DS

05/03/2024

Text Analysis:

Q What ~~steps~~ {feature eng.} steps are carried out in text analysis?

① **Flatening** (Raw text → Vector (A vector is a "flat" column))
- → Bag-of-words: Look up for freq. of words occuring in text. Order (or structure) NOT preserved. (Think map<string, int>)
- → Bag of n-grams: 2-gram E.g. → "The students"
  - Maintains some structure, more informative than bag of words

② **Filtering**
- → Stemming: E.g., Flower ≡ Flowers, both shd map to one word
- → Stop word removal: E.g. a, about, above, am, been

③ **Chunking**
- → Parsing & Tokenizⁿ
- → Collocation Extracⁿ

Q What can be the use case of {bag of words}?

① Document classificⁿ via frequency lookup

② Querying: "Student" appears in which docs?
{Informatⁿ Retreival}

Q What makes {bag of n-grams} a more costly approach comp. to bag of words?

k unique words

k^2 unique grams

Q What is the standard tool used to carry out {stemming}?

Porter Stemmer Tool

Q. Give an e.g. where stemming would give wrong results.

new vs news

Q. Explain {collocation extracⁿ for phrase detecⁿ}.

- More informative than bag of n-grams and bag of words.

Q. Explain {POS tagging}.

Ashok	knocked	on	the	door
↓	↓	↓	↓	↓
Noun	verb	Preposition	article	Noun

- Note: Chunking is done AFTER POS tagging.

E.g., "The quick brown fox jumps over dogs."

- POS Tagging followed by chunking leads to:
  - Noun Phrase: The quick brown fox
  - Verb Phrase: jumps over

(2) ~~Chunking~~

DS

06/03/2024

Document Classification :

• TF - IDF : Term Frequency - Inverse Document Frequency
Method

Q What is {term freq}?

$$\underset{\text{Bag of words}}{BoW}(w,d) = \frac{\text{\# No. of times word occurs in doc } d}{\text{Total no. of words in doc } d}$$

(which we query to find term freq)

Q What is {inverse doc. freq.}?

$$\log_e\left(\frac{\text{Total \# Num. of documents}}{\text{\# Num of ~~times~~ occurs in ~~the~~ ~~doc~~ ~~docs~~ ~~that~~ word}}\right) \quad \text{i.e. } \log\left(\frac{N}{n}\right)$$

\# Num of ~~doc~~ docs which contain that word

Q What is the eqn. for the TF-IDF method?

$$\underset{\text{Freq. of word } w \text{ in doc. } d}{\text{Normalized}} = \text{Bow}(w,d) \times \log_e\left(\frac{N}{n}\right)$$

Q Calc. the TF-IDF scores of each word in each document.

D1 (Docum.-1)	Person	Clock	cat
D2	Cat	clock	Cat
D3	Person	write	comment
D4	Cat	write	comment

IDF's : Person : $\log_e\left(\frac{4}{2}\right)$

Clock : $\ln\left(\frac{4}{2}\right)$

cat : $\ln\left(\frac{4}{3}\right)$

write : $\ln(4/2)$

Comment : $\ln(4/2)$

Make matrix of TF x IDF to get the normalized freq.

	D_1	D_2	D_3	D_4
Person	$\frac{1}{3} \times \ln(4/2)$	0	$\frac{1}{3} \times \ln(4/2)$	0
Clock				
Cat				
Write				
Command				

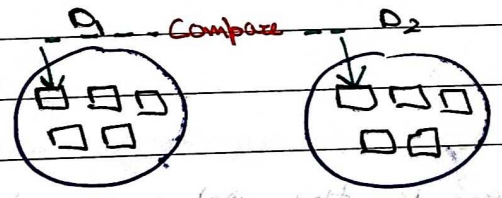
Finding Similar Items:

Q What are the use cases for finding similar items?

- Plagarism check
- Similar webpages
- Similar users : recommendation system

Q What is the simple approach to finding doc. similarity?

Scan two docs word to word, if all match then similar, else not similar



Q Explain the {Jaccard similarity} approach to finding document similarity.

Sentence 1 : He ran twice

S1 : { He, ran, twice }

Sentence 2 : He ran thrice

S2 : { He, ran, thrice }

$$Sim(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{2}{4} = 0.5$$

{ Jaccard Score }

1 2 3 4 5 6 ... 10

$\alpha - (k-1)$ $4-2+1$

$10-2+1$

Q Explain the disadv. of using Jaccard sim. with an example.

D1 : Will you do it

D2 : Do you will it

$$Sim(D_1, D_2) = \frac{|D1 \cap D2|}{|D1 \cup D2|} = \frac{4}{4} = 1$$

100% Similarity

But, sentences are NOT similar.

Q Explain {shingling of documents} as an approach to doc. similarity.

Instead of one word element in the set, we make a ~~set of {shingles}~~, i.e. set of {k-shingles}, i.e. k-words as one element and then calculate the typical Jaccard score over these sets.

Obviously, for $k = 1$, this is the same as Jaccard similarity. ∴ We need to choose k optimally.

```
Time    CLEAR
Labels: LCL     &M
&M      SET     0
        MOVER   AREG, '='0'
        ...
```

DS

09/03/2024

Q If there are x words in the sentence and we make k-sized shingles, how many shingles will we get?

$x - k + 1$ ⇒ No. of shingles grows linearly with no. of words in the doc.

(given all such substrings end up being unique)

#Min-Hashing:

- Create {hash signatures} of docs
- Preserve similarity

Used to: Overcome the large size of no. of shingles.

Goal: Sim of documents = Sim of their signatures

Makes comput^n more effective

Consider e.g of 16-shingles and 3 docs:

Shingle no.	Doc 1	Doc 2	Doc 3
1	1	1	1
2	1	0	1
3	1	0	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	1
8	0	1	0
9	0	1	0
10	0	1	0
11	0	1	0
12	0	0	1
13	0	0	1
14	0	0	1
15	0	0	1
16	0	0	1

Here, we minhash char. matrix of 16 rows into 4 signatures :→ 4 random hash values for each row

Step ①: Generate 4 columns of randomly permutated rows that are indep. of each other.

To generate this, we use the formula:

$$h(x) = (ax + b) \bmod c$$

→ Random no's <= max no. of Original rows

a, b must be unique in each signature

x → Row no. of your original char. matrix

c: a prime no. slightly larger than total no. of shingle sets

For above example row count = 16, ∴ c = 17 will suffice.

Doing this we will get 4 diff. hash values for each row in our char. matrix

	hash_1 (used one set of (a,b))	hash_2 (used another diff set of (a,b) & so on)	hash_3	hash_4
1	5	13	6	6
2	2	16	16	1
⋮				
16	11	7	3	16

New ordering

Step ②: The signature value is calculated per document (i.e. for a column)

(obey new ordering imposed by hash func^n)

- Iterate through the hash column in a sorted way (min value first, then 2nd lowest ...) and write the corresponding ~~matrix~~ original char matrix's index

(In this example assume:)

14 : 0 (hash_1), 1 (hash_2), 0 (hash_3), 9 (hash_4)

∴ Row 14 is the ~~base~~ first row (it had the lowest hash = 0)

- Look at ~~row that row's~~ that index's (here 14) entries for all docs in the original char. matrix and ask: "which document's entry at row 14 is a 1?"

⇒ Here Doc 3 is a 1 @ index 14.

∴ The signature value for document 3 generated by our first hash func^n is 0. And for the rest, we'll keep looking.

doc1, doc2

(Assume next entry is:

8 : 1 0 8 5

8 → OG Index, 1 → hash-1

We find ~~row~~ in OG char matrix:

8 : 0 1 0

0 → doc 1, 1 → doc 2, 0 → doc 3

∴ The signature value for doc 2 gen. by our first hash func^n is 1.

So, doc 2 & doc 3 accounted for doc 1 still left.

Next : 2 : 2 16 16 1

2 → OG Index, 2 → hash-1

OG Char matrix : 2 : 1 0 1

1 → doc 1, 0 → doc 2, 1 → doc 3

∴ Sig (doc 1) for hash_1 = 2

Doing this for all hash functions we get:

	doc_1	doc_2	doc_3
minhash_1	2	1	0
minhash_2	2	0	1
minhash_3	2	1	0
minhash_4	1	0	1

New table : n rows : n = No. of hash funcns

m columns : Same as OG char matrix.

Step-③ : Calculate similarities :

$$\text{Sim}(\text{Doc 1}, \text{Doc 2}) = \frac{(\text{\# of times their minhash values are same in a given row})}{(\text{\# of rows})}$$

So, $\text{Sim}(\text{Doc 1}, \text{Doc 2}) = \frac{0}{4} = 0$

$\text{Sim}(\text{Doc 1}, \text{Doc 3}) = \frac{1}{4} = 0.25$

$\text{Sim}(\text{Doc 2}, \text{Doc 3}) = \frac{0}{4} = 0$

Note: Minhash is a "fast approxim^n" of Jaccard similarity, not exactly equal to it. Results become more accurate & closer to Jaccard similarity in larger datasets.

AIF	LCL	REPT
AGO	GBL	IRP
ANOP	SET	

All of it falls under the umbrella of semantic expansion

Design of a Macro Preprocessor:

< We discussed diff. data structures & steps rep. in a macro preprocessor >

OS

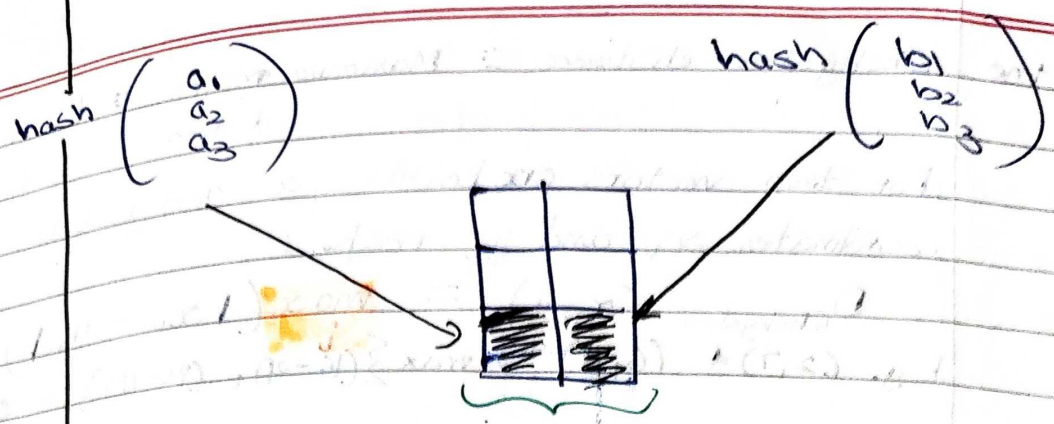
11/08/2024

Locality Sensitive Hashing: (LSH)

The LSH approach takes the output of minhashing as the input.

- Take the signatures and hash the segments / buckets of each signature.
- Hash collision informs ~~of~~ about similarity

Signature - 1	Signature - 2
a_1, a_2, a_3 } segment to be hashed	Segment { b_1, b_2, b_3
a_4, a_5, a_6 }	{ b_4, b_5, b_6



Collision = Same bucket → ∴ Signature - 1 & Signature - 2 are candidate pairs

Distance Measure:

• Jaccard Distance: Jaccard Distance = 1 − Jaccard Similarity

E.g.:

D_1	D_2
1	1
1	1
0	1
0	0
1	1
1	0
0	1

Jaccard Dist. $= 1 - \frac{3}{6}$

• Euclidean Dist: L_2-Norm

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

• Manhattan Dist.: L_1 Norm:

$$= \sum_{i=1}^{n} |x_i - y_i|$$

• L_∞-Norm:

L_∞-Norm $\neq$ ~~max |x_i − y_i|~~

$$= \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

L_∞-Norm distance $= \max\{|x_i - y_i|\}$ max over all (x_i, y_i) under consideration

Q Define the {Chebyshev distance}.

Chebyshev distance ≡ Maximum metric or distance ≡ L_∞ metric or distance

For two vectors ~~or~~ points x and y with std. coordinates x_i and y_i resp.:

$$D_{Chebyshev}(x,y) = \max_i (|x_i - y_i|)$$

E.g. $(2,7)$ & $(6,4)$ $\Rightarrow \max\{(|6-2|), (|4-7|)\} = 4$

Q Define {cosine distance}.

$$= \frac{x \cdot y}{(L_2 \text{ norm of } x \text{ from origin}) \cdot (L_2 \text{ norm of } y \text{ from origin})}$$

Dot Product

Q Find cosine dist. of
$x = [1, 2, -1]$
$y = [2, 1, 1]$

$x \cdot y = 2 + 2 - 1 = 3$

$$D_{L_2}(x,y) = \sqrt{(1-2)^2 + (2-1)^2 + (-1-1)^2}$$
$$= \sqrt{1+1+4}$$

$$D_{L_2}(x,y) = \sqrt{6}$$

$$\therefore D_{cosine}(x,y) = \frac{3}{\sqrt{6}} = \frac{3\sqrt{6}}{6} = \frac{\sqrt{6}}{2}$$

$$D_{cosine}(x,y) = \frac{3}{\sqrt{1^2+2^2+1^2} \cdot \sqrt{2^2+1^2+1^2}} = \frac{3}{\sqrt{6} \cdot \sqrt{6}}$$
$$= \frac{3}{6} = \frac{1}{2}$$

DS

13/03/2024

Contd.

Recom. Systems:

- Making Predictions: Say u hasn't watched a movie, what would be the expected rating u would give movie i? The higher the expected rating, more sense it makes to recommend

{User - User similarity}

$$r_{ui} = \frac{\sum_{v \in N} Sim(u,v) \cdot r_{vi}}{\sum_{v \in N} Sim(u,v)}$$

user ↳ movie

$$N = \{\underbrace{v_1, v_2, \ldots, v_n}\}$$

↳ All of them have watched movie i

Q What is the {user-user sim. weighted avg.} approach to predicting rating for predic^n?

Q Explain the {item - item sim.} approach to rating predic^n.

$$r_{xi} = \frac{\sum_{j \in N(i;x)} \overbrace{Sim(i,j)}^{\text{Similarity of item } i \text{ \& } j} \cdot r_{xj}}{\sum_{j \in N(i;x)} Sim(i,j)}$$

r_{xi}: x → user, i → movie

r_{xj} → Rating of user x on item j

$N(i;x)$ → Set of items rated by x

36 DS

18/03/2024

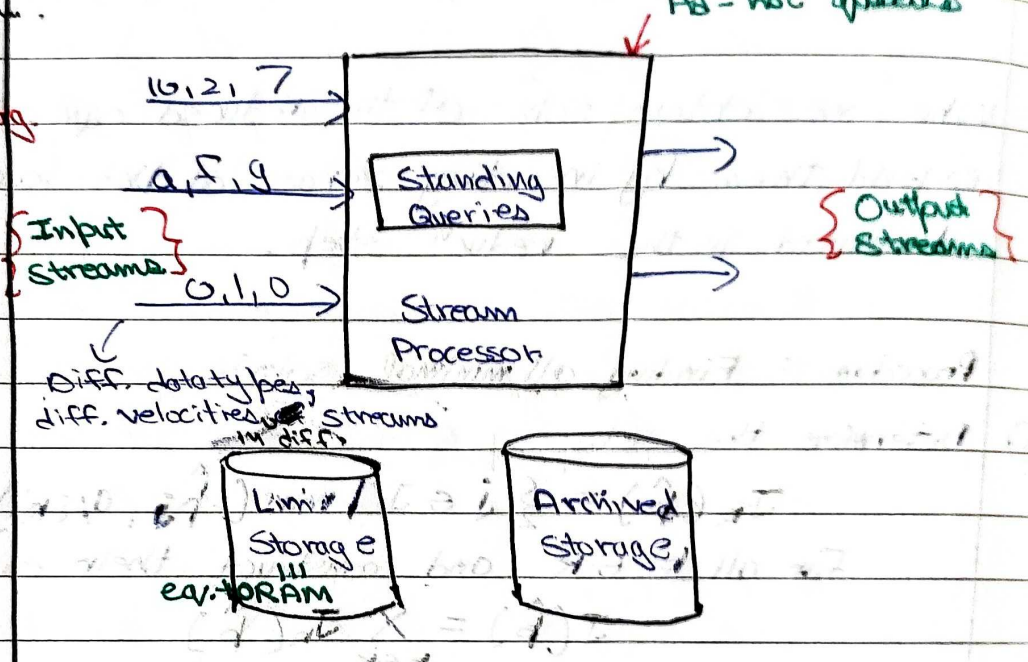
Mining Data Streams:

Data continuosly incoming — streams.
Calculations on the fly. Aim for good approx. rather than 100% acc.

Q Draw a diag. to illustrate a Datastream manag. system.

Ev. to DBMS but for streams

• Datastream Manag. System:



Q What are:

(i) Ad-hoc queries

(ii) Standing queries

• Ad-hoc queries : - Status of system at the time of query
- Executed only once
- E.g., max Temp. @ curr Time

• Standing queries : - Permanently executing on all data elements
- E.g., avg seen so far

• Standing Query:

Consider Give a detailed example of a standing query.

- Stream of integers
- Sliding window size = n (=3 here)
- Query: What is the avg. of integers which have come in so far.

... 10 2 7 6 5 4 → 0

Initial avg. = $\frac{6+5+4}{3} = \frac{15}{3} = 5$

new_value = $\frac{(\text{new_int_that_slid_in} - \text{old_int_that_slid_out})}{n}$

New_avg = Old_avg + new_val

~~The avg. of the new sliding window~~

• Here: new_avg = $5 + \frac{7-4}{3} = 6$

E.g.,

• 11 6 2 3 8 4

Intial_avg = $\frac{3+8+4}{3} = \frac{15}{3} = 5$

New_value = $\frac{(2-4)}{3} = \frac{-2}{3} = -0.66 \approx -0.67$

New_avg = $5 - 0.67$
= 4.33

E.g.,

• 17, 15, 3, 6, 10

initial_avg = $\frac{3+6+10}{3} = 6.34$

new_val = $\frac{(15-10)}{3} = 1.67$

new_avg = $6.34 + 1.67 = 8.01$

Q What are some problems that occur in processing data streams?

(i) Counting Bits : No. of set bits in the last k ($k \leq N$) bits of the no.

- Need an algo more eff. than $O(k)$

(ii) Sampling a stream : { E.g., google sampling search queries for trends }

- Sample should not be biased
- The sampling process preserve orig. answer of the queries we're processing

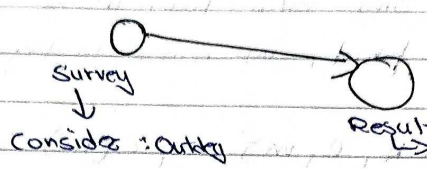
Q Name some appl. that are used IRL to work with data streams.

Apache Kafka, Node.js

• SimRank in Homog. Bipartite Network

⇒ Use case: In a citation network, two scientific papers might be similar as survey papers if they cite similar result papers, while two papers might be similar as result papers if they are cited by similar survey papers.

↳ Same scenario as heterogeneous bi-partite graph. Same calculations.



DS
19/03/2024

Contd.:

Sampling Streams:

Q What are the diff. {types of sampling}?

- Fixed Proportion Sampling
- Fixed Size Sampling
- Biased rerervoir sampling

Q Give real-world applic^n of fixed proportion sampling.

- Sentiment analysis of users of a social media website on a particular topic.
  - It'll have billions of streams.
  - ∴ Sample a fixed proport^n – say 10% (regardless of actual size of stream).
  - Note: Can be biased.

a. Give IRL e.g. of fixed size sampling.

- Say, an online store receives 10,000 orders / hr.
- Sample a fixed no., say 1000 orders.
- Clearly, less effective when data volume large, cuz it doesn't scale with the data.

b. Explain biased reservoir sampling.

The ~~[illegible]~~ ~~of~~ values we put in is our bias

{Say, a set of 100 values (temperature here)} → Avg. of the 100 values = μ

{Reservoir}

(Input stream) ... 15, 20, 30 ↑ val → $> \mu$: Probably anomaly

→ $\leq \mu$

wtf

Filtering Streams:

- There are a no. of tuples that need to be filtered.
- Certain criteria defines when a tuple is filtered.
- Filtered tuples are passed as stream to another process.

Q Explain {bloom filters}.

- A probabilistic data structure, based on hashing.
- Extremely space efficient
- Typically used to add elements to a set and test if an element is in a set. Though, the elements themselves are not added to a set, a hash of the elements is added.

Q How is bloom filter probabilistic?

When testing if an element is in the bloom filter, {false positives} are possible. It will either say that an element is definitely not in the set or that it is possible the element is in the set.

Basically a hash table that maps keys to a bucket, but doesn't store the key in the bucket, just marks the bucket as filled.

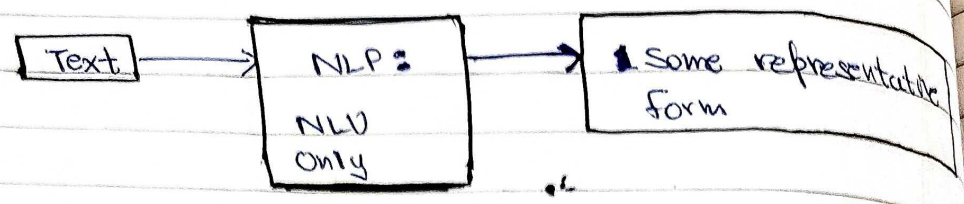
↳ ∴ The bloom filter will essentially be a {bit-array}.
filled : 1
not-filled : 0

Q How do bloom filters aid in privacy?

Bloom filter is implem. as a bit array.
∴ actual values not stored. Even if hacker obtains the data, won't know contents.

is ; what processes are involved in it etc.?

Our focus:



... < Boring stuff ; refer PPTs >

DS

20/03/2024

Contd:

Bloom Filters

Q Derive the formula for {P(false_positives)} for bloom filters.

Initially the bloom-filter is a bit array initialized with all zeros. (Here consider a 12 sized array) (Indx 0 - 11)

Consider two hash funcns : h_1 and h_2

h_1(brown) = 5 ; h_2(brown) = 6
h_1(pink) = 10 ; h_2(pink) = 5
h_1(green) = 0 ; h_2(green) = 9

Say we wanna search 'Red' now.

Let : $h_1(\text{Red}) = i$; $h_2(\text{Red}) = j$

$\therefore P(i \text{ not equal to } 5) = \frac{11}{12} = \frac{n-1}{n}$ ↳ generalized

$P(j != 6) = \frac{n-1}{n}$

$\therefore P(i != 5 \;\&\&\; j != 6) = \left(\frac{n-1}{n}\right)\left(\frac{n-1}{n}\right)$

$= \left(\frac{n-1}{n}\right)^k$ for k hash-funcns

This was for 1 word for m words, we get :

$P(\text{false_positive}) = \left(\frac{n-1}{n}\right)^{m \cdot k}$

n : Length of bit array
m : No. of words
k : No. of hash funcns

$\Rightarrow \therefore P(\text{false_positives}) = 1 - \left(\frac{n-1}{n}\right)^{m \cdot k}$

Q what is the effect of increasing n, m, and k on P(false_+ve)?

① n very large : $1 - \left(1 - \frac{1}{n}\right)^{m \cdot k}$

$1 - (1)^{mk}$

$= 0$

$\therefore P(\text{false_pos.}) \propto \frac{1}{n}$

② m ↑ : P(false_+ve) ↑

~~③ k ↑ : P(false_+ve) ↑~~

③ k : P(false_+ve) depends on the quality of the hash funcn.

→ Probabilistic algorithm : Gives an estimate, not the exact count

#Flajomet - Martin Algorithm : → Count distinct elements

→ Space Complexity : $O(\log_2 n)$ $n \to$ No. of integers

Q Run the Flajomet - martin algorithm on 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1

① Calculate hash func^n : here $h(x) = (6x+1) \bmod 5$

$h(1) = 2$
$h(2) = 3$
$h(3) = 4$
$h(4) = 0$

② ~~Determ~~ Write the binary equiv of the hash func^n values :

$h(1) = 2 \to 010$
$h(2) = 3 \to 011$
$h(3) = 4 \to 100$
$h(4) = 0 \to 000$

③ Count trailing zeroes in each binary rep.

$h(1) = 2 \to 010 \to 1$
$011 \to 0$
$100 \to 2$
$000 \to 0$

④ Determine the max no. of trailing zeros $= r$

Here, $r = 2$

⑤ No. of dist. elements $= 2^r$
$= 2^2 = \underline{\underline{4}}$

DS

02/04/2024

... some classes missed ...

Moments:

m_i : No. of times i occurs in the stream

k^{th} moment $\Rightarrow \sum (m_i)^k$

$k=0$: No. of distinct elem

$k=1$: Total no. of elem.

$k=2$: High val. signal uneven distribution.

$n=100$, Foll. values indicate frequencies ONLY

E.g., Stream 1: 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9

Stream 2: 90, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

2nd moment of Stream 1 $= 10^2 + 9^2 + 9^2 + \dots 9^2 =$

" " " " 2 $= 90^2 + 1^2 + 1^2 + \dots 1^2 =$

Decaying windows:
- → Older info : Less weight
- → Newer info : More weight

I/P stream : ① fifa, ② ipl, ③ fifa, ④ ipl, ⑤ ipl, ⑥ ipl, ⑦ fifa

Case 1 → ~~User searches~~ Query : fifa

"Decay factor" : c = 0.1

→ ∴ Don't add weight

f ① Initial : $1 \times (1-c) = 0.9$

② $0.9 \times (1-0.1) + 0 = 0.81$

f ③ $0.81 \times (1-0.1) + 1 = 1.729$

④ $1.729 (1-0.1) + 0 = 1.5561$

⑤

⑥

f ⑦ $1.2605 (1-0.1) + 1 = 2.135$

Case-2 : ~~User searches~~ Query : ipl

① $0 (1-c) = 0$

ipl ② $0 (1-c) + 1 = 1$

③

ipl ④

ipl ⑤

ipl ⑥

⑦ $3.764 (1-0.1) + 0 = 3.7264$

$$P(Y|X) = \frac{P(X \cap Y)}{P(X)}$$

Frequent Itemsets Analysis:

$X \rightarrow Y$; We're interested to find out if someone buys X, what is the prob. they'll buy Y

Q Define:

(i) {Support}

(ii) {Confidence}

wrt this analysis

$$\text{Support} = \frac{\text{freq}(X, Y)}{N}$$

$N \rightarrow$ Total no. of transactions

$$\text{Confidence} = \frac{\text{freq}(X, Y)}{\text{freq}(X)}$$

Transaction	Itemsets
T_1	A, B, C
T_2	A, C
T_3	A, D
T_4	B, E, F

Given that for us to establish an $X \rightarrow Y$ rel^n min support should be $\geq 50\%$ and min confid. $= 50\%$

Here, min. support $\Rightarrow \frac{1}{2} \times \underset{N}{4} = 2$

cond^n means freq. of item shld be at least,

Listing All candidates:

Candidate	Freq.
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

Suitable candidates:

{A}	3
{B}	2
{C}	2

C_2 :

$\{A, B\}$	1	✗
$\{B, C\}$	1	✗
$\{A, C\}$	2	✓

L_2 :

$\{A, C\}$	2

$$Confid_1 = \frac{freq(A,C)}{freq(A)} = \frac{2}{3} = 0.75 \geq 50\% \checkmark$$

$$Confid_2 = \frac{freq(A,C)}{freq(C)} = \frac{2}{2} = 1 \checkmark$$

Thus: $\left.\begin{array}{l} A \rightarrow C \\ C \rightarrow A \end{array}\right\}$ can be derived from the data