UNIT I
Introduction of NLP and Preprocessing of Text Data

By
Dr. S. S. Gharde
Dept. of Information Technology/ AIML
Government Poly technic Nagpur

Contents..

- Introduction to NLP
- Regular Expressions
- Words
- Corpora
- Text Normalization
- Minimum Edit Distance

Introduction to NLP

- **Definition:** Natural Language Processing (NLP) is a subfield of computer science and artificial intelligence that deals with the interaction between computers and human languages.

- **Goal of NLP: T**o enable computers to understand, interpret, and generate natural language, the way humans do.

- NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models.

Source: *https://www.geeksforgeeks.org/introduction-to-natural-language-processing/*

Introduction to NLP

- NLP tasks
  - **Speech recognition**, also called speech-to-text
  - **Part of speech tagging**, also called grammatical tagging
  - **Word sense disambiguation**
  - **Named entity recognition**
  - **Sentiment analysis**
  - **Natural language generation**

Source: https://www.ibm.com/topics/natural-language-processing

Introduction to NLP

- NLP tools and approaches /**Open source NLP libraries**
  - Python
  - Natural Language Toolkit (NLTK)
  - Statistical NLP, machine learning, and deep learning
  - Apache OpenNLP
  - Stanford NLP
  - MALLET

Introduction to NLP

- Main applications of NLP
  - language translation
  - speech recognition
  - sentiment analysis
  - text classification
  - information retrieval
  - Spam detection
  - Virtual agents and chatbots

Regular Expressions

- Formally, a regular expression is an algebraic notation for characterizing a set of string.

- Regular expressions are particularly useful for searching in texts, when we have a pattern to search for and a corpus of texts to search through.

- The corpus can be a single document or a collection.

Regular Expressions -Source: Dan Jurafsky and James Martin, “Speech and language Processing”

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

- Negations [^Ss]
  - Carat means negation only when first in []

Pattern	Matches (single characters)	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a^b	The pattern a carat b	Look up a^b now

Regular Expressions: More Disjunction

- Woodchuck is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	woodchuck
yours|mine	yours
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	Woodchuck

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

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

Example

- Find me all instances of the word “the” in a text.

  the

  Misses capitalized examples

  [tT]he

  Incorrectly returns other or theology

  [^a-zA-Z][tT]he[^a-zA-Z]

- More Examples to write regular expressions for given language.

Words

- we need to decide what counts as a word.
- He stepped out into the hall, was delighted to encounter a water brother
- 13 words if we don't count punctuation marks as words.
- 15 if we count punctuation
- Counting words like comma, period, etc are depends on the task

Words cont…

- The Switchboard corpus of American English telephone conversations between strangers was collected in the early 1990s;
- an utterance is the spoken correlate of a sentence:

  I do uh main- mainly business data processing

- This utterance has two kinds of disfluencies.
  - The broken-off word main- is called a fragment.
  - Words like uh and um are called fillers or filled pauses.

Words cont…

- A lemma is a set of lexical forms having the same stem, the same major part-of-speech, and the same word sense.

- The word-form is the full inflected or derived form of the word.

Words cont…

Herdan's Law or Heaps' Law

- Counting Words….
- **Types** are the number of distinct words in a corpus
- V - the set of words in the vocabulary
- |V| - the number of types is the vocabulary size
- N - number of Tokens

Words cont…

Herdan's Law or Heaps' Law

- Relationship between the number of types |V| and number of tokens N is called Herdan's Law or Heaps' Law

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

where k and β are positive constants,

and $0 < \beta < 1$.

- The value of β depends on the corpus size and the genre.
- For the large corpora β ranges from .67 to .75.

Words cont…

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Corpora

Words don't appear out of nowhere!

A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

Corpora

- **Language**: 7097 languages in the world
- **Variety**, like African American Language varieties.
  - AAE Twitter posts might include forms like "*iont" (I don't)*
- **Code switching**, e.g., Spanish/English, Hindi/English:

  S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)

  [For the first time I get to see @username actually being hateful! it was beautiful:)]

  H/E: dost tha or ra- hega ... dont wory ... but dherya rakhe

  [“he was and will remain a friend ... don’t worry ... but have faith”]

- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics**: writer's age, gender, ethnicity, SES

Corpora

- **Code switching** : It's also quite common for speakers or writers to use multiple languages in a single communicative act, a phenomenon called code switching.
- Code switching is enormously common across the world.

dost tha or ra- hega ... dont wory ... but dherya rakhe

["he was and will remain a friend ... don't worry ... but have faith"]

Corpora

Corpus **datasheets** or data statement: A datasheet specifies properties of a dataset like:

- **Motivation**: Why was the corpus collected?By whom? Who funded it?
- **Situation**: In what situation was the text written?
- **Language variety:** What language (including dialect/region) was the corpus in?
- **Collection process**: If it is a subsample how was it sampled? Was there consent? Pre-processing?
- **Annotation process:** What are the annotations, what are the demographics of the annotators, how were they trained, how was the data annotated?
- **Distribution**: Are there copyright or other intellectual property restrictions?

Text Normalization

- **Text normalization** is a pre-processing step aimed at improving the quality of the text and making it suitable for machines to process.

- Text normalization reduces the word to its base form by removing the inflectional part.

https://subscription.packtpub.com/book/data/9781800200937/1/ch01lvl1sec04/text-normalization
https://www.scaler.com/topics/nlp/what-is-text-normalization-in-nlp/

Text Normalization

- Every NLP task requires text normalization:
  1. Tokenizing (segmenting) words
  2. Normalizing word formats
  3. Segmenting sentences

Text Normalization

Tokenization

- In this process, we break down a piece of input text into smaller units called tokens.
- The text is commonly converted into individual tokens by splitting them from whitespaces.
- Regular expressions can also be used for tokenization.

Sentence: "I am a big fan of metal music!"

Whitespace tokenizer:
"I","am","a","big","fan","of","metal","music!"

Text Normalization

Tokenization

```
import re
my_string = "I love NLP!!!"
PATTERN = r"\w+"
re.findall(PATTERN, my_string)
```

Output

['I','love','NLP']

Text Normalization

- The most commonly used way to implement tokenization is by using the NLTK library in python.

```
from nltk import word_tokenize
sentence = 'I love NLP!!'
words = word_tokenize(sentence)
print(words)
```

Output

['I', 'love', 'NLP', '!', '!']

Text Normalization

Instead of

- white-space segmentation
- single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

- Three common algorithms:
  - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
  - **Unigram language modeling tokenization** (Kudo, 2018)
  - **WordPiece** (Schuster and Nakajima, 2012)
- All have 2 parts:
  - A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
  - A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Byte Pair Encoding (BPE) token learner

Let vocabulary be the set of all individual characters

= {A, B, C, D,…, a, b, c, d….}

- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until *k* merges have been done.

BPE token learner algorithm

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Byte Pair Encoding (BPE) Addendum

Most subword algorithms are run inside space-separated tokens.

So we commonly first add a special end-of-word symbol '__' before space in training corpus

Next, separate into letters.

BPE token learner

Original (very fascinating) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Add end-of-word tokens, resulting in this vocabulary:

vocabulary

_, d, e, i, l, n, o, r, s, t, w

BPE

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2 l o w e s t _	
6 n e w er _	
3 w i d er _	
2 n e w _	

Merge er _ to er_

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 l o w e s t _	
6 n e w er_	
3 w i d er_	
2 n e w _	

BPE

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 l o w e s t _	
6 n e w er_	
3 w i d er_	
2 n e w _	

Merge n e to ne

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2 l o w e s t _	
6 ne w er_	
3 w i d er_	
2 ne w _	

BPE

The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE token **segmenter** algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
- (test frequencies don't play a role)

So: merge every e r to er, then merge er _ to er_, etc.

- Result:
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"

Word Normalization

- Putting words/tokens in a standard format
  - U.S.A. or USA
  - uhhuh or uh-huh
  - Fed or fed
  - am, is, be, are

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Stemming and lemmatization

- Stemming and lemmatization differ in their approach and sophistication but serve the same objective.

- Stemming is a simpler, heuristic rule-based approach that chops off the affixes of words.

- The most famous stemmer is called the Porter stemmer, published by Martin Porter in 1980.

Stemming and lemmatization

"Stemming is aimed at reducing vocabulary and aid understanding of morphological processes. This helps people understand the morphology of words and reduce size of corpus."

After stemming using Porter's algorithm, this sentence will be reduced to the following:

"Stem is aim at reduce vocabulari and aid understand of morpholog process . Thi help peopl understand the morpholog of word and reduc size of corpu ."

Porter Stemmer

- Based on a series of rewrite rules run in series
  - A cascade, in which output of each pass fed to next pass
- Some sample rules:

$$\begin{aligned}
\text{ATIONAL} &\rightarrow \text{ATE} && \text{(e.g., relational} \rightarrow \text{relate)} \\
\text{ING} &\rightarrow \epsilon && \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\
\text{SSES} &\rightarrow \text{SS} && \text{(e.g., grasses} \rightarrow \text{grass)}
\end{aligned}$$

Stemming and lemmatization

- Lemmatization approaches this task in a more sophisticated manner, using vocabularies and morphological analysis of words.
- In the study of linguistics, a morpheme is a unit smaller than or equal to a word.
- When a morpheme is a word in itself, it is called a root or a free morpheme. Conversely, every word can be decomposed into one or more morphemes. The study of morphemes is called morphology.
- This base or dictionary form of the word is called a *lemma*, hence the process is called lemmatization. StanfordNLP includes lemmatization as part of processing.

Lemmatization

Represent all words as their lemma, their shared root

= dictionary headword form:

- *am, are, is* → *be*
- *car, cars, car's, cars'* → *car*
- Spanish quiero (‘I want’), quieres (‘you want’)

→ querer ‘want'

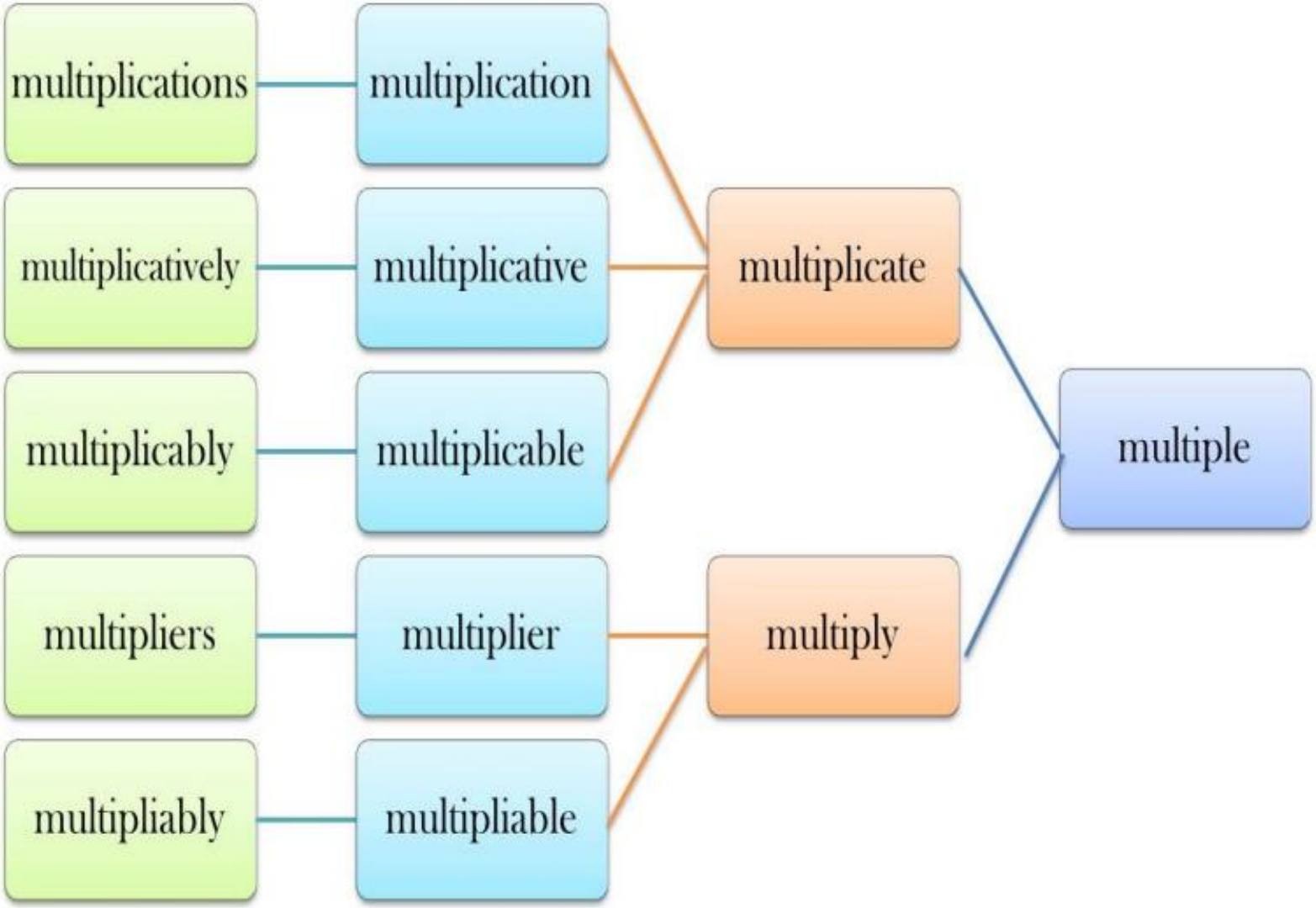
- *He is reading detective stories*

→ *He be read detective story*

Stemming and lemmatization

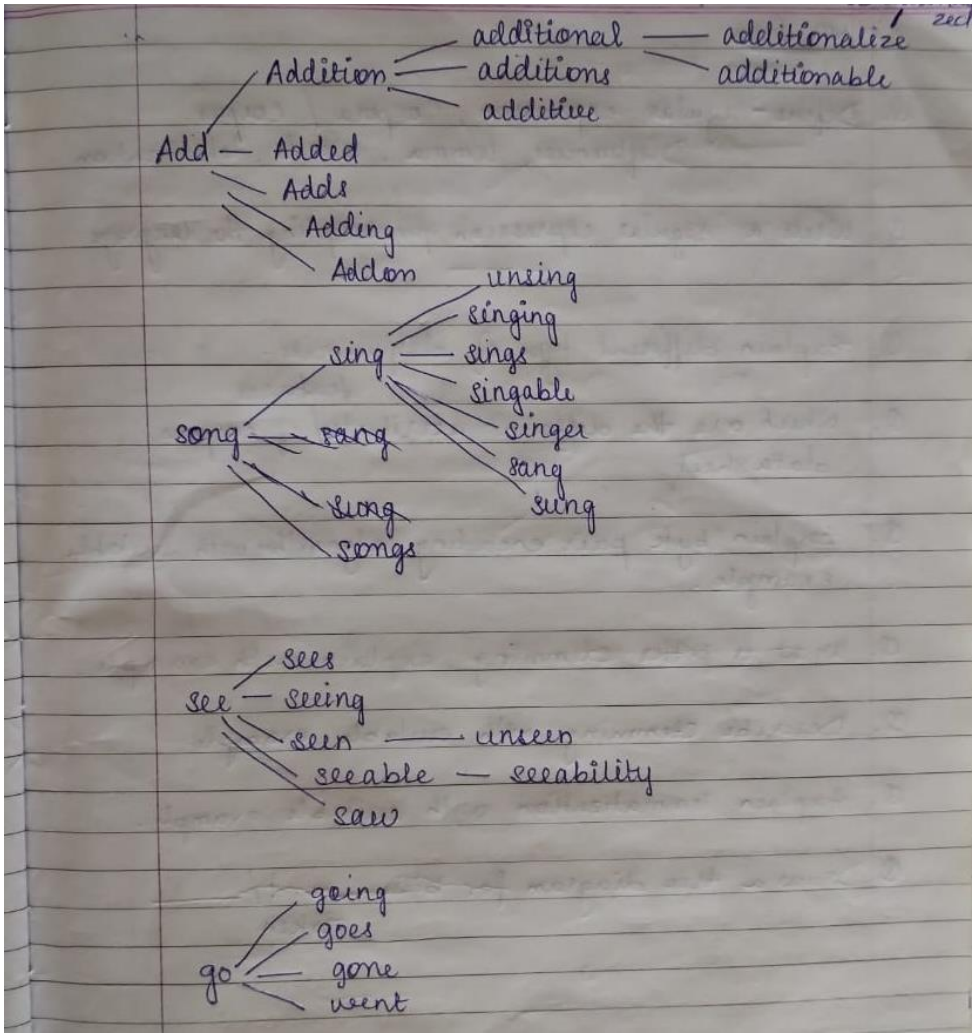
Sr. No.	Actual Word	Lemmatization	Stemming
1	Discribing	Describe	Discribe
2	Discribes	Describe	Discribe
3	Discribed	Describe	Discribe

Lemmatization



Lemmatization

- Prepare tree structure for lemmatizing of root word
- Add
- Song
- See
- Go



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*
  - Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

Sentence Segmentation

!, ? mostly unambiguous but **period** "." is very ambiguous

- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3

Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.

- An abbreviation dictionary can help

Sentence segmentation can then often be done by rules based on this tokenization.

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one into the other

Minimum Edit Distance

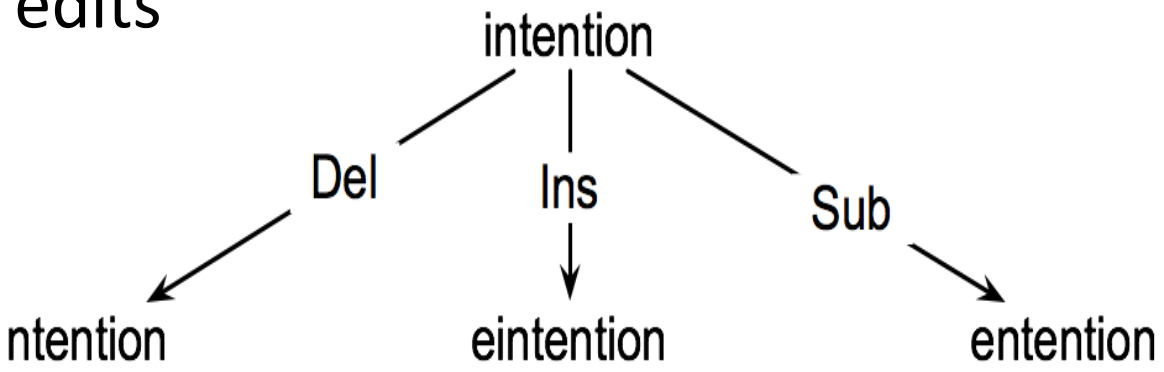
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

d – deletion
s – substitution
i - insertion

- If each operation has cost of 1
  - Distance between these is 5

How to find the Min Edit Distance?

- Searching for a path (sequence of edits) from the start string to the final string:
  - **Initial state**: the word we're transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we're trying to get to
  - **Path cost**: what we want to minimize: the number of edits



Defining Min Edit Distance

- For two strings
  - X of length *n*
  - Y of length *m*
- We define D(*i,j*)
  - the edit distance between X[1..*i*] and Y[1..*j*]
    - i.e., the first *i* characters of X and the first *j* characters of Y
  - The edit distance between X and Y is thus D(*n,m*)

References

1. Dan Jurafsky and James Martin, “Speech and language Processing”, 3rd Edition, Pearson Education, 2023.

2. NLP Introduction. URL = < *https://www.geeksforgeeks.org/introduction-to-natural-language-processing/* > , *Accessed on: 20/07/2023*

3. NLP Tasks. URL = < https://www.ibm.com/topics/natural-language-processing > *Accessed on: 24/07/2023*

4. Text Normalization. URL = < https://subscription.packtpub.com/book/data/9781800200937/1/ch01lvl1sec04/text-normalization > *Accessed on: 24/07/2023*

5. Text Normalization. URL = < https://www.scaler.com/topics/nlp/what-is-text-normalization-in-nlp/ > *Accessed on: 24/07/2023*

End of Unit I

CO Covered:

1. Students able to use basic concepts of Natural Language Processing