Subject:.....
Date:.....

Chap - 22 (NLP)

NLP is a branch of AI that helps to communicate with intelligent system using human language instead of programming laguage.

Purpose of NLP

i) Language Understanding : Help computers understand human language. Ex: Google search, Siri

ii) Language Generation: make computers generate text
Ex: ChatGPT writes essay, paragraph

iii) Machine Translation: Translate one language to another. Ex: Google translators

iv) Sentiment Analysis: detect emotion or opinion
Ex: social media review

Subject:.....

Date:.....

v) Speech recognition: Convert spoken language to text. Ex: Voice anistant (siri)

vi) Text summarization: large text → short text (meaning ful)

vii) Question Answering: answer question asked in natural language.

viii) Chatbots: conversational system respond like human.

Ex: ChatGpt

Subject:.....

Date:.....

N-gram character model

N-gram character model is a sequence of N characters that appear together.

⇒ 1-gram (Unigram) → one character ("a")

⇒ 2-gram (Bigram) → two character ("th")

⇒ 3-gram (Trigram) → Three character ("the")

A N-gram model learns how often these character groups appear in a language, helping to predict the next character.

* A N-gram model is defined a Markov chain of order N-1.

↳ next character depends on the previous N-1 characters.

Bigram → previous one Trigram → previous 2

$$P(c_1:N) = \prod_{i=1}^{N} P(c_1 | c_{1:i-1}) = \prod_{i=1}^{N} P(c_i | c_{i-2:i-1})$$

(N-gram) (Trigram)

Subject: .....

Date : .....

Uses

i) Language identification (→ detect language)

ii) Spelling correction

iii) Genre classification (→ identify the type of text (news, legal, scientific)

iv) Named Entity Recognition (NER) (→ find Names of people, drugs, place etc.)

Smoothing n-gram models

Problem: When we make an N-gram model, we count how often words or letters come together.

⇒ But sometimes, in our training text, some word combinations never appear.

Ex: "th" → this, that (common)

But "ht" → maybe never show

model makes its, probability = 0

But "http" word is available in reality.

Subject:.....
Date:.....

Solution: → smoothing

↳ we don't let anything have probability 0

Types of Smoothing

1) **Laplace Smoothing:** Just add 1 to every count

Ex: 'ht' appear 0 times, make it 1 time.

→ Simple but not very accurate.

2) **Backoff model:** If big model doesn't know something then backs off to smaller model.

Try 3-gram → "the cat" if missing

" 2-gram → "cat" "

" 1-gram → "the"

3) **Linear Interpolation:** Combine trigram, bigram, unigram together.

$$P(c_i | c_{i-2:i-1}) = \lambda_3 P(c_i | c_{i-2:i-1}) + \lambda_2 P(c_i | c_{i-1}) + \lambda_1 P(c_i)$$

where, $\lambda_3 + \lambda_2 + \lambda_1 = 1$

[λ → importance scores]

Subject:.....
Date:.....

Model evaluation

There are many ~~model~~ ways to build n-gram model (unigram, bigram, trigram). We need to know which one works best.

↳ solution model evaluation.

Step-1: Split the corpus

⇒ Training set — Teach the model

⇒ Validation set — Test how well it learned

Step-2: how to measure "goodness"

- **option-1:** Task-based evaluation –
  - Ex: language identification
  - → Accuracy predicts (%)
- **option-2:** Probability-based evaluation –
  - → give probability for validation test
  - → Higher probability → better model
  - **Problem** → for long texts probality became tiny
  - → solution: perplexity

Subject:.....
Date:.....

Step-3 : Perplexity

$$\text{perplexity}(c_{1:N}) = \frac{1}{P(c_{1:N})^{1/N}}$$

Model predicts well → probability ↑ perplexity ↓

Model confused → probality ↓ perplexity ↑

N-gram Word model

"Which word is likely come after another word?"

⇒ Unigram → single words ("The", "cat")

⇒ Bigram → 2 " ("The cat", "runs fast")

⇒ Trigram → 3 " ("The cat runs", "cat runs fast")

Problem: Word model can see new words, thats not in the training model.

Solution: use unknown token <UkN> which is replace with unknown word.

Then calculate probability.

Subject:.....
Date:.....

Ex:

Sentence	Model
I love pizza	I love pizza
I love xyzburger	I love <UNK>

→ give perfect result in Trigram.

Perplexity : Trigram > bigram > unigram

Text classification

Text classification is the process of teaching a computer to automatically sort a piece of text into a category like spam, positive/negative or news.

Example : Spam Detection

⇒ split emails into two folders

- Spam folder → all spam example
- Inbox (Ham) → all real emails

Subject:.....

Date:.....

⇒ Clues for Spam

Spam	Ham
"Buy now", "Cheap", "Viagra"	"Meeting at 3 pm"
All uppercase ("ALL CAPS")	Normally written
weirdly written (good-deserve)	Normal words

⇒ Computer build two model

$$P(\text{Message}|\text{spam}), \quad P(\text{Message}|\text{ham})$$

use bayes rule,

$$\underset{c \in \{spam, ham\}}{\text{argmax}} \; P(c|\text{message}) = \underset{c \in \{spam, ham\}}{\text{argmax}} \; P(\text{message}|c)\, P(c)$$

⇒ counting total ham and spam message

⇒ result

Subject:
Date:

Information Retrieval (IR)

Finding documents that are relevant to a user's information need.

Ex: In google — [AI Book]

Components

i) **Corpus of document:** The collection of documents the system search.

Ex: a paragraph, a page, text, file

ii) **Query in a query language:** specifies what the user wants.

Ex: [AI book] - keyword

iii) **Result set:** System thinks which documents is suitable for search.

iv) **Presentation:** How results shown. list or visualigation.

Subject:.....

Date:.....

IR Scoring Functions

→ a methods used by search engines to rank documents based on how relevant they are to a user's query.

→ BM25 is one of the best.

BM25: a formula that gives each search page a score.

High score → more relevant

BM25 three factors →

term frequency

i) **TF** → How many times a word appears

Ex: If a query is "farming"

Page A : 10 times ✓

Page B : 1 time ✗

Inverse document frequency

ii) **IDF** → How rare a word is!

Ex:

'in' — appears everywhere — low importance

"farmers" — rare — high importance

iii) **length documents :→**

long page may mention all words by chance → low score

Short, focused pages mention all query word → high score.

Subject:.....

Date :.....

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot \left(1 - b + b \cdot \frac{|d_j|}{L}\right)}$$

where,

$|d_j|$ = length of document d_j

L = avg document length in the corpus

$$L = \sum_i \frac{|d_j|}{N}$$

$k = 2$
$b = 0.75$ } parameters

$$IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

Subject:.....

Date:.....

IR System evolution

Subject:.....

Date:.....

The Page rank algorithm

PageRank is a Google's algorithm that decides which web page is most important.

The HITS algorithm —

function HITS (query) returns pages with hub and authority numbers

Pages ← EXPAND-PAGES (REVENT-PAGES) (query))

for each p in pages do

    p. AUTHORITY ← 1

    p. HUB ← 1

repeat until convergence do

    for each p in Pages do

        p. AUTHORITY ← $\sum_i$ INLINK$_i$(p). HUB

        p. HUB ← $\sum_i$ OUTLINK$_i$ (p). AUTHORITY

    NORMALIZE (Pages)

return pages

Subject:.....
Date:.....

The pageRank for a page p is defined as:

$$PR(p) = \frac{1-d}{N} + d \sum \frac{PR(in_i)}{C(in_i)}$$

where

N = Total Pages

d = damping factor (≈ 0.85)

i = Page linking to page p

C(i) = number of outgoing links on page i