Web" ore Concept of IR, is about finding relevant information in large collections, in response to a user query. Process Overview: Crawling->Parsing->Indexing-> Query Processing->Ranking->Evaluation.

🔹 **IR vs Database Systems**

Aspect	Information Retrieval	Database Systems
Data	Unstructured text	Structured (tables)
Queries	Free text	SQL (schema-based)
Results	Ranked list (approximate)	Exact matches
Matching	Keyword / semantic	Boolean logic

Key concept: **Document corpus**:collection of text units(web pages,PDF,etc) **Index**: maps content to locations for fast search. **Ranking**: orders documents by estimated relevance. **Feedback loop**: users' clicks improve future ranking

Web Search Engines – Architecture: 1.Crawler: Collects and revisits web pages**.2.Indexer:** Builds inverted index of terms**.3.Query Processor:** Parses user queries**.4.Ranker:** Scores documents via similarity metrics**.5.Ad Index / Meta Data:** Supports ad and topic-based retrieval. **User Behavior : Average Query length-2-3 words. Query types: informational:** history of AI**. Navigational: "**usc Viterbi website" **Transactional:** buy Iphone 15pro

Web Crawling:定义: A web crawler is a computer program that visits web pages in an organized way

Web Crawling Issue: How to crawl? Quality: how to find the "Best" pages first (一般用 BFS 而不是 DFS)**Efficiency**: how to avoid duplication **Etiquette**: behave politely by not disturbing a website's performance

How often to crawl?Freshness: How much has really changed? Crawling 流程：Place the page in a database，Extract the URLs within the page，Place the extracted URLs on a queue，Fetch a URL on the queue and repeat。 Crawling 的特点和面临的挑战-Crawling 的过程可以 distributed (可以处理大数据) Challenges-Handling/Avoiding malicious pages Some pages contain spam ● Some pages contain spider traps – especially dynamically generated pages

3.8. Normalizing URLs

3.8.1. **Why Normalizing URLs is Important?** 很多相似的link指向同一个page但是一旦他们有一点点不同他们的hash值就会不同(e.g. http://www.google.com; http://www.google.com/; https://www.google.com)

3.8.2. **4 rules of Normalizing URLs**

3.8.2.1. **Convert the scheme and host to lower case.** The scheme and host components of the URL are case-insensitive.

★ **Example:** HTTP://www.Example.com/ → http://www.example.com/

3.8.2.2. **Capitalize letters in escape sequences**. All letters within a percent-encoding triplet (e.g., "%3A") are case-insensitive, and should be capitalized.

★ **Example**: http://www.example.com/a%c2%b1b → http://www.example.com/a%C2%B1b

3.8.2.3. **Decode percent-encoded octets of unreserved characters.**

★ **Example:** http://www.example.com/%7Eusername/ → http://www.example.com/~username/

3.8.2.4. **Remove the default port.** The default port (port 80 for the "http" scheme) may be removed from (or added to) a URL.

★ **Example**: http://www.example.com:80/bar.html → http://www.example.com/bar.html

3.7. Representing URLs

3.7.1. 问题: URL太多太长了, 直接存太耗费空间.

3.7.2. 方法1: **To determine if a new URL has already been seen**

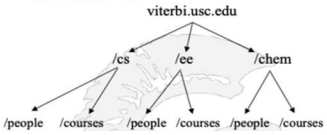
3.7.2.1. hash on host/domain name

3.7.2.2. Use a **trie data structure** (字典树) to determine if the path/resource is the same as one in the URL database

3.7.2.3. 最直接的比较两个URL是否相同的方式: 逐个字符比较, 复杂度**O(nk)**, n是URL个数, k是URL最大长度

3.7.2.4. 用了trie之后的复杂度: **O(k)**

★ **Example of tire:**



3.7.3. 方法2: **URLs are sorted lexicographically and then stored as a delta-encoded text file**

3.7.3.1. Each **entry is stored as the difference (delta) between the current and previous URL**; this substantially reduces storage

3.7.3.2. However, **restoring the actual URL is slower**, requiring all deltas to be applied to the initial URL

3.7.3.3. To improve speed, **checkpointing** (storing the full URL) is done periodically

Avoid Spider Traps: For the crawler to be careful when the query string "ID=" is present in the URL. Monitor the length of the URL and stop if the length gets "too long"

Distributed Crawling: Multi-Threaded Crawling: One bottleneck is network delay in downloading individual pages. It is best to have multiple threads running in parallel each requesting a page from a different host. (在一台机器上多线程爬虫)

Distributed Crawling Approaches:1.A centralized crawler controling a set of parallel crawlers all running on a LAN.2.A distributed set of crawlers running on widely distributed machines, with or without cross communication (MapReduce**)**

If crawlers are running in diverse geographic locations, how do we organize them? → Distributed crawlers must periodically update a master index (But incremental update is generally "cheap" because you need only send a differential update)优点: **scalability:** for large-scale web-crawls **costs:** use of cheaper machines **network-load dispersion and reduction**: by dividing the web into regions and crawling only the nearest pages 缺点: **overlap:** minimization of multiple downloaded pages 。**quality:** depends on the crawling strategy **communication bandwidth:** minimization

4.1. Precision, Recall, F1-score

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

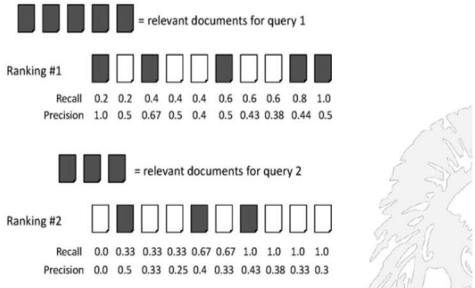
4.1.1. **relevant**指的是真实相关**(actual true), retrieved**是认为相关**(predicted true)**

4.1.2. **F1-score(F-measure)**是**Precision**和**Recall**的harmonic mean

Search Engine Evaluation:

4.2. Mean average precision

★ **Example:**



$$average\ precision\ query\ 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$

$$average\ precision\ query\ 2 = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$mean\ average\ precision = (0.62 + 0.44)/2 = 0.53$$

4.3.2. 公式: $DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{log_2(i+1)}$, 其中p为一个结果列表的排序位置, rel_i 代表第i个位置上文档的相关度 (i通常情况下是rank)

★ **Example:** 假设搜索到6个结果, 其相关性分数分别是3,2,3,0,1,2. 求DCG

i	rel_i	log2(i+1)	rel_i / log2(i+1)
1	3	1	3
2	2	1.58	1.26
3	3	2	1.5
4	0	2.32	0
5	1	2.58	0.38
6	2	2.8	0.71

$DCG = 3 + 1.26 + 1.5 + 0 + 0.38 + 0.71 = 6.86$

4.4. Normalized Discounted Cumulative Gain (nDCG∈[0,1])

4.4.1. 首先对语料库中所有相关文档的相关性排序, 再通过位置生成最大可能的 DCG, 称为IDCG. 对于一个 Query, nDCG的公式$nDCG = \frac{DCG}{IDCG}$

★ **Example:** 沿用上面那个🌰, 假设我们实际召回了8个文档, 除了上面的6个, 还有两个结果. 假设第7个相关性为3, 第8个相关性为0. 那么在理想情况下的相关性分数排序应该是: 3, 3, 3, 2, 2, 1, 0, 0. 计算IDCG

i	rel_i	log2(i+1)	rel_i / log2(i+1)
1	3	1	3
2	3	1.58	1.89
3	3	2	1.5
4	2	2.32	0.86
5	2	2.58	0.77
6	1	2.8	0.35

$IDCG@6 = 3 + 1.89 + 1.5 + 0.86 + 0.77 + 0.35 = 8.37$

$nDCG = \frac{DCG}{IDCG@6} = \frac{6.86}{8.37} = 0.819$

IR(Information Retrival): 目的: 读入用户的 queries, 根据已有的数据库推荐给用户他们想要看的内容 7.2. IR system 的特点 7.2.1. A retrieval model specifies the details of: 7.2.1.1. Document representation (vector of terms) 7.2.1.2. Query representation (vector of terms) 7.2.1.3. Retrieval function (TF-IDF + Similarity

How to Speed up processing?

Data structure: Use Inverted index to store data so that we can retrieve them quickly by using key-value structure. **Computational machinery:** we use distributed computing to divide the workload among multiple machines or processors. We use MapReduce to divide big data into many chunks to achieve the speedup of big data in distributed computing. **Disk space:** We use deduplication methods to reduce the amount of disk space needed to store data. **Bandwidth**: We use content delivery networks(CDN) to distribute traffic among multiple servers or locations. For example, if user in Japan wants to access a video on youtube, the CDN will first try to find the nearest server to find that video. If the server has the video, it will directly deliver it to the user; if not, It will request that video from the host server

What is the big/massive shift in IR?Traditional IR system matched exact words using inverted indexs and TF-IDF ranking.Modern systems now understand meaning rather than just match words. This represents a shift from retrieval of documents → retrieval of knowledge and intent, effectively merging search with reasoning and natural language understanding.

the current AI revolution in IR was made possible by the intersection of modern ML models, huge training data, and abundant compute power—conditions that simply didn't exist a decade ago

7.3. Document & Query representation

7.3.1. Documents和Queries都被表示成一堆keywords(aka terms)的集合

7.3.2. 语料库(vocabulary)是一个所有已知term的集合, 称为V

7.3.3. 假设我们用vector表示每个document和query

7.3.3.1. Size of V = Dimension

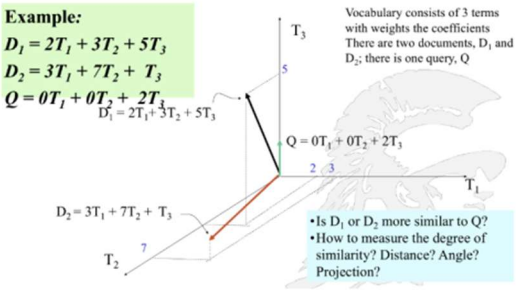
7.3.3.2. Document $D_i = (d_{i1}, d_{i2}, ..., d_{it})$, 其中$d_{ij}$表示第j个term在文档i中的权重(weight)是多少

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



★ 如何计算这个weight呢? → TF-IDF

7.4. TF-IDF

7.4.1. TF-IDF定义: Measure of Word Importance. **Item profile for a document** = set of words with **highest** TF-IDF scores, together with their scores.

7.4.1.1. Term Frequency: $TF_{i,j} = \frac{f_{i,j}}{max_k f_{k,j}} \leq 1$, 进行了一次**normalized**

$f_{i,j}$ = frequency of term (feature) i in document (item) j

$max_k f_{k,j}$ = maximum occurrences of any term in document j

7.4.1.2. Inverse Document Frequency: $IDF_i = log_2(\frac{N}{n_i})$

n_i – number of docs that mention term i

N = total number of docs

7.4.2. TF-IDF score: $w_{i,j} = TF_{i,j} \times IDF_i$

★ **Example:** 我们有2^{20}个document, term w出现在2^{10}个document里。

1. 假设w在document j里出现的次数是所有term里最多的。

A: $TF_{w,j} = 1, IDF_j = log_2 \frac{2^{20}}{2^{10}} = 10, w_{w,j} = 1 \times 10 = 10$

2. 假设w在document i里出现的次数为1, 且文档里出现最多的term出现了20次。

A: $TF_{w,i} = \frac{1}{20}, IDF_j = log_2 \frac{2^{20}}{2^{10}} = 10, w_{w,j} = \frac{1}{20} \times 10 = \frac{1}{2}$

7.4.3. 给定一个query, 它和一个document的分数$Score(q,d) = \sum(TF_{t,d} \times IDF_t)$, 其中 $t = Set(q) \cap Set(d)$

★ **Note:** in some cases we **normalize** each tf•idf value using the **L2** (sqrt of sum of squares), as opposed to **L1** ((absolute)sum) norm

★ 有了**TF-IDF**, 如何知道哪些**document**更应该推荐给用户呢? → **Cosine Similarity**

Cosine Similarity: 7.5.1. 用 Cosine Similarity 的原因: 我们用 vector 表 示 query 或者 document, 它们之间的相似度 就应该是两个向量之间的距离, 和向量的长短无关只和它们的方向有关 (degree of similarity) 7.5.2. 朴素版本 similarity: q 和 dj 之间的相似度就是他们的内积. 7.5.2.1. 公式$sim(d_j, q) = d_j \cdot q = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$**, 其中 wij 表示 term i 和 document j 的权重,而 wiq 为 term i 和 query 之间的权重 7.5.2.2. 对于 boolean vectors, 内积就表示哪些 query 中的 terms 在 document 中也出现了 (aka size of intersection/Hamming distance) 7.5.2.3. 对于 weighted term vectors, 内积表示所有共同出现的 terms 的加权乘积**

In IR, what are similarity measures for? Name and discuss 4 of them that we studied?1. Cosine Similarity: First we use TF-IDF to calculate the frequency of every term in two documents, then we measure the cosine of the angle between two vectors. We don't care about the length of the vectors, we only care about the angle. . Jaccard similarity: Frist use n-grams/shingling to divide each of the two documents to be many shingles. We will get term/shingle set from each document. Then we measure the ratio of the intersection to the union of two sets. 3 Weight similarity: First we use vector to represent all term in each on of the two documents with weight(eg. The frequency of the term). Then we measure the sum of the dot products of the weights of the matched terms.4. Edit distance: Definition: inimum number of edits (insertions, deletions, substitutions) to turn one string into another. Smaller distance = greater similarity. Spelling correction, query expansion, and fuzzy matching in search engines.

Text Processing (Classification)

1.Mannual Classification: tree(hierarchical structure)人为的给每个 URL 分类, 像是在维护一个图书馆. 需要 experts 才能很好地完成分类, 在数据规模很大的时候无法实现, 这个方法最早由 Yahoo!(DMOZ Open Diectory Project)提出, saty 在上课时对比了 Yahoo!和 Google 的方法. Google 的方法就是 index 法, 不需要分类, 在数据规模大的时候更有优势

2.Hand-coded-rule-based classifier:定义: 人为的制定某些规则进行分类 (e.g.如果你发的邮件符合某些特定的 pattern, 你的邮件就会被 NSA 捕捉), 多用于 news, government 和 enterprises. 特点: 如果规则制定的好准确度就会很高, 但是随着时间规则可能要一直改 变, 制定和维护好的规则是很 expensive 的 (1980 年的规则用在现在可能很差)

3.Supervised learning: **Naïve Bayes: :** 假定每个类别之间没有关联性 (比如一个水果具有 weight, shape, smell 等属性, 但它们之间我们假定是互相没有联系的, 也就是水果的 shape 不会影响它的 smell)一个应用: SpamAssassin **过程**: 得到一个邮件以后通过某些 feature 判断它是一个 spam email 的可能性 **特点**: feature 不仅限于 words, 还用 blacklist (黑名单), hand-crafted text pattern (符合某些结构的邮件很可能是 spam) **优点:** Very fast learning and testing (不需要神经网络) Low storage requirements (基本上只需要存一张概率表) Very good in domains with many equally important features (因为 Naive Bayes 不能对 feature 添加重要性这个概念) More robust to irrelevant features than many learning methods

KNN8.6.5.1. 过程: 每个文档作为一个向量, 一个新的文档进来时找出离它最近的 k 个邻 居进行投票, 得票最多的 class 作为新文档的 class (尽可能设 k 为奇数防止平 局情况, 万一真平了就同时属于多类) 8.6.5.2. 特点: ● Case-based, Memory-based learning ● Lazy learning (每次只需要计算 distance; no need for training) ★ ✴ ✴ Exam 题目 (saty 上课说的): 已知我们算距离的公式 $(x, 1-x2)2+(y1-y2)2$ 每次得到一个新的文档就要重新计算 k 次这个公式, 有什么办法可以加速 吗? ➢ 答案: 不计算根号. 因为我们最后是比较距离而不是真正需要这些距离, 有 没有根号没区别 (有点无厘头, 一个根号能省多少时间?) ● No feature selection necessary ● No training necessary ● 对于 large number of classes 能很好地 handle ● Small changes to one class can affect other classes ● Very expensive at test time (要算距离) ● In most cases accuracy > Naive Bayes and Rocchio