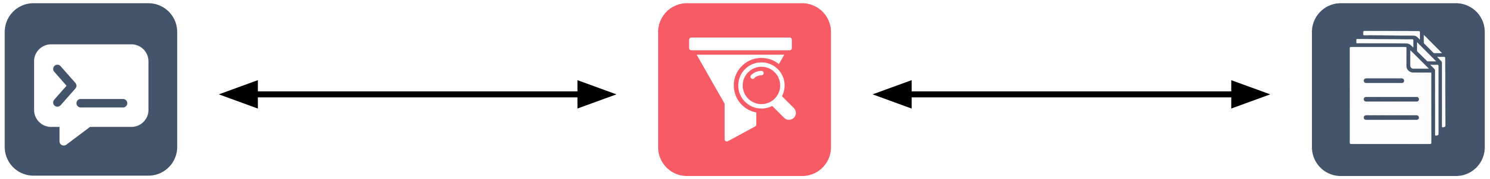
DeepLearning.AI

Module 2 introduction

Information Retrieval Foundations



Prompts
Unstructured
Conversational

Retriever
Needs to rapidly find
relevant documents
despite this mess!

Documents
Wide range of formats
Designed for humans to read

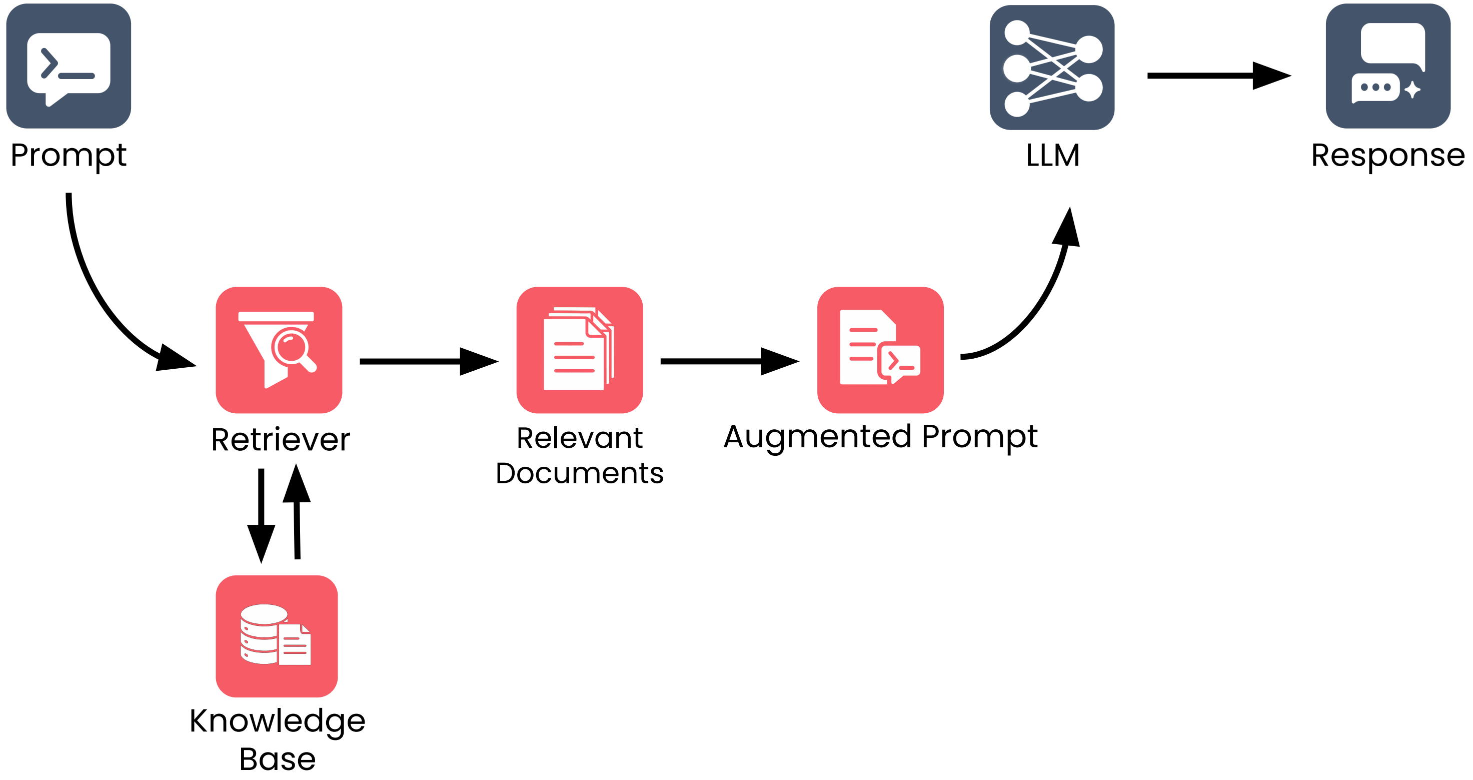
Module Topics

- Widely used retrieval techniques
- Theoretical foundations and relative strengths of each technique, plus how they’re used in combination
- Evaluation strategies
- Hands-on examples and a programming assignment

DeepLearning.AI

Retriever architecture overview

Information Retrieval Foundations


Prompt
Retriever
Knowledge Base
Relevant Documents
Augmented Prompt
LLM
Response

Two Search Approaches

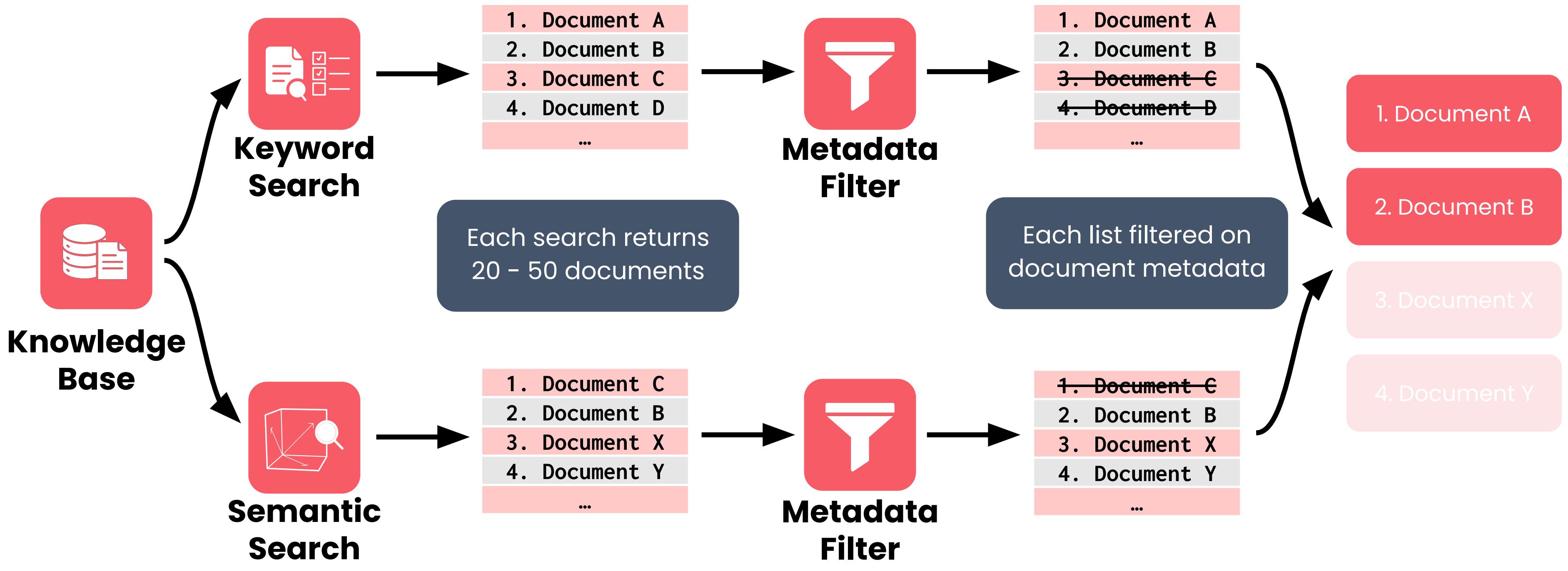
Keyword Search

Looks for documents containing the **exact words** found in the prompt.

Semantic Search

Looks for documents with **similar meaning** to the prompt.

Search Techniques



DeepLearning.AI

Zain Hasan

Hybrid Search

Keyword Search
Ensures sensitivity to exact words the user included in the prompt

Semantic Search
Finds documents with similar meaning, even without matching words

Metadata Filtering
Excludes documents based on rigid criteria

High-performing retrievers balance all three techniques based on project needs.

DeepLearning.AI

Zain Hasan

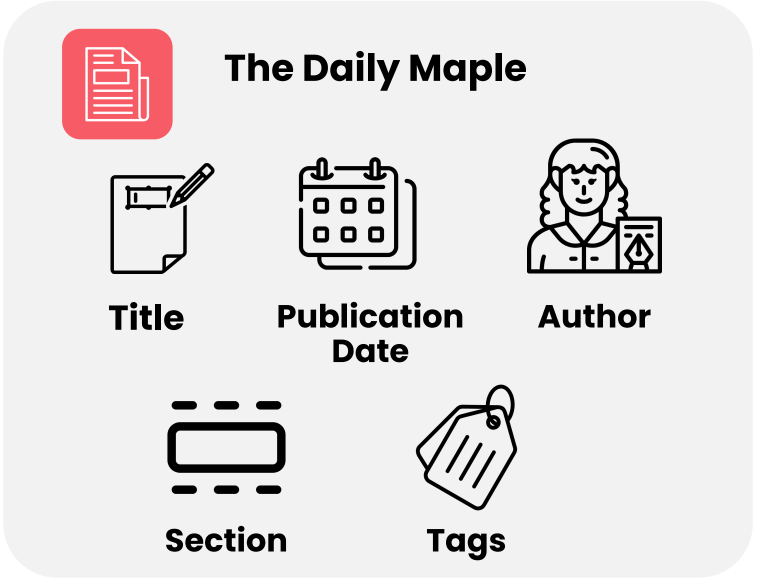
DeepLearning.AI

Metadata filtering

Information Retrieval Foundations

Metadata Filtering

Uses rigid criteria to narrow down documents based on metadata like title, author, creation date, access privileges, and more.



```
SELECT * FROM articles
WHERE publication_date = '2023-10-01';
```

Return all articles published on specific date

```
section = "Opinion"
author = "Michael Chen"
date = June to July 2024
```

Metadata Filtering In RAG

Metadata filtering doesn't perform retrieval, it **narrows down results from other techniques** based on user attributes, not query content.

Subscription Filtering

Filter: Exclude all articles with metadata "subscription = paid"

Geographic Filtering

Filter: Include only articles with metadata "region = North America"

Advantages and Limitations

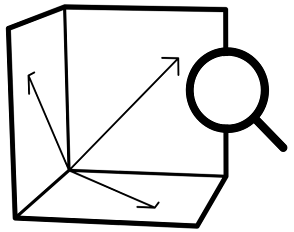
Pros	Cons
• Simple to understand and debug	• Not true search
• Fast, optimized, mature, and reliable	• Rigid, ignores content, and provides no way for ranking
• Enforces strict retrieval rules, matching exact filter criteria	• Useless alone

DeepLearning.AI

Keyword search - TF-IDF

Keyword Search TF-IDF

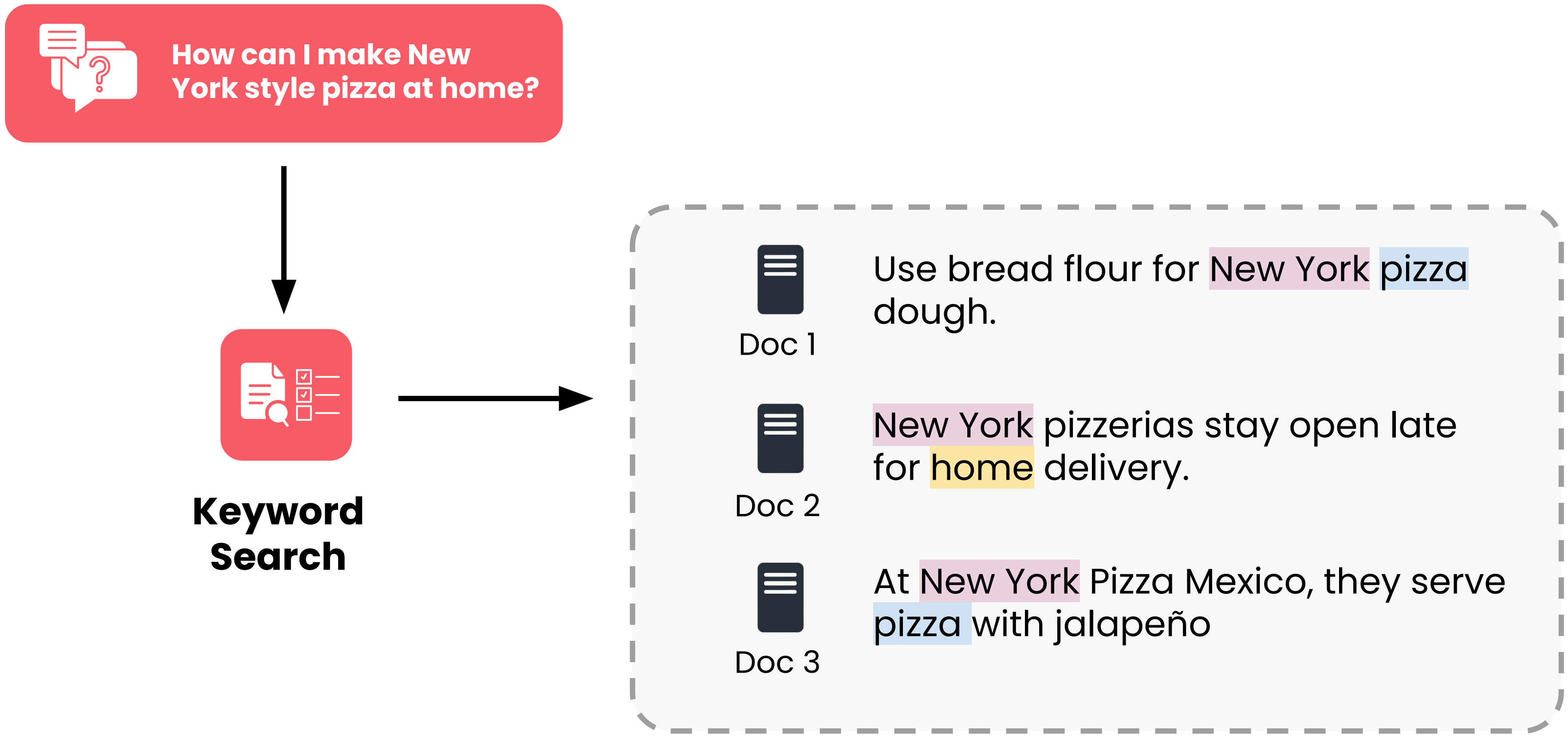
Introduction to Keyword Search



Keyword Search

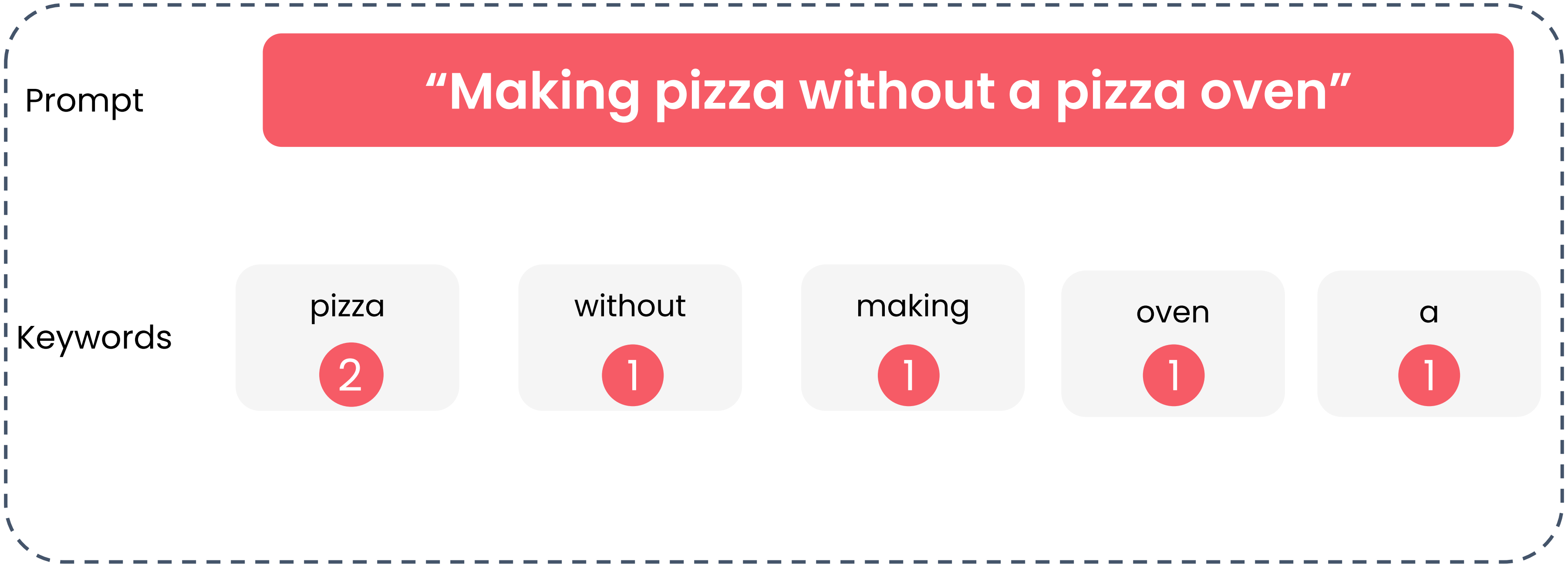
⚠️ **Decades of proven simplicity**

The simplicity and effectiveness that has powered retrieval for decades makes keyword search a key component of modern RAG systems.


How can I make New York style pizza at home?
Keyword Search
Doc 1
Use bread flour for New York pizza dough.
Doc 2
New York pizzerias stay open late for home delivery.
Doc 3
At New York Pizza Mexico, they serve pizza with jalapeño

Bag of Words

Word order is ignored, only word presence and frequency matter

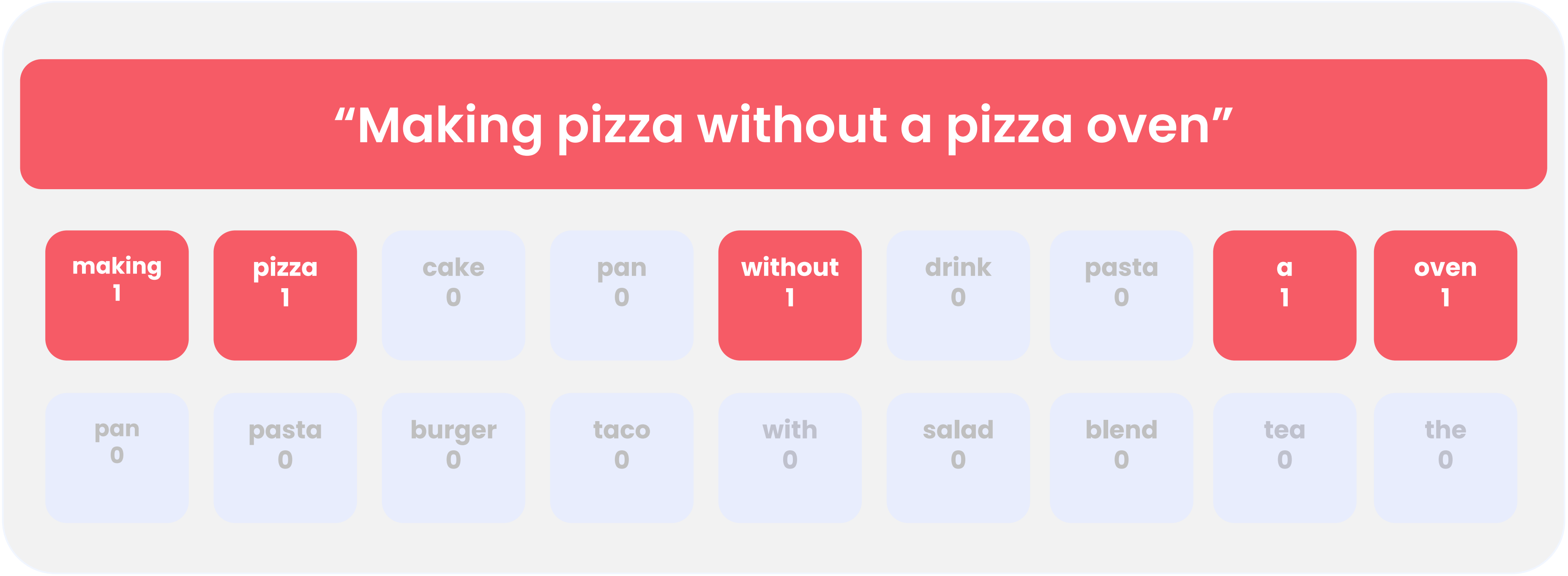


Bag of words

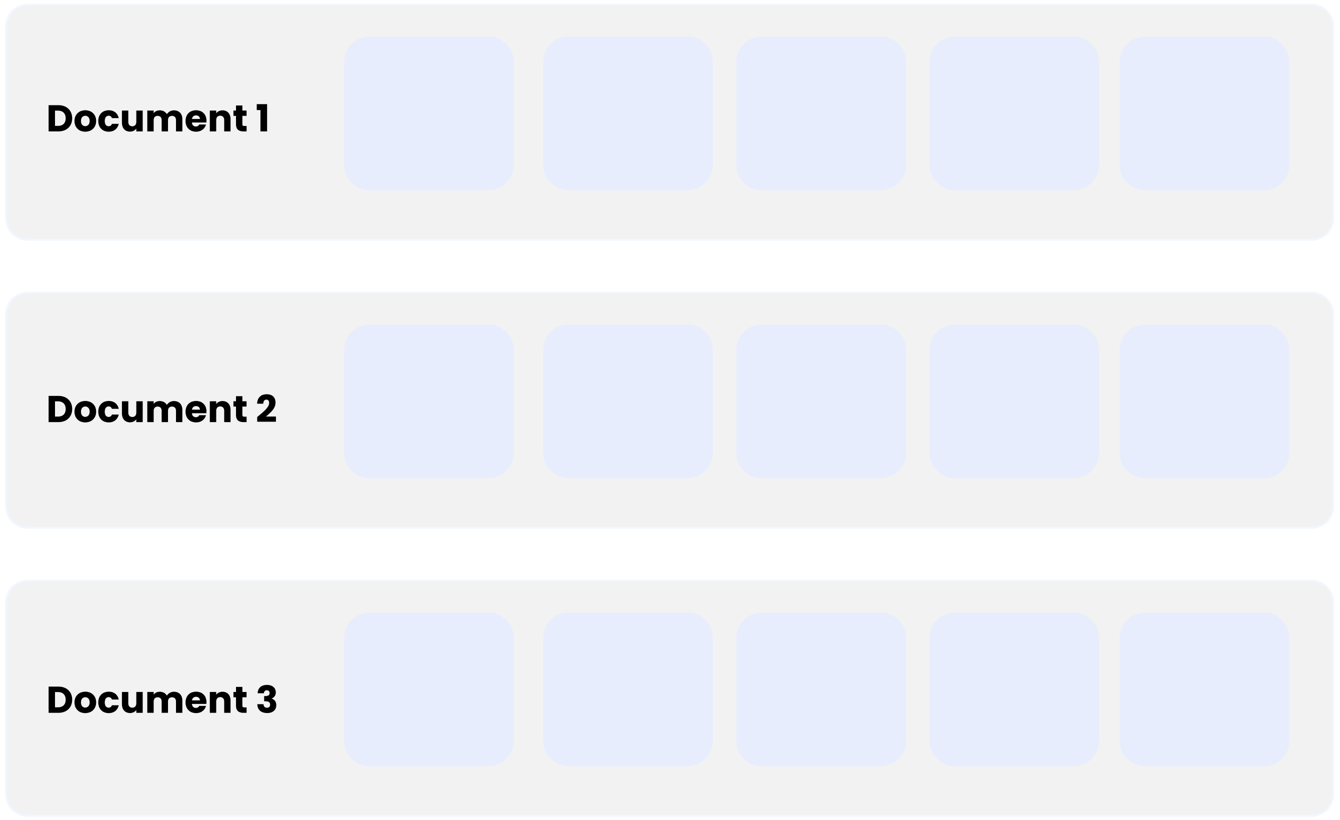
DeepLearning.AI

Zain Hasan

Sparse Vectors



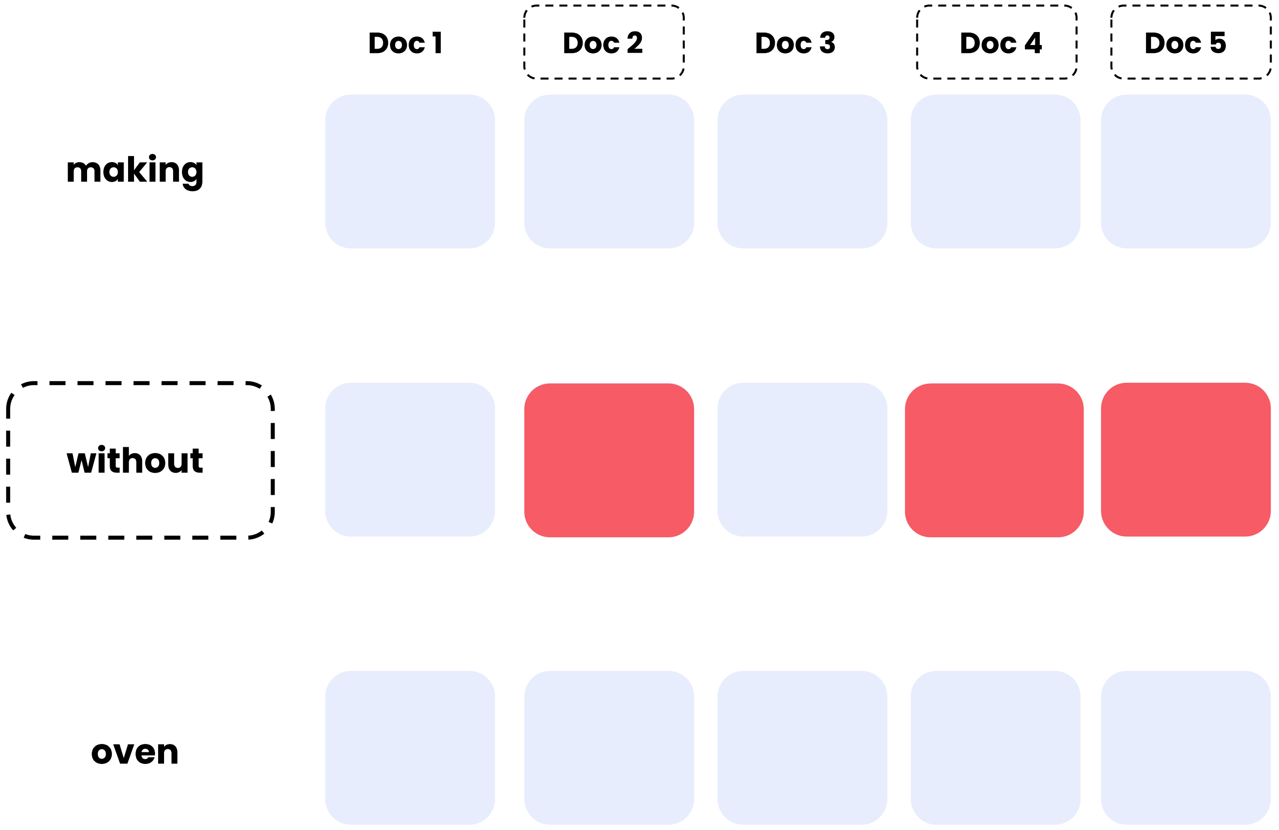
Most words aren't used. The bag of words is sparse, with few non-zero entries.

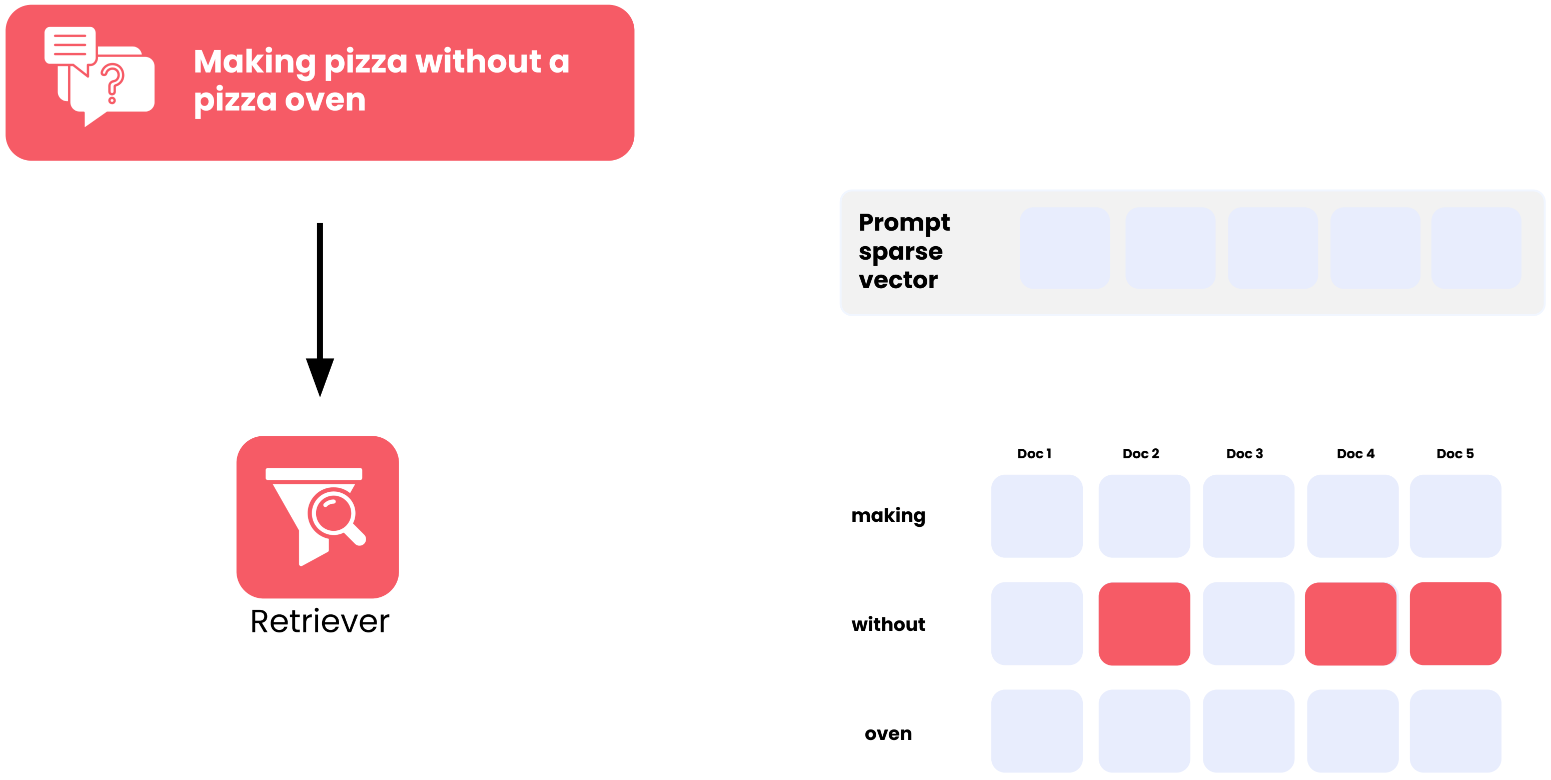

Document 1
Document 2
Document 3

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
making	⬜	⬜	⬜	⬜	⬜
without	⬜	🟥	⬜	🟥	🟥
oven	⬜	⬜	⬜	⬜	⬜

DeepLearning.AI

Zain Hasan


Doc 1
Doc 2
Doc 3
Doc 4
Doc 5
making
without
oven



	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
making					
without		🟥		🟥	🟥
oven					

“Making pizza without a pizza oven”

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
making					
pizza					
without					
a					
oven					

“Making pizza without a pizza oven”

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
making	+1			+1	+1
pizza					
without					
a					
oven					

“Making pizza without a pizza oven”

	5/5	3/5	1/5	4/5	4/5
	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**	**Doc 5**
making	+1			+1	+1
pizza	+1		+1		
without	+1	+1		+1	+1
a	+1	+1		+1	+1
oven	+1	+1		+1	+1

Frequency Based Scoring

Example Query: pizza oven

Document 1

Homemade pizza in oven is better than frozen pizza

Contains: Pizza (2x) Oven (1x)

Document 2

Wood-fired oven is a better oven than a stone oven for cooking pizza

Contains: Pizza (1x) Oven (3x)

Simple Scoring = 2 points

TF Scoring = 3 points

Simple Scoring = 2 points

TF Scoring = 4 point

DeepLearning.AI

Zain Hasan

Normalized TF Scoring

Longer documents may contain **keywords many times** simply because they are **longer**.

Solution: Normalize by document length

Score = (Number of keyword occurrences) / (Total words in document)

TF-IDF

Basic TF scoring treats **all words equally,** whether they're common filler words or rare, meaningful terms.

Solution: Weight terms using "inverse document frequency" (IDF).

Score = TF(word, doc) × log(Total docs / Docs containing word)

1 **Count Documents**

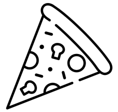
For each word **making** **pizza**

$$\frac{\text{docs word appears in}}{\text{total docs}}$$

DeepLearning.AI

Zain Hasan

1 Count Documents



Pizza

Appears in 5 out of 100 documents

DF = 5/100 `0.05`

T

The

Appears in all 100 documents

DF = 100/100 `1.0`

1 **Count Documents**

Pizza Appears in 5 out of 100 documents

DF = 5/100 `0.05`

The Appears in all 100 documents

DF = 100/100 `1.0`

2 **Flip to reward rare words**

Pizza High Score

IDF = 1/0.05 `20`

The Low Score

DF = 1/1.0 `1.0`

DeepLearning.AI

Zain Hasan

1 Count Documents

Pizza
Appears in 5 out of 100 documents
DF = 5/100 `0.05`

The
Appears in all 100 documents
DF = 100/100 `1.0`

2 Flip to reward rare words

Pizza
High Score
IDF = 1/0.05 `20`

The
Low Score
DF = 1/1.0 `1.0`

3 Apply log

Pizza
Still higher
log(1/20 **)** `1.30`

The
Too common, no weight
log(1 **)** `0`

TF

- **making:** 0.4 (appears in 3/5 docs)
  moderately common
- **pizza:** 0.7 (appears in 2/5 docs)
  less common
- **without:** 0.2 (appears in 4/5 docs)
  very common
- **a:** 0.1 (appears in 4/5 docs)
  very common
- **oven:** 0.2 (appears in 4/5 docs)
  very common

TF-IDF

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
making	0.4			0.4	0.4
pizza	0.7		0.7		
without	0.2	0.2		0.2	02
a	0.1	0.1		0.1	0.1
oven	0.2	0.2		0.2	0.2

“Making pizza without a pizza oven”

High Scoring Document

Pizza	1.3
oven	1.1

Low Scoring Document

a	0
without	0.1

Documents with **rare keywords** score **higher** than documents with common words

DeepLearning.AI

Zain Hasan

TF-IDF
Refined into
BM25

Modern systems use a slightly refined version called BM25

DeepLearning.AI

Keyword search - BM25

Information Retrieval Foundations

BM25 Scoring

BM25 (Best Matching 25) was named as the 25th variant in a series of scoring functions proposed by its creators.

$$IDF * \frac{TF * (k_1 + 1)}{TF + k_1 * (1 - b + b * (\frac{\text{document length}}{\text{average document length}}))}$$

- This gives the score for a **single keyword**
- **Sum scores** across all keywords for total relevance score for a document

1 Term Frequency Saturation

✗ TF-IDF

"pizza" 10 times = Score X
"pizza" 20 times = Score 2X

✓ BM25

"pizza" 10 times = Score X
"pizza" 20 times = Score 1.3X

Term Frequency Saturation

2 Document Length Normalization

✗ TF-IDF

Short Doc = Good Score
Long Doc = Heavy Penalty

Too aggressive

✓ BM25

Short Doc = Good Score
Long Doc = Smaller Penalty

Document length normalization

BM25 Tunable Parameters

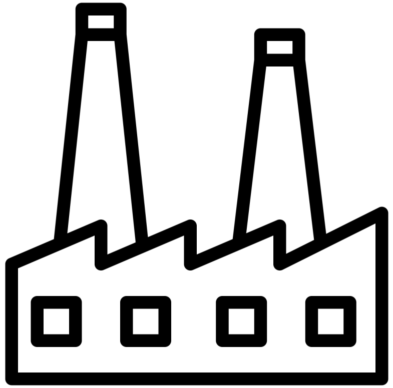
k_1 - Term Frequency Saturation

- **Controls:** How much term frequency influences the score.
- **Range:** Typically between 1.2 and 2.0.
- **Effect:** Higher values increase the impact of term frequency; lower values reduce it.

b - Length Normalization

- **Controls:** The degree of normalization for document length.
- **Range:** Between 0 (no normalization) and 1 (full normalization).
- **Effect**: Balances favoring shorter vs. longer documents.

TF-IDF vs BM25



BM25 = Standard keyword search algorithm in production retrievers

better performance + **same cost** + **more flexibility** than TF-IDF

Keyword Search Overview

Match documents by **keyword frequency**

$$\begin{tikzcd}\text{Sparse Vectors} \arrow[r] & \text{Scored} \arrow[r] & \text{Ranked}\end{tikzcd}$$

TF-IDF

- Keyword rarity
- Term frequency
- Document length

BM25

`Most commonly used`

- Document length normalization
- Term Frequency Saturation

Keyword Search Strengths

Simplicity

Guaranteed keyword matching

DeepLearning.AI

Semantic search introduction

Information Retrieval Foundations

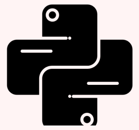
In order to improve on lexical search it's necessary to capture not only the presence of words, but their **meaning.**

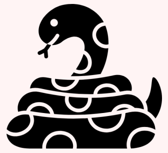
Cannot match synonyms

"happy"

"glad"

Incorrectly matches different meanings





DeepLearning.AI

Zain Hasan

Semantic Search vs. Keyword Search

- Prompt and documents each get a vector
- Vectors compared to generate scores
- The main difference is how vectors are assigned
    - **Keyword Search:** count words
    - **Semantic Search:** use embedding model

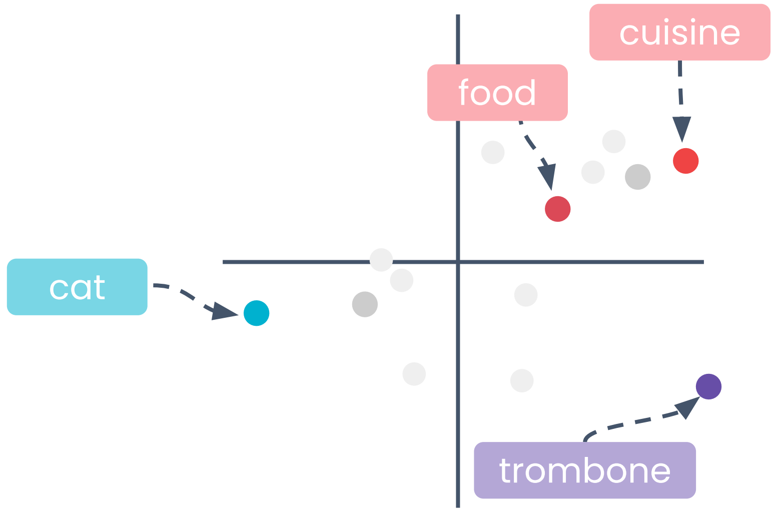
Understanding Embedding Models

Embedding models map tokens, to a location in space. This location is represented by a **vector.**

"Pizza" $\longrightarrow$ vector [3, 1]

"Bear" $\longrightarrow$ vector [5, 2]

In two dimensions, these can be represented as points



Vector Space

DeepLearning.AI

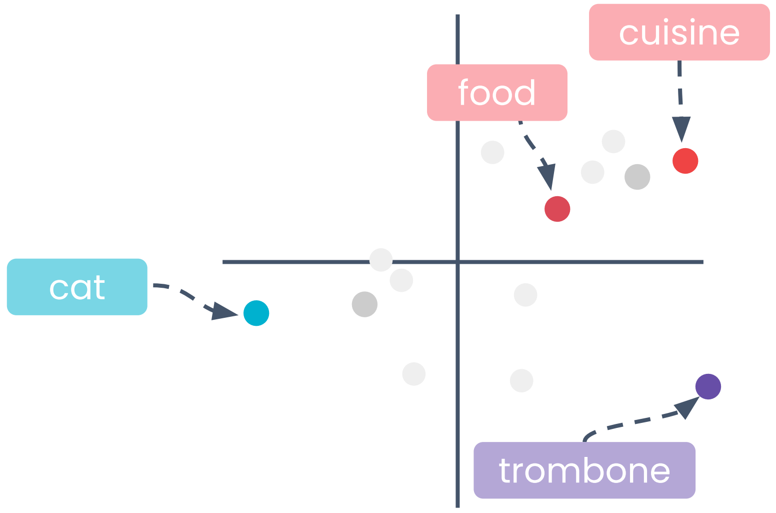
Zain Hasan

Understanding Embedding Models

No simple interpretation of X and Y axis

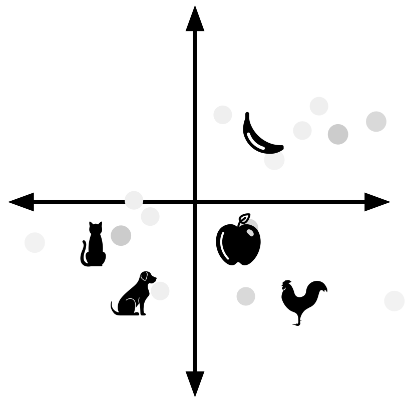
...instead...

points "float around" in space and similar words cluster together

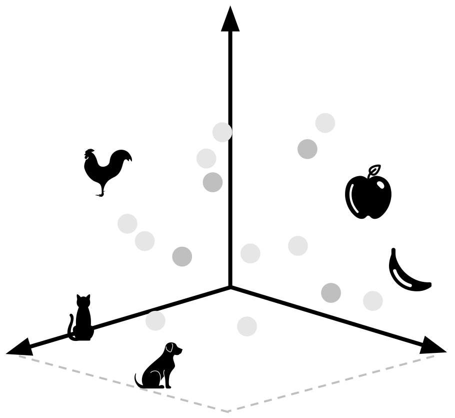


Vector Space

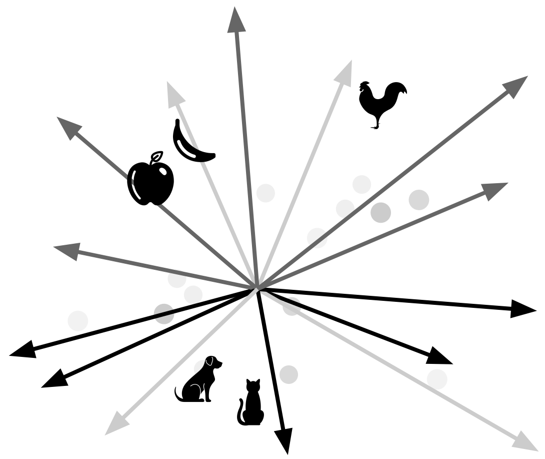
Understanding Embedding Models



2 dimensions



3 dimensions



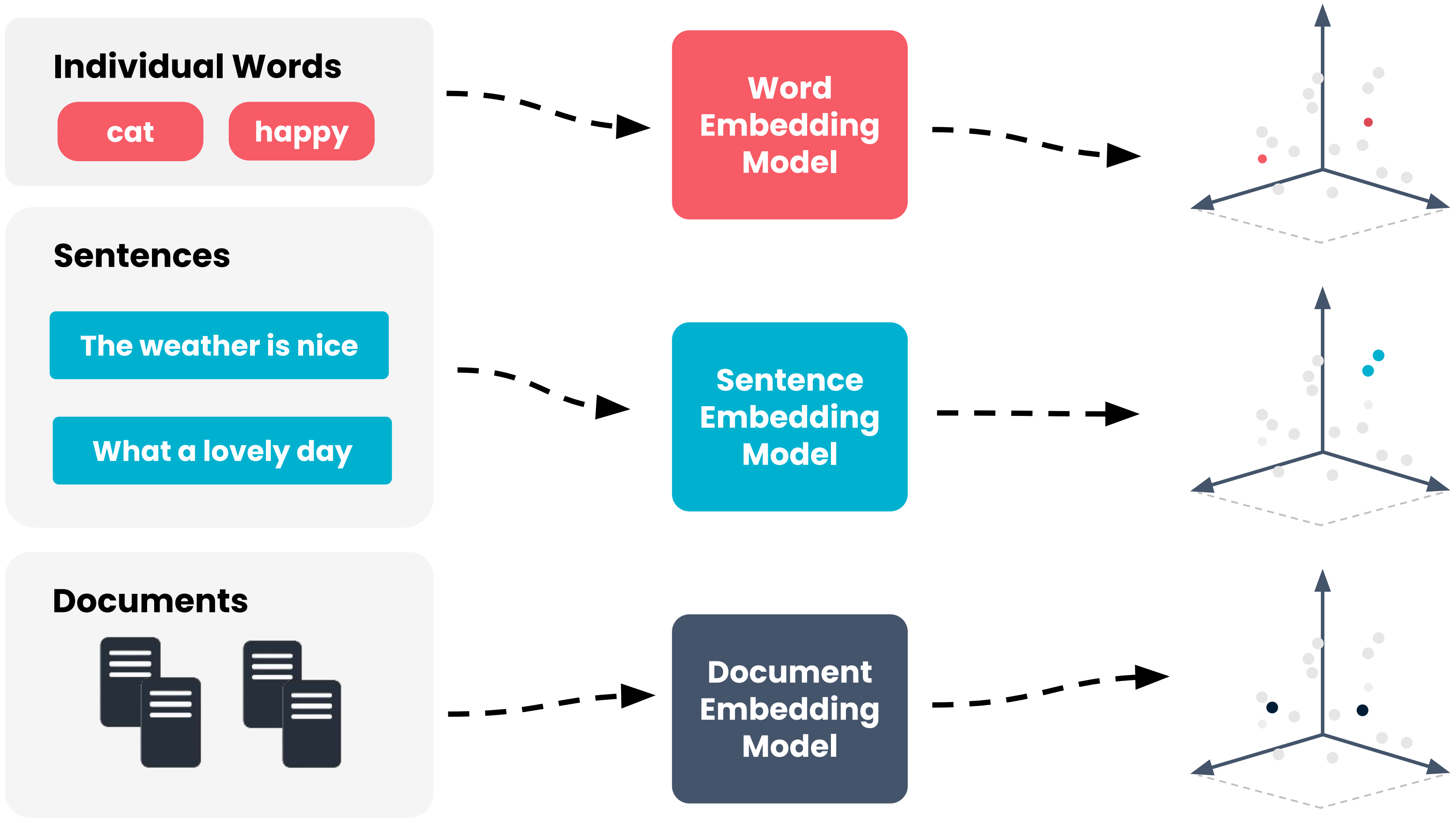
100 - 1,000+ dimensions

More dimensions means more room to form clusters and capture nuanced relationships

Same principles hold
Close vectors, similar meanings

DeepLearning.AI

Zain Hasan



DeepLearning.AI

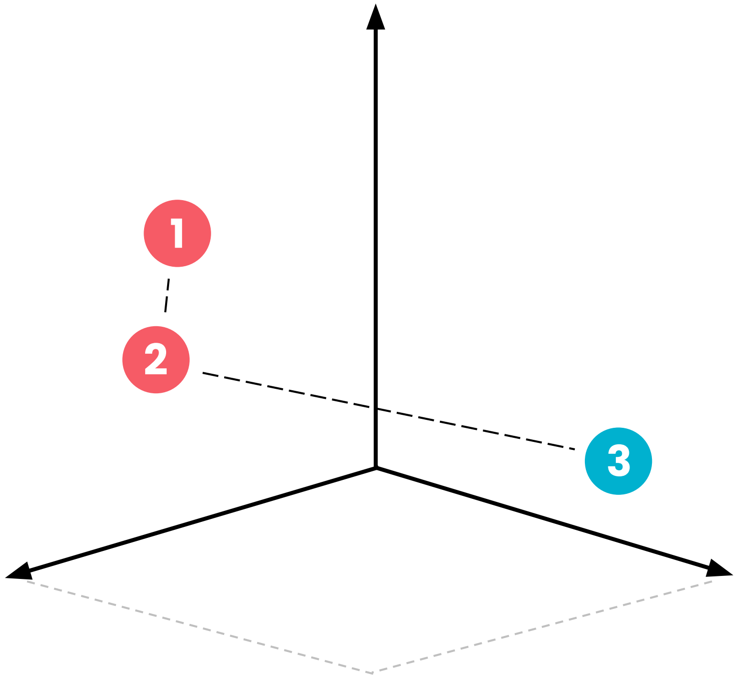
Zain Hasan

Sentence Embedding Example

1 He spoke softly in class

2 He whispered quietly during class

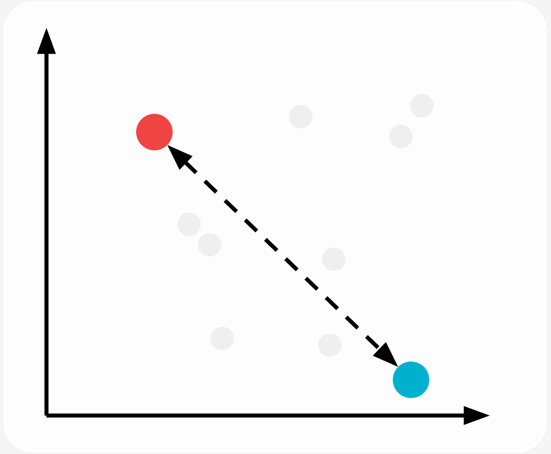
3 Her daughter brightened the gloomy day



DeepLearning.AI

Zain Hasan

Measuring Vector Distance



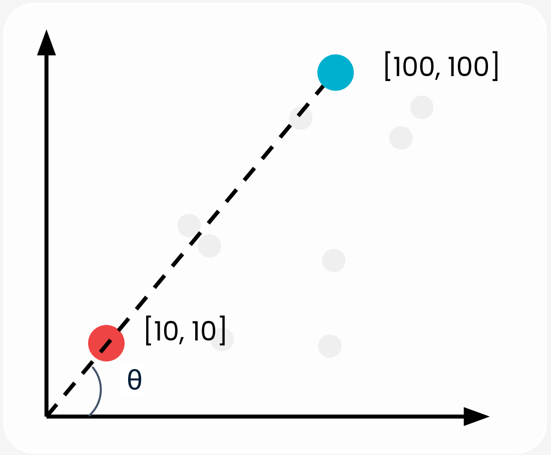
Euclidean Distance

Measures how far apart two vectors are by drawing a straight line from one vector to the other - the shortest possible distance between them.

DeepLearning.AI

Zain Hasan

Measuring Vector Distance

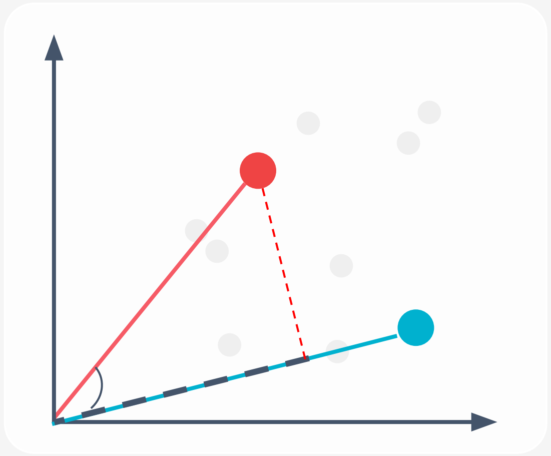


Cosine Similarity

Looks at the similarity in the **direction** of two vectors, regardless of whether they're close to one another in space

-1	0	1
Opposite direction	Perpendicular	Same direction

Measuring Vector Distance

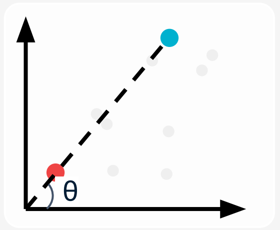


Dot product

Measures the length of the projection of one vector onto another.

`Negative`	`0`	`Positive`
Opposite direction	Perpendicular	Same direction

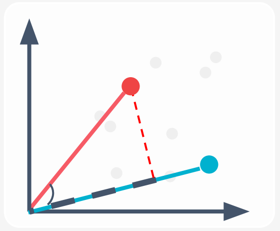
Measuring Vector Distance



Cosine Similarity

-1	0	1
Opposite direction	Perpendicular	Same direction

Higher values, closer vectors



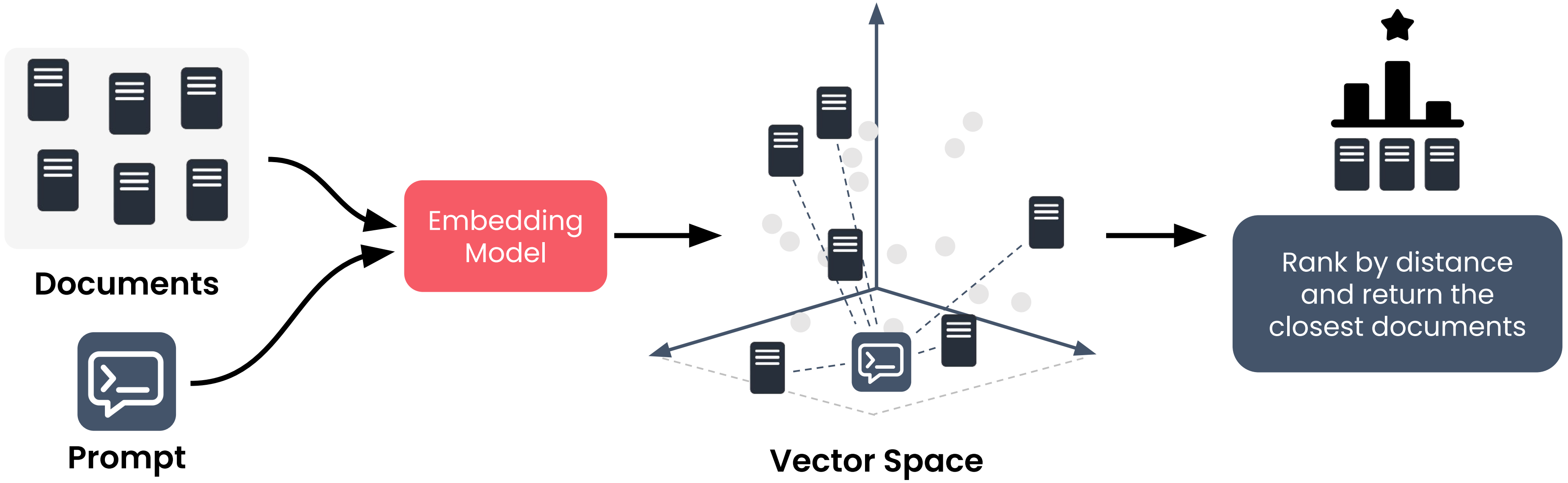
Dot product

Negative	0	Positive
Opposite direction	Perpendicular	Same direction

DeepLearning.AI

Zain Hasan

Semantic Search



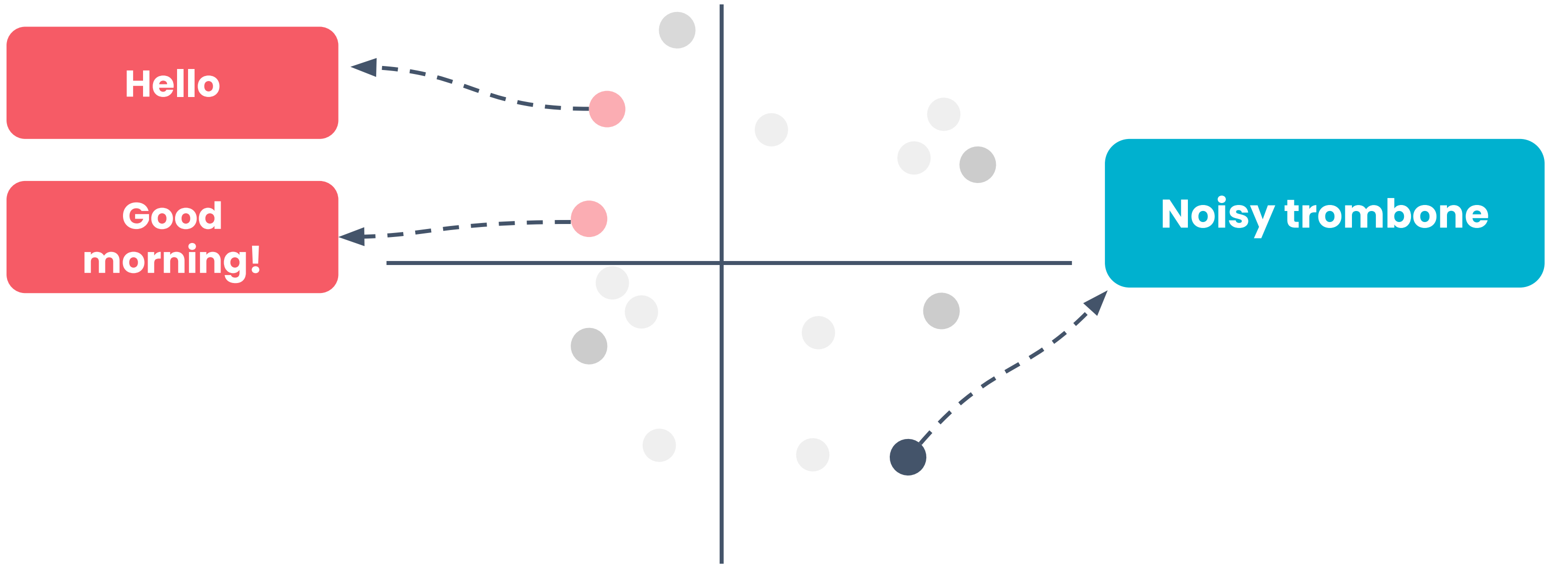
Zain Hasan

DeepLearning.AI

Semantic search embedding model deepdive

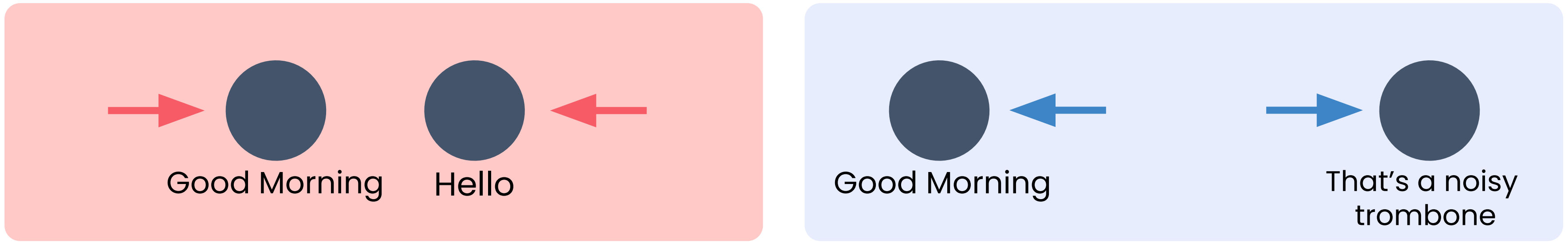
Information Retrieval Foundations

Embedding Models



How can an embedding model know to place similar text together, and dissimilar text farther apart?

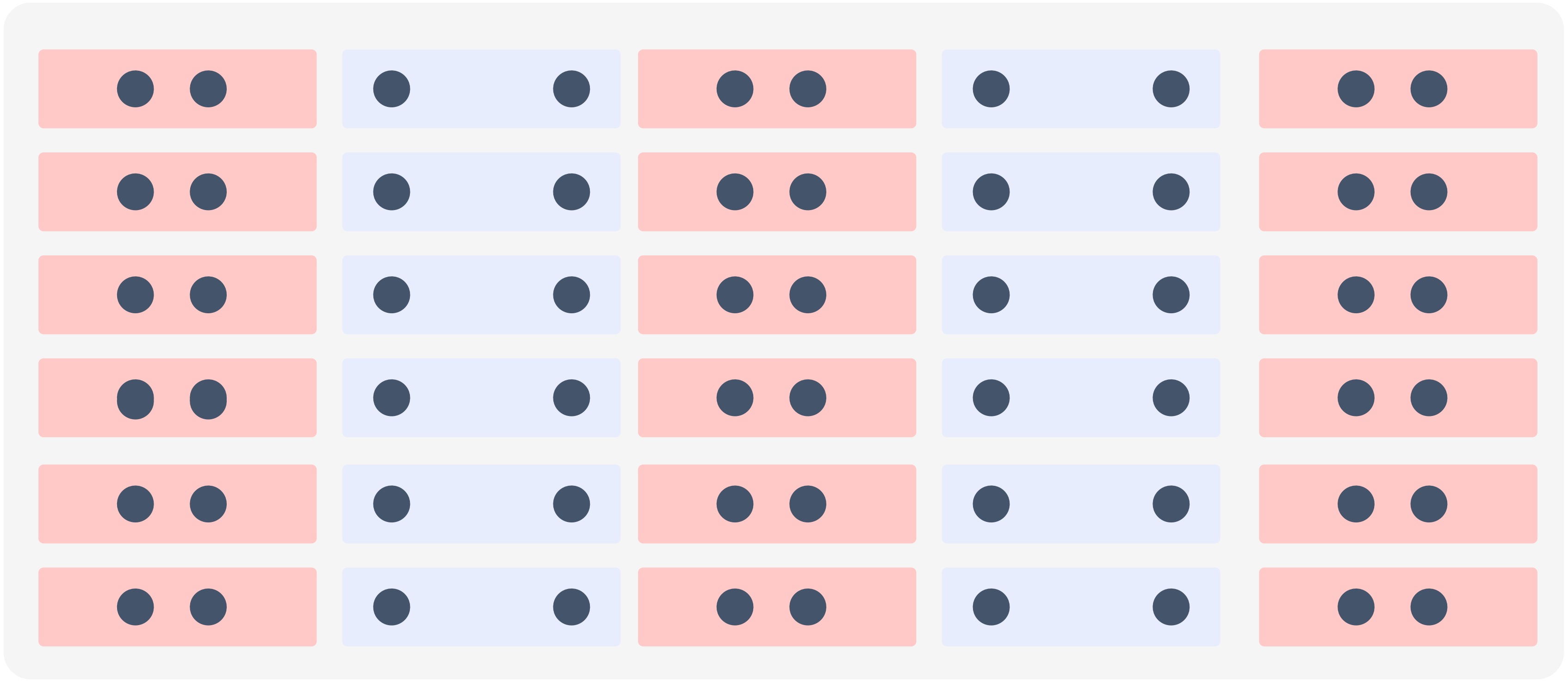
Positive and Negative Examples in Training



Positive Pair

Negative Pair

Positive and Negative Examples in Training

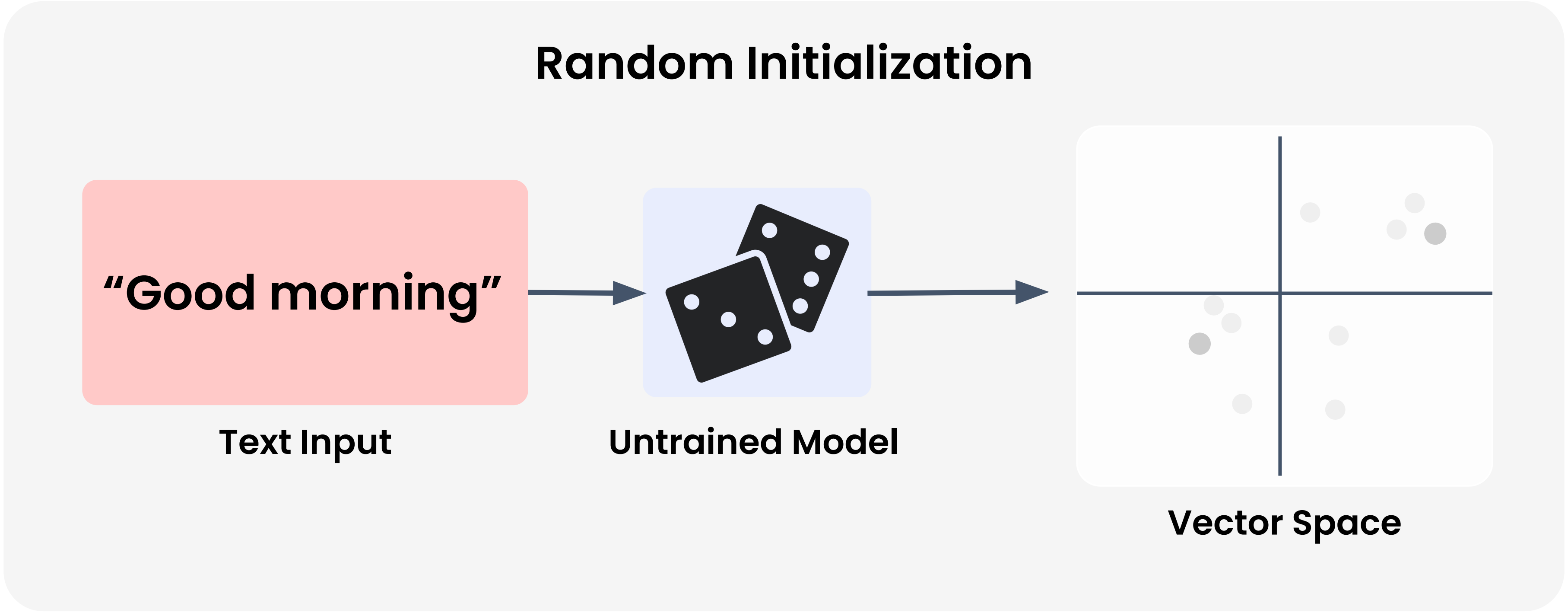


Compile massive training dataset of positive and negative pairs

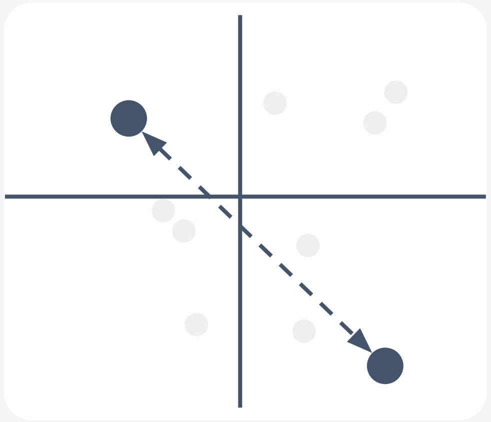
DeepLearning.AI

Zain Hasan

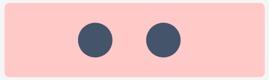
Initial Random Vectors in Embedding Models



Contrastive Training Process



Look at where each pair was placed in vector space

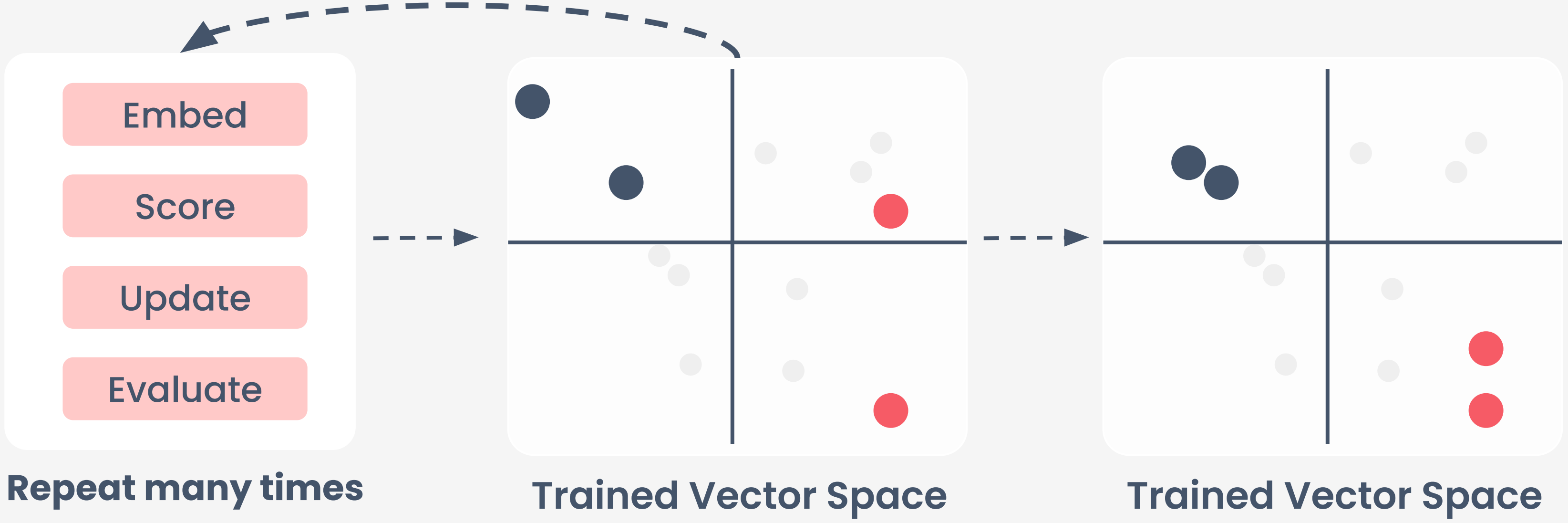


Scores better when closer together

Scores better when farther apart

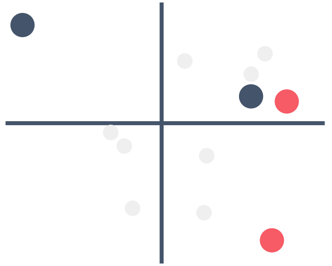
Score Results

Contrastive Training Process

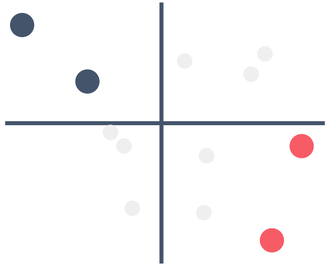


Contrastive Training Process

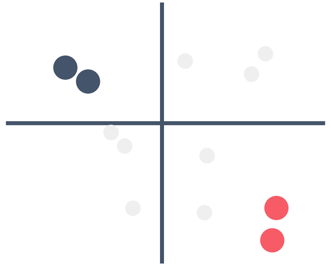
- Update internal parameters based on scoring the positive and negative pairs
- Repeat the process: Embed → Score with Pairs → Update parameters
- Iteratively repeat the process, improving the model



Early in training

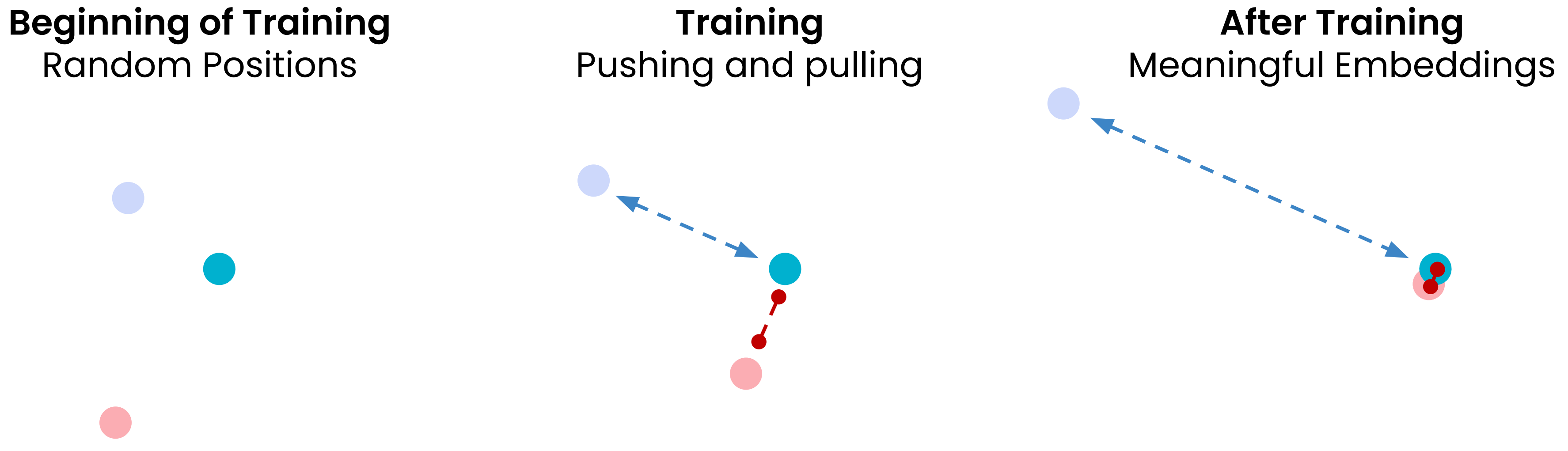


Mid-training



Trained model

Contrastive Training Process



● "He could smell the roses" ● "A field of fragrant flowers" ● "The lion roared majestically"

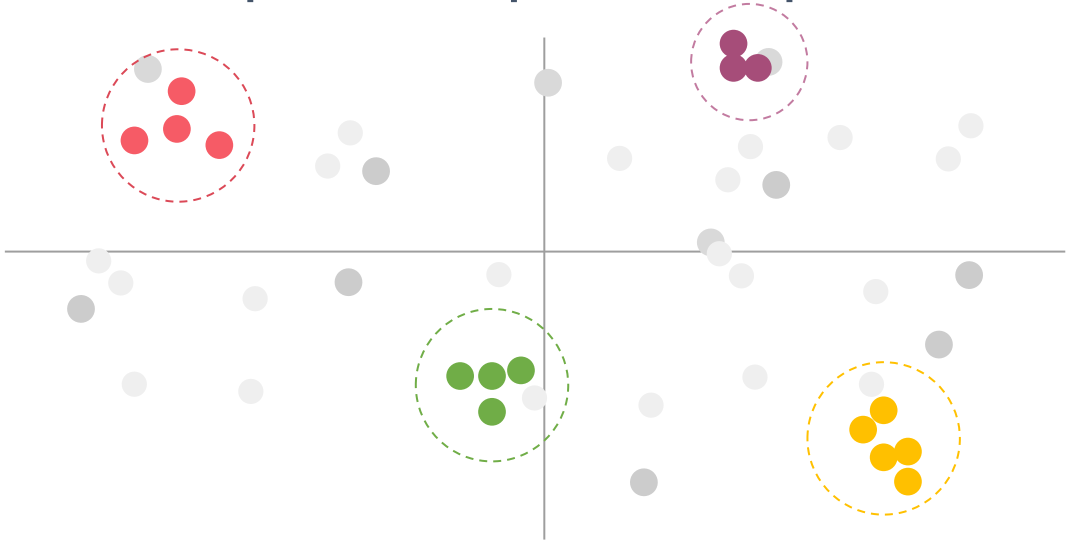
DeepLearning.AI

Zain Hasan

Scaling up Contrastive Learning

- In reality every vector is simultaneously pushed and pulled in many directions
- Using 100s or 1,000s of dimensions creates more space in which to push and pull vectors
- Eventually vectors pulled near similar words or text

Millions of positive and negative pairs with complex relationships!



DeepLearning.AI

Zain Hasan

Key Takeaways

- Semantic vectors are abstract and somewhat random
- Before training: locations in space have no meaning
- After training: locations have meaning because clusters of similar text have formed
- Only compare vectors from same embedding model

DeepLearning.AI

Vector embeddings in RAG

Hybrid Search in Information Retrieval

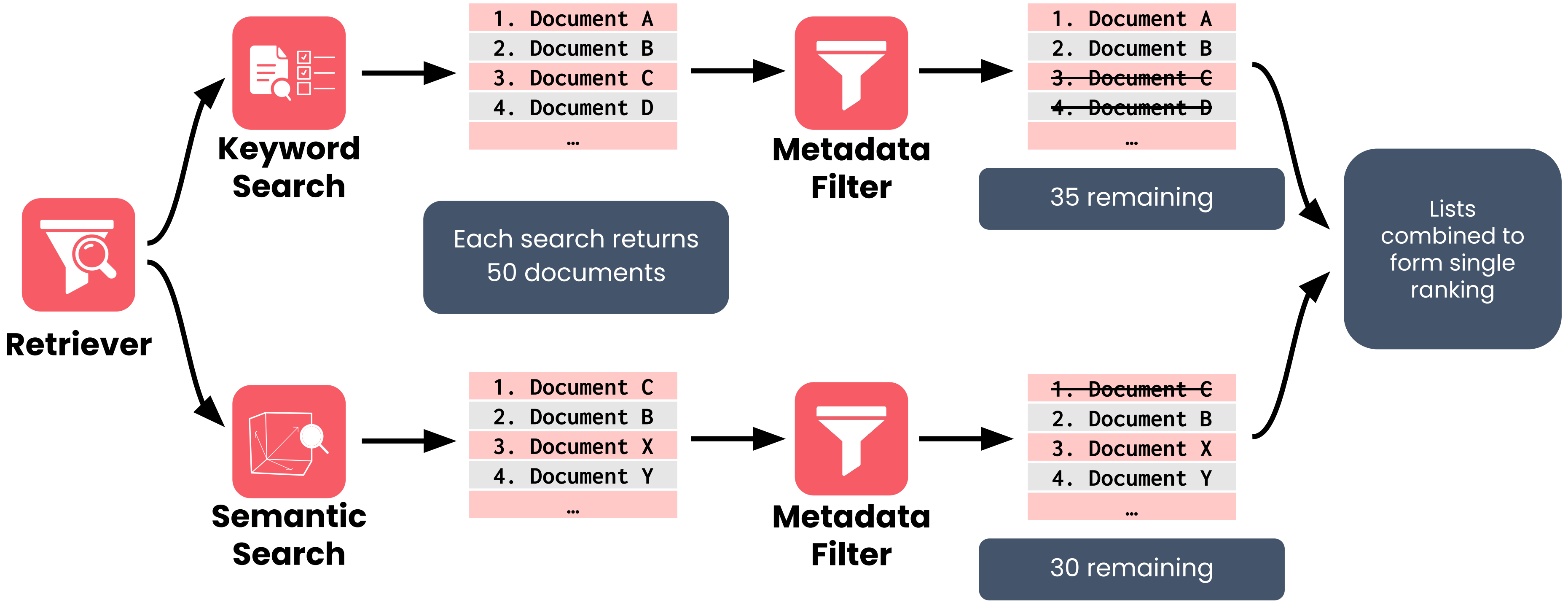
Key Strategies

Metadata Filtering
Uses rigid criteria stored in document metadata to narrow down search results
Fast, easy, yes-no filter, but can't be used alone

Keyword Search
Scores documents based on having the same keywords found in the prompt
Fast, performs especially well when keywords matter, but relies on exact matches

Semantic Search
Scores and ranks documents based on having similar meaning to the prompt
Slower, computationally expensive, but more flexible

Hybrid Search



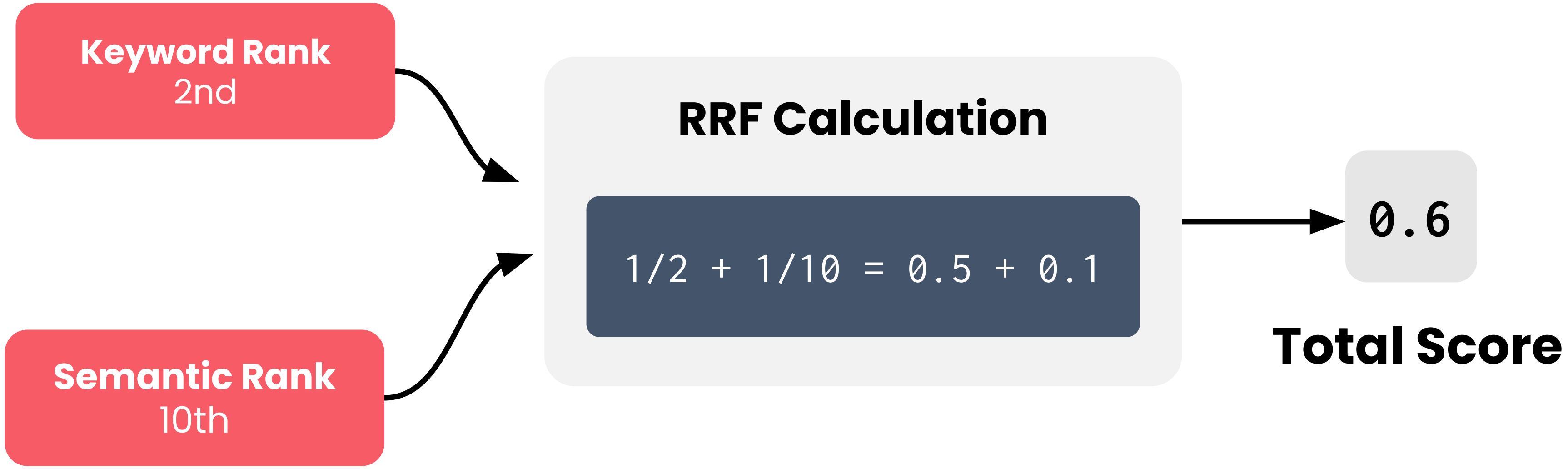
DeepLearning.AI

Zain Hasan

Reciprocal Rank Fusion

- Rewards documents for being highly ranked on each list
- Control weight of keyword vs. semantic ranking
- Score points equal to reciprocal of ranking
  1st = 1 point, 2nd = 0.5 points, etc.
- Total points from all ranked list used to perform final ranking

$$\frac{1}{k + \text{rank in list 1}} + \frac{1}{k + \text{rank in list 2}} + \ldots + \frac{1}{k + \text{rank in list n}}$$



DeepLearning.AI

Zain Hasan

Reciprocal Rank Fusion

$$\frac{1}{k + \text{rank in list 1}} + \frac{1}{k + \text{rank in list 2}} + \dots + \frac{1}{k + \text{rank in list n}}$$

When k = 0

Top ranked document shoots to top of overall ranking
1st vs 10th: 10x difference

When k = 50

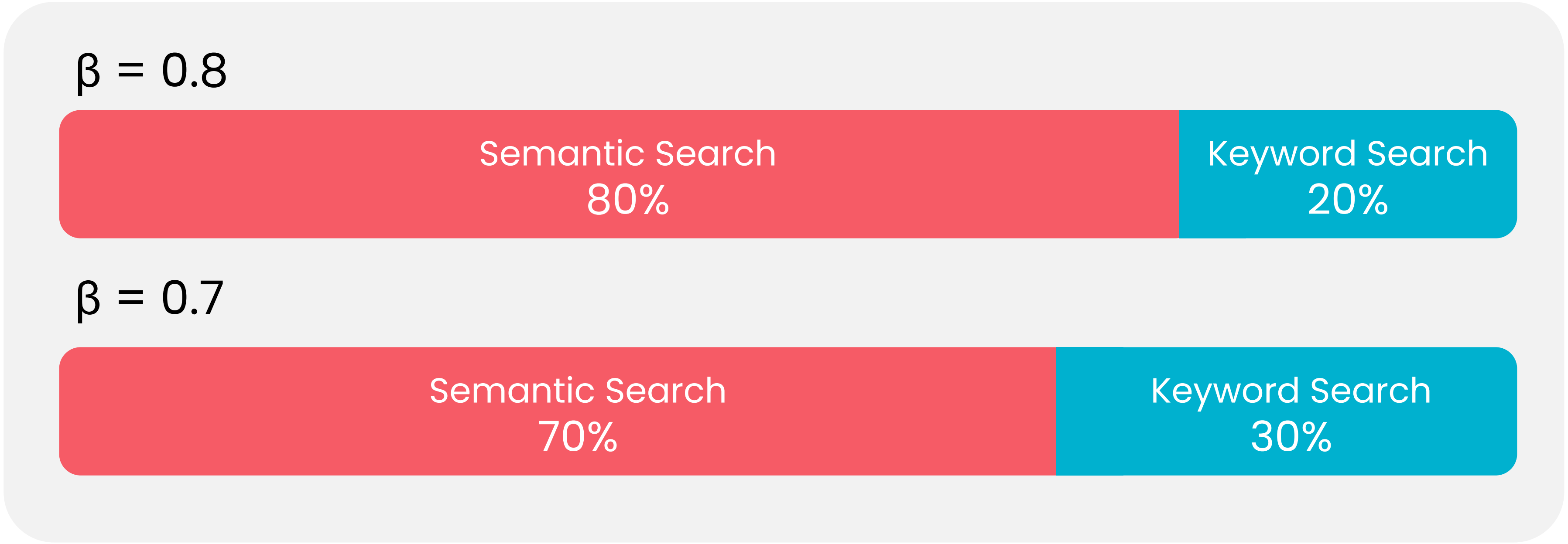
Single high rank doesn’t dominate overall ranking
1st vs 10th: 1.2x difference

RRF only cares about ranks, not scores

DeepLearning.AI

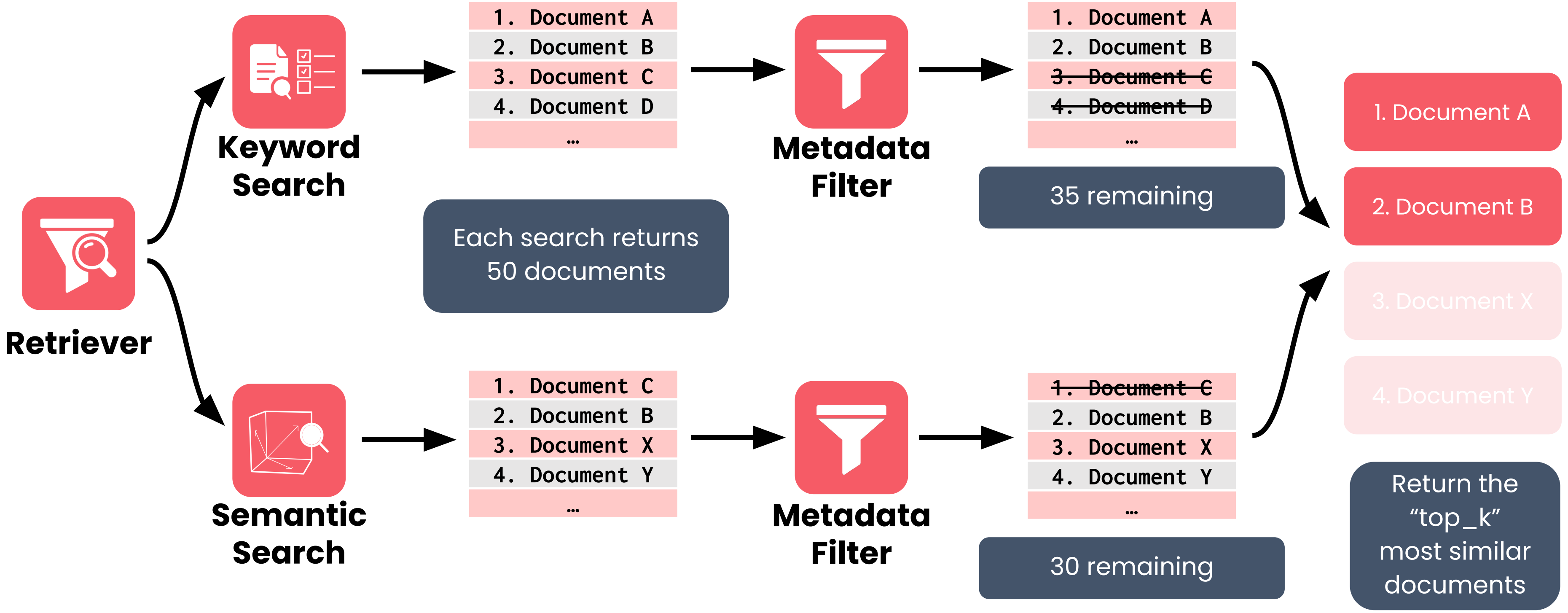
Zain Hasan

Beta: Weighting Semantic vs. Keyword



If exact keyword matching is important, set a lower beta

Hybrid Search



DeepLearning.AI

Zain Hasan

DeepLearning.AI

Hybrid search

Information Retrieval Foundations

Retrieval Quality Metrics

Common ingredients to most retriever quality metrics:

The Prompt
The specific prompt being evaluated

Ranked Results
Documents returned in ranked order

Ground Truth
All documents labeled as relevant or irrelevant

If you want to evaluate your retriever
you need to know the correct answers

Precision and Recall

Precision
Measures how many of the returned documents are relevant

```
Relevant Retrieved / Total Retrieved
```

Recall
Measures how many of the relevant documents are returned

```
Relevant Retrieved / Total Relevant
```

Example

10 Relevant Documents in Knowledge Base

First Run

Retrieved: 12 Documents

Relevant: 8 Documents

Precision (8/12) 66%

Recall (8/10) 80%

Second Run

Retrieved: 15 Documents

Relevant: 9 Documents

Precision (9/15) 60% ↓

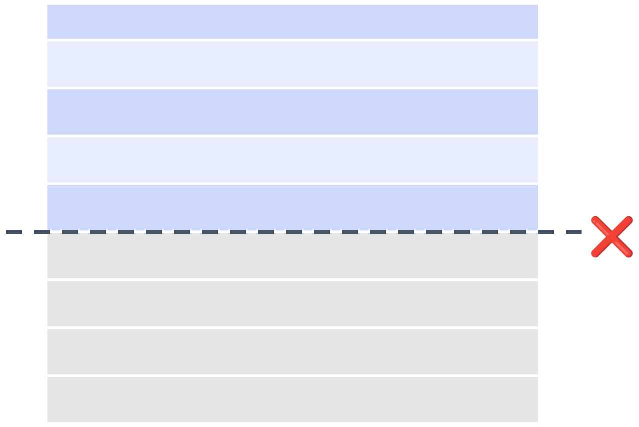
Recall (9/10) 90% ↑

Precision penalizes for **returning irrelevant** documents

Recall penalizes for **leaving out relevant** documents

100% Precision & Recall: Rank relevant documents most highly and only return those

Top k



Top K

- Retrieval metrics are influenced by **how many documents the retriever returns**
- Metrics are discussed in terms of **top-k** documents

Example

Rank	Relevant
1	relevant
2	-
3	-
4	relevant
5	-
6	relevant
7	relevant
8	-
9	relevant
10	relevant

Precision @5
2 out of 5

40%

Precision @10
6 out of 10

60%

Recall@10
6 out of 8 total relevant

75%

Top-5 or Top-1 is stricter

Top-5 to Top-15 often used

Mean Average Precision

MAP@K evaluates average precision for relevant documents in first K documents. It is built off a related metric called "average precision".

Rank	Item	Precision@K
1	relevant	1/1 **1.0**
2	-	1/2 **0.5**
3	-	1/3 **0.3**
4	relevant	2/4 **0.5**
5	relevant	3/5 **0.6**
6	-	3/6 **0.5**

Rewards ranking relevant documents highly

1. **Sum precisions** for relevant docs only.

   1 + 0.5 + 0.6 = 2.1

2. **Divide** by number of relevant documents

   2.1 / 3 = 0.7

3. This calculation gives **Average Precision** or AP, for Mean Average Precision you find the average AP value across many prompts

Reciprocal rank

Measures the rank of the first relevant document in the returned list

```
Reciprocal Rank = 1 / Rank
```

First relevant at rank 1	1.0
First relevant at rank 2	0.5
First relevant at rank 4	0.25

The later the first relevant document appears, the worse the reciprocal rank

Mean Reciprocal Rank (MRR) averages over many prompt

Mean Reciprocal Rank

Search 1	**First relevant at rank 1**	1.0
Search 2	**First relevant at rank 3**	0.33
Search 3	**First relevant at rank 6**	0.17
Search 4	**First relevant at rank 2**	0.5

1. **Sum all ranks**

   `1.0 + 0.33 + 0.17 + 0.5 = 2.0`

2. **Divide by number of searches**

   `2.0 / 4`

MRR = 0.5

How to use retriever metrics

Recall or recall@K

Most cited metric, captures fundamental goal of finding relevant documents

Precision & MAP

Asses irrelevant documents and ranking effectiveness

Mean Reciprocal Rank

How well model performs at the very top of ranking

Metrics help:

- Evaluate retriever performance
- Check if adjustments improve results

All metrics depend on having ground truth relevant documents

DeepLearning.AI

Module 2 Conclusion

Information Retrieval Foundations

Conclusion

- **Keyword Search**
  Ranks by keyword frequency exact matches
- **Semantic Search**
  Ranks by meaning, flexible
- **Metadata Filtering**
  Excludes by criteria
- **Hybrid Search**
  Combines all three techniques

Evaluation Metrics

Precision & Recall

MAP

Mean Reciprocal Rank

Measure improvement from adjusting tunable parameters in hybrid search