Linear Interpolation

$$q(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 \times q_{ML}(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 \times q_{ML}(w_i \mid w_{i-1}) + \lambda_3 \times q_{ML}(w_i)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_i \geq 0 \ \forall i$

(V.V.V. Imp.)

$\boxed{Q}$ - Is $q(w_i \mid w_{i-2}, w_{i-1})$ ~~as~~ a valid ~~po~~ estimator?

Ans: Our estimate ~~o~~ correctly defines a distribution (define $V' = V \cup \{STOP\}$)

$$\sum_{w \in V'} q(w \mid u, v)$$

$$= \sum_{w \in V'} \left[\lambda_1 \cdot q_{ML}(w \mid u, v) + \lambda_2 \cdot q_{ML}(w \mid v) + \lambda_3 \cdot q_{ML}(w) \right]$$

$$= \lambda_1 \sum_{w} q_{ML}(w \mid u, v) + \lambda_2 \sum_{w} q_{ML}(w \mid v) + \lambda_3 \cdot \sum_{w} q_{ML}(w)$$

$$= \lambda_1 + \lambda_2 + \lambda_3$$

$$= 1$$

$= 1$ ∵ MLE itself correctly defines the distribution

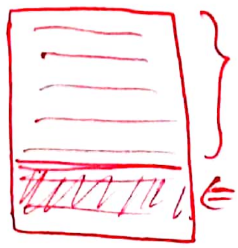
(can ~~also~~ show also that $q(w \mid u, v) \geq 0 \quad \forall \ w \in V'$)

Trivial

How to estimate the λ values?

- Hold out part of training set as "validation" data.
- Define $c'(w_1, w_2, w_3)$ to be the # times the trigram (w_1, w_2, w_3) is seen in validation set.
- Choose $\lambda_1, \lambda_2, \lambda_3$ to maximize:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1, w_2, w_3} c'(w_1, w_2, w_3) \log q(w_3 \mid w_1, w_2)$$



⇐ 5% (VALIDATION)

Distributing Indexes : Term Partitioning vs Doc. Partitioning

[V.Imp. for compre] 5

Term partitioning

- Dict. of idx terms partitioned into subsets.
  ↳ Each subset at a node.

  Along with terms, keep posting lists also

- Query routed to terms corresponding to query terms ⇒ Greater concurrency

BUT: partitioning by vocab-terms non-trivial (in practice)

- Large cost of merging → Outweighs concurrency
- Another problem: Load Balancing
  ↳ governed by distr. of query terms and their co-occurrences

☆ - makes impl. of dynamic indexing more difficult

Doc. Partitioning [Better]

- Each node contains idx for a subset of all docs.
- Each query distributed to all nodes — results from various nodes are merged
- ☹ More local disk seeks but ☺ less inter-node comm.
- One difficulty: 'idf' must be computed across entire doc. collection even though each node contains a subset of docs.
  [computed by distributed bcg processes]

- How to partition docs to nodes?
  → Assign all pages from a host to a single node.

13/04/2020

Langg. Modelling Appl.

- Machine Transl.
- Speech Recog^n
- QA / Summarization
- Query Completion

##

Imp. Note (7)

Langg. Models

Other methods used to improve langg. models:

- "Topic" or "long-range" features
- syntactic models

⇒ Linear interpolation is a kind of smoothening

↓

Another approach : $\boxed{\text{Discounting Methods}}$

Define: $\text{count}^*(x) = \text{count}(x) - 0.5$ ↳ Now, estimate $= \dfrac{\text{count}^*(x)}{\text{Count}(w)}$ ↳ "the"

"Discounted" counts

→ "missing probability mass": $\alpha(w_{i-1}) = 1 - \sum_{w} \dfrac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$

e.g. in our example, $\alpha(\text{the}) = 10 \times \dfrac{0.5}{48} = \dfrac{5}{48}$

Basic idea: Distribute these missing mass prob. values to ~~values~~ words that were never seen together with 'the' in the training corpus.

* **Katz Back-Off Models** (BO) (Bigrams)

- $A(w_{i-1}) = \{ w : \text{Count}(w_{i-1}, w) > 0 \}$
- $B(w_{i-1}) = \{ w : \text{count}(w_{i-1}, w) = 0 \}$ → Never seen in training data

- A bigram model

$$q_{BO}(w_i, w_{i-1}) = \begin{cases} \dfrac{\text{count}^*(w_{i-1}, w_i)}{\text{count}(w_{i-1})} & ; \text{ if } w_i \in A(w_{i-1}) \\[2ex] \alpha(w_{i-1}) \cdot \dfrac{q_{ML}(w_i)}{\sum_{w \in B(w_{i-1})} q_{ML}(w)} & ; \text{ if } w_i \in B(w_{i-1}) \end{cases}$$

In this way $q_{BO}(w_i, w_{i-1})$ would never be 0.

* Katz Back-Off Model (Trigrams)

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in A(w_{i-2}, w_{i-1})} \frac{\overset{\text{Count}(w_{i-2}, w_{i-1}, w_i) - 0.5}{\overset{\shortparallel}{\text{Count}^*(w_{i-2}, w_{i-1}, w)}}}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$A(w_{i-2}, w_{i-1}) = \{ w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0 \}$$

$$B(w_{i-2}, w_{i-1}) = \{ w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0 \}$$

* q_{ML} of unigram is well-defined since we only consider the words that are not in vocab. (But not for 'bigram)

$\therefore$ we use q_{BO} (bi) for q_{BO} (tri) $\rightarrow$ Remember

- A trigram model

$$q_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \dfrac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & ; \text{ if } w_i \in A(w_{i-2}, w_{i-1}) \\ \alpha(w_{i-2}, w_{i-1}) \cdot \dfrac{q_{BO}(w_i \mid w_{i-1})}{\sum_{w \in B(w_{i-2}, w_{i-1})} q_{BO}(w \mid w_{i-1})} & ; \text{ if } w_i \in B(w_{i-2}, w_{i-1}) \end{cases}$$

* LM: Summary

##

3 steps:
- Expand $p(w_1, w_2, \ldots, w_n)$ using chain rule
- Make Markov Independence Assumptions
- Smooth the estimates using lower order counts.

Evaluation of Langg. Models

Perplexity:

- test data : m sentences : $S_1, S_2 \ldots, S_m$

  For each sentence, assign a prob. value

- Prob. under model (log prob.) = $\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$

  [This prob. value should be higher]

Disadv. ☆
Depends on test data size (m)
V.V. Imp

- Perplexity $= 2^{-l}$

  where $l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$ → Higher if better

Normalizing by M
Total # words in the test data

$\boxed{\text{perplexity} \rightarrow \text{lower if better}}$

Intuition

- Vocab V : , $N = |V| + 1$ (STOP word)

  Model : $q(w \mid u, v) = \frac{1}{N}$ (Uniform prob.)

  $\forall\, w \in V \cup \{STOP\}$ $\forall\, u, v \in V \cup \{*\}$

In this case, perplexity $= 2^{-l}$. where $l = \log \frac{1}{N}$

$= \boxed{N}$

☆ Perplexity is a measure of effective "branching factor"

Lower perplexity value ⇒ More context

↳ Lower in case of trigrams (compared to bigrams and unigrams)

* Langg. Modelling Variants

- $P(x_1, x_2, \ldots, x_n) \equiv P(x_m \mid x_1, x_2, x_3, \ldots, x_{n-1})$

  ⇔ (chain rule)

- $P(x_i \mid x_1, x_2, \ldots, x_{i-1}, x_{i+2}, x_{i+5}, \ldots)$

  Predicting middle value based on left and right

- LM at char. level in place of words.

  ↓

  Consider word as a seq. of char.

  ⇒ whole sentence is a seq. of char.

* State-of-the-art in LM

- Tri, Bi → 20 yrs. ago. (Rarely used)

(SOTA)

- Models the whole seq. of words (rather than tri-gram or n-gram)

  (No Markov assumption)

  Any word in sentence is dependent on whole seq. of words appearing before it.

- Trained on huge datasets (~ 1 B tokens)
- Around half-a-million parameters

* **BERT**: Bidirectional Encoder Representations from Transformers

- MASK: Word that is missing
- Transformer Encoder: A layer of NN archi.
- Classification layer: Fully-connected layer + GELU + Norm
  (Lots of parameters and expensive to train)

Back to Machine Translation

$p(t)$ - The langg. Model [Done - Trigrams]

$p(s|t)$ - The Translation Model [Now]

IBM Model 1: Alignments

$s_1, s_2, \cdots s_m$ m

$\updownarrow$

$t_1, t_2, \cdots t_l$ l

- Modelling $p(s|t)$
  - problem: Data sparsity

- Alignment 'a' identifies which target word each src word originated from
  - There are total $\boxed{(l+1)^m}$ possible alignments
  - Formally, an alignment ~~is~~ 'a' is $\{a_1, \dots, a_m\}$ where each $a_j \in \{0, \dots, l\}$.

In IBM model 1, all alignments 'a' are equally likely:

$$\boxed{p(a|e,m) = \frac{1}{(l+1)^m}} \leftarrow \text{Major simplifying assumption}$$

Scanned with CamScanner

15/04/2020

$p(s|t)$: translation model

src : French (f)
tgt : English (e)

$f = f_1 f_2 f_3 \dots f_m$ $\quad e = e_1 e_2 e_3 \dots e_l \Rightarrow p(f|e)$

Two choices involved :
1. The length of the French sentence (m)
2. The choice of words $f_1 \dots f_m$

~~Let's assume~~

Alignment :
1. Each Fr. word is aligned to exactly 1 En. word
2. Alignment is many-to-one.
3. A French word can be generated by no En. word
   ↳ Special case : e_0 (NULL)

Now we attemp to model $p(f_1 \dots f_m \mid e_1 \dots e_l, \text{NULL})$ using alignment variables.

Joint distribution (Marginal)

$$p(f_1 \dots f_m \mid e_1 \dots e_l, m) = \underbrace{\sum_{a_1=0}^{l} \sum_{a_2=0}^{l} \dots \sum_{a_m=0}^{l} p(f_1 \dots f_m, a_1, \dots a_m \mid e_1 \dots e_l, m)}_{\text{How to estimate this?}}$$

IBM Model 2 def^n for model parameters

- $t(f|e) \rightarrow$ cond^{n}al. prob. of generating a French word f from an Eng. word e.

$$f \in F, \quad e \in E \cup \{NULL\}$$

- $q(j|i, l, m) \rightarrow$ prob. of alignment var. a_i taking the value j, conditioned on lengths l & m.

$$a_i \rightarrow j$$

$$l \in \{1, \ldots L\}, \quad m \in \{1, \ldots M\}, \quad i \in \{1 \ldots m\}, \quad j \in \{\cancel{1} \ldots l\}$$

$0 \ldots l$ (0: NULL)

Given these definitions:

$$p(f_1, \ldots f_m, a_1 \ldots a_m | e_1 \ldots e_l, m)$$

$$= \prod_{i=1}^{m} \underbrace{q(a_i | i, l, m)}_{\text{Fixes the index}} \cdot \underbrace{t(f_i | e_{a_i})}_{\text{Fixes the distribution of words}}$$

(V. Imp)

e.g. e = And the programme has been implemented

f = Le programme a ete mis en application

Assumptions [V. Imp]

① $q(a_i | i, l, m) \Rightarrow$ a_i does not depend on any Eng. word or any other alignment values.

② $t(f_i | e_{a_i}) \Rightarrow$ The word f_i is only dependent on the Eng. word it is aligned to

Parameter Estimation in fully observed data

(f^k, e^k, a^k) ← Training corpus consists of tuples

- a^k → alignment
- e^k → English
- f^k → French

$c(e, f)$ → # times word e is aligned to word f in the training data

$c(e)$ → # times e is aligned to any French word

$$t_{ML}(f|e) = \frac{c(e,f)}{c(e)}$$

ML: Maximum Likelihood

Remember: We are estimating this from the whole corpus (Not a single sentence)

$c(j|i, l, m)$ → # times we see Eng. sent. of length l, French sent. of len m, where word 'i' in Fr. sent. is aligned to word j in Eng

w/o specific word

$$q_{ML}(j|i, l, m) = \frac{c(j|i,l,m)}{c(i,l,m)}$$

→ Algo 5.2 [From Notes]

Applying IBM Model 2 [Imp.]

Once we have estimated $t(f|e)$ & $q(j|i,l,m)$

$\Downarrow$

$p(f,a|e)$

$\Downarrow$

$\arg\max_{e} p(e)\,p(f|e) \Leftarrow p(f|e) = \sum_{a} p(f,a|e)$

Parameter Estimation

$t(f|e)$ $\qquad$ $q(j|i,l,m)$

Data: parallel corpus

500,000 pairs of sentences

$f^1 \rightarrow e^1$

$f^2 \rightarrow e^2$

$\vdots$

$f^N \rightarrow e^N$

$[f_1 f_2 \ldots f_m] \Leftrightarrow [e_1 e_2 \ldots e_l]$

$\uparrow m \qquad \uparrow l$

Issue: In the parallel corpus, we don't know the alignment for each training example.

$\Downarrow$

Partially observable

Sol^n: ① HIRE annotators (Expensive)

IBM Model - 1 : Alignments

- All alignments 'a' are equally likely :

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

* Translation probabilities

- Next step: come up with an estimate for : $p(f \mid a, e, m)$
- In model 1, this is:

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

e.g.

e = NULL the dog barks (Eng.)

$m=3$ le   chien   chara (french)

f_1 f_2 f_3

$a = \{1, 2, 3\}$

$$p(f \mid a, e, m) = t(\text{le} \mid \text{the}) \times t(\text{chien} \mid \text{dog}) \times t(\text{chara} \mid \text{barks})$$

* The Generative Process

To generate a French string f from an Eng. string e :

- Step-1: Pick an alignment 'a' with prob. $\frac{1}{(l+1)^m}$
- Step-2: Pick the French words with prob.

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

ie. The same french word may substitute for diff. Eng word in diff. contexts

The final result:

$$\cancel{P(f)}\ p(f, a \mid e, m) = p(a \mid e, m) \times p(f \mid a, e, m) = \underbrace{\frac{1}{(l+1)^m}}_{\text{uni. prob. of alignment}} \underbrace{\prod_{j=1}^{m} t(f_j \mid e_{a_j})}_{\text{Each Fr. word an Eng. word is aligned to}}$$

~~e.g.~~ How to estimate $t(f|e)$? → Through a parallel corpus

* IBM Model-2 (Extension of model-1)

- Only difference: We now introduce alignment or distortion parameters

- $q(i|j,l,m)$ = Prob. that j^{th} French word is connected to i^{th} Eng. word, given sent. length of e is l and f is m.

- Define: $p(a|e,m) = \prod_{j=1}^{m} q(a_j | j, l, m)$

  where $a = \{a_1, \ldots, a_m\}$ is the alignment

- Result: ~~p(f|~~ $p(F, a | e, m) = \prod_{j=1}^{m} q(a_j | j, l, m) \cdot t(f_j | e_{a_j})$

e.g.

$l = 6$
$m = 7$
e = And the prog. has been implemented (len = 6)
f = $F_1 F_2 - F_3 - - - F_7$ (len = 7)
$a = \{2, 3, 4, 5, 6, 6, 6\}$

$p(a|e,7) = q(2|1,6,7) \times$
$q(3|2,6,7) \times$
$q(4|3,6,7) \times$
$q(5|4,6,7) \times$
$q(6|5,6,7) \times$
$q(6|6,6,7) \times$
$q(6|7,6,7)$

$p(f|a,e,7)$
$= t(F_1|\text{the}) \times$
$t(F_2|\text{program}) \times$
$t(F_3|\text{has}) \times$
$t(F_4|\text{been}) \times$
$t(F_5|\text{implemented}) \times$
$t(F_6|\text{implemented}) \times$
$t(F_7|\text{implemented})$

The Generative Process

To generate a French string f from an Eng. string e:

- Step-1: Pick an alignment $a = \{a_1, a_2, \dots a_m\}$ with prob.:

$$\prod_{j=1}^{m} q(a_j \mid j, l, m)$$

- Step-2: Pick the French words with prob.

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

The final result:

$$p(f, a \mid e, m) = p(a \mid e, m) \cdot p(f \mid a, e, m) = \prod_{j=1}^{m} q(a_j \mid j, l, m) \cdot t(f_j \mid e_{a_j})$$

Recovering Alignments

Given a sent. pair $e_1, e_2, \dots e_l$, $f_1, f_2, \dots, f_m$ define

$$a_j = \arg\max_{a \in \{0, \dots l\}} q(a \mid j, l, m) \times t(f_j \mid e_a)$$

* EM Training of Models 1 and 2

The Parameter Estimation Problem:

Input: $(\underbrace{e^{(k)}}_{\text{Eng. sent.}}, \underbrace{f^{(k)}}_{\text{Fr. sent.}})$ for $k = 1 \dots n$

O/P: parameters $t(f \mid e)$ and $q(i \mid j, l, m)$

Challenge: We don't have alignments on our training examples
(In ideal situation, sb would have annotated)

Parameter Estn if the Alignments are Observed

$$t_{ML}(f \mid e) = \frac{\text{Count}(e, f)}{\text{Count}(e)}, \quad q_{ML}(j \mid i, l, m) = \frac{\text{Count}(j \mid i, l, m)}{\text{Count}(i, l, m)}$$

Summary

- Key ideas in the IBM translation models:
  - Alignment variables
  - Translation parameters e.g. $t(\text{chien} \mid \text{dog})$
  - Distortion parameters e.g. $q(2 \mid 1, 6, 7)$

- The EM algorithm: an iterative algo for training the q and t parameters

- Once the parameters are trained, we can recover the most likely arg alignments on our training examples.

* **Recommender Systems**: reduce info. overload by estimating relevance

Paradigms: ① Personalized recommendations
② Collaborative (Popular among peers) ✓
③ Content-based: (More of what I've liked)
④ Knowledge-based: (what fits based on my needs)

Hybrid: Combⁿ of various inputs and/or composition of diff. mechanisms

* **Collaborative Filtering** (CF) → Pros: No knowledge engg. reqd.

Approach: Use "wisdom of the crowd" to recommend items

Basic assumptions (V. Imp.) (2)

① Users give ratings to catalog items (implicitly or explicitly)
② Customers who had similar tastes in the past, will have similar tastes in the future.

Pure CF approaches:

I/p: Only a matrix of given user-item ratings

O/p: • A (numerical) predⁿ indicating to what degree the current user will like or dislike certain items
• A top-N list of recommended items

① **User-based NN_CF**

• "Active user" → Alice Item → I (Not yet seen by Alice)

Predⁿ: Alice's rating for item I:

= Avg. of the ratings of her peers who liked the same items as Alice in the past and who have rated item I.

- Do this for all items not seen by Alice and recommend the best-rated.

(a) Measuring user similarity

Pearson Correlation (ρ) →

- Linear correlation
- Lies b/w -1 and 1

-1: neg.

1: Total +ve linear corr.

0: No corr.

a, b: Users

$r_{a,p}$: Rating of user a for item p

P: Set of items, rated both by a and b

$$\rho = sim(a,b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \cdot \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}$$

☆ **Note**: $\boxed{\rho \text{ is calculated on raw user ratings}}$

Reason: Users have different bias while rating. Some rarely give 5 ☆ while some do it easily (frequently).

→ Measures relative distance from the avg. rating

Numerator: COVARIANC

Denominator: Std. Dev. of user A * Std. Dev. of User B

Pearson correlation

- Takes differences in rating behv. into account
- Works well in usual domains, compared with alternate measures such as cosine similarity.

(b) Making Predictions

Neighbour's bias

weighted similarity

$$pred(a,p) = \underbrace{\bar{r}_a}_{\text{user's avg. rating}} + \left[\frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} sim(a,b)} \right]$$

Normalization

N → Neighbours decided on similarity threshold

IMPROVING PREDICTION

Better similarity, weighting metrices and neighbour selection

→ Give more weights ~~to~~ items that have a higher variance.

→ # Co-rated items : Use "significance weighting"
e.g. linearly reducing the weight when the # co-rated items is low.

→ Case amplification : Give more weight to "very similar" neighbours i.e. where similarity value is close to 1.

→ Nbd selection : Use similarity threshold ~~of~~ or fixed No. of neighbours.

* ITEM BASED COLLABORATIVE FILTERING

Basic idea : Use similarity b/w items (and not users) to make predictions

Cosine similarity → produces better results in item-to-item filtering

Ratings → n-dimensional space

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

→ Issue (V. Imp.) : Does not take into account user's bias.

- **Adjusted cosine Similarity**

Avg. user rating taken into account

$$\text{sim}(a, b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r})^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

- **Making predictions**

$$\text{pred}(u, p) = \frac{\sum_{i \in \text{ratedItems}(u)} \text{sim}(i, p) * r_{u,i}}{\sum_{i \in \text{ratedItems}(a)} \text{sim}(i, p)}$$

Most real-life datasets : 20-50 neighbours work well.

- **Pre-processing for item-based filtering** [That's why they are preferred]

Approach :
- Calculate all pairwise item similarities in advance.
- The nbd. to be used at run-time is typically small
- Item similarities are supposed to be more stable than user similarities.

Memory Req: ($N \approx 1$ million)

- # items = N

$\Rightarrow$ $\boxed{N^2}$ pairwise similarities to be memorized

$$\begin{array}{c|ccc} & I_1 & I_2 & I_3 \cdots \\ \hline I_1 & & & \\ I_2 & & & \\ I_3 & & & \\ \vdots & & & \end{array}$$

↓
Stable
(won't change much)

- Easily expandable ✓

In practice, matrix would be sparse, so space can be further reduced.

while user based matrix changes on small change

(7) ISSUES WITH CF (Collaborative Filtering)

* **Data Sparsity Problems**
  - Cold start problem
    - How to recommend new items? What to recommend to new users?

soln

- straightforward approach: (Imp.)
  - Force users to ~~sto~~ rate a set of items.
  - Use other methods like content-based, demographic or non-personalized in the initial phase.

in initial phase

The Cold-Start Problem [Coursera] - Univ. of Minnesota

- New user
- New item
- New system

- New user :
  - Problem: No profile, No preferences
  - No issue if system is non-personalized
  - Else: provide useful default personalized options

    To personalize further

    e.g
    - Offer popular items — Get data from its ratings
    - Demographically relevant (or age, sex) → Domain dependent
    - Product association - Browsing history or shopping cart
    - Trust-based social network data (or Referral data)

Scanned with CamScanner

- New items

→ Challenge - Can't recommend - because we don't know whether user will like or not

- Need ratings quickly

- Options
  - Use content-based approach (Degrading our rec. system)
  - Recommend to random, (or) well-chosen set of users (people with diverse tastes, tolerance, interest, influence)

- New systems
  - Syndicated data (get data from another source)
    - └ could be content data or rating data
  - Design a non-recommender system that captures data to feed a later recommender

[End-of-Coursera]

* Alternative approach to Data sparsity problems

- "Transitivity" of neighbourhoods :
- Recursive CF : Assume there is a v.close neighbour n of u who has not rated the item i yet

(This approach works better when data is sparse)

- Graph-based methods
  - "Spreading activation" : Exploit the supposed "transitivity" of customer tastes and thereby augment the matrix with add'nal. info.

[Nearest nb. based CF]

Idea : Use paths of lengths > 3 to recommend items.
(Extendable)

Scanned with CamScanner

- <u>Association Rule Mining</u>
  - Market-basket (Baby food ⇒ Diapers)
  - <u>Algo</u> :

    Trans. t_i = { Set of items purchased during that transaction }

    - Detect rules ($A \to B$) from $D = \{t_1, t_2, .. t_n\}$
    - Two qty : support and confidence (<u>Both should be high</u>)
    - Let X : set of items

      $c(X)$: How many trans. (t_i), X is present

$$\text{Support} = \frac{c(X \cup Y)}{|D|} \qquad \text{Confidence} = \frac{c(X \cup Y)}{c(X)}$$

Total # trans.

- <u>Simple approach</u> : Transform 5-point ratings into binary ratings (1 = above user avg.)

- <u>Making recommendations</u>
  - Determine "relevant" rules based on user's transactions
  - Determine items not already bought by USER.
  - Sort the items based on the rules' <u>confidence</u> values.

* Evaluating Recommender Systems

(3 types of Evaluation Metrices)

① Comparing Values

- Measure how close the predicted ratings are to the true user ratings (for all the ratings in the test set)

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |p_i - r_i|$$

p_i → predicted rating
r_i → actual rating

- Root Mean Square Error (RMSE) : similar to MAE, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - r_i)^2}$$

② Comparing Recommendations : Precision and Recall

$$Precision = \frac{tp}{tp + fp} \qquad Recall = \frac{tp}{tp + fn}$$

Rec. ≈ Info. Retr. task

(h) $Prec = \frac{|\text{good movies rec.}|}{|\text{all rec.}|}$ $\qquad$ $Recall = \frac{|\text{good movies rec.}|}{|\text{all good movies}|}$

Typically, a rec. system is tuned to increase precision
(Recall decreases as a result)

③ Metrices : Rank-posn matters

- Rank metrices extend recall and precision to take the posn of correct items in a ranked list into account.

- Relevant items are more useful when they appear earlier in the rec. list
- particularly imp. in rec. systems as lower ranked items may be overlooked by users

* Metrics : Normalized Discounted Cumulative Gain

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}$$ (Logarithmic reduc^n factor)

nDCG

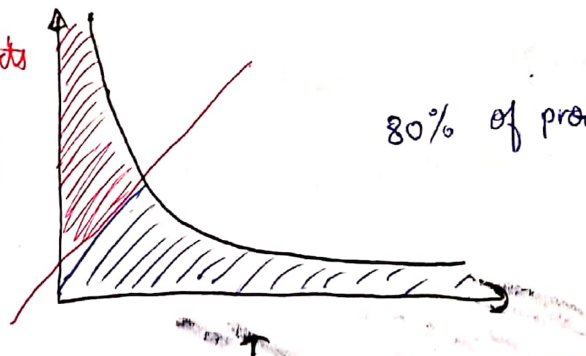
- Divide by the ideal recommendations DCG
- Normalize to the interval [0..1]

Same as discussed earlier

(Normalize DCG at rank n by the DCG value at rank n of the ideal ranking)

* An Imperfect World : Concentration Effects (long tail)

20% of products get 80% of +ve ratings



80% of products get 20% of +ve ratings

A good rec. system should be able to recommend several products from this long tail

24/04/2020

Multimedia IR

Appl.: Medical Imgs (X-rays)

* Use cases:

- Images
  - Find look-alike pictures (Google Reverse Img search)
  - Recognize landmarks (Google Lens)
  - People wearing white suits (text-based img. retrieval) → Most common

  → Two categories: ① Img to Img search
  ② Text to Img. search

- Audio
  - Finding similar sounds [Music to Music]
    - Music
    - Animal sounds
    - Copyrights
  - Retrieve relevant speech or podcasts or music [Text to Music]

- Videos
  - Finding similar videos [Video to Video]
    - Copyrights (e.g. if someone uploads full movie on YT)
  - Retrieve relevant news telecasts
  - Find specific actions: Kathak dance
  - Lecture

  } [Text to video]

* Basic IR Working Intuition

2 cases

Doc: [Images]
q: [Images]

Case-1
Doc & q → both are multimedia

Doc: [Images]
q: [Text]

Doc → Rep.
q → Rep.
} This representation should be comparable through some similarity measure

Img → RGB → N×M×3

Even if N = M = 100, N×M×3 = 30 k is huge # parameters to compare.

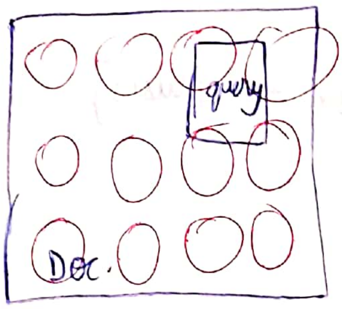
What we do? : Img is treated as a tensor of N×M×3

↓

Encoded into lower-dimen"al vector (1000)

Now, vector-based comparision b/w doc & query.

Challenge:



→ query is a small part of doc.

Divide img (doc) into smaller frame and then encode

Case-II: Doc is multimedia (Img) and query is Text.

Doc → Text & Text

↓

Now text-text comparision

How?

↓

Suppose: Image www (Source + Webpages)

Caption/title (while crawling, retrieve it)

Image Captioning   Android Photo   party   friends

ML Classifier

↓

Assign tag to images

IR system now uses this tag.

→ Same logic can be applied to audio and video.

* Piggyback Retrieval [Imp.]
  - Convert everything to text
  - Use text based retrieval

→ . Use metadata (TITLE) or loose annotations (automated or manual)
  • Image caption (Text surrounding images on web page)

* Info. Loss (while converting img/audio/video → text)

Because it may not to annotated considering all possible scenarios.

  ◦ Semantic gap

  Features: segmented-blobs; Salient regions, pixel-level histograms, Fourier descriptors etc.

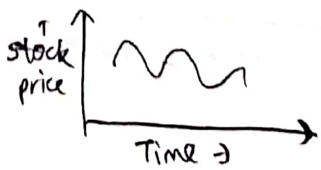
* Example: Sub-image matching

27/04/2020 CONTENT BASED RETRIEVAL (Multimedia IR)

Objective: Design a fast searching method that will search database of multimedia objects to locate objects that matches the query object, exactly or approximately.

e.g.

  ◦ Img similar to a given query img.
  ◦ Companies whose stock prices move similarly [Same stock price variation]
    ↳ MM i/p obj: Time series data



  ◦ X-ray img. that contains sth, that has a texture of a ~~tumour~~ tumor.

TERMINOLOGIES

→ **Distance or Dissimilarity**

$D(O_1, O_2)$ = Dist. b/w 2 obj. O_1 & O_2 . e.g. Euclidean dist.

→ **Whole match**: Given a collection of N obj; $O_1, O_2 \ldots O_n$ and a query Q, find those obj. that are within dist. ε from Q.

Note: Query and objs are of same type : img - img
audio - audio etc.

→ **Sub-pattern match** : Query is allowed to specify only part of the object.

e.g. • Objs. are 512 × 512 or larger size img. (medical X-rays)
• Query is 32 × 32 img. (X-ray of a tumor)

* 4 Key Requirements (V. Imp)

- Fast (distance calculation)
- Correct (Return all qualifying objects. — No false negatives)
  Note : False positives — acceptable
  They can be discarded in the post-processing step
- Small space overhead
- Dynamic (Easy to insert, delete and update objects)

* GEMINI (GEneric Multimedia object INdexIng)

Two key ideas : (To avoid 2 disadv. of sequential scanning)

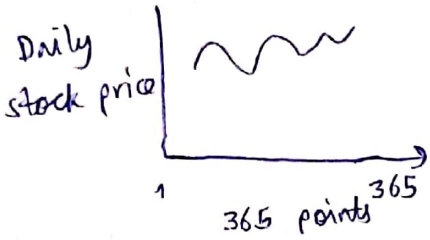
1) a '*quick-and-dirty*' test, to discard quickly the vast majority of non-qualifying objects (possibly allowing some false alarms)

2) use of *spatial access methods*, to achieve faster-than-sequential searching.

Quick and Dirty test (QD test) — Idea

To categorize obj. with a single or multiple numbers, which can help us discard many non-qualifying obj. sequences.

e.g.
If obj. is a series of Nos, a single number can be avg.



$\longrightarrow$ [. . .] f-dime^n^al vector

$\downarrow$ called

Features

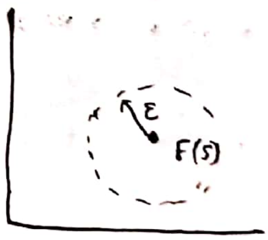
(Numbers that contain some info. about the multimedia objects)

Def^n : F() → mapping of obj. to f-dim. pt.

F(O) → f-D pt.

Then use feature space to apply QD test.

* Algo 1 : Search



1. Map the query obj. Q into a pt. F(Q) in feature space.
2. Using a spatial access method, retrieve all points within a desired tolerance ε from F(Q).
3. Retrieve the corresponding objects, compute their actual distance from Q and discard the false alarms.
   (By computing actual distance D of these points)

• No False Negatives Guarantee

Lemma (Lower Bounding) : To guarantee no false dismissals for whole-match queries, the feature extraction f^n F() should satisfy

Scanned with CamScanner

the foll. formula :

$$D_{Feature}(F(O_1), F(O_2)) \leq D(O_1, O_2)$$

* **Spatial Access Methods (SAM)**

The mapping provides the key to improve the 2^{nd} drawback of sequential scanning.

↳ R-trees (A data-structure)

↓

an index tree-structure derived from the B-tree that uses multi-dimensional indexes

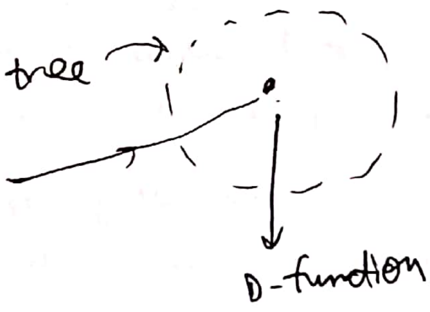
* **Algo 2 : (GEMINI)**

1. Determine the distance f^n D() b/w 2 objects
2. Find 1 or more numerical feature-extraction functions, to provide a 'quick-and-dirty' test.
3. Prove that the distance in feature space lower-bounds the actual distance D(), to guarantee correctness.
4. Use a SAM (e.g. an R-tree) to store and retrieve the f-D feature vectors.

Overview :

$\forall$ Obj. → F(O) ⇒ R-tree

Q ⇒ F(Q)



* More about D() and F()

D(): Distance function
F(): Feature function

"Good" dist. f^n

- A domain expert decides (independent of GEMINI)
- The GEMINI methodology focuses on speed of search only.

"Good" Feature f^n

- should facilitate step-3 (distance lower-bounding)
- should capture most of the ~~ct~~ characteristics of the objects,

* 1-D Time Series (e.g.)

- Variation of stock prices
- Dist. f^n: Euclidean dist., Time warping

→ Bad feature: Only 1^{st} day value of the company stock

→ Good feature: - Avg.
- coeff. of DFT (Discrete Fourier Trans. gives n-coeff., keep top-k coeff.)

$F_{DFT}(o)$:

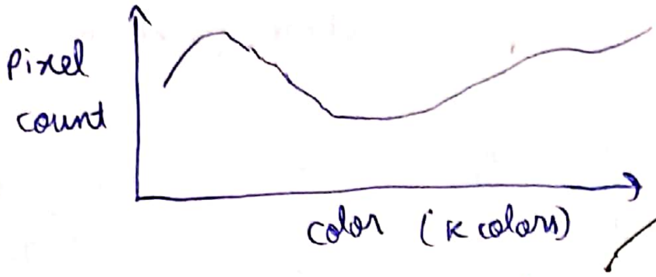
Parseval's theorem: Coeff. of DFT and dist. computed in feature space using these coeff. will be a lower bound on the Euclidean distance.
(Hence GEMINI algo ~~cal~~ can be used)

* 2-D Color Images

Distance: k-element color histogram

[Take 64 or 32 diff. colors. Given an img., plot a histogram
x: axis – colors, y-axis: # pixels]

Scanned with CamScanner



Weight matrix

How much is color i similar to color j

$$d^2_{hist.}(\vec{x},\vec{y}) = (\vec{x}-\vec{y})^t \underbrace{A}_{} (\vec{x}-\vec{y}) = \sum_i^k \sum_j^k \underbrace{a_{ij}}_{} \underbrace{(x_i - y_i)(x_j - y_j)}_{}$$

How much 2 colors are similar

square of differences

→ 3-D Feature vector

$$\vec{x} = (R_{avg.}, G_{avg}, B_{avg.})^t$$

$$G_{avg.} = \left(\frac{1}{P}\right) \sum_{p=1}^{P} G(p) \quad \Bigg| \quad R_{avg.} = \left(\frac{1}{P}\right) \sum_{p=1}^{P} R(p) \quad \Bigg| \quad B_{avg.} = \left(\frac{1}{P}\right) \sum_{p=1}^{P} B(p)$$

$$\boxed{d^2_{avg.}(\vec{x},\vec{y}) = (\vec{x}-\vec{y})^t (\vec{x}-\vec{y})}$$

Lower-Bound : Quadratic Distance Bounding Theorem

* Conclusion

Two key elements to speed up the ~~in~~ retrieval of multimedia objects are :

- QD test ← F( )
- Spatial Access methods (SAM) ← R-Trees