018; Jinfeng et al 2019]

Embedding Generation - Model & Loss Function

• Neural Networks for Embedding Generation



[Han et al 2018]



[Jinfeng et al 2019]

Embedding Based Retrieval

Embedding Generation - Evaluation Metrics & Embedding Granularity

- Evaluate Recall@K

$$\text{recall}@K = \frac{\sum_{i=1}^K d_i \in T}{N}$$

d_i : the i-th document from top K retrieved docs

T: the target Result Set {t_1, t_2, ..., t_N}

- Embedding Granularity
 - Word level
 - Entity level
 - Sentence level
 - Document level

Embedding Based Retrieval

Index Building

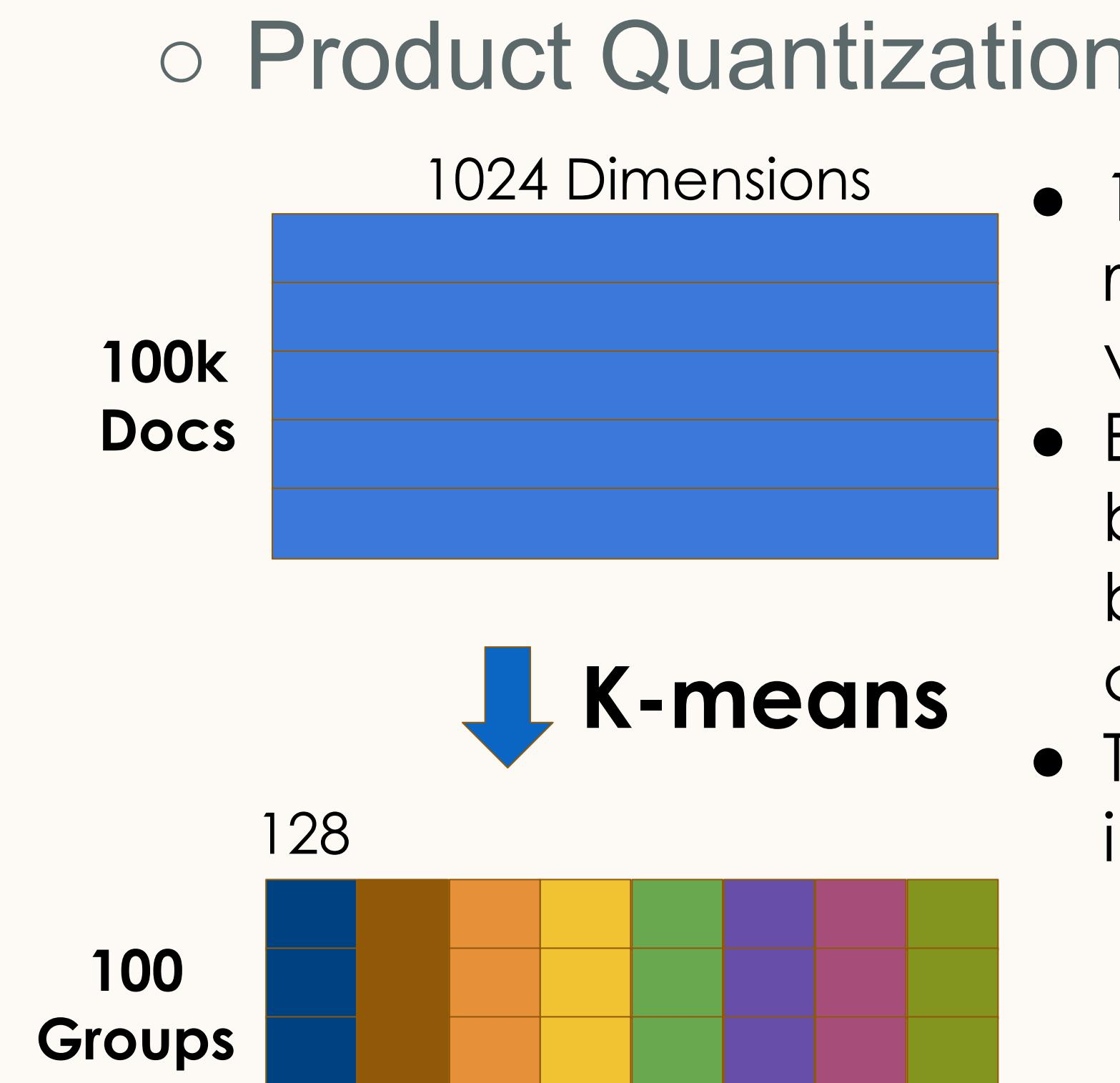
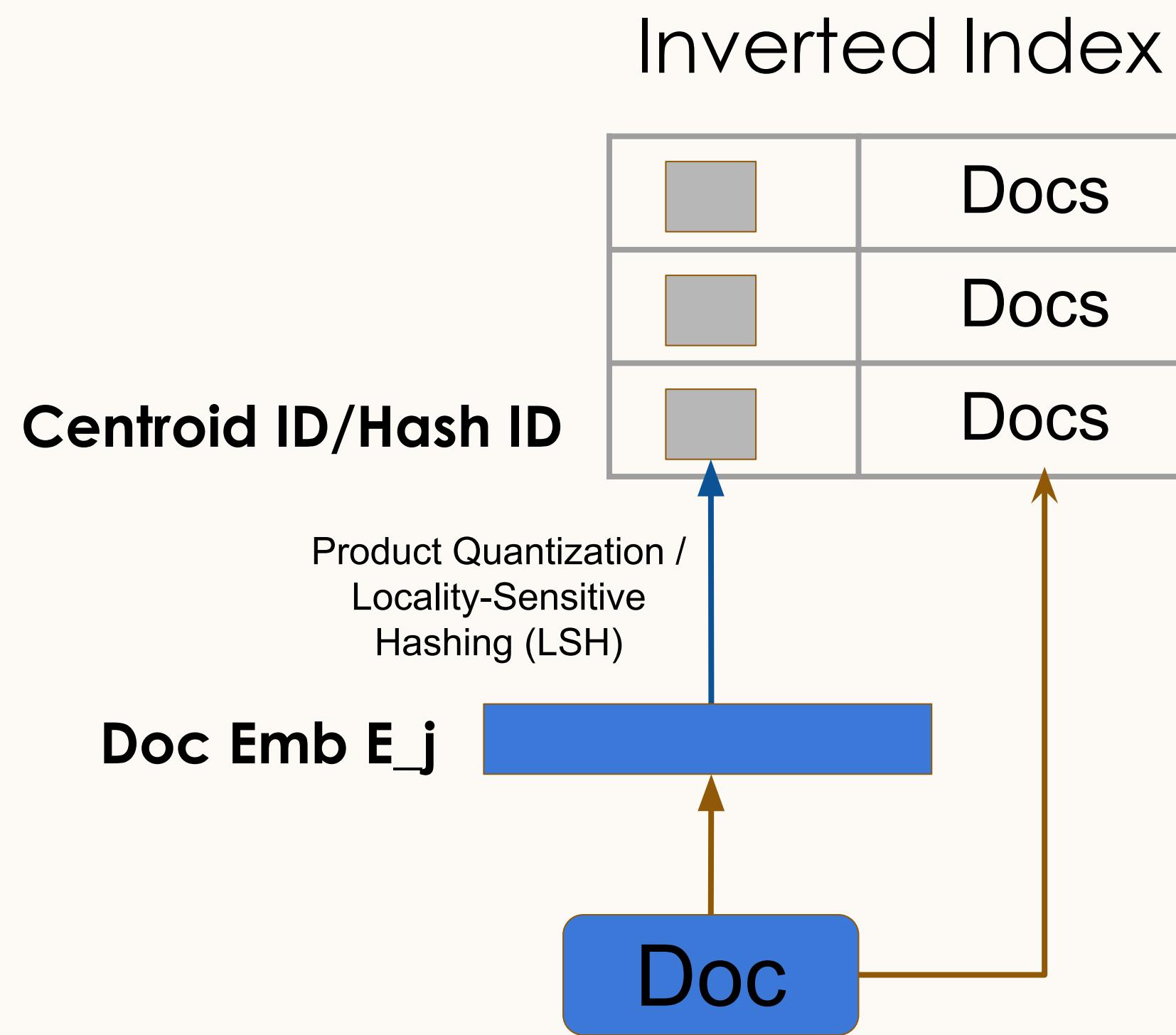
- **Compatible with Inverted Index Systems**
 - Product Quantization
 - Locality-Sensitive Hashing (LSH)
- **Non-compatible with Inverted Index Systems**
 - Hierarchical Navigable Small World (HNSW)

Embedding Based Retrieval

[Herve et al 2010; Chris et al 2017]

Index Building

- Compatible with Inverted Index Systems



- Locality-Sensitive Hashing (LSH)

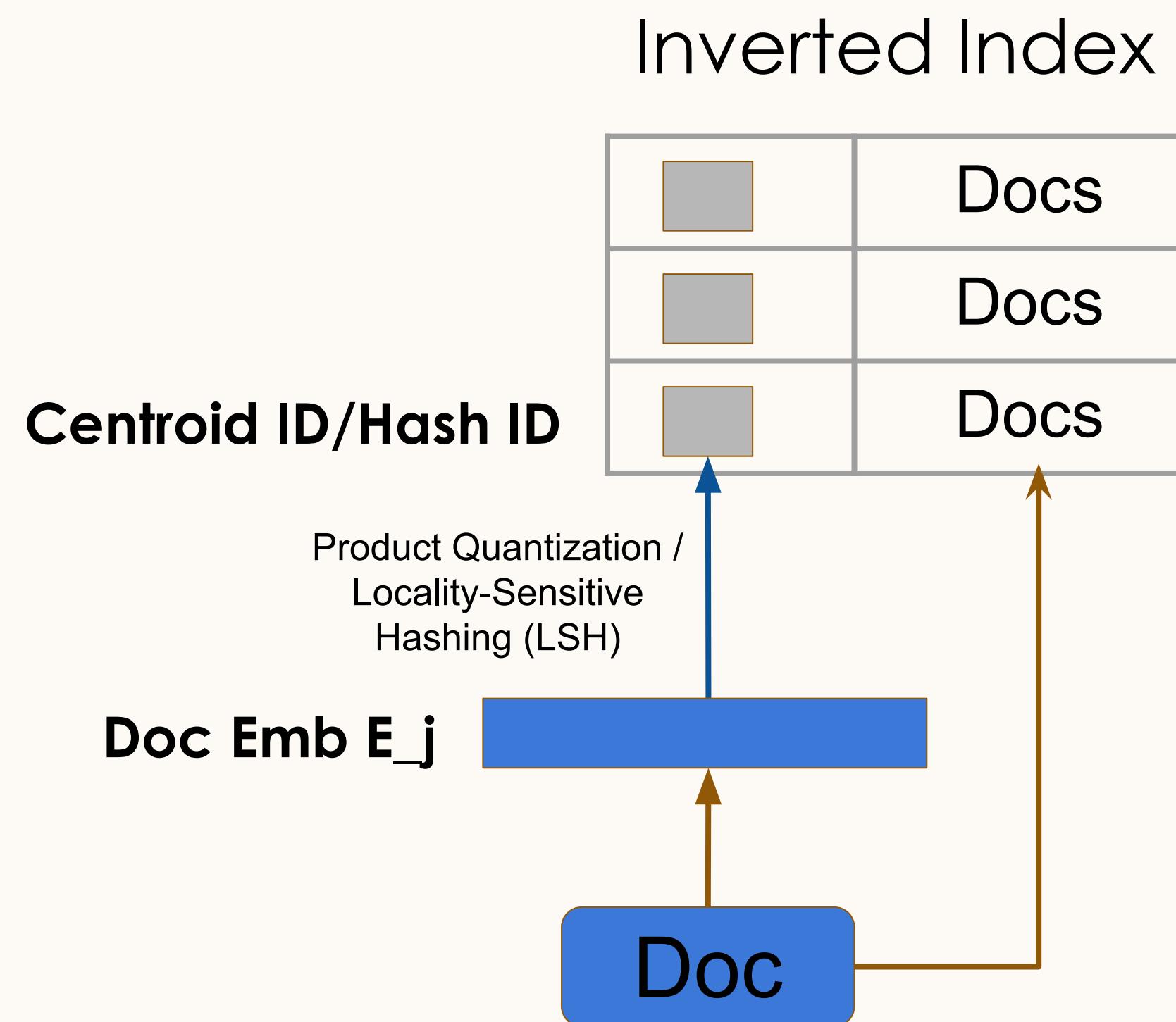
KDD 2020

- 100 x 8 centroids, each centroid is represented by a 128 dimension vector
- Each embedding can be represented by 8 centroid embedding vectors, based on its distance to each centroid
- The centroids can be used as keys in inverted index

Embedding Based Retrieval

Index Building

- **Compatible with Inverted Index Systems**

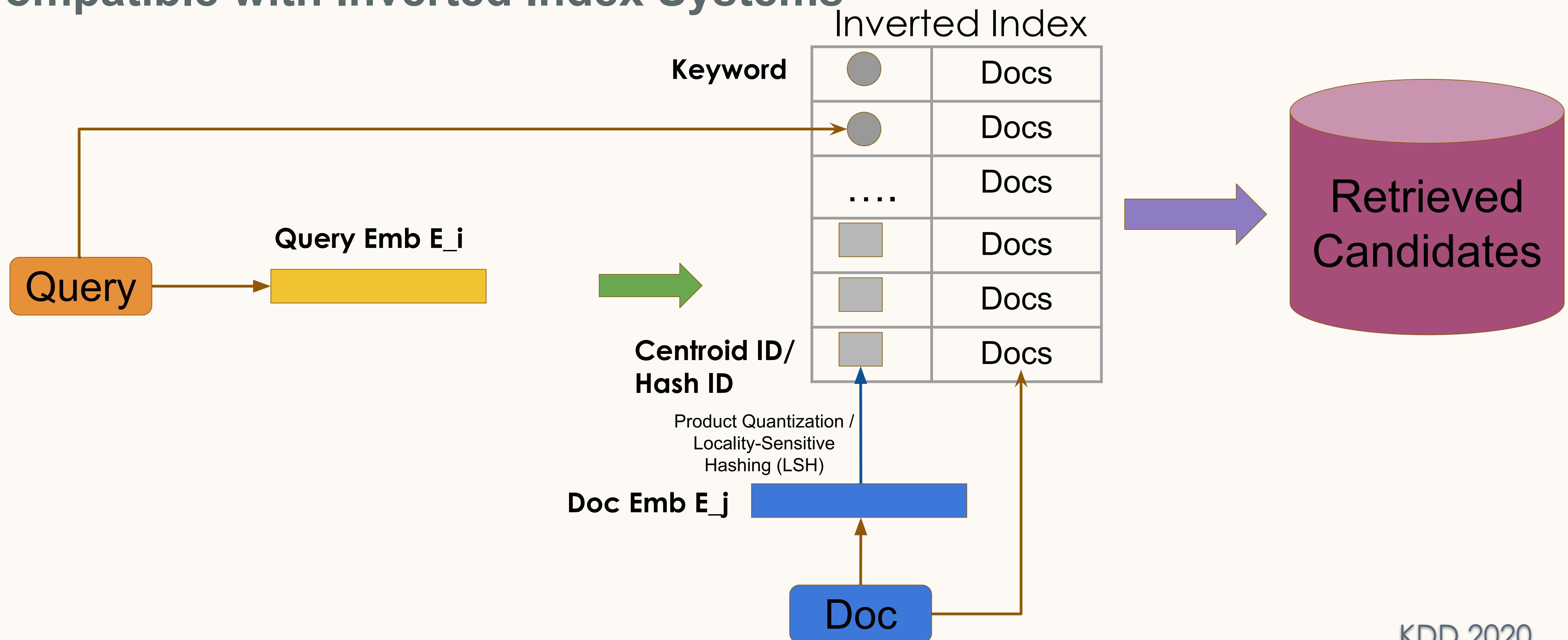


- Product Quantization
- Locality-Sensitive Hashing (LSH)
 - Hash functions are specifically designed so that hash value collisions are *more likely* for two input values that are *close together* than for inputs that are *far apart*. Therefore, similar docs have a high probability to map to the same hash id.

Embedding Based Retrieval

KNN Retrieval

- Compatible with Inverted Index Systems

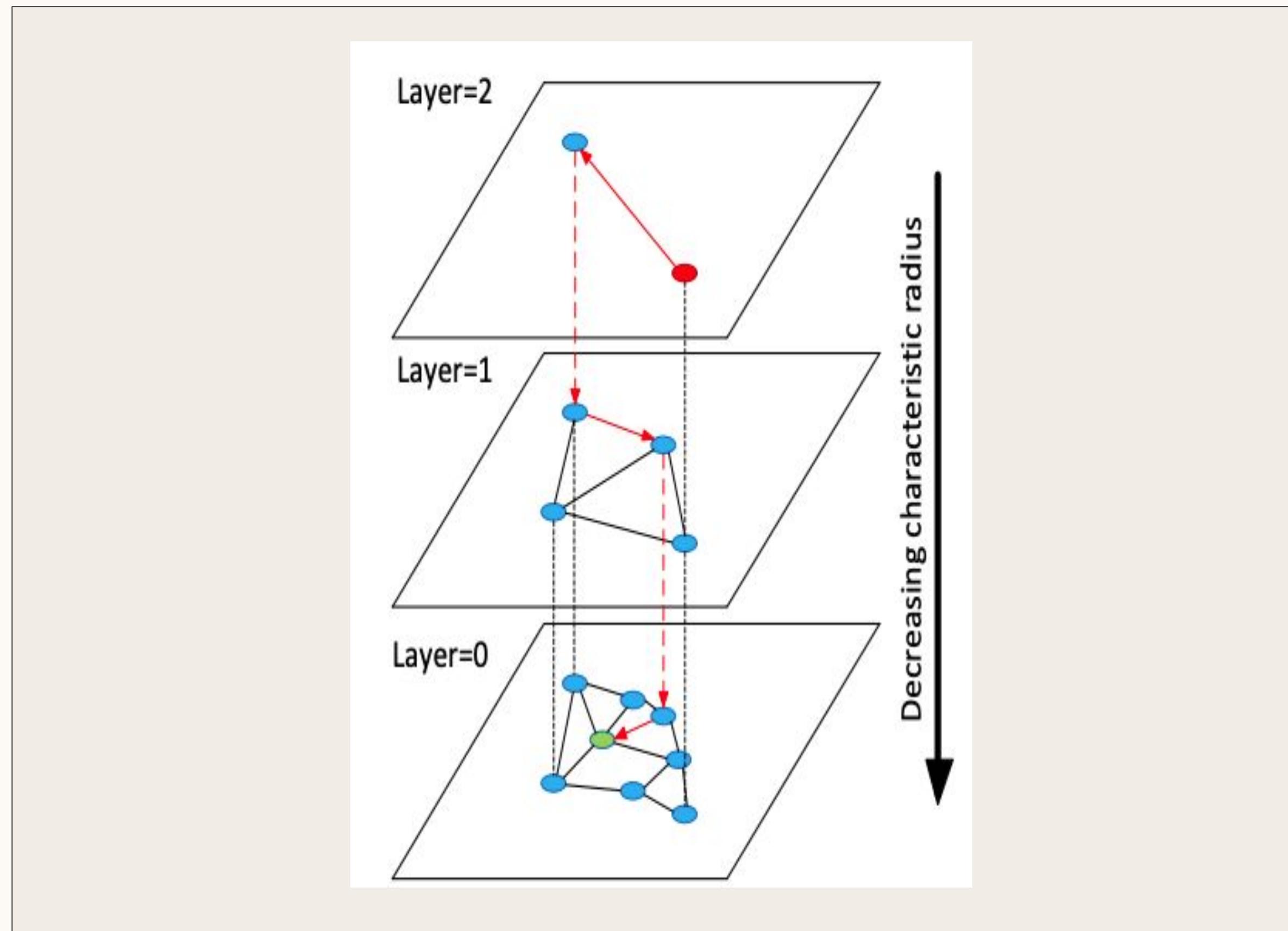


Embedding Based Retrieval

(Yu et al 2016; Qi et al 2018)

Index Building

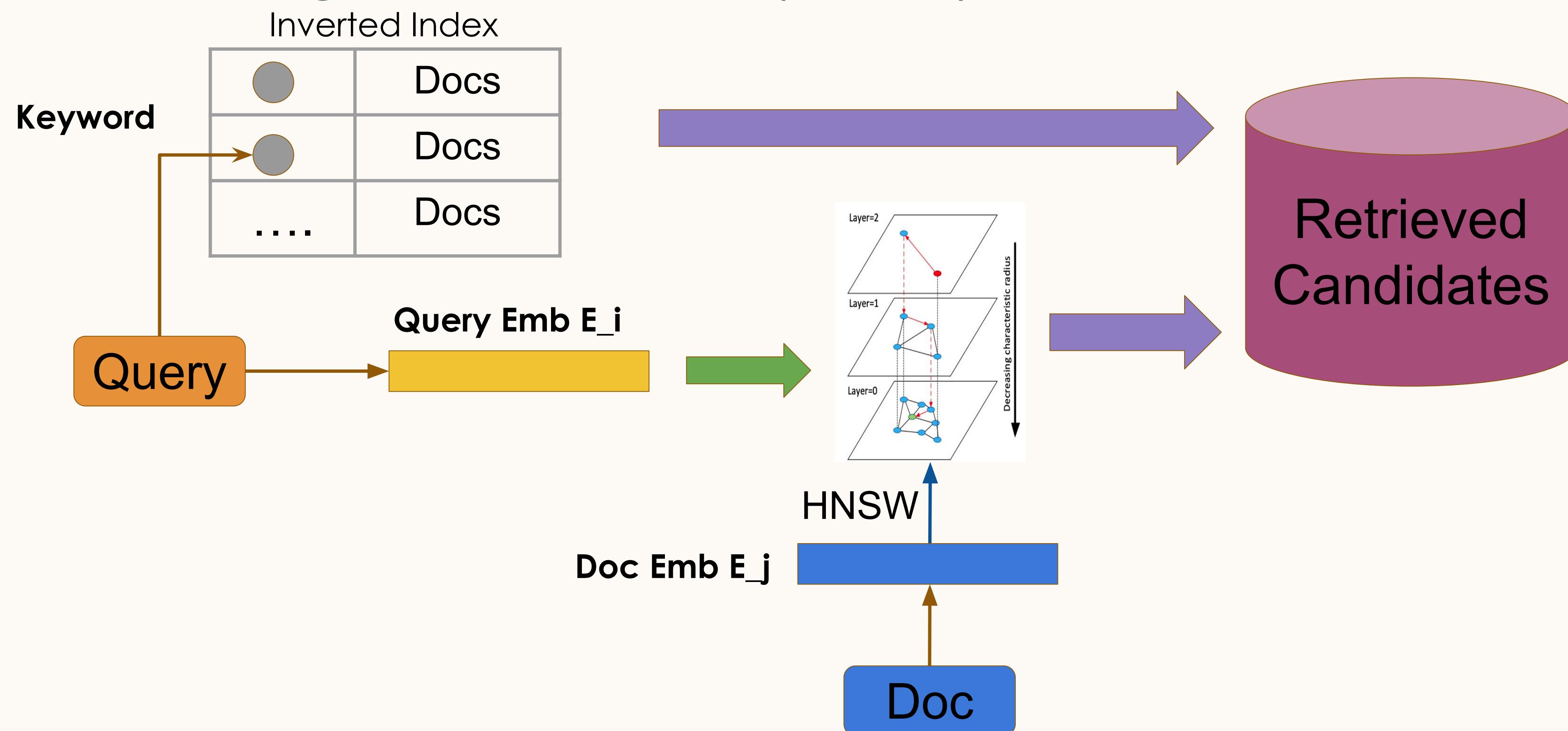
- Non-compatible with Inverted Index Systems
 - Hierarchical Navigable Small World (HNSW)



Embedding Based Retrieval

KNN Retrieval

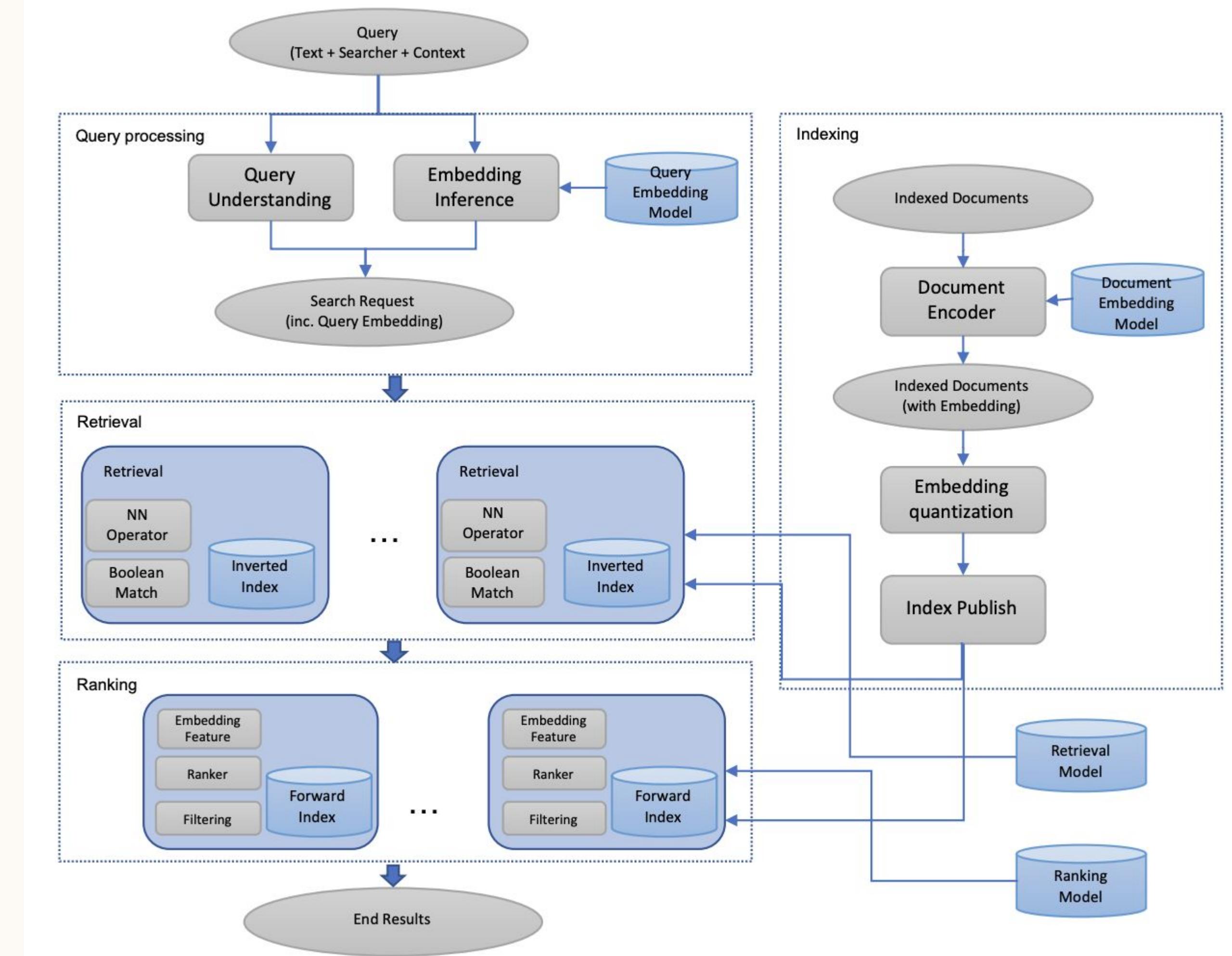
- Non-compatible with Inverted Index Systems
 - Hierarchical Navigable Small World (HNSW)



Embedding Based Retrieval

The end-to-end System

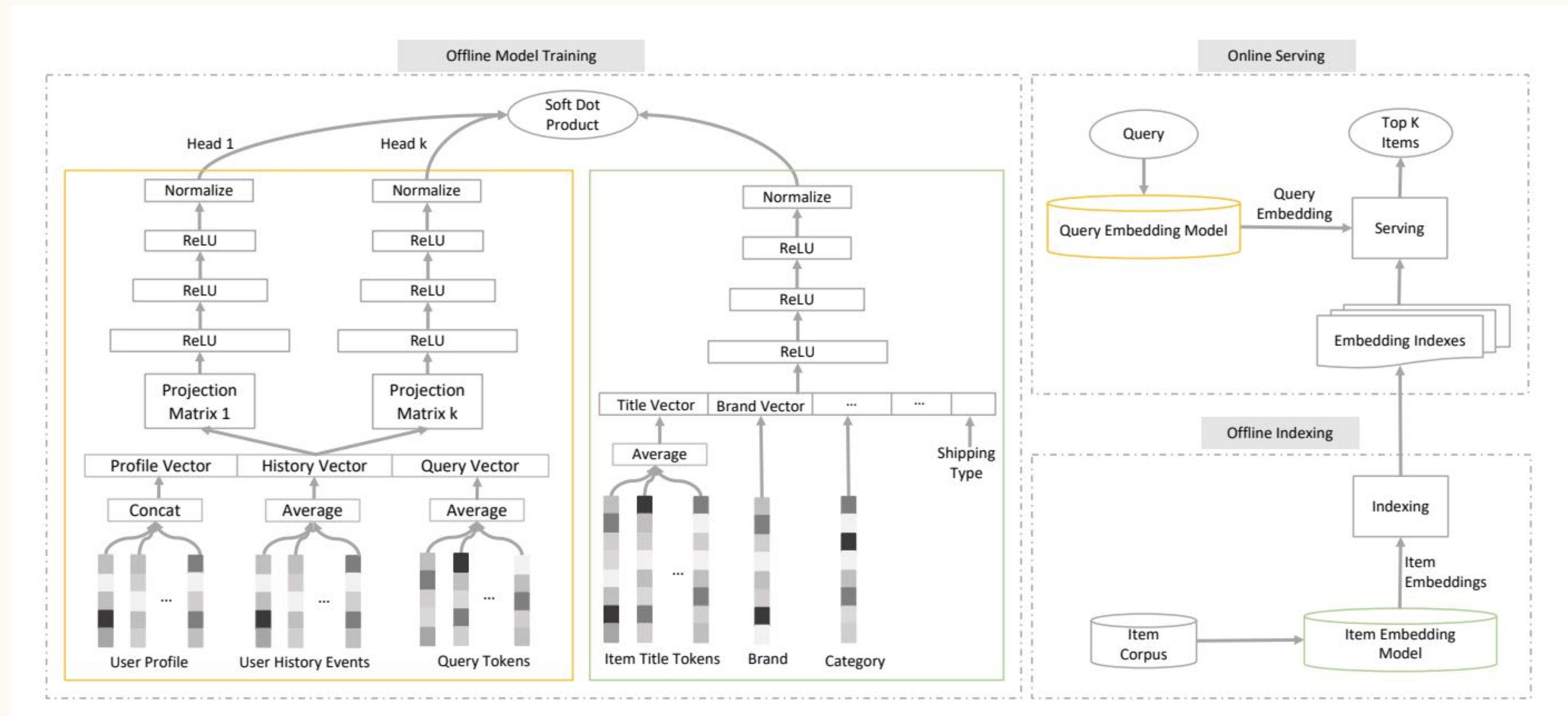
[Jui-Ting et al 2020)



Embedding Based Retrieval

The end-to-end System

[Han et al 2020]



Candidate Retrieval

References

- Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen)Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, Bing Yin. Semantic Product Search. In KDD 2019.
- Jinfeng Zhuang, Yu Liu. PinText: A Multitask Text Embedding System in Pinterest. KDD 2019.
- Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, Linjun Yang. Embedding-based Retrieval in Facebook Search. KDD 2020.
- Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, Kun Gai. Learning Tree-based Deep Model for Recommender Systems. arXiv:1801.02294, 2018.
- Paul Covington, Jay Adams, Emre Sargin. Deep Neural Networks for YouTube Recommendations. RecSys 2016.
- Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. RecSys 2019.
- Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, Wen-Yun Yang. Towards Personalized and Semantic Retrieval: An End-to-End Solution for E-commerce Search via Embedding Learning. arXiv:2006.02282, 2020.
- Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, Kun Gai. Joint Optimization of Tree-based Index and Deep Model for Recommender Systems. NeurIPS 2019.
- Yu. A. Malkov, D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:1603.09320, 2016.
- Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, Jingdong Wang. SPTAG: A library for fast approximate nearest neighbor search. <https://github.com/Microsoft/SPTAG>, 2018.
- Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, Wen-Yun Yang. Learning Tree-based Deep Model for Recommender Systems. In KDD 2018.
- Herve Jégou, Matthijs Douze, Cordelia Schmid. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010.
- Chris McCormick. Product Quantizers for k-NN Tutorial. <http://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1>, 2017



Candidate Retrieval

Hands-on: Deep Learning Retrieval



Zhoutong Fu



Deep Learning for Search and Recommender Systems in Practice

-

Learning to Rank



Jun Shi



Learning to Rank

1 System Overview

2 Metrics for Evaluation

3 Ranking Models

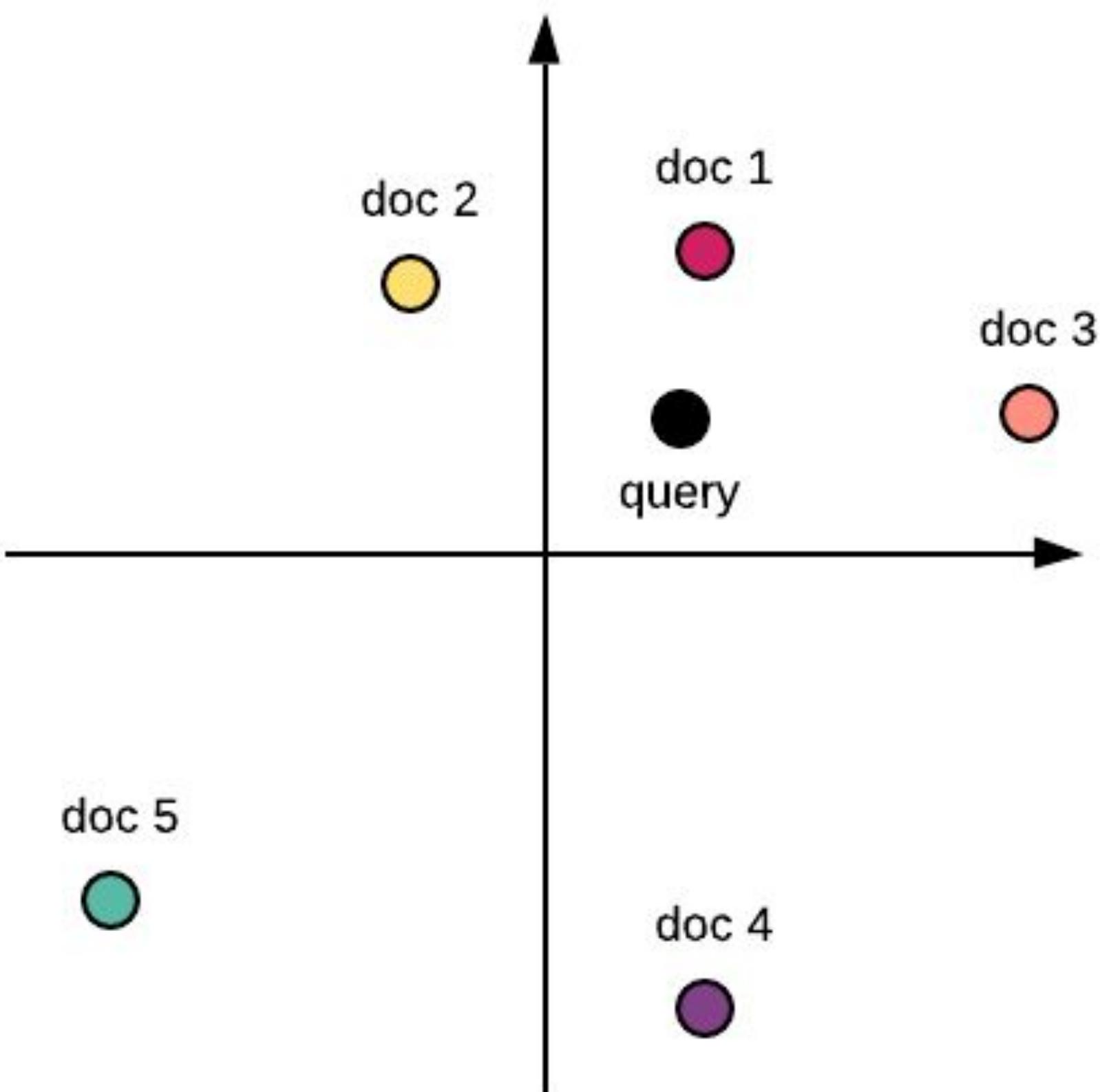
System Overview

- Document Retrieval
- Scoring and Ranking
- Personalization and Re-ranking

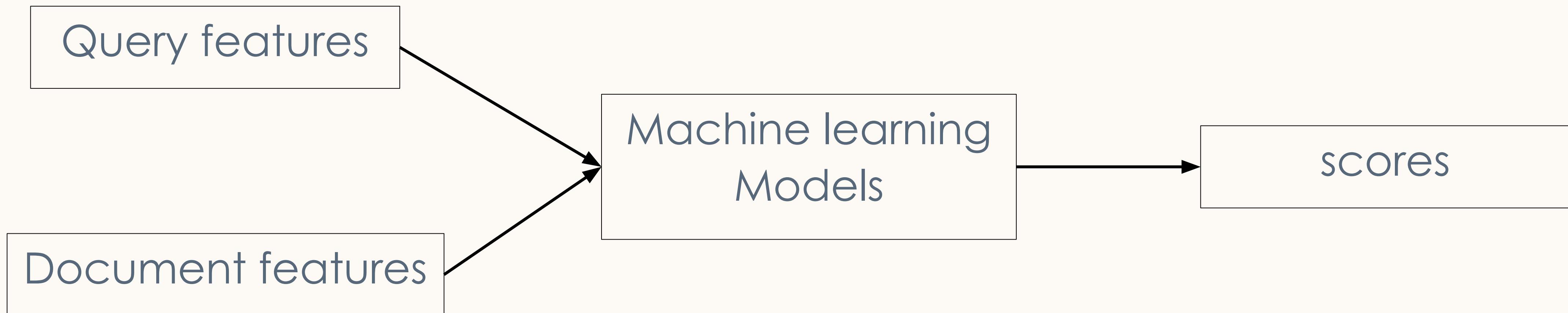


Document Retrieval

- Simple regex based retrieval
- Traditional inverted index based retrieval
- Embedding based retrieval



Scoring and Ranking





Learning to Rank

1 System Overview

2 Metrics for Evaluation

3 Ranking Models

Metrics for Evaluation

- Multiple level of relevance
 - NDCG (Normalized Discounted Cumulative Gain)
- Binary relevance
 - MAP (Mean Average Precision)
 - MRR (Mean Reciprocal Rank)

Normalized Discounted Cumulative Gain

Discounted Cumulative Gain

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2(i+1)}$$

Normalized Discounted Cumulative Gain

$$nDCG_p = \frac{DCG_p}{IDCG_p},$$

where

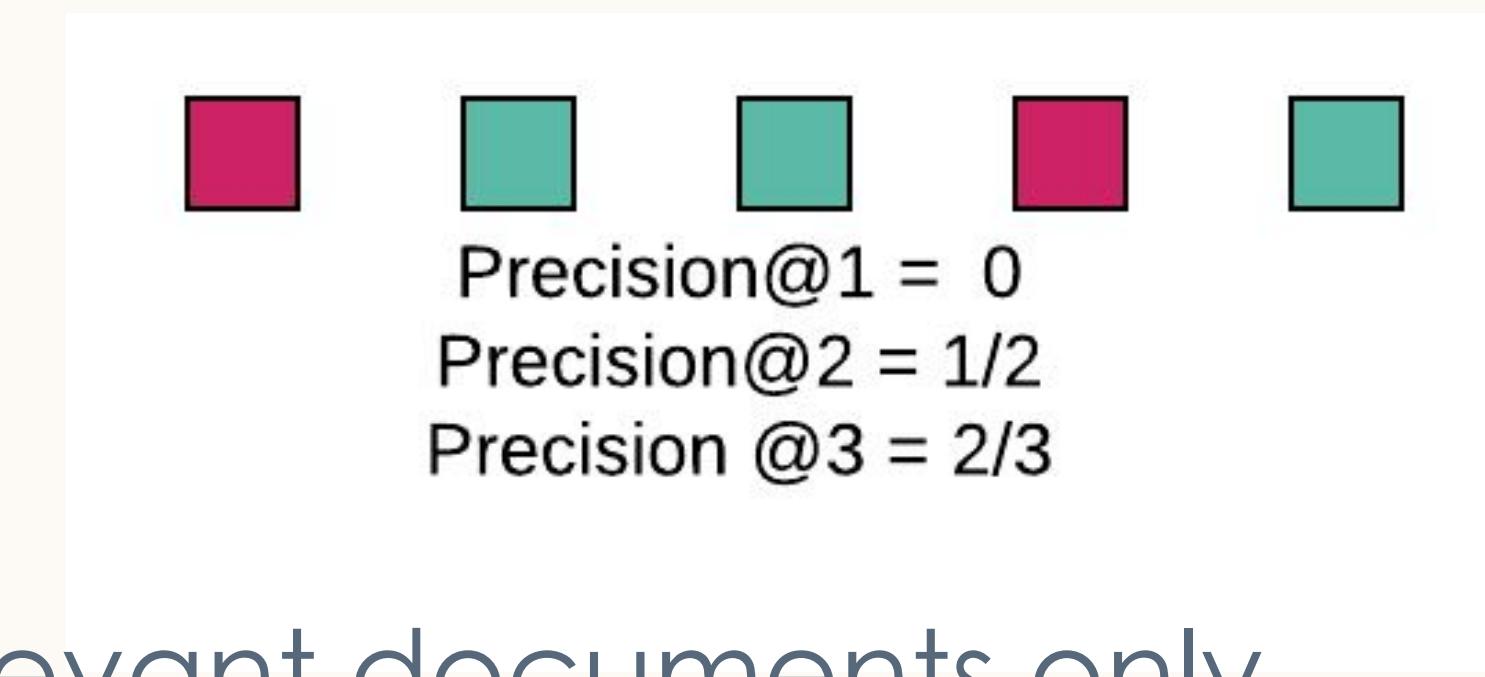
$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

https://en.wikipedia.org/wiki/Discounted_cumulative_gain

KDD 2020

Mean Average Precision

Precision@K: Relevant documents up to rank K / K



Average Precision: Average over K for relevant documents only

Precision@2 = $\frac{1}{2}$, Precision@3 = $\frac{2}{3}$, Precision@5 = $\frac{3}{5}$

$$\text{Average Precision} = \left(\frac{1}{2} + \frac{2}{3} + \frac{3}{5}\right)/3 = 0.59$$

Mean Average Precision: Average precision across all queries.

Mean Reciprocal Rank

Reciprocal Rank

$$\frac{1}{\text{rank}_i}$$

Mean Reciprocal Rank

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

Learning to Rank

1 System Overview

2 Metrics for Evaluation

3 Ranking Models

Ranking Models

- Vector space models
 - Latent semantic indexing (co-occurrence vector -> latent semantic space vector)
- Probabilistic models
 - BM25 (bag-of-word, TF-IDF based scoring function)
 - Statistical language model (Probability of the query given a document)
- Learning to rank models
 - Ordinal regression
 - Ranking SVM
 - LambdaRank

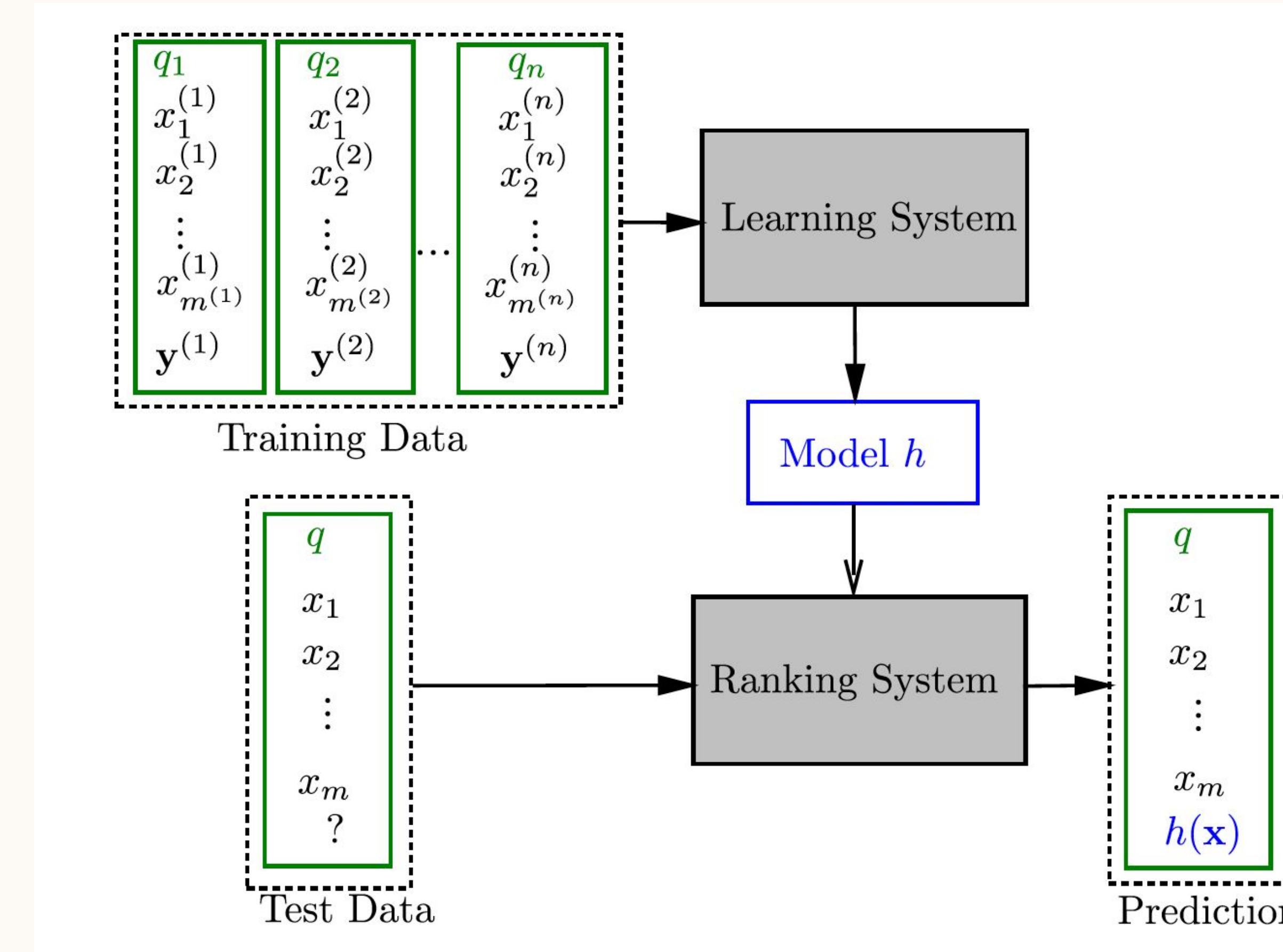
Learning to Rank

- Pointwise ranking
- Pairwise ranking
- Listwise ranking

Ranking model

key components:

- Input
- Label
- Loss function



Pointwise Ranking

Loss function is based on a single (query, document) pair.

- Regression based Algorithms
- Classification based Algorithms
- Ordinal Regression based Algorithms

Regression based pointwise ranking

Input (q, x) feature vector responding to the query and a document.

Label: y , relevance of the document.

Loss function: $L(q, x, y) = |f(q, x) - y|^2$

Scoring function tries to approximate the labels by minimizing the mean squared error

Classification based pointwise ranking

Input (q, x) feature vector responding to the query and a document.

Label: y , relevance of the document.

Loss function: cross entropy loss between label and predicted probability.

Logistic regression example (binary relevance labels):

$$P(R|x_j) = \frac{1}{1 + e^{-c - \sum_{t=1}^T w_t x_{j,t}}}.$$

Ordinal regression based pointwise ranking

Input (q, x) feature vector responding to the query and a document.

Label: y , relevance of the document.

Loss function: negative log likelihood.

Ordinal regression outputs the weights and the thresholds.

$$\Pr(y < i | q, x) = \text{sigmoid}(\theta_i - \mathbf{w}^T [\mathbf{q}, \mathbf{x}] + c),$$

where θ_i is the i th threshold

Crammer, K., Singer, Y.: Pranking with ranking. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 641–647 (2002)

Summary of pointwise ranking

Pros

- Simple, considering one document at a time.
- Available algorithms are rich. Most regression/classification algorithms can be used.

Cons

- Pointwise ranking ignores other documents in the list that will affect the ranking result.
- Pointwise ranking does not achieve the correct objective.
Ranking is about relative position, not absolute scores of individual documents.

Pairwise Ranking

Loss function is based on query and a pair of documents.

- RankNet
- Ranking SVM
- LambdaRank

Commonality of pairwise ranking algorithms

- Input: query \mathbf{q} and a pair of documents \mathbf{d}_i and \mathbf{d}_j ,
Label: \mathbf{d}_i ranks before or after \mathbf{d}_j . Binary $\{-1, +1\}$ if we ignore ties.
- Scoring function: function of the query and the pair of
documents, e.g. $f(\mathbf{q}, \mathbf{d}_i, \mathbf{d}_j)$ but in most cases, it is a function of
query and one document, e.g. $f(\mathbf{q}, \mathbf{d}_i, \mathbf{d}_j) = s(\mathbf{q}, \mathbf{d}_i) - s(\mathbf{q}, \mathbf{d}_j)$

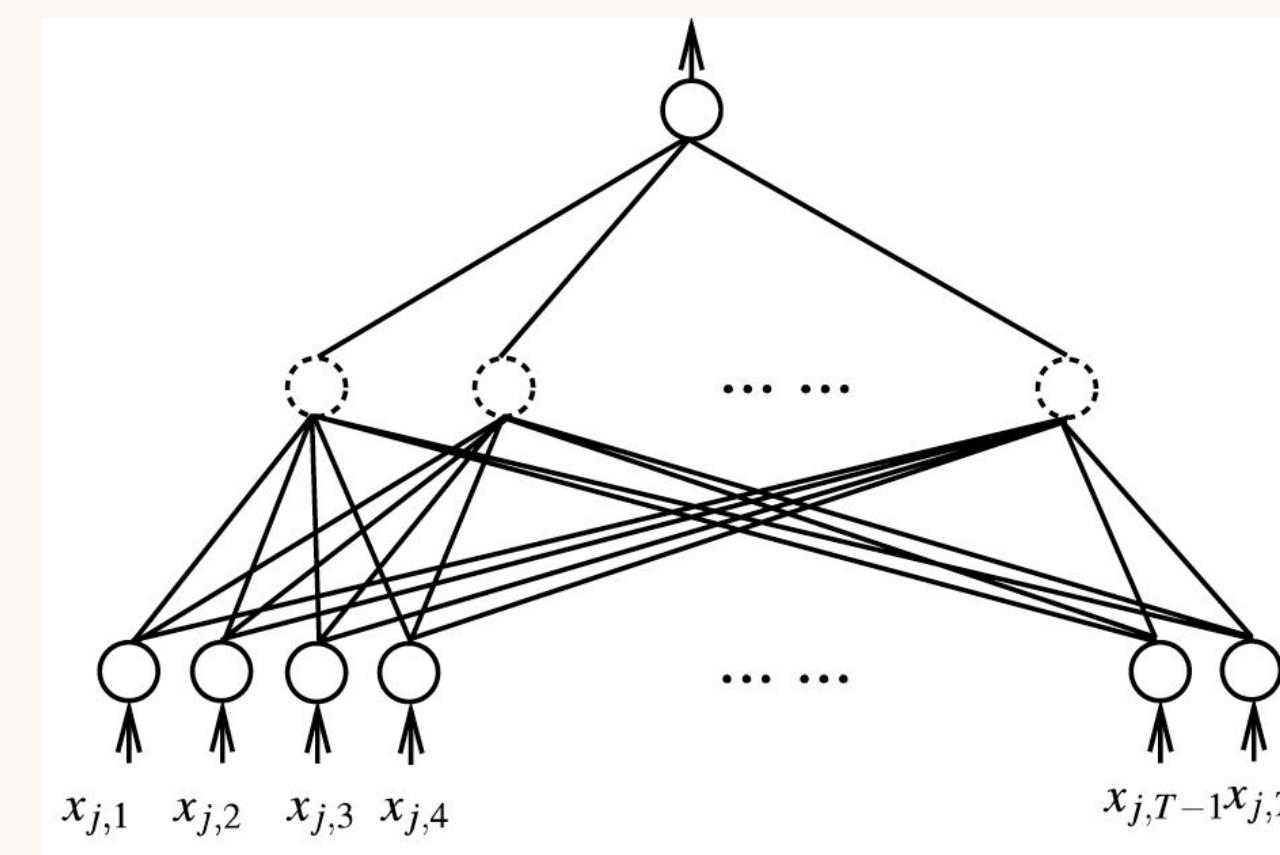
RankNet

- Input: query and a pair of documents (\mathbf{q} , \mathbf{d}_i , \mathbf{d}_j).
- Label: $y_{ij} = +1$ if \mathbf{d}_i ranks higher than \mathbf{d}_j , -1 otherwise.
- Loss function: cross entropy loss function

$$L(\mathbf{q}, \mathbf{d}_i, \mathbf{d}_j) = -y_{ij} * \log P_{ij} - (1-y_{ij}) * \log(1-P_{ij})$$

$$\text{where } P_{ij} = \text{sigmoid}(s(\mathbf{q}, \mathbf{d}_i) - s(\mathbf{q}, \mathbf{d}_j))$$

- RankNet uses neural networks as the scoring function



- Burges, C.J., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G., Learning to rank using gradient descent. Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 89–96 (2005)
- T.-Y. Liu, Learning to Rank for Information Retrieval Foundations and Trends in Information Retrieval, Vol. 3, No. 3 (2009) 225-331

Ranking SVM

Apply SVM to a binary classification problem.

- +1 \mathbf{d}_i ranks higher than \mathbf{d}_j
- -1 \mathbf{d}_i ranks lower than \mathbf{d}_j

Linear scoring function $s(\mathbf{q}, \mathbf{d}) = \mathbf{w}^\top \mathbf{x}$

$$\begin{aligned} & \min \frac{1}{2} \|\mathbf{w}\|^2 + \sum_k \sum_{i,j: y_{i,j}^k=1} \xi_{i,j}^k \\ \text{s.t. } & \mathbf{w}^T (\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{i,j}^k, y_{i,j}^k = 1 \\ & \xi_{i,j}^k \geq 0, k = 1, \dots, n \end{aligned}$$

Herbrich, R., Obermayer, K., Graepel, T., Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers, pp. 115–132 (2000)

LambdaRank / LambdaMART

- The loss function is not explicit known, use evaluation metrics instead.
- Select a metric, e.g. NDCG. compute the NDCG for the current list.
- Select two documents, d_i and d_j , assuming d_i is more relevant than d_j .
- Swap d_i and d_j 's positions in the list, compute Delta NDCG.
- Compute Lambda that is proportional to the Delta NDCG.
- Treat the Lambda as gradient and update the weights.
- LambdaMART uses gradient boosting trees models.



Summary of pairwise ranking

Pros:

- Considering relative relevance of pairs. Closer to the actual problem than pointwise ranking.
- Better relevance performance.

Cons:

- Complexity is higher than pointwise ranking.
- Only looking at pairs, not the entire list. Loss of information.

Listwise Ranking

Loss function is based on the query and a list of documents.

- AdaRank
- ListNet
- ListMLE

AdaRank

Motivation: commonly used evaluation metrics are not differentiable. So it is not easy to optimize directly. AdaRank minimizes the exponential loss, $E()$ below can be NDCG.

$$E(\pi_i, \mathbf{y}_i) \rightarrow \exp(-E(\pi_i, \mathbf{y}_i))$$

where π_i is the predicted list

\mathbf{y}_i is the ground truth list

The usual AdaBoost algorithm applies afterwards.

ListNet / ListMLE

Map list of scores to a probability distribution by Plackett–Luce model.

- Permutation probability, where $s()$ is the scoring function.

$$P_s(\pi) = \prod_{j=1}^n \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^n \phi(s_{\pi(k)})}$$

- Top-1 probability

$$P_s(j) = \frac{\phi(s_j)}{\sum_{k=1}^n \phi(s_k)},$$

Loss function

- ListNet: K-L divergence or cross entropy loss between predicted probability and ground truth probability.
- ListMLE: negative log likelihood loss.

Summary of listwise ranking

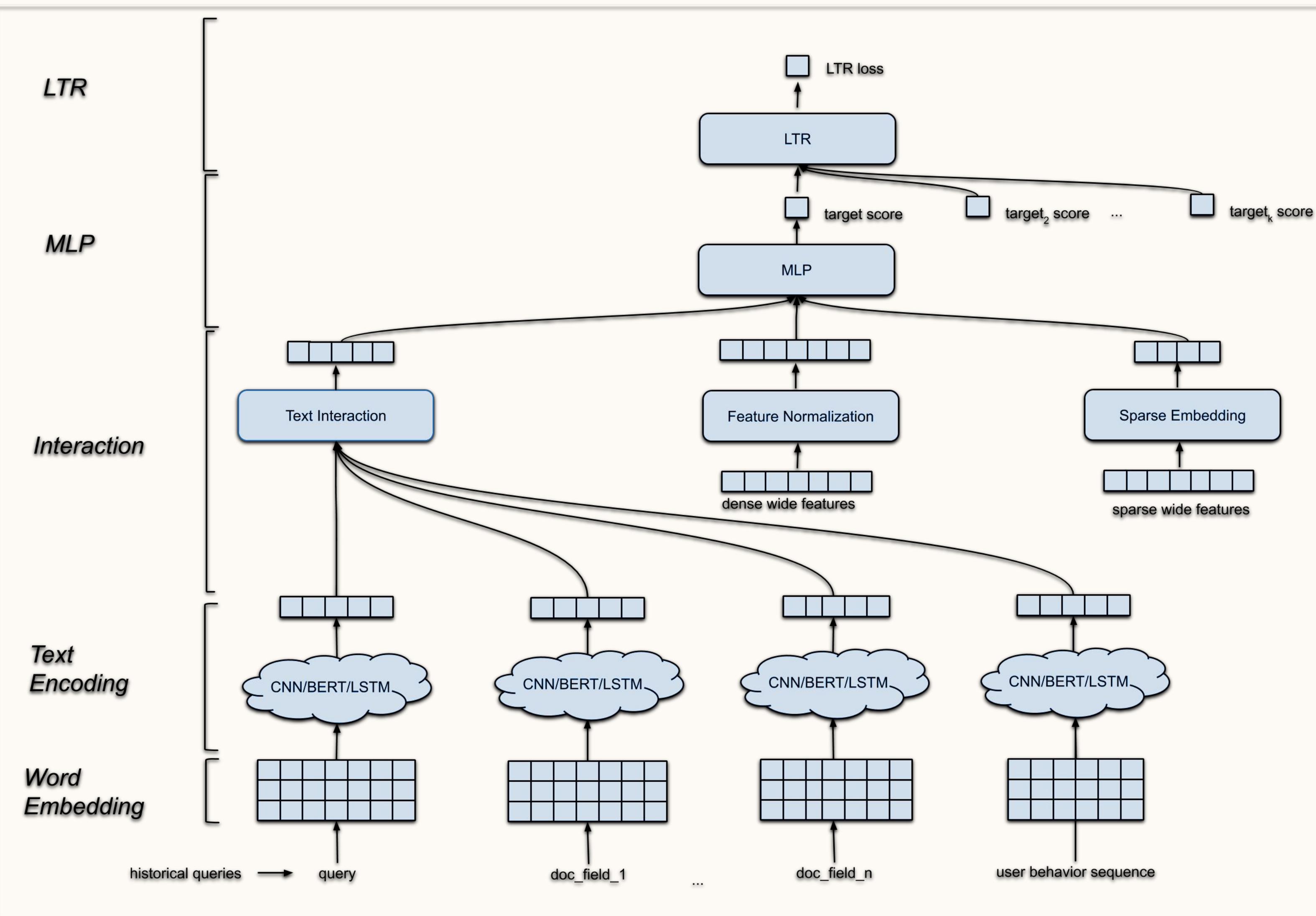
Pros

- Considers the entire list, uses more information for ranking.
- Usually offers the best performance.

Cons

- Higher complexity than algorithms in the other two categories.

DeText: a Deep Learning Ranking Framework

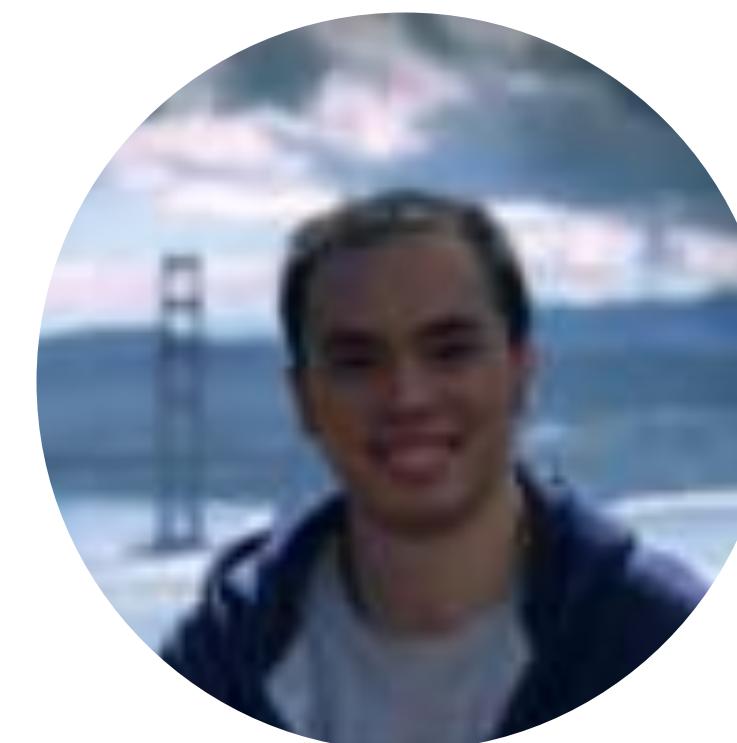


- Deep Learning + Ranking
- Supports pointwise, pairwise and listwise functions.
- Compatible with TF-ranking
- Automatic text feature incorporation
- Find more at <https://github.com/linkedin/detext>



Learning to Rank

Hands-on: Generalized Deep Mixed Model and DeText



Mingzhou Zhou



Sandeep Jha

Thank you