ML System Design Youtube Video Search

https://www.youtube.com/watch?v=NE9JIIaNhQk&list=PL_b_MRp1BnEj4UuHv1jAGeO1a0VTRXsAk&index=8

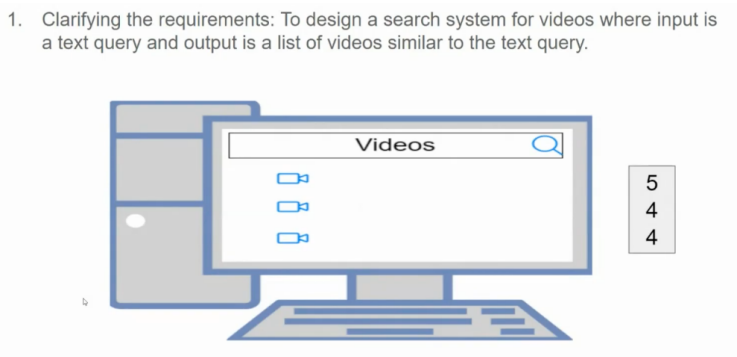
ML System Design:
YouTube Video Search

Clarifying Requirement

- How many languages needs to be supported?
- Can we use video metadata? Yes
- What are the system requirements (such as response time, accuracy, scalability, and integration with existing systems or platforms)?

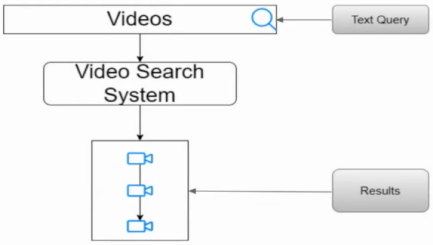
Requirements : response time, accuracy, scalability (50M DAU)

Data: sources and availability Sources: videos (1B), text 10M pairs of <video, text_query>. Videos have metadata (title, description, tags) in text format



Framing as ML task

2. Framing the problem as an ML task:
1. Defining the ML objective - To give relevant videos based on the user's text query.
2. Specifying the system's input and output - input is a text query and the output is ranked list of videos sorted by relevance.

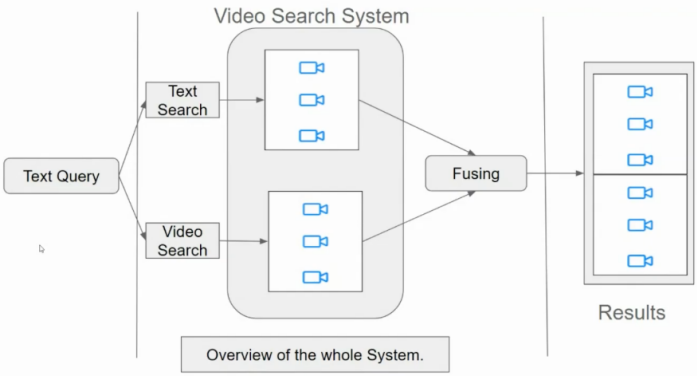


ML Objective: retrieve videos that are relevant to a text query

ML I/O: I: text query from a user, O: ranked list of relevant videos on a video sharing platform

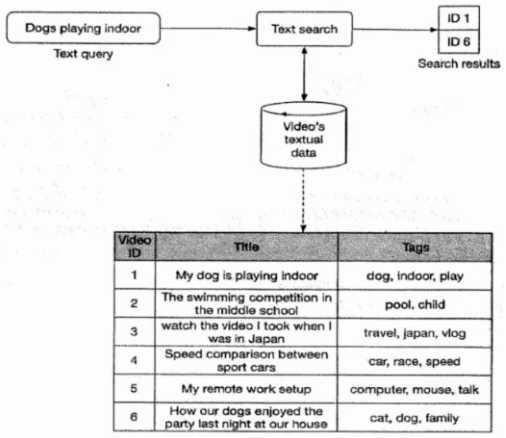
ML category: Visual search + Text Search systems

3. Choosing the right ML category:



Text Search:

- Deals with textual aspect of the videos such as titles, description, tags etc.



Video ID	Title	Tags
1	My dog is playing indoor	dog, indoor, play
2	The swimming competition in the middle school	pool, child
3	watch the video I took when I was in Japan	travel, japan, vlog
4	Speed comparison between sport cars	car, race, speed
5	My remote work setup	computer, mouse, talk
6	How our dogs enjoyed the party last night at our house	cat, dog, family

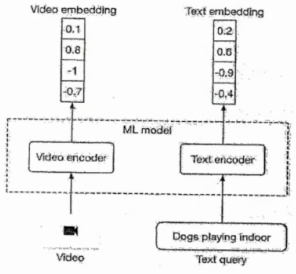
- Inverted Index, where each of the terms (words) in search query points to documents or web pages that contain those terms. We will use Elasticsearch.

Visual Search:

- Representation learning is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data.
- Representation learning works by reducing high-dimensional data to low-dimensional data, making it easier to discover patterns and anomalies while also providing a better understanding of the data's overall behaviour.
- Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space. An embedding can be learned and reused across models.

Visual Search: (Cont.)

- The user enters a text query and model outputs a list of videos.
- Rank of the videos are based on similarity of the text query and videos' visual content.
- Similarity scores are calculated by computing the dot product of the text and each video in the embedding space. And then rank the video based on the similarity score.

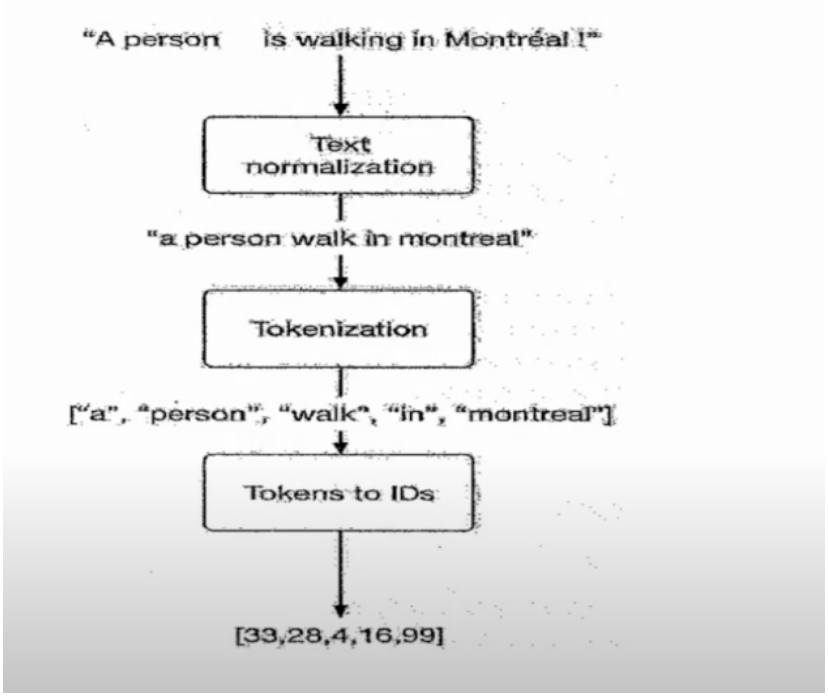


4. Data Preparation:

1. Data Engineering - We have an annotated dataset.

Video name	Query	Split type
76134.mp4	Kids swimming in a pool!	Training
92167.mp4	Celebrating graduation	Training
2867.mp4	A group of teenagers playing soccer	Validation
28543.mp4	How Tensorboard works	Validation
70310.mp4	Road trip in winter	Test

2. Feature Engineering - Unstructured data is converted into numerical representation during this step.
    a. Preparing the Text data :
        i. Text Normalization
        ii. Tokenizations
        iii. Tokens to IDs.



I. Text Normalization - It ensures the words and sentences are consistent. Words like “dog”, “dogs” and “DOGS!” can mean different even though they mean the same thing.
Some of the text normalization techniques are:
   a. Lowercasing - making all the letters lowercase does not change the meaning of the sentence.
   b. Punctuation removal - removing common punctuations from the texts like comma, exclamations, full-stop (period), question marks etc.
   c. Time whitespace - trim leading, trailing and multiple whitespace.
   d. NFKD - decomposes combined gramphemes into combination of simple ones. Eg. "œ" = "oe”
   e. Strip accents - é, è = e, â = a, î = i and ô = o.
   f. Lemmatization and Stemming - grouping morphologically similar words together.
      Eg. lemmatization - walking, walked , walks -> walk.
      Eg. of stemming - studies -> studi and studying -> study.

II. Tokenization - Breaking a piece of text into smaller units called tokens.
Eg. “Let’s eat an icecream” -> “Let’s”, “eat”, “an”, “icecream”.
A. Subword tokenization - splitting the text into subwords. Eg. the word “unwanted” might be split into “un”, “want”, and “ed”.

B. Character tokenization - splitting the text into set of characters. Eg. “Let us learn tokenization.” -> “[“L”, “e”, “t”, “u”, “s”, “l”, “e”, “a”, “r”, “n”, “t”, “o”, “k”, “e”, “n”, “i”, “z”, “a”, “t”, “i”, “o”, “n”, “.”]”.

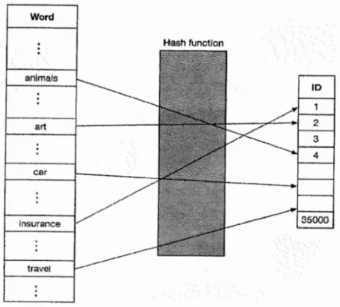
III. Token IDs - Converting the tokens into numerical values.

a. Lookup table - each token is mapped to an ID creating a 1:1 mapping.
b. Hashing - is a memory-efficient method which uses hash functions to obtain IDs without keeping a lookup table.

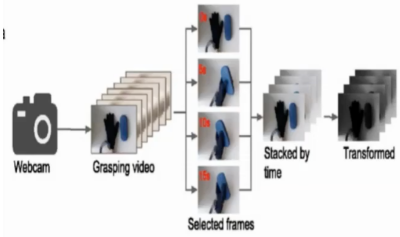
Lookup Table.

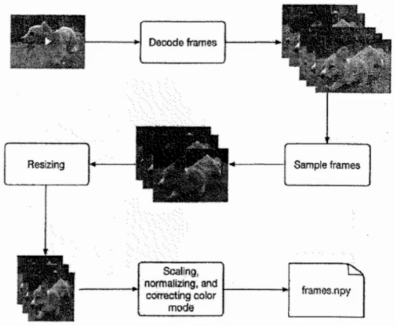
Word	ID
⋮	
animals	18
⋮	
art	35
⋮	
car	128
⋮	
insurance	426
⋮	
travel	1239
⋮	

Hashing.



Preparing the Video data:





5. Model Development:

- Text encoder and video encoder.

Text Encoder:

I. Statistical Method.

- Bag of Words (BoW) - converts the sentence into a fixed length vectors. It models sentence word occurrences by creating a matrix with rows are sentences and columns by words.

	best	holiday	is	nice	person	this	today	trip	very	with
this person is nice very nice	0	0	1	2	1	1	0	0	1	0
today is holiday	0	1	1	0	0	0	1	0	0	0
this trip with best person is best	2	0	1	0	1	1	0	1	0	1

Limitations-

i. The method does not consider order.
ii. Obtained representation does not capture semantic and contextual meaning about the sentence.
iii. Because the unique vector representation is equal to the total number of tokens we have for sentence, the vectors are very large and sparse and mostly filled with zeroes.

5. Model Development -> Text Encoder-> Statistical method: (Cont.)

2. Term Frequency - Inverse Document Frequency (TF-IDF):

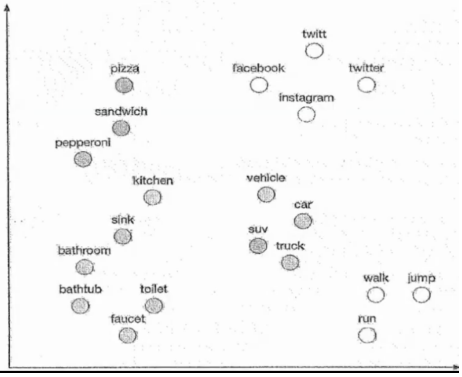
- Works similar to Bag of Words (BoW) but normalizes the matrix based on frequency of words.
  - Gives less weight to more making the representations better than Bag of Words (BoW).
  - Limitations:
    - Normalization is needed.
    - Does not consider order of words.
    - The Obtained representation does not contain the semantic meaning of the sentence.
    - Representations are sparse.

- The statistical methods are fast but they do not consider the context and semantics of the sentence and have sparse representations.

5. Model Development -> Text Encoder: (Cont.)

2. ML based Methods: Sentences are converted into meaningful word embeddings so that distance between two embeddings reflects to the similarity of words. Embeddings of similar words such as “Rich” and “Wealth”, they will be closer in the embedding space.

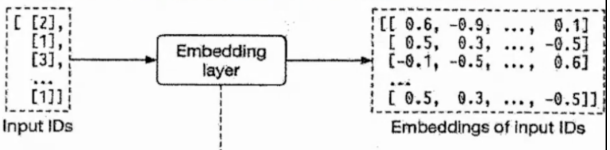
- Three common ML based approaches for transforming words Into embeddings are:
  1. Embedding (lookup) layer.
  2. Word2vec.
  3. Transformer based architectures.



5. Model Development -> Text Encoder: (Cont.)

2. ML based Methods:

1. Embedding (lookup) layer: Maps each ID to an embedding vector. Using an embedding layer helps convert sparse features into fixed-size embedding.



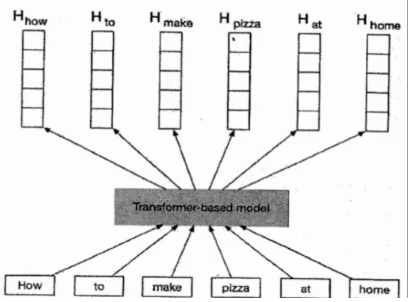
Lookup Table	
ID	Embeddings
1	[0.5, 0.3, ..., -0.5]
2	[0.6, -0.9, ..., 0.1]
	...
	...
N	[0.4, 0.4, ..., -0.5]

5. Model Development -> Text Encoder: (Cont.)
2. ML based Methods:
2. Word2vec:

- word2vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large datasets.
- Word2vec is a family related model used to produce word embeddings.
- The model uses shallow neural network architecture and utilises co-occurrences of words in local context to learn word embeddings.
- In training phase-
  - The model learns to predict center words from surrounding words.
  - After training, the model is capable of converting words into meaningful embeddings.
- There are two main models based on Word2vec:
  1. Continuous Bag of Words (CBOW) - the inputs are the context words and the output is the target word.
  2. Skipgram - In skip-gram, the input is the target word and the output are the context words.

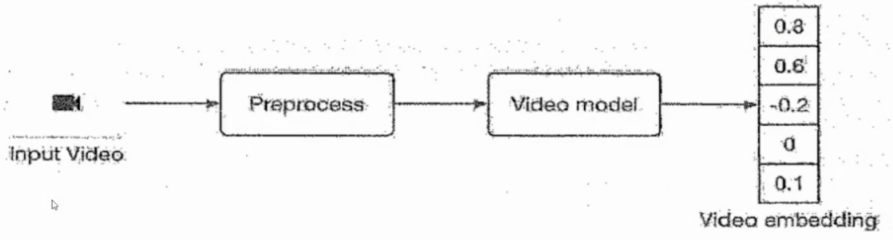
5. Model Development -> Text Encoder: (Cont.)
2. ML based Methods:
3. Transformer-based Architectures:

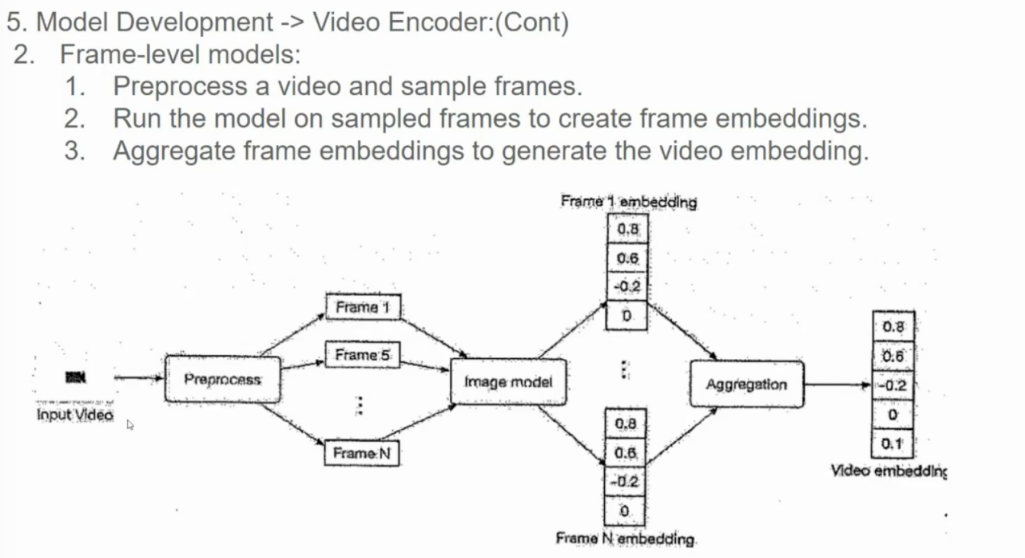
- Transformer based architecture considers the context of words in the sentence when converting them into embeddings. As opposed to word2vec they produce different embeddings for the same word depending on the context.
- Produces an embedding for each word from the input sentence, a group of words.
- For our system, we will use Bidirectional Encoder Representations from Transformers (BERT).



5. Model Development: (Cont)

- Video Encoder:
  1. Video-level models - processes the whole video to create an embedding. Since it uses the whole video to process it is computationally expensive.





1. Preprocessing:

- **Video Input:** The input is a video clip.
- **Frame Sampling:** Key frames are extracted from the video. These frames can be sampled uniformly or based on motion or scene changes.

2. Frame-Level Model:

- **Image Model:** Each sampled frame is processed by an image model (e.g., a Convolutional Neural Network) to extract visual features.
- **Frame Embedding:** The image model generates a dense vector representation (embedding) for each frame. This embedding captures the semantic and visual information of the frame.

3. Aggregation:

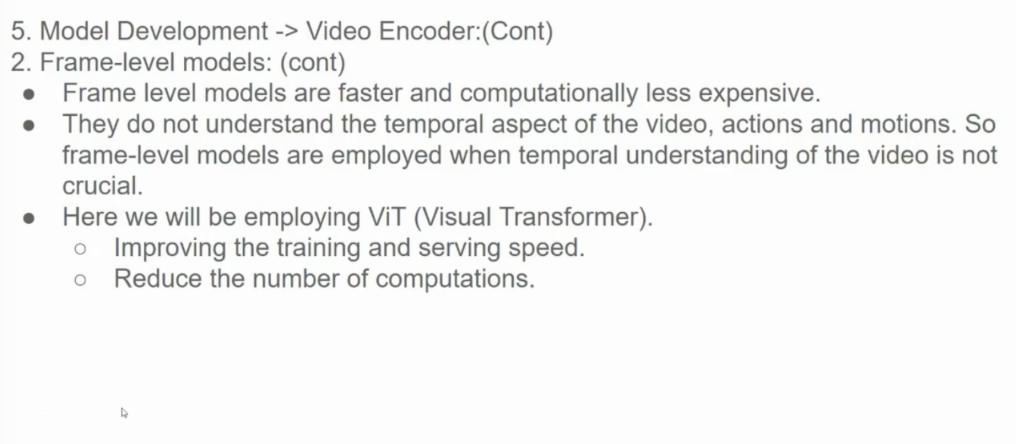
- **Frame Embeddings:** The embeddings of all the sampled frames are combined to form a video-level representation.
- **Aggregation Techniques:** Various techniques can be used to aggregate frame embeddings, such as:
    - **Average Pooling:** Calculating the average of all frame embeddings.
    - **Max Pooling:** Selecting the maximum value from each dimension of all frame embeddings.
    - **Attention-Based Pooling:** Weighting frame embeddings based on their importance to the overall video content.

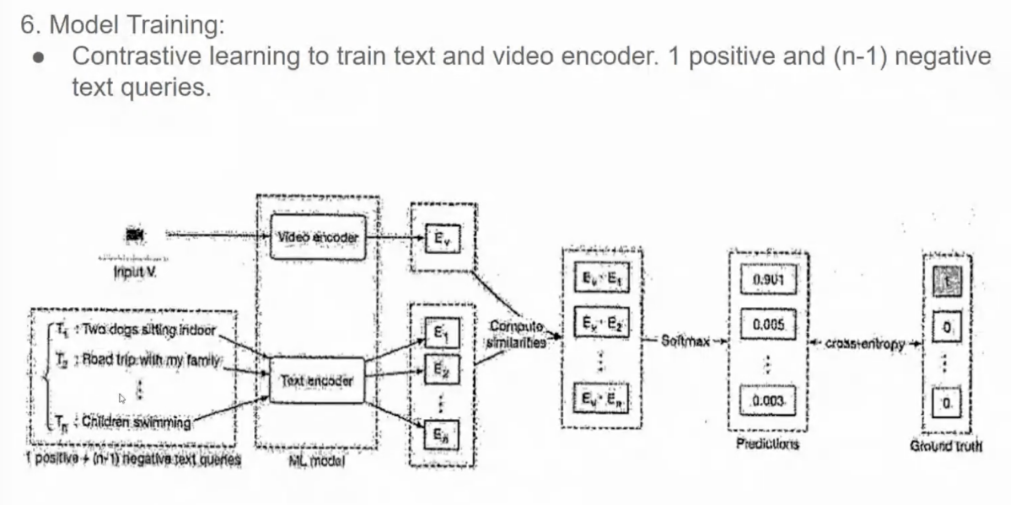
Output:

- **Video Embedding:** The final video embedding is a dense vector representation that captures the overall semantic and visual content of the video.

Why Frame-Level Models?

- **Fine-Grained Information:** Frame-level models can capture detailed information about the video content, such as object motion, scene changes, and temporal relationships between frames.
- **Flexibility:** They can be adapted to various video analysis tasks, including action recognition, video classification, and video retrieval.
- **Transfer Learning:** Pre-trained image models can be used as a starting point, saving time and computational resources.





Input: A video clip (V) and a set of text queries (T1, T2, T3,...). One of these queries (T1) is the positive query, and the others are negative queries.

 Encoding:

- The video clip (V) is passed through a video encoder to obtain a video embedding (Ev).
- Each text query (Ti) is passed through a text encoder to obtain a text embedding (Et).

 Similarity Calculation:

- The similarity between the video embedding (Ev) and each text embedding (Et) is calculated using a similarity function (e.g., cosine similarity).

 Contrastive Loss:

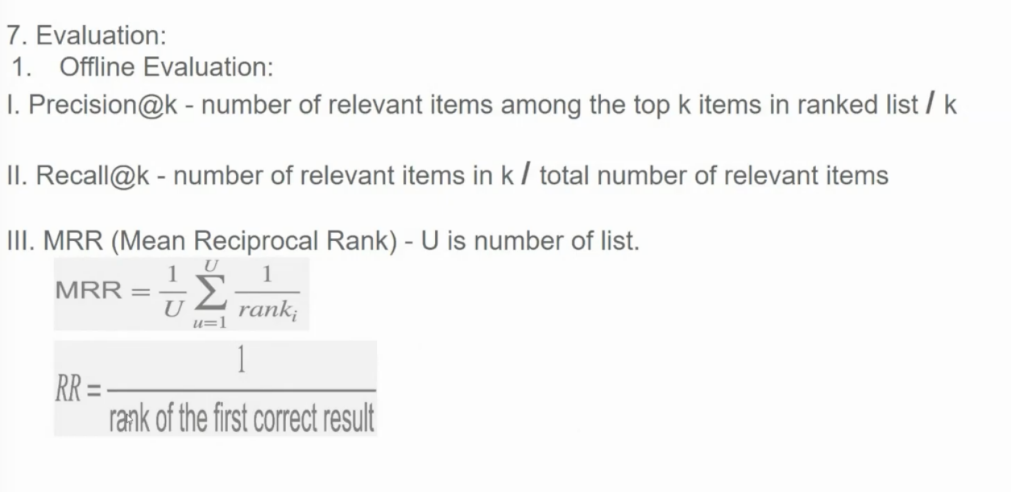
- A contrastive loss function is used to train the model. The goal is to maximize the similarity between the video embedding and the positive text query embedding, while minimizing the similarity between the video embedding and the negative text query embeddings.
- This loss function encourages the model to learn representations that capture the semantic and visual relationships between videos and text queries.

 Model Training:

- The model is trained iteratively on a large dataset of videos and text queries to minimize the contrastive loss.

 Video Search:

- At inference time, a user provides a text query.
- The text query is passed through the text encoder to obtain a query embedding.
- The query embedding is then compared to the embeddings of all videos in the database using the similarity function.
- The videos with the highest similarity scores are returned as search results.



Understanding Offline Evaluation Metrics for Recommendation Systems

The provided image outlines several key offline metrics used to evaluate the performance of recommendation systems. Let's break down each metric:

1. Precision@k and Recall@k

- **Precision@k:** Measures the proportion of relevant items among the top k items in a ranked list.
    - **Formula:** Precision@k = Number of relevant items in top k / k
- **Recall@k:** Measures the proportion of relevant items that are retrieved among the top k items.
    - **Formula:** Recall@k = Number of relevant items in top k / Total number of relevant items

Example:

Consider a search engine that returns 10 results for a query. If 5 of the top 10 results are relevant, then:

- Precision@10 = 5/10 = 0.5
- Recall@10 depends on the total number of relevant items in the entire dataset. If there are 20 relevant items in total, and 5 of them are in the top 10, then Recall@10 = 5/20 = 0.25

2. Mean Reciprocal Rank (MRR)

- Measures the average reciprocal rank of the first relevant item in a ranked list.

- A higher MRR indicates that relevant items are ranked higher.

Formula:

MRR = (1/rank_1 + 1/rank_2 + ... + 1/rank_u) / U

Where:

- U is the number of user queries.
- rank_i is the rank of the first relevant item for the i-th query.

Example:

If for the first query, the relevant item is ranked 3rd, for the second query, it's ranked 1st, and for the third query, it's ranked 2nd, then:

- MRR = (1/3 + 1/1 + 1/2) / 3 = 11/18

3. Reciprocal Rank (RR)

- A simpler metric that measures the reciprocal rank of the first relevant item in a single ranked list.

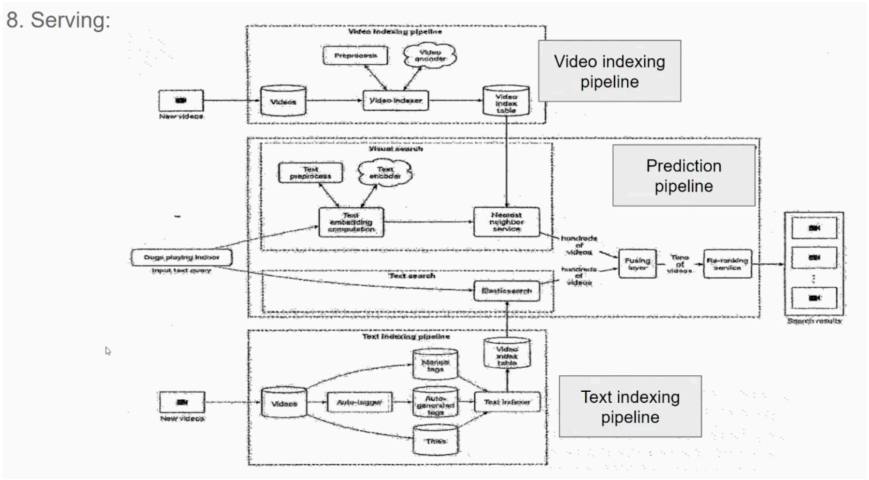
Formula:

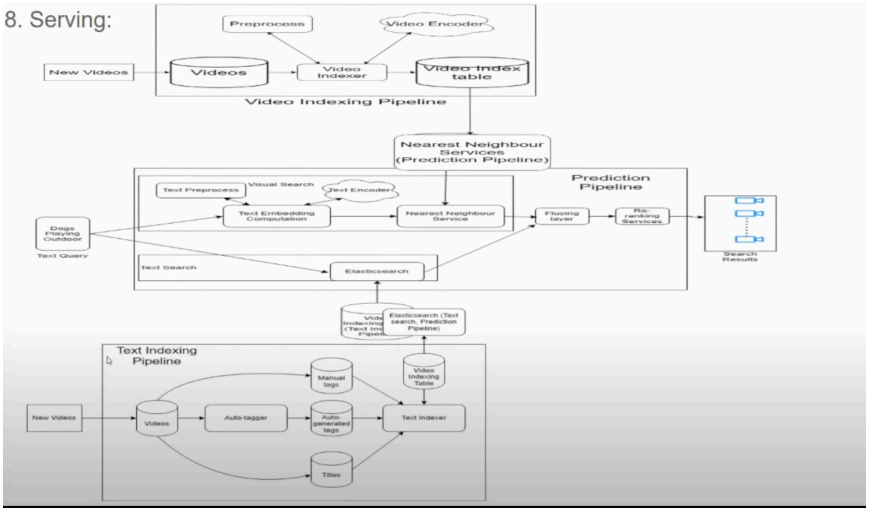
RR = 1 / rank

Where rank is the rank of the first relevant item.

7. Evaluation: (Cont)
2. Online Metric:
- Click-through rate (CTR) - Shows often users click on retrieved videos/results.
- Video Completion Rate - Metric shows how many results appear in search results and are watched by the user till the end.
- Total watch time of searched result - measures the total watch time of user spend on watching the videos returned in search result.

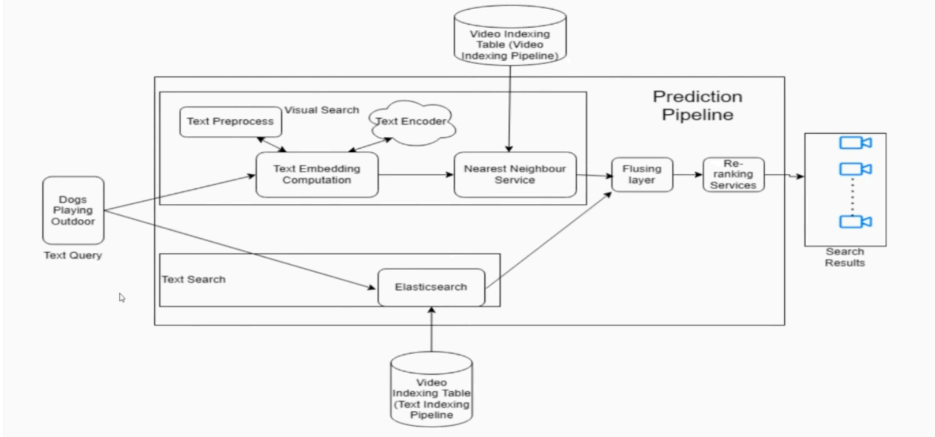
8. Serving:





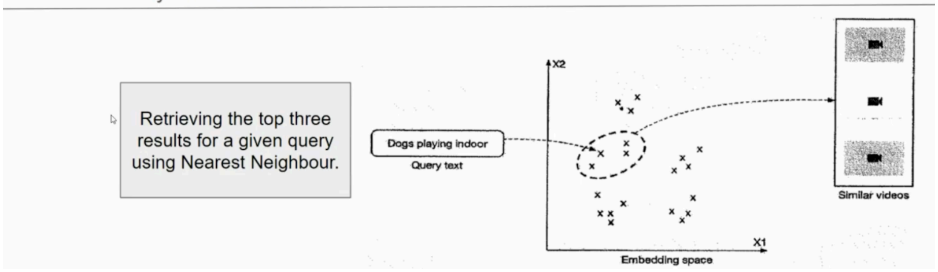
8. Serving: (Cont.)

- Prediction Pipeline.



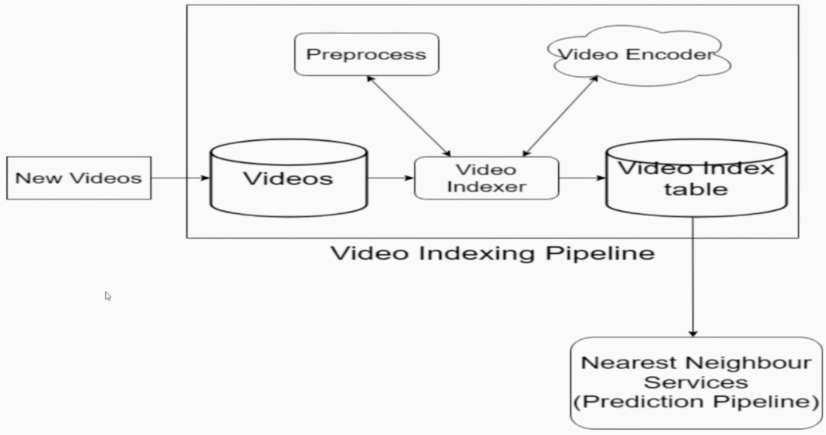
8. Serving: (Cont.)

- Prediction Pipeline:
  - Visual search - Approximate Nearest Neighbour algorithms.
  - Text Search - Using Elasticsearch for finding videos that matches the titles and tags that overlap the search query.
  - Fusing Layer - Using search results of Visual search and Text search to produce combined list of new videos.
  - Re-ranking Services - incorporates business level logic as well as policies to modify the ranked list.



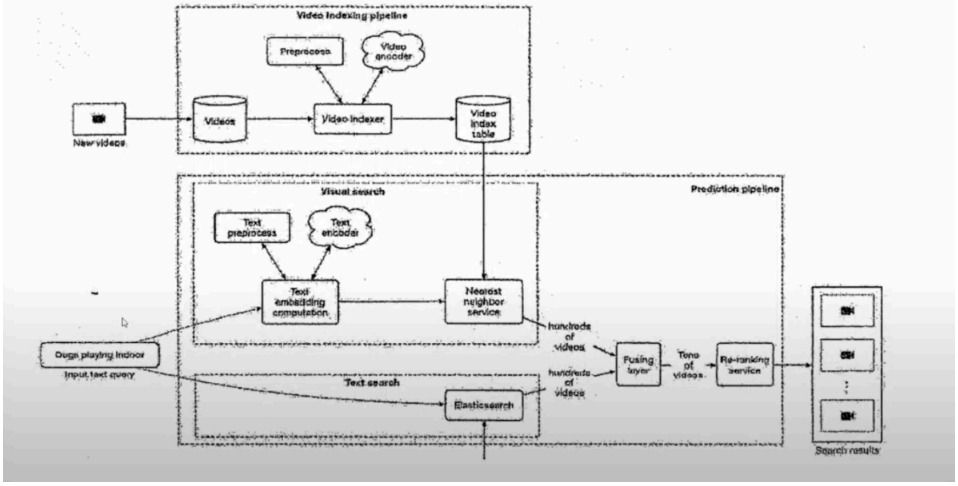
8. Serving: (Cont.)

- Video Indexing Pipeline.



8. Serving: (Cont.)

- Video Indexing Pipeline.

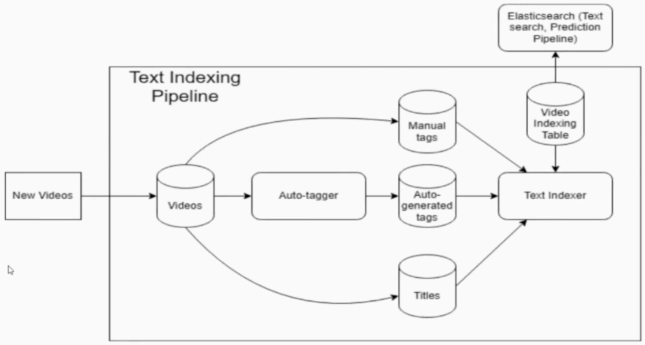


8. Serving: (Cont.)

- Video Indexing Pipeline - Video embeddings generated by the video encoder.
- These embeddings will be indexed which will be used Nearest Neighbour Services.

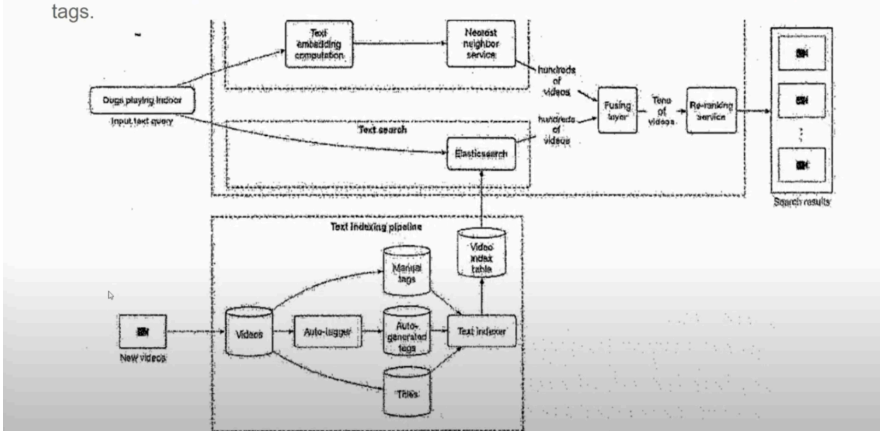
8. Serving: (Cont.)

- Text Indexing Pipeline -Uses Elasticsearch for indexing titles, manual tags and autogenerated tags.

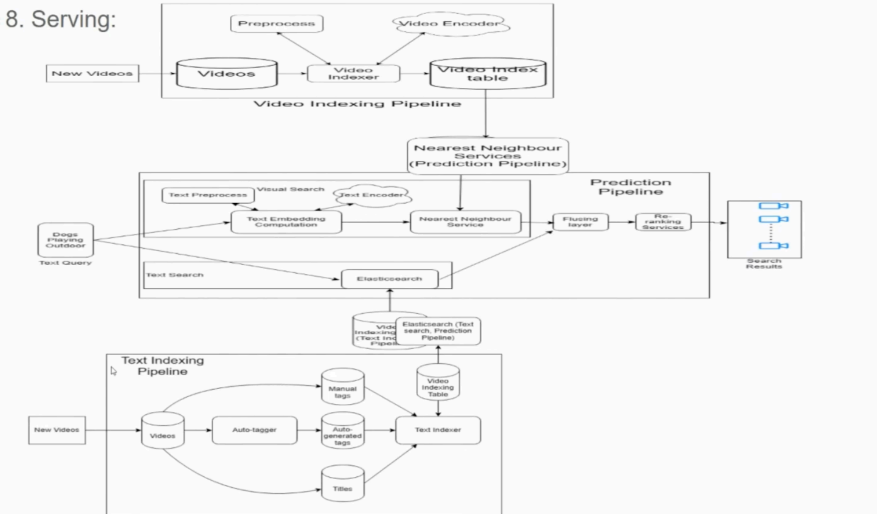


8. Serving: (Cont.)

- Text Indexing Pipeline -Uses Elasticsearch for indexing titles, manual tags and autogenerated tags.

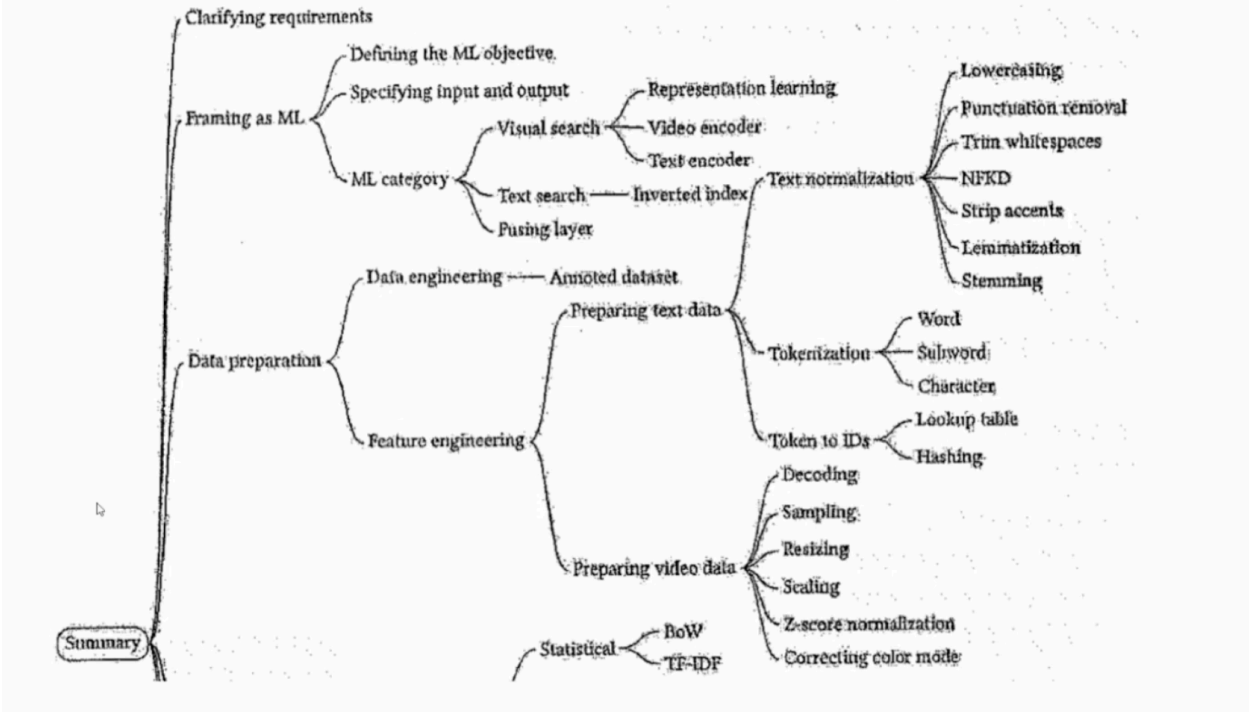


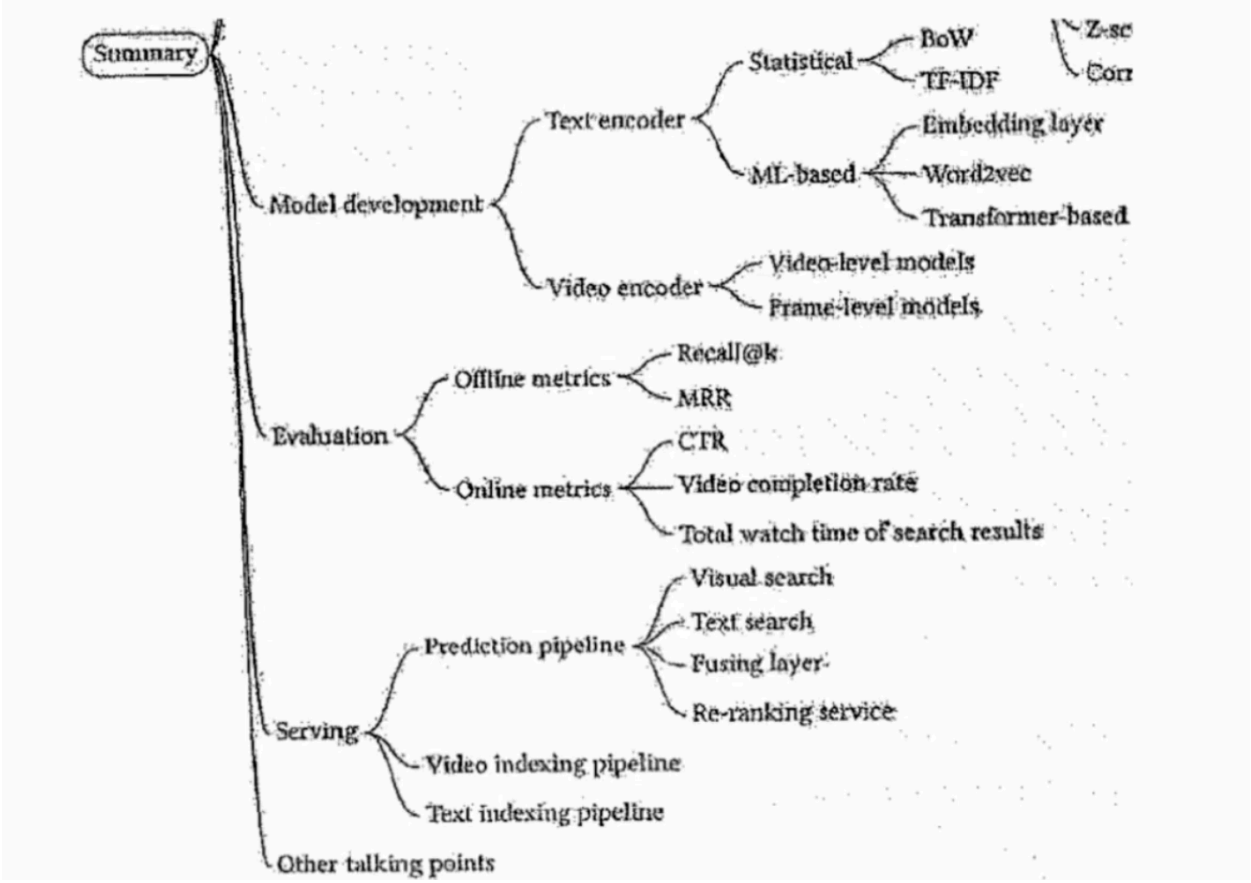
8. Serving:

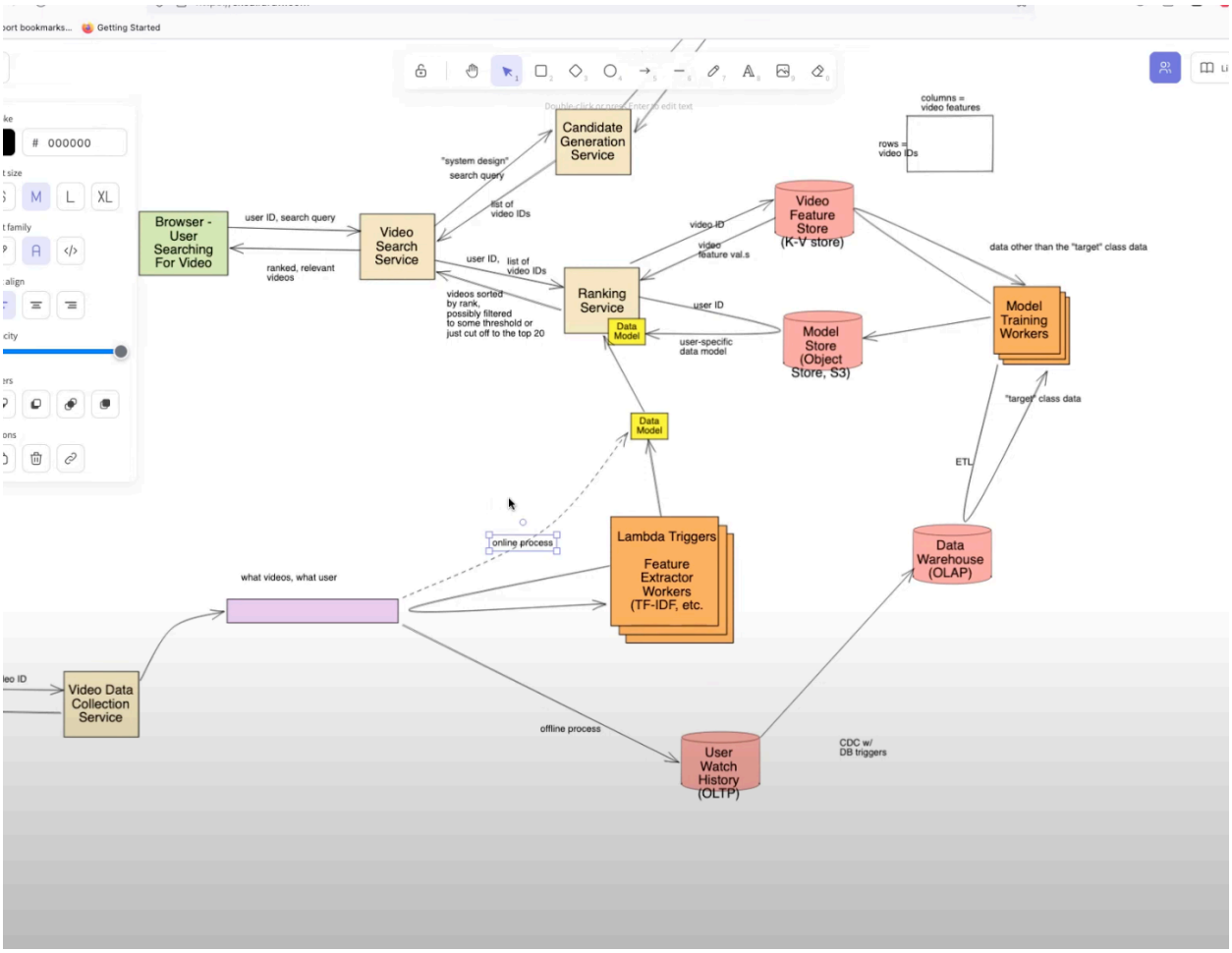


Additional Talking Points:

- The system can be multi-stage design.
- Using video length, video popularity etc. as more features.
- Instead of relying on annotated dataset we can use interactions like “like”, “Clicks” etc. and generate the dataset which can be used for continuously training the model.
- Using an ML model which can find titles and tags which are semantically similar to the text query.




Summary
Model development
Text encoder
Statistical
BoW
TF-IDF
Z-sc
Corr
ML-based
Embedding layer
Word2vec
Transformer-based
Video encoder
Video-level models
Frame-level models
Evaluation
Offline metrics
Recall@k
MRR
Online metrics
CTR
Video completion rate
Total watch time of search results
Serving
Prediction pipeline
Visual search
Text search
Fusing layer
Re-ranking service
Video indexing pipeline
Text indexing pipeline
Other talking points



https://www.youtube.com/watch?v=nprPYgn4E58&list=PLlvnxKilk3aKx0oFua-HTtFf-d_inQ8Qn&index=3