Learning to Rank & Semantic Matching

Information Retrieval - IR0

Yubao Tang

y.tang3@uva.nl

Credit: The majority of this content is derived from the lecture slides created by Philipp Hager and Mohammad Aliannejadi. Some modifications and supplementary content have been added by Yubao Tang.

About me

- Postdoc researcher at UvA (IRLab & AIRLab)
- Previously:
  - Ph.D. & M.Sc. Chinese Academy of Sciences, China
  - B.Sc. Sichuan University, China
- Research focus: Generative Information Retrieval & Recommendation Systems
- Happy to discuss research or course questions — feel free to reach out!
- Homepage: https://yubaotang11.github.io/

Recap

- Text processing
- Advanced ranking methods
- Bag of words
- Inverted indexing for these methods
- Vector Space Model
- Term importance weighting
- Probabilistic ranking methods

Today

- Part 1: Learning to rank
  - Pointwise methods
  - Pairwise methods
  - Listwise methods
- Part 2: Semantic matching
  - Vocabulary mismatch
  - Semantic matching
  - Distributed representation

Part 1: Learning to Rank

Google

attractions in amsterdam central

I amsterdam

https://www.iamsterdam.com › explore › centrum › thi...

Things to do in Amsterdam City Centre

10 Jul 2025 — Wandering along the cobbled streets, you'll find **world-class museums and age-old churches**, market squares and tasting rooms, tranquil courtyard gardens hidden ...

Tripadvisor

https://www.tripadvisor.com › Attr... · Translate this page

Things to Do near Amsterdam Centraal Station

Top Attractions in Amsterdam. **Rijksmuseum** · Anne Frank House · Van Gogh Museum · The Jordaan · A'dam Lookout · Body Worlds · Vondelpark · Moco Museum Amsterdam ...

Stromma

https://www.stromma.com › en-nl › amsterdam › blog

5 x things to do near Amsterdam Central Station

10 Mar 2025 — **Shopping, dining near or on the water, breathtaking sky terraces** and snapping the perfect shot! All within a 10-minute walk from Central ...

Signals in Web Search

- **Textual Signals**
  - Query content: text
  - Document content: title, page content

- How well does the query text match the document text? [6]
  - BM25
  - TF-IDF / vector space models
  - Semantic matching with LLMs or topic models

Signals in Web Search

- **But there are many signals beyond text**
  - Query: type, language
  - Document: URLs, images, structure
  - User context: location, date, device, search history
  - Metadata: popularity, recency, page quality, spam, adult content, …
  - External stakeholders: advertisers, auctions, content creators, …

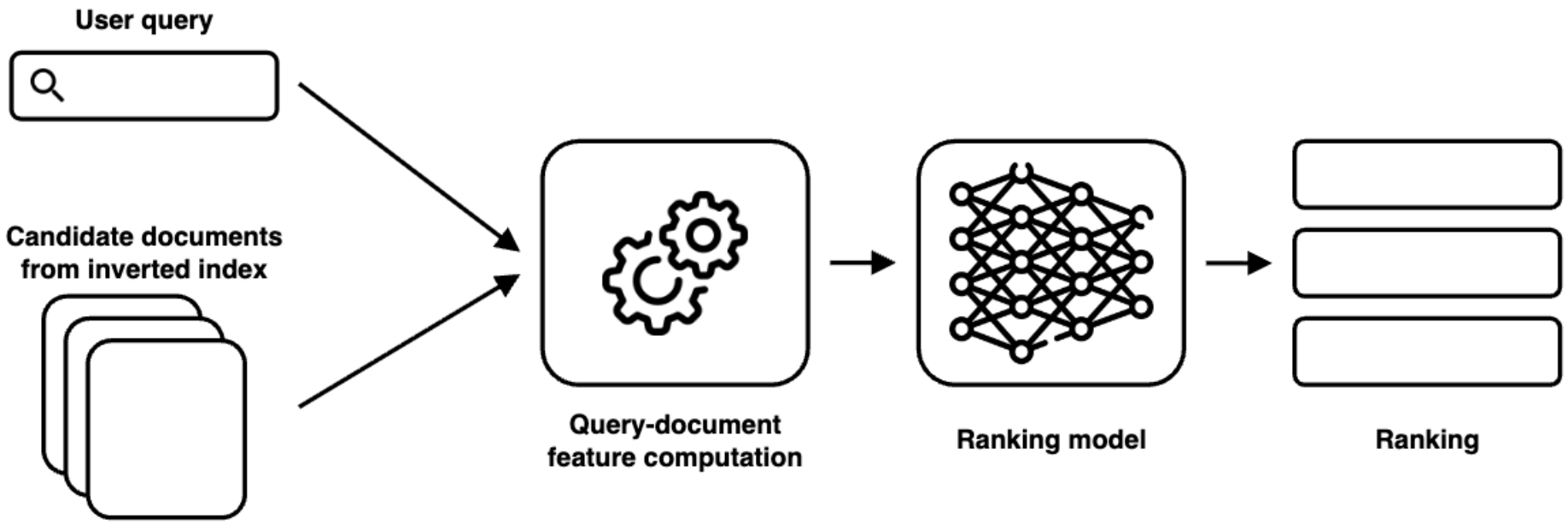
Signals in Web Search

- **Modern search engines use a lot of features**
  - Airbnb [10]: > 195 features
  - Bing [15]: > 136 features
  - Istella [7]: > 220 features
  - Yahoo [2]: > 700 features

- **How do we combine all of these signals?**

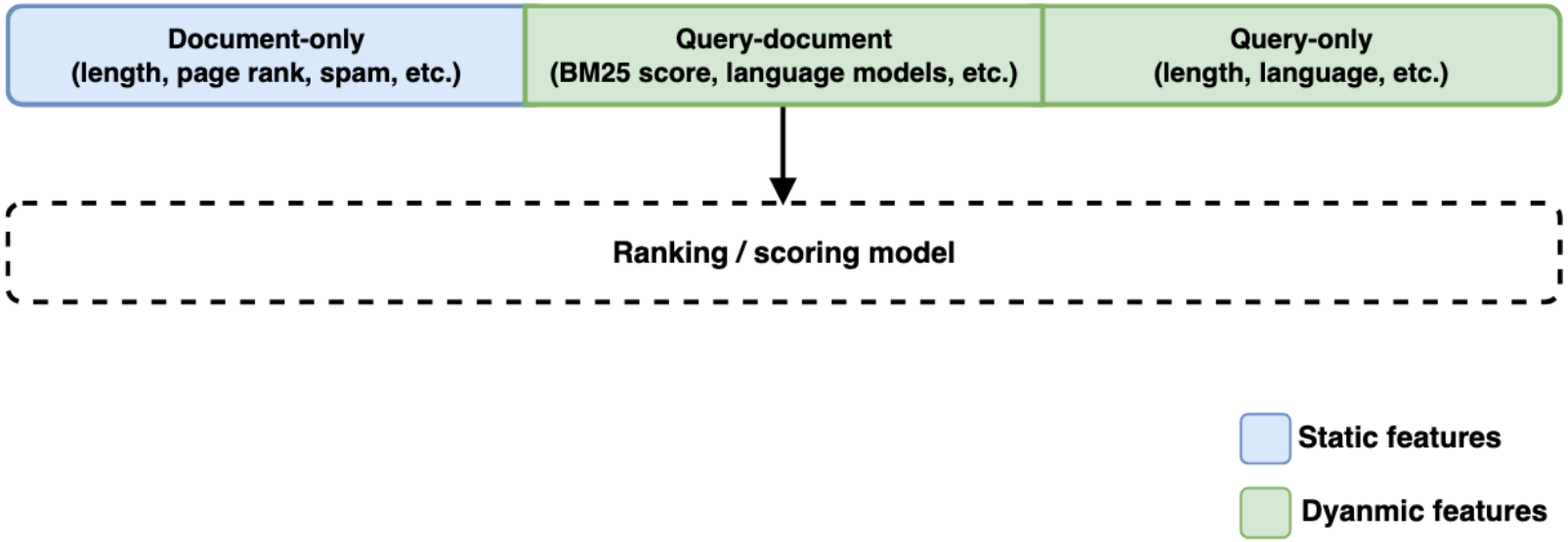
- Learning to Rank (LTR) is
  - “. . . a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance.” – Liu [13]

Learning to Rank

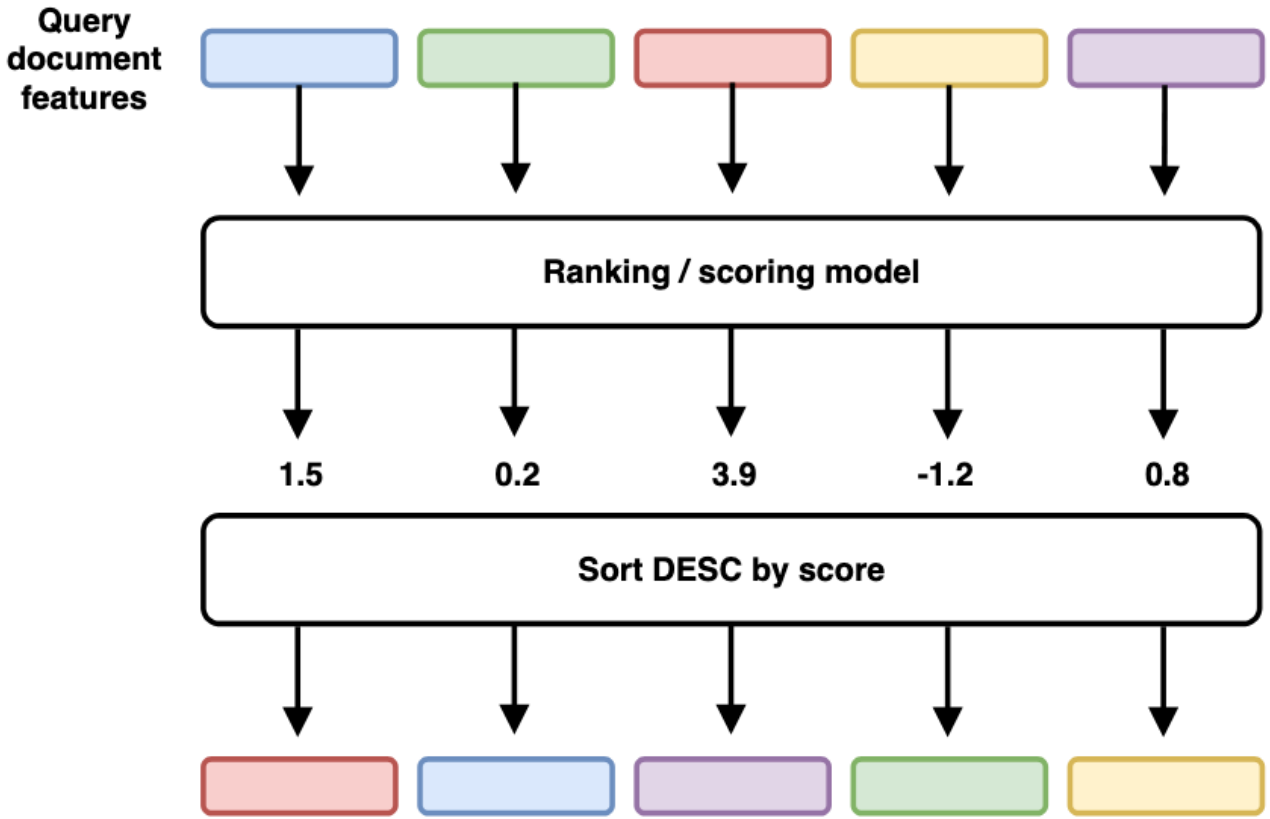


Features

- Traditionally LTR used hand-crafted numerical features
- Nowadays, we also use deep learned features
- We can categorize features into:



Problem Formulation



The goal of learning to rank

- Thus, we have:
  - a **feature vector** for each query-document pair: $\vec{x}_{\{q,d\}} \in \mathbb{R}^m$
  - a **relevance judgment** for each query-document pair, e.g.: $y_{\{q,d\}} \in \{0, 1, 2, 3, 4\}$
- A ranking model $f: \vec{x} \rightarrow \mathbb{R}$ scores each query-document pair to optimize the order of items when sorting descendingly by $f\left(\vec{x}_{\{q,d\}}\right) = s_{\{q,d\}}$

- **How can we learn a ranking model f?**

Pointwise methods

Pointwise LTR

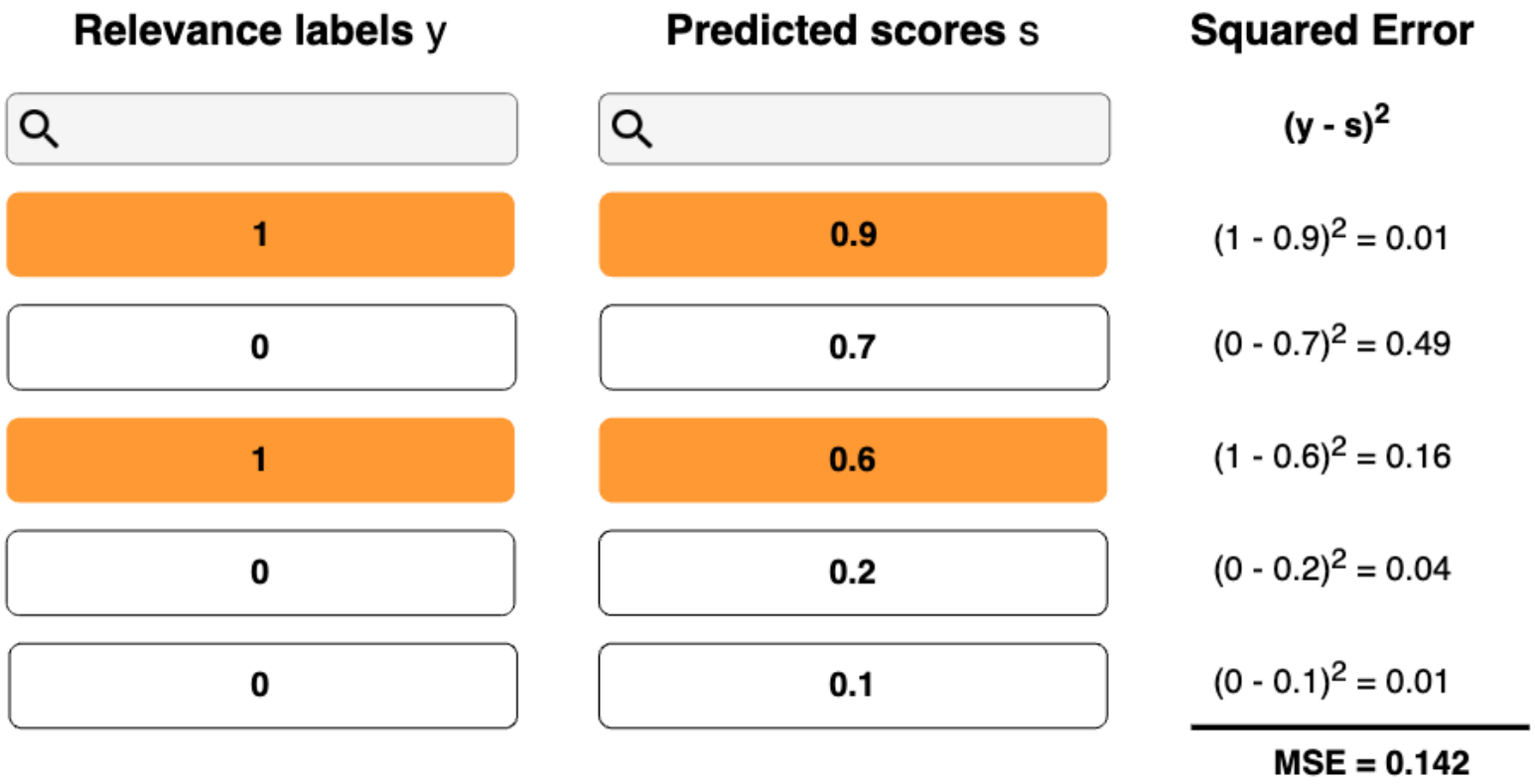
- Regression: Relevance as a real-valued score [4,9]
- We can use linear regression for our ranking model:

$$f(x) = w_0 + w_1 x_1 + \cdots + w_n x_n = s$$

- Each feature has a weight w, learned to minimize prediction error
- Usually we quantify how far off our predictions are using the mean squared error (MSE) loss:

$$L_{mse} = \frac{1}{n} \sum (y_i - s_i)^2$$

Pointwise LTR – MSE



Pointwise LTR

- Regression: Relevance as a real-valued score [4, 9]
- Classification: Relevance as unordered categories [3, 14]
- Ordinal regression: Relevance as ordered categories [5, 16]

- **What are challenges with these approaches?**

Problems of pointwise LTR

- Some (solvable) challenges include:
  - Class imbalance: We have way more irrelevant than relevant documents
  - Feature normalization: Feature distributions can differ greatly between queries

Problems of pointwise LTR

- A more fundamental problem:
  - Pointwise methods predict a score for each query-document **independently**
  - But document scores are fundamentally dependent on each other in a ranking
- Minimizing a pointwise loss does not always lead to a better ranking

Pointwise LTR: A lower loss does not imply a better ranking

$$L_{mse} = \frac{1}{n}\sum_{i=1}^{n}(y_i - s_i)^2$$

Pointwise LTR: A lower loss does not imply a better ranking

L_{mse} =1.16 / 5
MRR=1, nDCG=1

Pointwise LTR: A lower loss does not imply a better ranking

$L_{mse} = 0.77 / 5$

MRR=0.2, nDCG=0.39

Pairwise methods

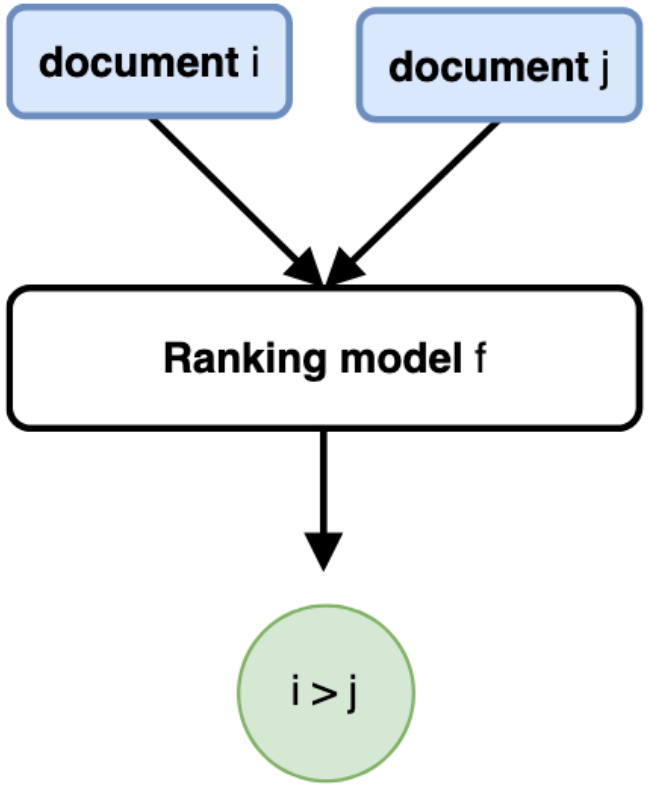
Pairwise LTR

- **Observation**: For a good ranking, we only require relative relevance levels:

$$y_i > y_j \rightarrow s_i > s_j$$

- How can we optimize a model with pairs of items?

Naïve pairwise model



$$P(i > j) = f(x_i, x_j)$$

Naïve pairwise model

- Let's (naively) change the ranking model to take document pairs as input:

$$P(i > j) = f(x_i, x_j)$$

- But pairwise document inputs are not a good idea:
  - This method has quadratic complexity $O(N^2)$ during **training** and **inference** and thus does not scale to many documents
  - Pair-wise preferences have to be aggregated and can lead to paradoxical situations:

$$s_1 > s_2$$
$$s_2 > s_3$$
$$s_3 > s_1$$

Pairwise LTR

- **A better idea: Let's keep the ranking model unchanged:**

$$f(\vec{x}) = \mathrm{s_i}$$

- But the loss function is based on pairs of documents:

$$L_{pairwise}(s, y) = \sum_{y_i > y_j} \emptyset(s_i - s_j)$$

- We still score one document at-a-time and can sort according to scores, but the model is optimized over item pairs

Pairwise LTR

- Pairwise loss functions minimize the number of incorrectly ranked pairs:

  where $y_i > y_j$, but our model falsely predicts $s_i < s_j$.

- Pairwise loss functions generally have the following form:

$$L_{pairwise}(s, y) = \sum_{y_i > y_j} \emptyset(s_i - s_j)$$

  where $\emptyset$ can be the:

  - Hinge function in RankingSVM [11, 12]: $\emptyset(z) = \max(0,1 - z)$ (z) = max(0, 1 − z)
  - Exponential function in RankBoost [8]: $\emptyset(z) = e^{-z}$
  - Logistic / Sigmoid function in RankNet [1]: $\emptyset(z) = \log(1 + e^{-z})$

RankNet

RankNet

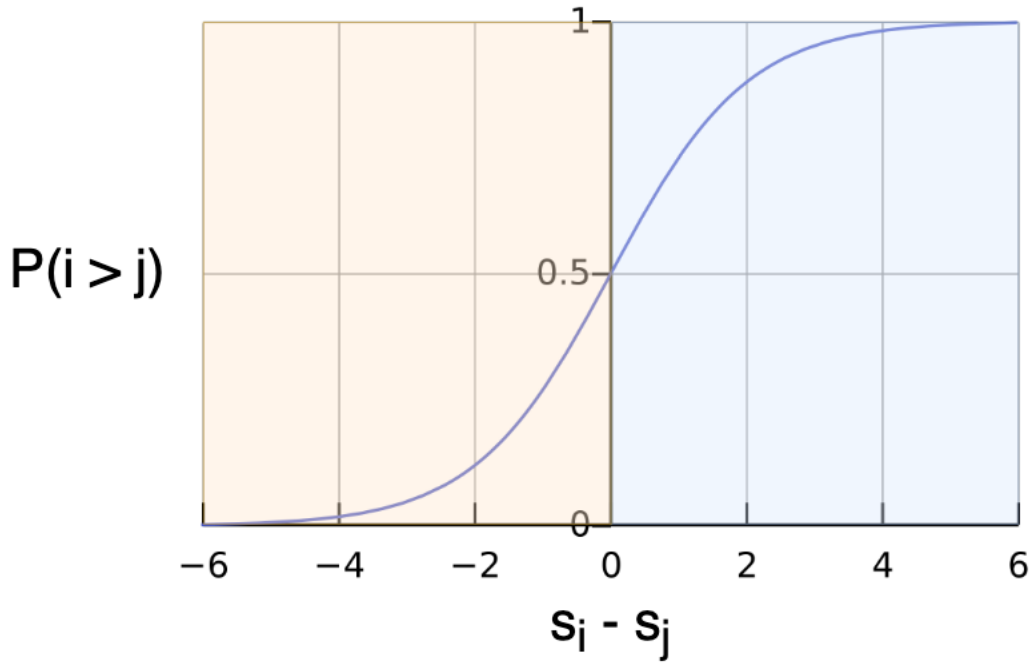
- Introduced by Burges et al. [1] in 2005 to train neural ranking models
- Popular in industry applications and won the ICML 2015 test of time award
- RankNet defines the probability that document i should be ranked over document j as:

$$P(i > j) = sigmoid(s_i - s_j)$$

- RankNet then uses the log loss between the predicted probabilities for each pair and their true/target probability: $\bar{P}(i > j)$

https://icml.cc/2015/index.html%3Fp=51.html

RankNet



Mapping the difference in scores between to items to the predicted probability $P(i > j)$ using the sigmoid function

RankNet

- Now that we have a model prediction for each item pair, where do we get the target probability of $\bar{P}(i > j)$ from?
  1. Ask annotators to judge pairs of items (infeasible, it requires $O(N^2)$ annotations)
  2. We make up probabilities based on relevance judgments:
     - If $y_i > y_j$, set $\bar{P}(i > j) = 1$
     - If $y_i = y_j$, set $\bar{P}(i > j) = 0.5$
     - If $y_i < y_j$, set $\bar{P}(i > j) = 0$
- These made-up/virtual target probabilities were chosen mainly for convenience as they simplify the final loss to:

$$L_{RankNet}(s, y) = \sum_{y_i > y_j} log(1 + e^{-(s_i - s_j)})$$

https://icml.cc/2015/index.html%3Fp=51.html

Pairwise LTR

What are problems of pairwise methods?

The made-up target probabilities $\bar{P} \in \{0, 0.5, 1\}$ are quite crude, since any difference in relevance labels is treated equally. For example:

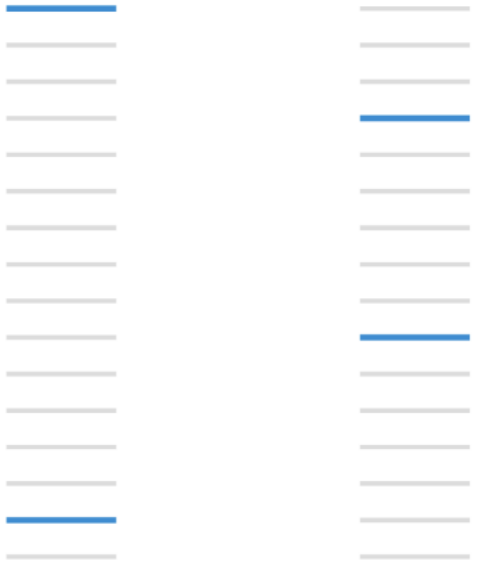
$$P(4 > 1) = 1.0$$

$$P(4 > 3) = 1.0$$

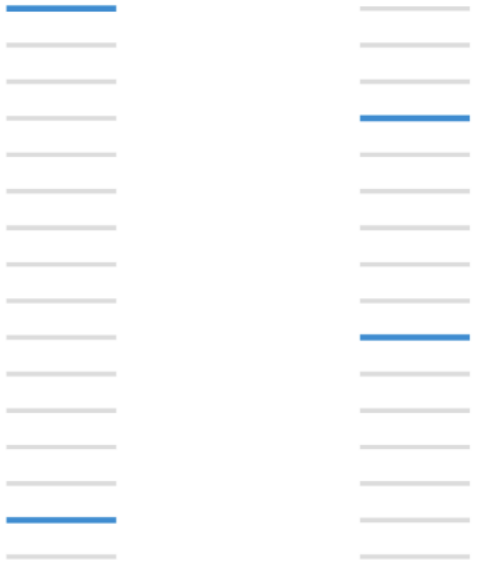
Not very elegant, but works well in practice. . .

A more important limitation: We treat all item pairs as equally important, but are they?

Pairwise LTR: Minimizing pairwise errors



Pairwise LTR: Minimizing pairwise errors



Reducing pairwise errors from 13 (left) to 11 (right),

while top-heavy measures like MRR and nDCG degrade [1, Figure 1].

Pairwise LTR: Minimizing pairwise errors

The **black** arrows denote the RankNet gradients,

while what we'd arguably want are the **red** arrows [1, Figure 1].

Optimization Units and Objectives in LTR Methods

Method	Optimization Unit	Optimization Objective	Required Label Type
Pointwise	Individual item	Predict scores close to ground-truth relevance	Absolute labels (e.g., 0/1 or grades)
Pairwise	Item pair	Learn which item is more relevant	Relative order between items

Evaluation Metrics and What They Measure

Metric	Evaluated Unit	What It Measures
NDCG	The entire ranked list	Penalizes relevant documents that appear lower in the list
MRR	Position of the first relevant item	How early the first relevant item appears
Hits@K	Top-K retrieved items	Whether at least one relevant item appears in the top K

There is a **misalignment**:
pointwise and pairwise methods optimize for local accuracy (on items or pairs), while ranking metrics care about **global ordering**, especially at the top of the list.

Listwise methods

LambdaRank

- Motivation: Can we directly optimize IR metrics such as nDCG, Precision, and MRR?
- Reciprocal Rank: Reciprocal of the rank of the first relevant item after sorting by scores:

$$RR = \frac{1}{rank_i}$$

- Discounted Cumulative Gain:

$$DCG = \frac{1}{n}\sum_{i=1}^{n}\frac{2^{y_i}-1}{\log(i+1)}$$

- Non-smooth and discontinuous
  - Ranking metrics typically only depend on the rank of an item, not on its score
  - Model scores change smoothly, the ranks of documents change abruptly

Listwise LTR

- Non-differentiable
  - Ranking metrics rely on a sorting operation that is non-smooth and discontinuous w.r.t. to model parameters θ:

$$\frac{\partial RR}{\partial \theta} = ? ? ?$$

$$\frac{\partial DCG}{\partial \theta} = ? ? ?$$

- Thus, ranking metrics are either flat (with zero gradient) or discontinuous
- Holy grail of LTR: Finding methods that (indirectly) optimize listwise IR metrics

LambdaRank

LambdaRank

- Observations:
  - To train a model, we don’t need the costs just the gradients (of the costs w.r.t model scores)
  - Gradients should be larger for pairs that have a greater impact on our metric

LambdaRank

- Idea: Scale the RankNet gradients for each document pair based on the change in nDCG we would observe after swapping the two items:

$$L_{LambdaRank}(s, y) = \sum_{y_i > y_j} \Delta NDCG(i, j) \log(sigmoid(s_i - s_j))$$

Conclusion

Summary

- Today’s search and recommender systems use many signals for ranking.
- These signals can be computed beforehand (static features) or have to be computed when the user submits their query (dynamic features).
- Learning to rank is a method to learn models that automatically combine these query and document features into a single ranking.

Summary

Method	Optimization Unit	Optimization Objective	Required Label Type	Alignment with Ranking Metrics
Pointwise	Individual item	Predict scores close to ground-truth relevance	Absolute labels (e.g., 0/1 or grades)	Ignores relative or positional ranking
Pairwise	Item pair	Learn which item is more relevant	Relative order between items	Treats all pairwise errors equally, regardless of position
Listwise	Entire ranked list	Optimize the quality of the overall ranking (e.g., NDCG)	Full relevance list or rankings	Directly accounts for ranking quality and position impact

Summary

Aspect	RankNet	LambdaRank
Method type	Pairwise	Pairwise-based listwise methods (metric-aware)
Loss function	Cross-entropy on pairwise probabilities	No explicit loss; gradients defined directly
Uses sigmoid?	Yes, to model pairwise preference probability	No need for sigmoid
Ranking metric aware?	No, all pairs treated equally	Yes, swaps weighted by impact on metric (e.g., NDCG)
Gradient computation	From loss function	Directly defined by ΔNDCG (or other metric changes)
Improvement over?	Pointwise methods	RankNet

Part 2: Semantic Matching

Contents

- Vocabulary mismatch.
- Semantic matching.
- Distributed representation.

Document 1

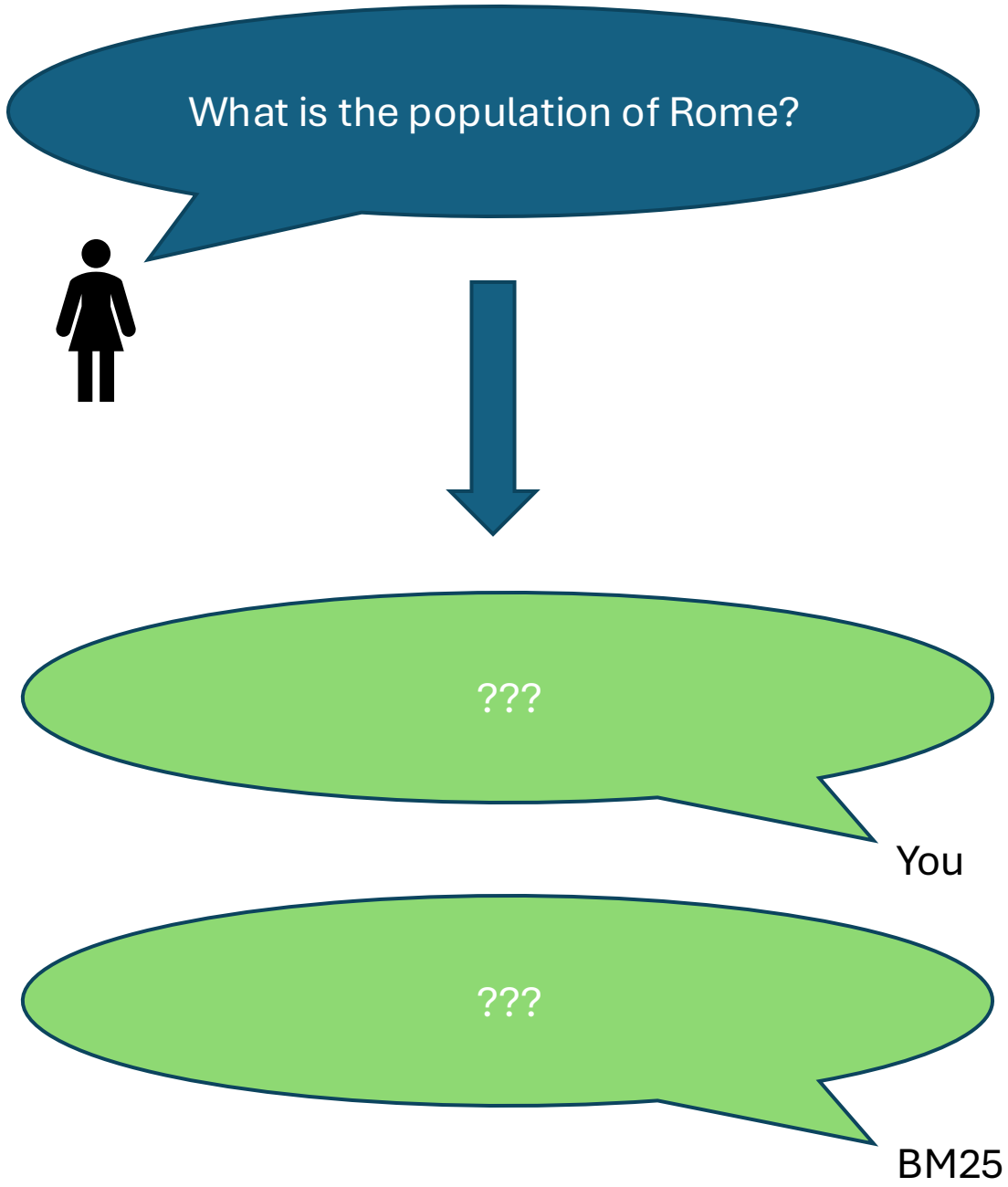
Stolen artefacts recovered by police go on display in new Rome museum. Italy has been so successful in regaining ancient artworks and artefacts illegally exported from the country.

Document 2

According to the latest statistics, the number of people living in Italy's capital has risen by 10%. As of 2017, it has reached 2.9 million.

Document 3

Popular Italian cities banning water in worst drought in 70 years. A severe heatwave and lack of rainfall have led to disaster in Italy, especially in cities with high population, like Rome.



Document 1

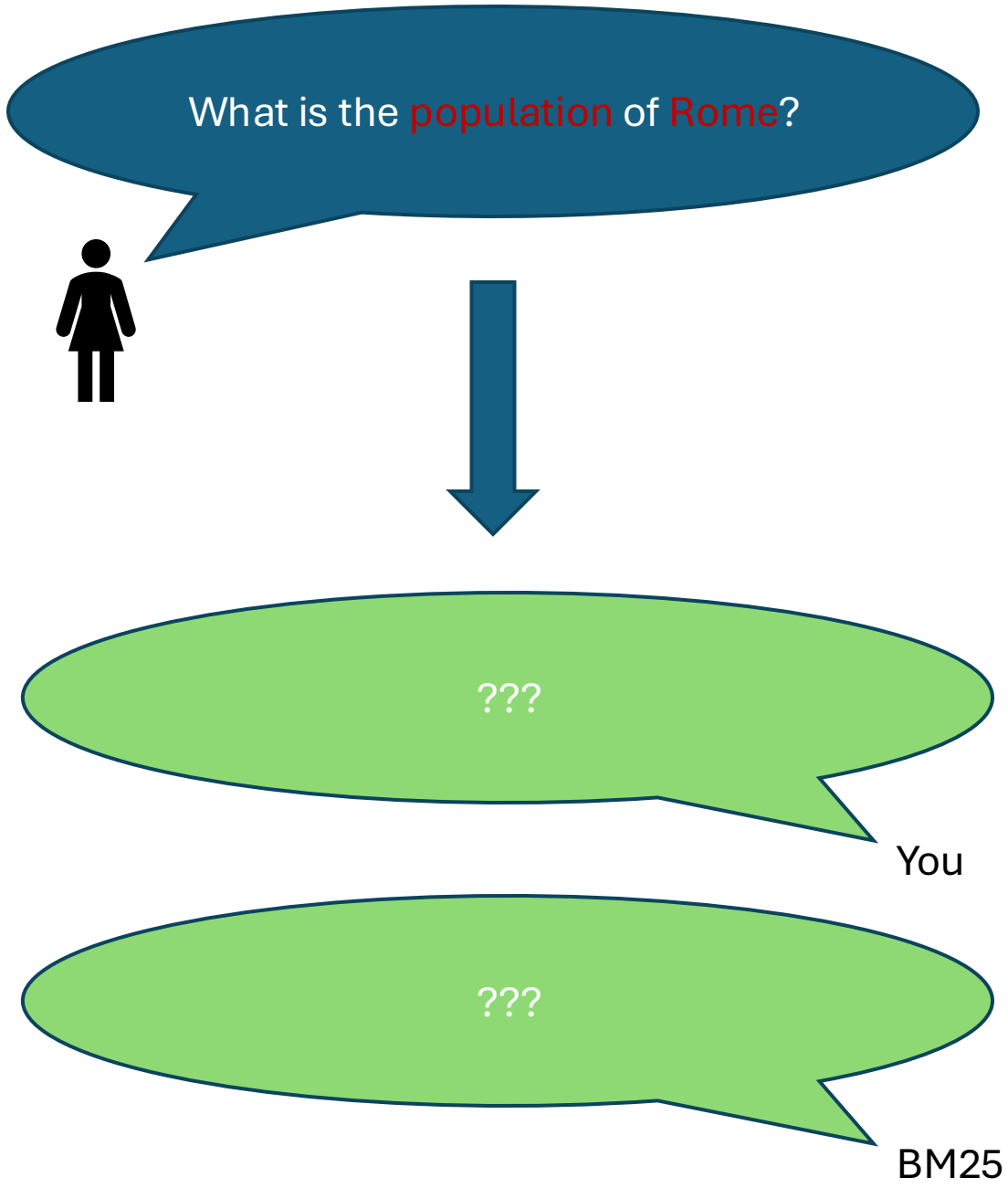
Stolen artefacts recovered by police go on display in new Rome museum. Italy has been so successful in regaining ancient artworks and artefacts illegally exported from the country.

Document 2

According to the latest statistics, the number of people living in Italy's capital has risen by 10%. As of 2017, it has reached 2.9 million.

Document 3

Popular Italian cities banning water in worst drought in 70 years. A severe heatwave and lack of rainfall have led to disaster in Italy, especially in cities with high population, like Rome.



Dimensionality Reduction

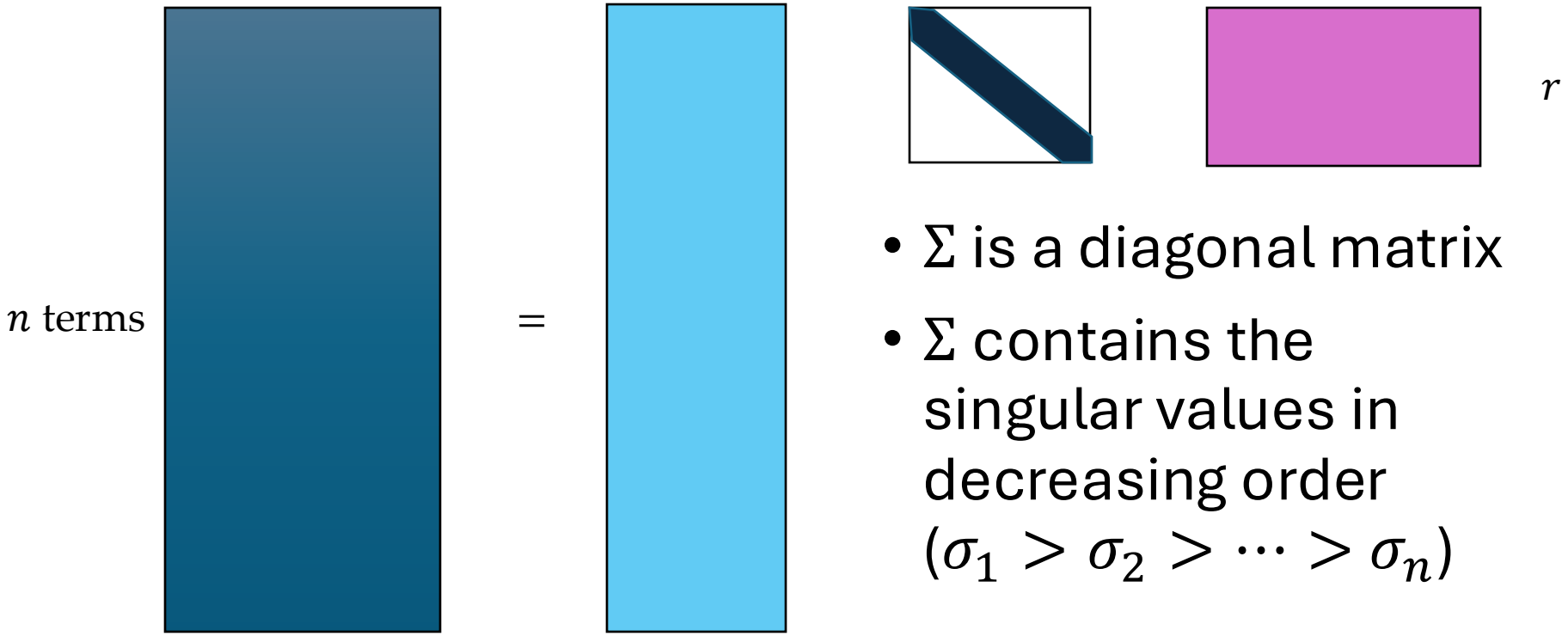
- Vector Space Model reminder: each term is a dimension.
- High dimensional data:
  - Vectors in a high-dimensional space.
  - Not all possible vectors appear in the data set (sparse).
  - This data has intrinsic dimensionality k (where $k \ll n$).

- Latent semantic analysis (LSA): relationship between documents and terms
  - Singular Value Decomposition (SVD)

Singular Value Decomposition

$\mathbf{X}_{n \times m}$ $U_{n \times r}$ $\Sigma_{r \times r}$ ${V_{m \times r}}^T$

m documents r r m



- Σ is a diagonal matrix
- Σ contains the singular values in decreasing order ($\sigma_1 > \sigma_2 > \cdots > \sigma_n$)

An example of SVD: The matrix X

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

An example of SVD

$\mathbf{X}_{n \times m}$

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$U_{n \times r}$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

$\Sigma_{r \times r}$

$V_{m \times r}{}^{T}$

d_1	d_2	d_3	d_4	d_5	d_6
−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
−0.29	−0.53	−0.19	0.63	0.22	0.41
0.28	−0.75	0.45	−0.20	0.12	−0.33
0.00	0.00	0.58	0.00	−0.58	0.58
−0.53	0.29	0.63	0.19	0.41	−0.22

An example of SVD: The matrix U

ship	–0.44	–0.30	0.57	0.58	0.25
boat	–0.13	–0.33	–0.59	0.00	0.73
ocean	–0.48	–0.51	–0.37	0.00	–0.61
wood	–0.70	0.35	0.15	–0.58	0.16
tree	–0.26	0.65	–0.41	0.58	–0.09

- One row per term.
- Think of the dimensions as semantic dimensions that capture distinct topics like politics, sports, economics.
  - Each number u_{ij} in the matrix indicates how strongly related term i is to the topic represented by semantic dimension j .

An example of SVD: The matrix Σ

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

- This is a square, diagonal matrix.
- The magnitude of the singular value measures the importance of the corresponding semantic dimension.
- We'll make use of this by omitting unimportant dimensions.

An example of SVD: The matrix V

d_1	d_2	d_3	d_4	d_5	d_6
–0.75	–0.28	–0.20	–0.45	–0.33	–0.12
–0.29	–0.53	–0.19	0.63	0.22	0.41
0.28	–0.75	0.45	–0.20	0.12	–0.33
0.00	0.00	0.58	0.00	–0.58	0.58
–0.53	0.29	0.63	0.19	0.41	–0.22

- One column per document.
- These are again the semantic dimensions from the term matrix U that capture distinct topics like politics, sports, economics.
  - Each number v_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension j.

How to use SVD in LSA

- Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details:
  - may be noise – in that case, reduced LSA is a better representation because it is less noisy;
  - make things dissimilar that should be similar – again reduced LSI is a better representation because it represents similarity better.

Latent Semantic Analysis

$\mathbf{X}_{n \times m}$

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

ship	–0.44	–0.30	0.57	0.58	0.25
boat	–0.13	–0.33	–0.59	0.00	0.73
ocean	–0.48	–0.51	–0.37	0.00	–0.61
wood	–0.70	0.35	0.15	–0.58	0.16
tree	–0.26	0.65	–0.41	0.58	–0.09

$U_{n \times r}$

$=$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

$\Sigma_{r \times r}$

$V_{m \times r}{}^{T}$

d_1	d_2	d_3	d_4	d_5	d_6
–0.75	–0.28	–0.20	–0.45	–0.33	–0.12
–0.29	–0.53	–0.19	0.63	0.22	0.41
0.28	–0.75	0.45	–0.20	0.12	–0.33
0.00	0.00	0.58	0.00	–0.58	0.58
–0.53	0.29	0.63	0.19	0.41	–0.22

Choice of k (= # of dimensions)

- The choice of k is critical.
- Ideally, we want a value of k:
  - large enough to fit all the real structure in the data;
  - small enough so that we do not also fit the sampling error or unimportant details.
- The proper way to make such choices is an open issue.
- Typically, we use different values of k and compare performance based on evaluation measures.

Ranking

1. Apply SVD.
2. Reduce dimensionality.
3. Reconstruct matrix with reduced dimensionality.
4. Query representation.
5. Ranking.

Example

- Extracted topics (dimensions):
  - Topic 0: jpeg image file gif images graphics format color
  - Topic 1: edu graphics pub data mail ftp 128 ray 3d send
  - Topic 2: jehovah god lord jesus christ father earth people
  - Topic 3: space earth planet vanue spacecraft solar surface

```
print(lsa[0])
```

```
[0.37179178 0.11877906 0.54492488 0.31503806]
```

```
print(dataset.data[0])
```

```
My point is that you set up your views as the only way to believe.  Saying
that all eveil in this world is caused by atheism is ridiculous and
counterproductive to dialogue in this newsgroups.  I see in your posts a
spirit of condemnation of the atheists in this newsgroup bacause they don'
t believe exactly as you do.  If you're here to try to convert the atheists
here, you're failing miserably.  Who wants to be in position of constantly
defending themselves agaist insulting attacks, like you seem to like to do?!
I'm sorry you're so blind that you didn't get the messgae in the quote,
everyone else has seemed to.

Tammy
```

Summary

- LSA can capture semantics of terms.
- SVD can be used to reduce dimensionality.
- Address vocabulary mismatch.
- Represent documents and queries using LSA topics.
- Compute similarity and rank based on that.

Distributed Representation

Local Representation

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

Distributed Representation

- The features may have direct and obvious relations to the original input but they have comparative values.

d_1	d_2	d_3	d_4	d_5	d_6
–0.75	–0.28	–0.20	–0.45	–0.33	–0.12
–0.29	–0.53	–0.19	0.63	0.22	0.41
0.28	–0.75	0.45	–0.20	0.12	–0.33
0.00	0.00	0.58	0.00	–0.58	0.58
–0.53	0.29	0.63	0.19	0.41	–0.22

Intuition of Distributional Word Similarity

tesgüino

Intuition of Distributional Word Similarity

A bottle of ***tesgüino*** is on the table
Everybody likes ***tesgüino***
Tesgüino makes you drunk
We make ***tesgüino*** out of corn.

Intuition of Distributional Word Similarity

A bottle of ***tesgüino*** is on the table
Everybody likes ***tesgüino***
Tesgüino makes you drunk
We make ***tesgüino*** out of corn.

- From context words humans can guess **tesgüino** means
  - an alcoholic beverage like **beer**
- Intuition for algorithm:
  - Two words are similar if they have similar word contexts.

Vector Models

Sparse vector representations:

1. Mutual-information weighted word co-occurrence matrices.

Dense vector representations:

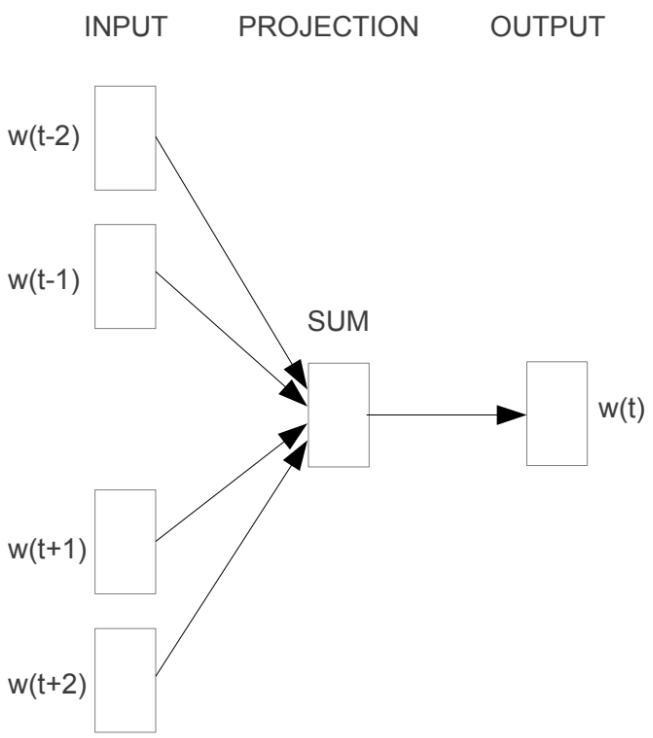
2. Singular value decomposition (and Latent Semantic Analysis).
3. Neural-network-inspired models (skip-grams, CBOW).

Prediction-based Models

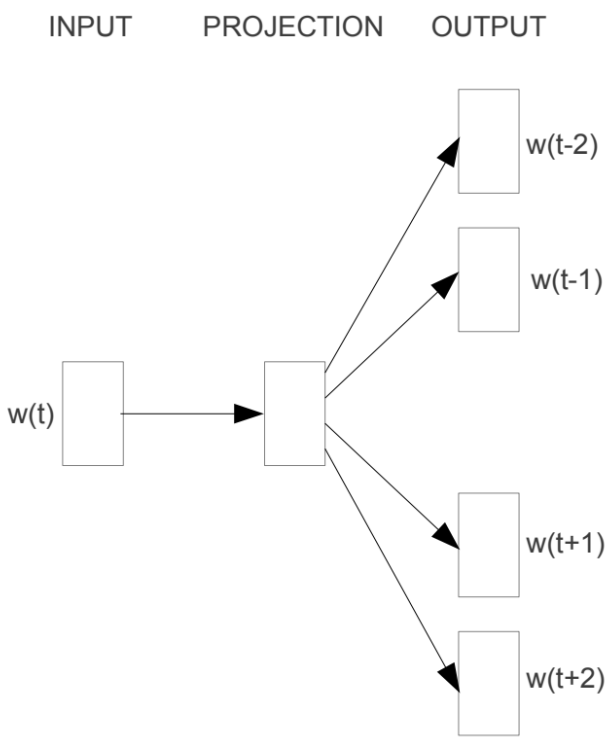
- Skip-gram, Continuous Bag of Words (CBOW) Learn embeddings as part of the process of word prediction.
- Advantages:
  - Fast, easy to train.
  - Available online in the word2vec package.
  - Including sets of pretrained embeddings!

Mikolov et al., “Distributed representations of words and phrases and their compositionality,” NeurIPS’13
Mikolov et al., “Efficient estimation of word representations in vector space,” arXiv, 2013

CBOW vs. Skip-Gram



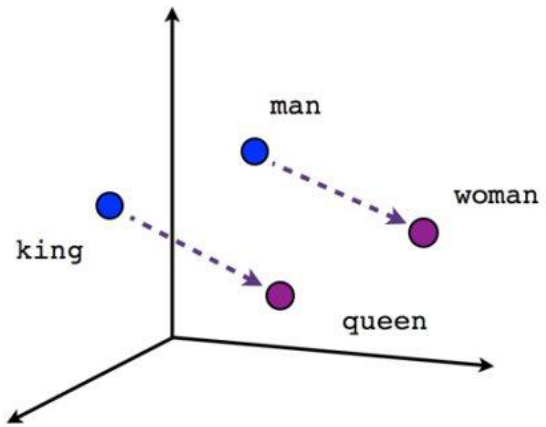
CBOW



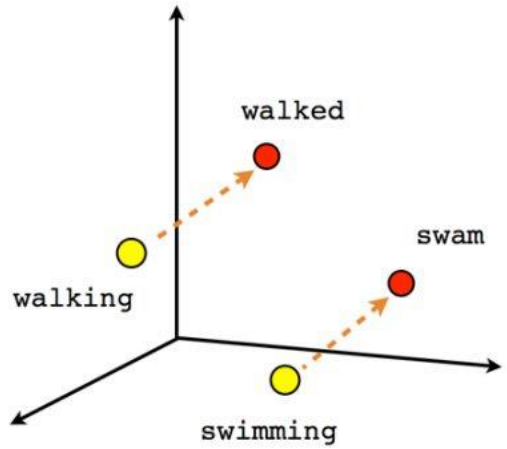
Skip-gram

CBOW vs. Skip-Gram

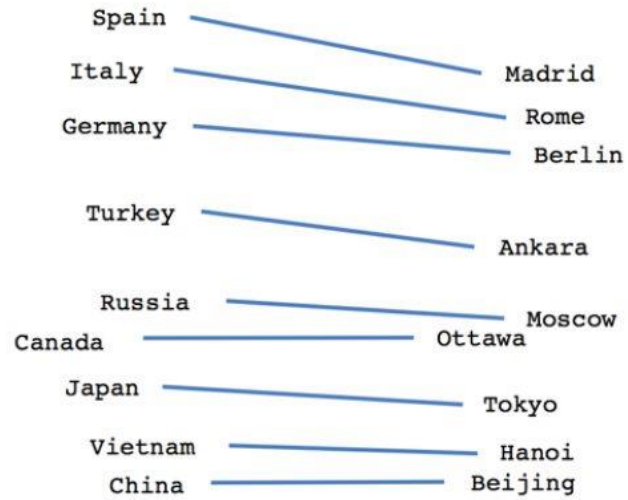
Feature	CBOW (Continuous Bag of Words)	Skip-Gram
Input	Context words	Target (center) word
Output	Target (center) word	Context words
Prediction direction	Context → Target	Target → Context
Suitable for	High-frequency words	Low-frequency words
Training speed	Faster (context is averaged)	Slower (more training samples per word)
Representation goal	Learn embeddings that predict a word from its context	Learn embeddings that predict context from a word



Male-Female



Verb tense



Country-Capital

[[king]] – [[man]] + [[woman]] = [[queen]]

[[paris]] – [[france]] + [[germany]] = [[berlin]]

Average Word Embedding

- Distributed word embeddings can benefit document ranking.
- Even simple models perform very well.
- Average Word Embedding:
  - Compute the average word embedding of the query.
  - Compute the average word embedding of candidate documents.
  - Compute similarity and rank based on that.

Average Word Embedding

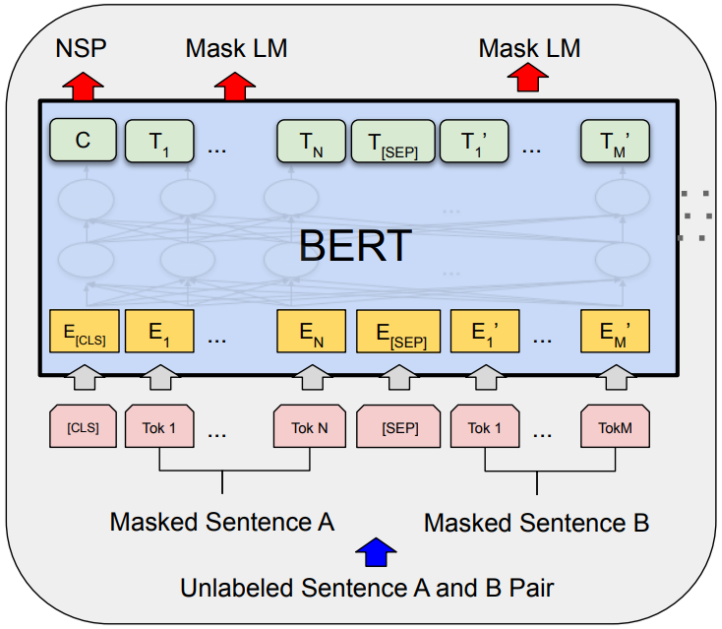
- Limitations:
  - Ignores word order
  - Context is not used
  - Polysemy (multiple meanings of the same word) cannot be resolved

“The bank of the river was flooded.”

“bank” = ??? (financial or river?)

Contextualized Word Embeddings

- Each word's meaning depends on its context.
- Embeddings are computed dynamically for each word in a sentence.
- Implemented using Transformer-based models, like BERT.



Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019: 4171-4186.

Contextualized Word Embeddings

- **Advantages:**
  - Context-sensitive
  - Captures syntax and semantics
  - Strong performance on retrieval, QA, ranking, etc.

Summary

- Local vs. distributed repsenetation.
- Representation as part of prediction.
- Capture word semantics more effectively; based on their context.
- Average word embedding.
- Contextualized Word Embeddings.

- Thanks for listening. Are there any remaining questions?

References

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In Proceedings of the International Conference on Machine Learning (ICML), pages 89–96, 2005. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

[2] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In Proceedings of the Learning to Rank Challenge, volume 14 of Proceedings of Machine Learning Research (PMLR),pages 1–24, 6 2011. URL https://proceedings.mlr.press/v14/chapelle11a.html.

[3] William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. Probabilistic retrieval based on stagedlogistic regression. In Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 198–210, 1992. doi:10.1145/133160.133199. URL https://doi.org/10.1145/133160.133199.

[4] David Cossock and Tong Zhang. Subset ranking using regression. In Proceedings of the Annual Conference on Learning Theory (COLT), pages 605–619, 2006. doi: 10.1007/11776420 44. URL https://doi.org/10.1007/11776420_44.

[5] Koby Crammer and Yoram Singer. Pranking with ranking. In Advances in Neural Information Processing Systems (NIPS), volume 14, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/5531a583481622228 0f20d1ef9e95f69-Paper.pdf.

[6] W Bruce Croft, Donald Metzler, and Trevor Strohman. Search engines: Information retrieval in practice, volume 520. Addison-Wesley Reading, 2010.

References

[7] Domenico Dato, Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. The istella22 dataset: Bridging traditional and neural learning to rank evaluation. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 3099–3107, 2022. doi: 10.1145/3477495.3531740. URL https://doi.org/10.1145/3477495.3531740.

[8] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research (JMLR), 4:933–969, 2003. ISSN 1532-4435.

[9] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. ACM Transactions on Information Systems (TOIS), 7(3):183–204, 1989. ISSN 1046-8188. doi: 10.1145/65943.65944. URL https://doi.org/10.1145/65943.65944.

[10] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, and Thomas Legrand. Applying deep learning to Airbnb search. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 1927–1935, 2019. doi: 10.1145/3292500.3330658. URL https://doi.org/10.1145/3292500.3330658.

[11] Ralf Herbrich, Thore Graepel, and Klause Obermayer. Large margin rank boundaries for ordinal regression. In Advances in Large Margin Classifiers, chapter 7, pages 115–132. The MIT Press, 1999. URL http://www.herbrich.me/papers/nips98_ordinal.pdf.

[12] Thorsten Joachims. Optimizing search engines using clickthrough data. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 133–142, 2002. doi: 10.1145/775047.775067. URL https://doi.org/10.1145/775047.775067.

[13] Tie-Yan Liu. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3):225–331, 2009. doi: 10.1561/1500000016. URL https://doi.org/10.1561/1500000016.

References

[14] Ramesh Nallapati. Discriminative models for information retrieval. In Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 64–71, 2004. doi: 10.1145/1008992.1009006. URL https://doi.org/10.1145/1008992.1009006.

[15] Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. CoRR, abs/1306.2597, 2013. URL http://arxiv.org/abs/1306.2597.

[16] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In Advances in Neural Information Processing Systems (NIPS), volume 15, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/ 51de85ddd068f0bc787691d356176df9-Paper.pdf.