The most important concepts

1. Search: BM25 (takes into account both tf and doc length normalization to determine the relevance)

   Components: 1. TF (How often term occurs)
   2. Length of doc
   3. IDF (measure the significance of a term across the entire doc collection.

2. Embeddings: Word2vec, FastText, GloVE, BERT
3. Precision/Recall

① Introduction to Multimedia Retrival

- What is Retrival?
- What do we want to achieve?
-

1. Task
2. General Retrival Process
3. Retrival models (7)
4. Metadata (Two ways for extraction)

② Evaluation

1. How to validate a search algorithm ?
2. How to compare boolean retrival model?
3. How to compare model with order ?
4. How to compare Retrival graded ?
5. Confusion Matrix: def ?
6. How to find treshhold for model ?

1. Benchmarks (Sparse and dense)
2. Boolean Retrival
   a) What
   b) Precision and recall
   c) F-measure
   d) Micro- and macro evalution

$$\frac{F_q}{R_q} \cdot \frac{R}{A}$$

3. Retrival with order
   a) Def
   b) MRR
   c) P-R curve
   d) „Analysing P-R" :
      1) System efficiency 2) P@k
      3) R-precision 4) AP 5) MAP
4. Retrival with Graded Relevance
   1) def
   2) CG-k = Σrel_i
   3) DCG-k = Σ
   4) Normalized DCG
5. Confusion Matrix (TP, TN, FP, FN, ACC, Sensetivity, Specifity)
6. Optimizing Hyperparametrs
   1) Treshhold
   2) Pdf (sensetivity + specifity)
   3) ROC-curve, AUC

③ Classic Text Retrival

1. Fundamentals of Retrival

I. Offline phase: (Aim: create vec. with features

1. Extract:
   - (metadata, url, text, html tags) - Clean document
2. Split (Fixed sizes, min. tokens, meladat or structure, semantic spliting)
3. Tokenize (Seq of characters)
   (Chars, words, n-grams and phrases)
   * Stemming → Porter (5 steps with ~20 rules)
     $[C](VC)^m[V]$
   * Tagging of tokens

d) Summarazing → creating vector with features.

$tf(D_i, t_j)$ number of occurences of term t_j in D_i

1. Create vector
   1) set-of-words 2) bag-of-words
   * inverted file method to store.
2. Eliminate unimportant terms
   1) Stop words
   2) Zipf's Law $p(r) = \frac{c(M)}{r}$, $c = \frac{1}{\sum \frac{1}{r}}$, M - # distinct terms in vocabluary

   3) „Similarities". C - centroid doc all M terms with average freq
      $dp(t) = Q_t - Q$ (without doc); $dp(t)$ large ⇒ import; $Q = \sum_{i=1}^{N} sim(D_i, C)$
   4) Inverse document frequency (idf)
      $idf(t) = \log \frac{N+1}{df(t)+1}$ (document freq); => weighted doc $idf(t) \cdot tf(D_i, t)$

③ Classical Text Retrival

2. Text Retrival Models

1. Standard Boolean Model

+ranking
+partial matches

1) 2)

2. Extended Boolean Model (Similarity)

- no theoret. background
- hard to express complex info

- 1) use $tf(D_i, t_j)$ and $idf(t_j)$ + $sim(req, D_i) = \frac{1}{1+d}$ $d_{ij}(k,k)$
- Fuzzy Algebraic, Fuzzy set, p-norm-model, soft boolean

3. Vector Space Retrival

- a lot theoret. background
- independence of terms, but it is not always true.

- Most popular
- $d_{i,j} = tf \cdot idf$. - term-doc-matrix A

$q_j = tf \cdot idf$

- Search: 1. $sim(Q, D_i) = q \cdot d_i = \sum q_j \cdot d_{ij}$ – inner vector product

2. $Sim = \frac{q \cdot d_i}{\|q\| \cdot \|d_i\|}$ – cosine

- Actually we use only direction of vectors!

4. Probabilistic Retrival (Formal approach)

- assumptions
- feedback

1) BIR: 3 assumptions => $sim(Q, D_i) \sim \sum_{c} c_j$, $c_j = \log \frac{r_j \cdot (1 - n_j)}{n_j (1 - r_j)}$

c - relevant, p - irrelevant

1. Initial step (frequency)
2. Feedback step (user annotation)

5. BM25 (Vector space model + probabilistic approach)

family of functions with diff params

- Take into account length of doc + frequency

3. Index Structure

- Inverted index

1. Boolean model
2. BIR: a) doc-at-a-time (DAAT) b) term-at-a-time (TAAT)

+ we can add predicates

3. For Vector Space model

same, but + (freq, sim, $\|d\|_1$, $\|d\|_2$)

④ Lucene scalability

1) segment level 2) sharding 3) replicate shards
4) multiple regions.

④ Advanced Text Retrival

① Chunking Text

1) Split into fixed-sized chunks (langchain library)
2) Spliting at sentence boundaries (NLTK: Punkt; spaCy) — unsupervised DL
3) Splitting on structure (Merge small parts + overlaping for semantics) — semantic, size control
4) Semantic splitting (1. Def semilarity (Vectore space model or embeddings; 2.

Set min/max chunk size, split into sentences NLTK or spaCy; merge chunks based on similarity

② Tokenization Revisted

1) Divide by words: naive, nltk, spaCy, for scripts continua — Amazon CodeWhisperer
2) Continious stream: HMM, overlapping seq + sub-seq search
3) n-grams: 1. Pointwise Mutual Information (PMI); $\log_2 \frac{tf(t_1,t_2)}{tf(t_1)\cdot tf(t_2)}$
   2. Likelihood Ratios (LHR); • H_1: $P(t_2|t_1) = P(t_2|\neg t_1) = p$
   • H_2: $P(t_1|t_2) = p_1$, $p_2 = P(t_2|t_1)$

⚠ One-hot vectors → Embedding + positional encoding

4) BPE tokenization (transformers library): split → merge
   1) initialization (chars or subword units)
   2) freq count (bag-of-words)
   3) merge most freq pair
   4) repeat 5) tokenize text

5) WordPiece Tokenization (Extended BPE in 2 ways: 1. position (beginning + middle) 2. pairs frequency) until size

③ Lemmatization and Linguistic Transformation

1. Rule based stemmers (Porter, Lankaster, Snowball)
2. Dict-based stemmers (WordNet, spaCy) ← include POS part-of-speech tag

* Words combination: split ← rule based + freq. of the components

* Synonyms → extened dict with predef synonyms
* Homonyms → POS, ML, most common meaning
* Hypernyms (broad)
* Hyponyms (narrow)

weights; Faceted search, expend queries, relevance ranking

* Spelling mistakes: spellchecker

④ Part of Speech (POS and NER (name entity recognition))

1. Rule-based POS tagging (like -ing → verb)
2. Stochastic POS tagging (HMM)
3. Transformation-based POS (rule-based + correcting the errors iteratively)
4. Dl POS tagging

⑤ Latent Semantic Analysis (LSA)

1. Latent Semantic Indexing (LSI)
   1. SVD: $A = USV^T$ (term × doc = U × S × doc)
      $A = s_1(u_1 v_1^T) + s_2(u_2 v_2^T) + \dots + s_r(u_r v_r^T)$
      - LSI treats queries as mini-docs: $\hat{q}^T = q^T U_k S_k^{-1}$
      - sim fun (dot-product or cosine measure) to compare the query
2. Embeddings (gensim library, spiCy)
   1. Word2vec
      a) Skip-Gram Model (x ± word x x x)
      b) Continuous Bag of Words (CBOW) Model (word x word)
   2. GloVe (self-supervised, considers word co-occurrences within context windows.
   3. fastText (sub-word representations for center word) → word2vec skip-gram →
   4. BERT (NN based embedding; transformer based)
3. Embeddings in Text Retrival
   1. Retriever-Ranker (BM25 + ranker embedding)
   2. Semantic Search with Reader
   3. Retrival Augmented Generation (RAG)
      (q → Retriever → Generator)

4. Language Detection
   - Naive Bayes ($\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$) multinomial
5. Sentiment Analysis
   1. Naïve Bayes (Bernoulli distr.)
   2. Text-CNN
   3. Transformer-based classification

5 Web Social

- Types of search queries:
  1. Navigational search queries (reach specific website)
  2. Transactional search q. (ended purchasing smth)
  3. Informational search q (info)

① Additional Features (compare to classical text retrival)

1) Proximity of query terms in docs
2) Penalties for low-quality pages
3) language and location awareness
4) Term boosting if it occurs in anchor text link
5) Term occurrence weighting based on tag
6) Proximity of query terms in doc

② PageRank ( )

- rank represents the likelihood of a random surfer visiting that page. $PR(p) = \frac{1-d}{N} \cdot \sum \frac{PR(q)}{L(q)}$ ($L(q)$: outgoing link)
- solving interatively → fast due to adjacency M sparce matrix → linear

③ Hyperlinked-Induced Topic Search (HITS)

- Authorities (Wiki) $a(p)$ and Hubs (Yahoo) $h(p)$; Problems: 1. Overpowered Authorities and Hubs 2. Automated, non-related Links 3. Query Term Frequency

↓ modification 1. One domain – one vote

2. Filter unrelated pages, using reference doc; cosine and threshold.

④ Other relationships

1. Co-citation analysis 
2. Stochastic Approach for link Structure Analysis (SALSA) ← HITS extention
   - Bipartite graph → two-step markov chain → probability
3. Sim Rank (Similarity between nodes, based on their connections)

⑥ Vector Search

Motivation:

Idea: each doc D_i yields an N-dim embedding vector d_i. we have query vec. q. We want to compare.

1. Best Match Problem:

Given d_i and q, find $\vec{d}^*$ that optimizes the similarity fun.

$$d^*_{dot} = \text{argmax}_{\vec{q}} \cdot d_i \quad ; \quad d^*_{cos} = \text{argmax} \frac{q \cdot d_i}{\|q\| \|d_i\|}$$

2. Normalizatio of vectors: (Gaussian normalization)

- $\hat{v}_j = \frac{v_j - \mu_j}{\sigma_j} \rightarrow (\mathbb{E}[0,1], \mathbb{E}[0,1000])$
- Using eigenvalues and eigenvalues we can rotate and scale the original space do get normalized space

3. $MS(q) = \text{argmax}\ Sim(p_i, q)$, $NN(q) = \text{argmin}\ dist(p_i, q)$ (Nearest Neighbor Problem)

3. Low-dim Search Structure

- Voronoi Cell
- Grid File
- R-Tree

- Optimal Search Method by Samet (using priority queue)
- We can utilize lower-upper bounds do „navigate" a search

4. Curse of Dimensionality

- As the number of dims increases, the data becomes increasingly sparce. In high-dim spaces, most data points concentrate near the corners of the hypercube representing the feature space.

5. Quantization based NN-search

- VA-File → grid with points

6. Approximate NN-Search (FAISS)

1. Vector Transformers (
2. Coarse quantizers
3. Encodings (Fine quantizers)
4. Refiners

1] VA-File – approximation using lower bit precisions, while maintaining a small error.

1	0.6	0.8
2	0.0	1.0

→

10	11
00	11

⑦ Basic Image Retrival

Def • Illumination - measure energy per sec, accounting for the sensetivity of the human eye to different light wavelength.

• Image Normalization → Image Segmentation → Feature Extraction → Feature Aggregation

• Global feature extraction:

1. Feature Set 2. Feature Concatenation 3. Feature Summary

• Segment images:

1. Global feature 2. Static Segmentation 3. Object Segmentation

1. Image Normalization

1. Gray scale $i(x,y): \mathbb{N}^2 \to [0,1]$
2. Color image $i(x,y): \mathbb{N}^2 \to [0,1]^3 = \begin{bmatrix} r \\ g \\ b \end{bmatrix}$
3. Quantization to replace fixed-point numbers in image representat:
   - True color (32 bit): $\hat{f}(x,y): \mathbb{N}^2 \to [0,255]$ $f(x,y) = \frac{\hat{f}(x,y)}{255}$
   - Deep color (64) : $\to [-655\,35$
4. Standardize the dataset
   1. Rotate-invariant features
   2. Histogram normalization
   3. Grayscale transformation
   4. Scaling
   5. Affine Transformation
   6. Convolution (Kernels, remember to normali to get colors in range)

2. Color Information

Convert Cartesian RGB to cylindrical coordin. → compression, printing

1. Common color systems: 1. CIE 2. RGB 3. HSL/HSV 4. YUV 5. CMYK
2. Color : Brightness + Chromaticity (Hue, Saturation)

$X = \_ \cdot R + \_ \cdot G + \_ \cdot B$

CIE - based on human perception, XIY-color space

RGB - most common, most used green/blue range, sRGB

device independent; CIE Lab color space: L, a (Green-Red), b (Blue-Yellow), device independent, easy to perform color manipulations.

3. Color Features

1. Color Histogram

Compare: • Manhattan, Euclidean, Quadratic ⇒ to consider similarity
• Also consider luminance
• Consider only chromaticity.

2. Color Moments (Lab) as hist. resistant to noise, scaling, transform. do not account for spatial relat. between colors not independent.

3. Texture information:

1. Structural approach

1. Edge magnitude and directions (Sobel operator on smoothed Gaussian image.

2. Gradient Histogram (2 values per pixel - direction magnitude)

3. Gradient Moments

4. Law's Texture Energy (Combine 5 1D pattern to create 2D texture 9 filters. This filters capture dif. aspects of texture, like edges, spots, waves).

2. Statistical approach

3. Fourier approach

1. Gabor Moments (Use to capture various frequency ranges and directions. Reveals info about the precences of selected freq in frame) $g(x,y) = \frac{e^{-\frac{\bar{x}^2+\gamma^2\bar{y}^2}{2\sigma^2}}}{2\pi\sigma^2} \cdot e^{i(2\pi\frac{\bar{x}}{\lambda}+\psi)}$ ← before we use rotation

- We use it through convolution, each pixel mapped to energy value based on chosen G.filter. filter's direction and scale.

4. Shape information

- Edge Detection
- Contour Detection
- Region - based Segmentation

Once shape is avalible we can calculate:

- Area - Centroid - Different Ratio of length, shape resembles a circle etc.

1. HOG (Histogram of Oriented Gradients)

2. SIFT (Scale - Invariant Feature Transform)

⑧ Advanced Image Retrival

1. Perceptron concept
2. SVM
3. Feed Forward
4. Error Function
5. SGD
6. Backpropogation (Vanishing gradient or exploding)
7. Overfitting and underfitting
8. Occam's razor
9. Activation Function (ReLU, leaky Relu ($\{0.01 \cdot s,\ s$ ← never 0), parametric ReLU ($\{\alpha \cdot s,\ s$ ← helps with negative values), exponential Linear Unit ($\{\alpha(e^s - 1),\ s$ ← smoothing derivative)
10. Generative Adversarial Network (GAN)
11. $L_1 \backslash L_2$ regulization (L_1 ← absolute weights; L_2 ← in loss function)
12. Normalization
    1. Batch
    2. Layer
    3. Weight Standardization
13. Deep learning
    1. Convolution
    2.