EE6041

TEXT ANALYTICS AND NATURAL LANGUAGE PROCESSING

Syllabus:

The course will cover the following aspects:
1. Fundamentals: students will be introduced to basic concepts of text analytics, such as application domains, tasks, NLP aspects of text analysis, text pre-processing (tokenization, stemming, stop-word removal), text representation and modelling (Ngrams, bag of words, bag of concepts, word embedding), benchmarking and evaluation methods (standard datasets, precision, recall, F-measure, ROC curve).
2. Tasks & Tools: Consideration of the main text analytics tasks and techniques such as, text categorization, text clustering, concept/entity extraction, sentiment analysis, text summarization, and language detection. In this part students will practice with some of the popular open-source tools that implement text analytics tasks (e.g., RapidMiner, Weka, Carrot2, LingPipe, NLTK, Ontotext, GATE).
3. Big Data & Cloud: Students will gain a practical knowledge of big data text analytics by learning to conduct text analytics experiments and pipelines in commercially available cloud platforms. Students will also gain practice in the use of well known chaining text analytics cloud APIs such as: IBM Watson's Natural Language Understanding, Google Cloud Natural Language, Amazon Comprehend, and Microsoft Azure Text Analytics API.
4. Emerging trends in text analytics and NLP.
The practical experiments and platforms that are considered in this module will reflect the current state of the art in Text Analytics and NLP.

Prime Texts:

S. Struhl (2015) Practical Text Analytics , Kogan Page ISBN: 0749474017

LESSON 1

REGULAR EXPRESSIONS

- Formal language for specifying text strings
- Disjunctions
  - Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1 2 3 4 5 6 7 8 9 0]	Any digit

  - Ranges

Pattern	Matches
[A-Z]	An upper case letter
[a-z]	A lower case letter
[0-9]	A single digit

check: regexpal.com

- Negations

carat (^) means negation only when first in []

Pattern	Matches
[^A-Z]	Not an upper case letter
[^Ss]	Neither 'S' nor 's'
[^e^]	Neither e nor ^
a^b	The pattern a carat b

- The pipe | for disjunction

Pattern	Matches
groundhog | woodchuck	The strings- groundhog or woodchuck
yours | mine	The strings- yours or mine
a|b|c	= [abc]
[gG]roundhog |[wW]oodchuck	groundhog, woodchuck, Groundhog, woodchuck

- ? * +

Pattern	Matches	
colou?r	optional prev char	color colour
oo*h!	0 or more of prev char	oh! ooh! oooh!
o+h!	1 or more of prev char	oh! ooh! oooh!
baa+		baa baaa baaaa
beg.n	Any character	begin begun beg3n

- Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

To check for a period(.), we use the esc. character '\'

- **Errors**

  Type 1: False Positives

  Matching strings we should not have matched

  Type 2: False Negatives

  Not matching strings we should have matched

- Reducing error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision - min false positives
  - Increasing coverage or recall - min false negatives

2. WORDS AND CORPORA

- **Fragments:** Incomplete sentences, words, or phrases that occur in spoken or written language. They often result from interruptions, self-corrections, or unfinished thoughts.

  e.g., I was just thinking, um maybe we should- no, never mind.

- **Filled Pauses:** non-lexical utterances or words used by speakers to fill gaps while they think or hesitate. They are common in spoken language and often considered disfluencies.

  e.g., um, uh, er, etc.

sample sentence: "Seuss's cat in the hat is different from other cats!"

- **Lemma:** canonical or base form of a word, as it appears in a dictionary or lexicon. It is the form of the word used to represent all its variations.

  In the sample text,
  cat and cats = same lemma

Lemmatization is the process of reducing words to their lemma. This helps in normalizing text for analysis, ensuring that different inflected forms of a word are treated as the same.

- **Wordform** : Any specific inflected or derived version of a word that appears in actual use.
  Wordforms are the surface-level realizations of a lemma based on grammatical or contextual requirements

In the sample text,
cat and cats = different word forms

Identifying wordforms is crucial for understanding sentence structure, as wordforms provide context-sensitive grammatical information

- **Type** : An element of the vocabulary.
- **Token** : An instance of that type in running text.

- **Representation**

N = number of tokens

V = vocabulary = set of types

$|V|$ = size of vocabulary

- **Heaps Law**
(or Herdan's Law)

$$|V| = kN^{\beta}$$

where often $0.67 < \beta < 0.75$

It states that as the length of a text increases, the growth of the vocabulary slows down.
This is due to the fact that many words start to repeat rather than new words being formed.

3. WORD TOKENIZATION

- Every NLP task requires text normalization:
  1. Tokenization (segmenting) words
  2. Normalizing word formats
  3. Segmenting sentences
- Space-based tokenization
- Issues in Tokenization:
  - can't blindly remove punctuation
    e.g., m.p.h., Ph.D., $45.55, hashtags, URLs, etc.
  - clitic: a word that doesn't stand on its own
    "are" in "we're"

4. WORD NORMALIZATION

- Putting words / tokens in a standard format.
- case folding
- Lemmatization is done by Morphological Parsing
  - Morphemes:
    The small meaningful units that make up words
    Stems: The core meaning-bearing units.
    Affixes: Parts that adhere to stems, often with grammatical functions.
  - Morphological Parsers:
    Parse cats into two morphemes cat and s
- Stemming
  Reduce terms to stems, chopping off affixes crudely

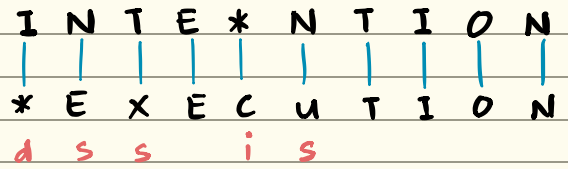
  accurate → accur
  soundings → sound
  exception → except
- Ponter stemmer
  Based on a series of rewrite rules run in series

  Some sample rules:
  ATIONAL → ATE (relational → relate)

LESSON 2

I. MINIMUM EDIT DISTANCE

- **Edit distance** is the min number of editing operations (insertion, deletion, substitution) needed to transform one into the other.
- Two strings & their alignment:



- If each operation has cost of 1, edit distance is 5
- If substitutions cost 2 (Levenshtein), edit distance is 8
- Applications of edit distance:
  1. Alignment in Computational Biology
     Given two sequences, align each letter to a letter or gap
  2. Evaluating machine translation and speech recognition.
  3. Named Entity extraction & entity conference
- How to find the min edit distance?
  - Searching for a path (sequence of edits) from the start string to the final string

- For two strings X (len = n) and Y (len = m)

  We define $D(i,j)$ = the edit dist b/w $X[1 \dots i]$ and $Y[1 \dots j]$

  ∴ The edit dist b/w X and Y is $D(n,m)$

II. COMPUTING MINIMUM EDIT DISTANCE

- Dynamic Programming : A tabular computation of $D(n,m)$
- computation (Levenshtein):

1. Initialization

   $D(i,0) = i$

   $D(0,j) = j$

2. Recurrence Relation

   For each $i = 1 \dots M$

   For each $j = 1 \dots N$

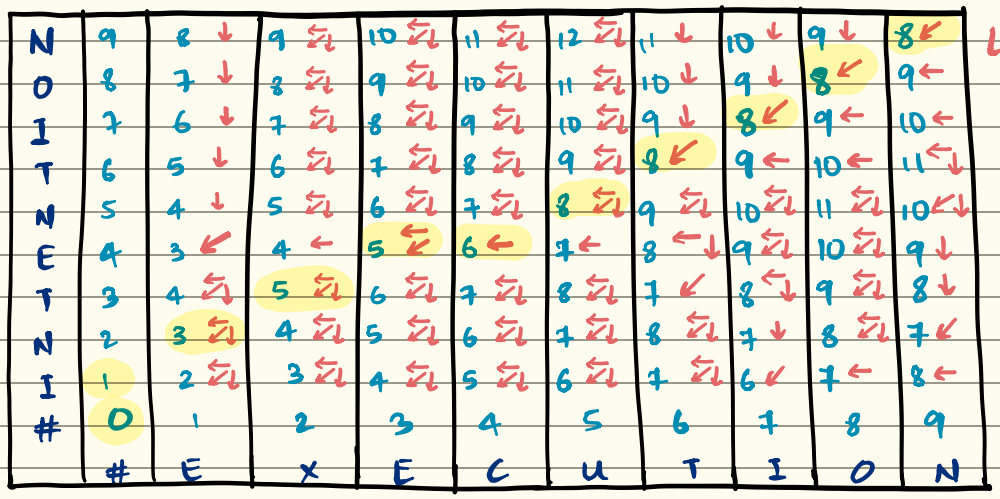
$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 ; & \text{if } X(i) \neq Y(j) \\ 0 ; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

3. Termination:

   $D(N,M)$ is distance

N	9	8	9	10	11	12	11	10	9	8	L = 8
O	8	7	8	9	10	11	10	9	8	9	
I	7	6	7	8	9	10	9	8	9	10	
T	6	5	6	7	8	9	8	9	10	11	
N	5	4	5	6	7	8	9	10	11	10	
E	4	3	4	5	6	7	8	9	10	9	
T	3	4	5	6	7	8	7	8	9	8	
N	2	3	4	5	6	7	8	7	8	7	
I	1	2	3	4	5	6	7	6	7	8	
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	

III BACKTRACE FOR COMPUTING ALIGNMENTS



- Every non-decreasing path from (0,0) to (M,N) corresponds to an alignment of the two sequences
- An optimal alignment is composed of optimal subalignments
- Performance:
  - Time: O(nm)
  - space: O(nm)
  - Backtrace: O(n+m)

IV. SOUNDEX

Phonetic algorithm used in NLP to encode words or names based on their sounds rather than their spelling.

LESSON 3

I. INTRODUCTION TO N-GRAMS

- **Probablistic Language Models**

Goal: Compute the probability of a sentence or sequence of words.

$P(W) = P(w_1, w_2, w_3, w_4, w_5, ..., w_n)$

↳ Joint probability of the whole string

$P(w_5 | w_1, w_2, w_3, w_4)$

↳ conditional probability of the last word given the previous words

- Language Models → determine $P(W)$, $P(w_5 | w_1, w_2, w_3, w_4)$

- **How to compute P(W)**

  - Intuition: Chain rule of probability

  - compute this joint prob: P(its, water, is, so, transparent, that)

  - **Chain Rule of Conditional Probability**

    For a set of n events $A_1, A_2, A_3, ..., A_n$, the chain rule states:

$$P(A_1, A_2, ..., A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1, A_2) \cdot ... \cdot P(A_n|A_1, A_2, A_3, ..., A_{n-1})$$

Prob that first event occurred

Prob that 2nd event occured given that the 1st event occured

Prob that 3rd event occured given that the 1st, 2nd events occured

P("its water is so transparent") =

P(its) · P(water | its) · P(is | its water) · P(so | its water is) · P(transparent | its water is so)

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

- **Markov Assumption**

To calculate the probabilities

P(the | its water is so transparent that) ≃ P(the | that)

or maybe

P(the | its water is so transparent that)

≃ P(the | transparent that)

$$P(w_1 w_2 \dots w_n) \simeq \prod P(w_i \mid w_{i-k} \dots w_{i-1})$$

we approximate each component in the product.

- **Unigram Model**

Simplest case of a Markov model

$$P(w_1 w_2 \dots w_n) \simeq \prod P(w_i)$$

- Bigram Model

condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \simeq P(w_i \mid w_{i-1})$$

II. ESTIMATING N-GRAM PROBABILITIES

- Estimating Bigram Probabilities
  - The maximum likelihood estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

example:

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(I \mid <s>) = \frac{c(<s>, I)}{c(<s>)} = \frac{2}{3} = 0.67$$

Bigram prob. calculation:

$$P(<s> I \text{ want english food } </s>) = P(I \mid <s>) \times P(\text{want} \mid I) \times P(\text{english} \mid \text{want}) \times P(\text{food} \mid \text{english}) \times P(</s> \mid \text{food})$$

- we do the math in log space

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

  - To avoid underflow
  - Adding is faster than multiplying

III. EVALUATION & PEPLEXITY

- Extrinsic Evaluation of N-gram models

To compare model A & model B:
- Put each model in a task (like spelling correction, speech recognizer, etc)
- Run the task, get accuracies for A & B
- Compare accuracies

- Disadvantages of Extrinsic Evaluation
  - Time consuming

- Intrinsic Evaluation: Perplexity

This is bad approximation unless:
- The test data looks just like the training data.
- Generally, only useful in pilot experiments.

- Perplexity

Perplexity is the probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \dots w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

$$= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 \cdots w_{i-1})}} \quad (N \text{ gram})$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid P_{w_{i-1}})}} \quad (\text{bigram})$$

Minimizing Perplexity is the same as maximizing probability.

- The Shannon Game Intuition for perplexity

A call routing phone system gets 120K calls and has to recognize
- "operator" (occurs 1 in 4 calls)
- "sales" (1 in 4)
- "Technical Support" (1 in 4)
- 30,000 different names (each name occuring 1 time in the 120K calls)

To get the perplexity of this sequence of length 120K:

1. Multiply 120K probabilities
   90K of which are $\frac{1}{4}$ and 30K of which are $\frac{1}{120K}$

2. Take the inverse 120,000th root

$$PP = \left(\underbrace{\frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} \times \ldots \times}_{\substack{30K \times + 30K \times + 30K \times \\ \text{total } 90K \text{ times}}} \underbrace{\frac{1}{120K} \times \frac{1}{120K} \ldots}_{30K \text{ times}} \right)^{\frac{-1}{120,000}}$$

can be arithmetically simplified to just N=4

operator ($\frac{1}{4}$)

sales ($\frac{1}{4}$)

tech support ($\frac{1}{4}$)

30K names ($\frac{1}{120,000}$)

$$PP = \left(\frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{1}{120K} \right)^{\frac{-1}{4}}$$

$$= 52.6$$

IV. GENERALIZATION AND ZEROES

• Shannon Visualization Method

- The Shannon visualization method for Bigrams involves simulating text generation based on a bigram language model to understand how probabilities guide word selection.
- Demonstrates how sequence of words can be generated by repeatedly sampling the next word based on the probabilities provided by the bigram model until an end-of-sentence token (EOS) is reached.

• Zero Probability Bigram

Bigrams with zero probability to the test set

⇒ cannot compute perplexity (can't divide by 0)!

V. ADD-ONE (LAPLACE) SMOOTHING

- Used for handling zero probabilities in language models.
- Ensures that no event has a zero probability.
- Pretend we saw each word one or more times than we did.
- Max Likelihood Estimate (MLE):

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate

$$P_{Add\text{-}1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

LESSON 4

1. THE SPELLING CORRECTION TASK

- Spelling Tasks
  1. Spelling Error Detection
  2. Spelling Error Correction
     - Autocorrect
     - Suggest correction
     - Suggestion lists
- Types of Spelling Errors
  - Non-word Errors [graffe → giraffe]
  - Real word Errors
    - → Typographical Errors [three → there]
    - → Cognitive Errors [piece → peace, too → two] (homophones)
- Non-word Spelling Errors
  - Detection:
    - Any word in a dictionary is an error
  - correction:
    - Generate candidates: real words similar to error.
    - choose the best on based on:
      1. Shortest weighted edit distance
      2. Highest noisy channel probability.

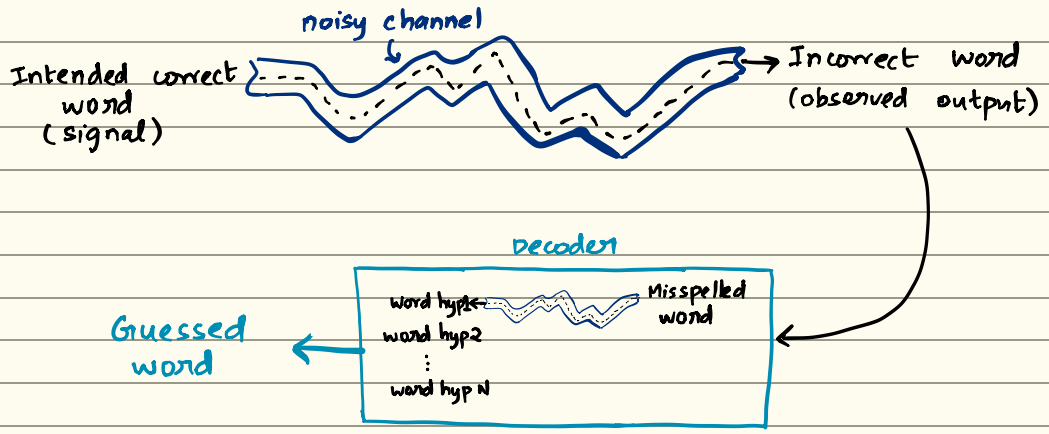
- Real word Spelling Errors

For each word w, generate candidate set:
- Find candidate words with similar pronunciations
- Find candidate words with similar spelling.
- Include w in candidate set.

Choose best candidate based on
- Noisy channel
- classifier

II. NOISY CHANNEL MODEL OF SPELLING

- The noisy channel model of spelling is a framework often used in natural language processing to correct spelling errors
- It treats the problem of spelling correction as one of communication over a "noisy channel"
- In the noisy channel, the intended word (the signal) may be distorted into an incorrect word (the observed output) due to errors (the noise).



Using Bayes' theorem, the model computes the most probable correct word (w) given the observed misspelled word (o):

$$P(w|o) = \frac{P(o|w) \cdot P(w)}{P(o)}$$

P(w|o): Probability that w is the correct word given the observed word is o.

P(o|w): Probability of observing o given that the intended word was w (error model)

P(w): Prior probability of the word w (language model)

P(o): Probability of the observed word o

LESSON - 5

I. WHAT IS TEXT CLASSIFICATION

- Spam or not spam
- Authorship attribution - which author wrote which text
- Gender identification - male or female author
- Using Naive Bayes classifier
- Positive or negative movie review

- **Text classification : Definition**
  - Input :
    - a document d
    - a fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$
  - Output : a predicted class $c \in C$

- **Naive Bayes classifier**
  - Representation: Bag of words

Baye's Rule

For a document d and a class c

$$P(c|d) = \frac{P(d|c)\,P(c)}{P(d)}$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(c|d)$$

$$= \underset{c \in C}{\operatorname{argmax}} \; \frac{P(d|c)\,P(c)}{\cancel{P(d)}}$$

$$= \underset{c \in C}{\operatorname{argmax}} \; P(d|c)\,P(c)$$

Document d represented as features $x_1, x_2, \ldots, x_n$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(x_1, x_2, \ldots, x_n | c)\,P(c)$$

- **Multinomial Naïve Bayes Independence Assumptions**

$$P(x_1, x_2, \ldots, x_n | c)$$

1. Bag of words assumption: Assume position doesn't matter

2. Conditional Independence: Assume the feature probabilities $P(x_i | c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n | c) = P(x_1|c) \cdot P(x_2|c) \cdot P(x_3|c) \cdot \ldots \cdot P(x_n|c)$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(x_1, x_2, \ldots, x_n | c)\,P(c)$$

$$c_{MNB} = \underset{c \in C}{\operatorname{argmax}} \; P(c_j) \prod_{x \in X} P(x|c)$$

causes integer underflow

we do everything in log space

$$C_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i \mid c_j) \right]$$

This makes it a linear model

	Doc	words	class
Training	1	London is the capital of GB 6	GB
	2	Oxford is a city in GB 4	GB
	3	Dublin is the capital of Ireland 2	IE
	4	Limerick is a city in Ireland 1	IE
Test	5	University of Limerick wolves	

V = {London, is, the, capital, of, GB, oxford, a, city, in, Dublin, Ireland, Limerick}

|V| = 13

1. Prior from Training

$P(GB) = \frac{2}{4} = 0.5$ $\quad$ $P(IE) = \frac{2}{4} = 0.5$

2. Drop "university", "wolves"

3. likelihoods from training

$$P(w_i \mid C) = \frac{count(w_i, c) + 1}{\sum_{w \in V} count(w, c) + |V|}$$

$$P(of \mid GB) = \frac{count(of, GB) + 1}{\text{count of all words in GB} + |V|}$$

$$= \frac{1+1}{12+13} = \frac{2}{25}$$

$$P(Limerick \mid GB) = \frac{count(Limerick, GB) + 1}{\text{count of all words in GB} + |V|} = \frac{0+1}{12+13}$$

$$= \frac{1}{25}$$

$$P(of \mid IE) = \frac{count(of, IE) + 1}{\text{count of all words in IE} + |V|} = \frac{1+1}{12+13} = \frac{2}{25}$$

$$P(limerick \mid IE) = \frac{count(limerick, IE) + 1}{\text{count of all words in IE} + |V|} = \frac{1+1}{12+13} = \frac{2}{25}$$

4. Scoring the test set

$$P(GB)\,P(S \mid GB) = 0.5 \times \frac{2}{25} \times \frac{1}{25} = \frac{1}{25^2}$$

$$P(IE)\,P(S \mid IE) = 0.5 \times \frac{2}{25} \times \frac{2}{25}$$

$$= \frac{2}{25^2}$$

- Binary Multinomial Naive Bayes
  - clip out word counts at 1
  - First remove all duplicate words from d

Then compute NB using same equation

$$C_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in \text{positions}} P(w_i \mid c_j)$$

Four Original Docs:

- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

After per-document Binarization

- it was pathetic the worst part boxingscenes
- no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

Binarization is within doc

QP AUTUMN - 2022

Q1.

(a) Write a regular expression for validating UL student email addresses in this format: studentID@studentmail.ul.ie, where studentID is an 8-digit number

```
^[0-9]{8}@studentmail\.ul\.ie$
```

(b) How many Types and Tokens are in the sentence:

"NLP stands for Natural Language Processing!"

- Token:
  A token is any individual unit in a sentence, often separated by spaces or punctuation

(c) Explain the difference between lemmatization and stemming and provide an example for each

Lemmatization and stemming are two common techniques used in NLP to reduce words to their root form.

- Stemming:
  - Rule based process of stripping suffixes (and sometimes prefixes) from a word to reduce it to its stem or base form.
  - Like removing -ing, -ed, -s
  - does not necessarily produce a valid word.
  - Use case: When speed is prioritized over precision e.g., Information Retreival System.

- running → run

  runner → runner

  runs → run

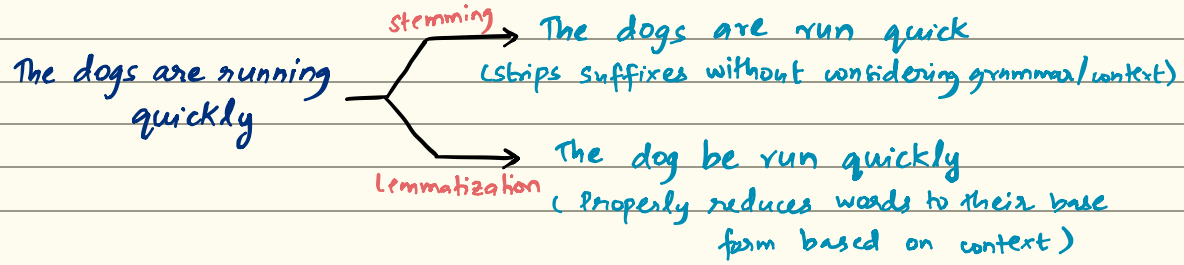
- **Lemmatization:**

  Lemmatization reduces a word to its lemma, which is its dictionary or base form.

- It takes into account the word's meaning and grammatical context.
- Produces valid words that belong to the language
- use case: When precise word forms matter e.g., linguistic or semantic analysis.

  running → run
  runner → run
  runs → run

Aspect	Stemming	Lemmatization
Output	May not be valid word	Always a valid word
Technique	Rule-based (heuristic)	vocabulary-based + grammar
speed	Faster	Slower
Accuracy	Less accurate	More accurate
Context	Ignores word context	considers word context



Q2.

(a) With the help of an example, explain the difference between Hamming distance and Levenshtein distance.

- **Hamming Distance:** Number of positions at which the corresponding characters in two strings of the same length differ.
  - Requires strings to be of equal length.
  - Only substitutions are considered (no insertions or deletions)

  e.g., For the strings kitten, sitten
  H.D = 1
  - Application: error detection in fixed-length codes.

- **Levenshtein Distance:** Minimum number of insertions, deletions, or substitutions required to transform one string into another.
  - works for strings of different lengths.
  - Consider all three operations: insertion, deletion, and substitution

  k i t t e n
  s i t t i n g
  s s i

  Levenshtein dist = 3

  - Application: Spelling corrections, DNA sequencing, NLP.

(b) Explain the advantage of using Dynamic Programming over Recursion for computing the Minimum Edit Distance.

DP is advantageous over recursion for computing min edit distance because it avoids redundant computations by storing intermediate results in a table, ensuring each subproblem is solved only once.

This reduces the time complexity

(c)

	" "	B	A	R
" "	0	1 ←	2 ←	3 ←
C	1 ↑	2	3	4
A	2 ↑	3	2	3
R	3 ↑	4	3	2

3. without add-one smoothing:

The bigram probability for a sentence is calculated as:

$$P(w_1, w_2, w_3, .., w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdot \ldots \cdot P(w_n|w_{n-1})$$

$$P(w_i|w_{i-1}) = \frac{\text{Bigram count } (w_{i-1}, w_i)}{\text{unigram count } (w_{i-1})}$$

"I want lunch"

1. $P(I) = \frac{\text{Unigram count}(I)}{\text{Total unigrams}}$

$$= \frac{2533}{2533+927+2417+746+158+1093+341+278}$$

$$= \frac{2533}{8493}$$

$$= 0.2983$$

2. $P(want \mid I) = \frac{C(want, I)}{C(I)}$

$= \frac{827}{2533}$

$= 0.3265$

3. $P(lunch \mid want) = \frac{C(lunch, want)}{C(want)}$

$= \frac{5}{927} = 0.0054$

$P(\text{"I want lunch"}) = P(I) \cdot P(want \mid I) \cdot P(lunch \mid want)$

$= 0.2933 \times 0.3265 \times 0.0054$

Add-One Smoothing

$|V| = 8$

$P(w_i \mid w_{i-1}) = \frac{count(w_i, w_{i-1}) + 1}{count(w_{i-1}) + V}$

$P(i) \rightarrow$ same

$P(want \mid i) = \frac{count(want, i) + 1}{count(i) + V}$

$= \frac{827 + 1}{2533 + 8}$

$P(lunch \mid want) = \frac{count(lunch, want) + 1}{count(want) + V}$

$= \frac{5 + 1}{927 + 8}$

4. Priors:

$$P(FR) = \frac{2}{4} = 0.5$$

$$P(IE) = \frac{2}{4} = 0.5$$

Conditional Prob:

$$P(word|class) = \frac{\text{Word count in class} + 1}{\text{Total words in class} + V}$$

V = { Paris, is, the, capital, of, France, Nice, a, city, in, Dublin, Ireland, Limerick }

$|V| = 13$

	Truth: Y	Truth: N
classifu: Y	T.P	F.P
classifu: N	F N	T N

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

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

$Acc = \frac{TP + TN}{TP + FP + TN + FN}$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall = prop of actual +ve correctly identified

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

$$F_1 = 2 . \frac{Pr . Re}{Pr + Re}$$