Natural Language Processing

INT404

Natural Language Processing

- Language is meant for Communicating about the world.
- By studying language, we can come to understand more about the world.
- If we can succeed at building computational mode of language, we will have a powerful tool for communicating about the world.
- We look at how we can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems.
- NLP problem can be divided into two tasks:

- **Processing written text**, using lexical, syntactic and semantic knowledge of the language as well as the required real world information.

- **Processing spoken language**, using all the information needed above plus additional knowledge about phonology as well as enough added information to handle the further ambiguities that arise in speech.

- The Problem : English sentences are incomplete descriptions of the information that they are intended to convey.

- Some dogs are outside is incomplete – it can mean
  - Some dogs are on the lawn.
  - Three dogs are on the lawn.
  - Moti, Hira & Buzo are on the lawn.

- The good side : Language allows speakers to be as vague or as precise as they like. It also allows speakers to leave out things that the hearers already know.

- The Problem : The same expression means different things in different context.
  - Where's the water? (In a lab, it must be pure)
  - Where's the water? (When you are thirsty, it must be potable or drinkable)
  - Where's the water? (Dealing with a leaky roof, it can be filthy)
- The good side : Language lets us communicate about an infinite world using a finite number of symbols.
- The problem : There are lots of ways to say the same thing :
  - Mary was born on October 11.
  - Mary's birthday is October 11.
- The good side : When you know a lot, facts imply each other. Language is intended to be used by agents who know a lot.

Components of NLP

There are the following two components of NLP -

1. Natural Language Understanding (NLU)

- Natural Language Understanding (NLU) helps the machine to understand and analyze human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.
- NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks -

- It is used to map the given input into useful representation.
- It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

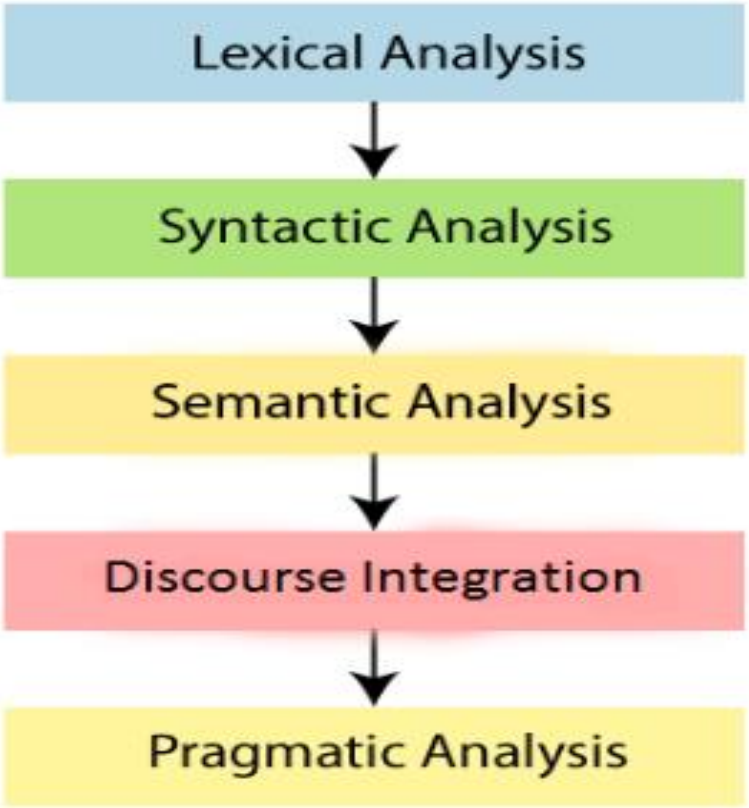
- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation.
- It mainly involves Text planning, Sentence planning, and Text Realization.

Difference between NLU and NLG

NLU	NLG
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

Phases of NLP

There are the following five phases of NLP:



1. Lexical Analysis and Morphological

- The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes.
- It divides the whole text into paragraphs, sentences, and words.

2. Syntactic Analysis (Parsing)

- Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

- In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

- Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

4. Discourse Integration

- Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

- Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example: "Open the door" is interpreted as a request instead of an order.

Why NLP is difficult?

NLP is difficult because Ambiguity and Uncertainty exist in the language.

Ambiguity

There are the following three ambiguity -

- **Lexical Ambiguity**

Lexical Ambiguity exists in the presence of two or more possible meanings of the sentence within a single word.

Example:

- Manya is looking for a **match**.
- In the above example, the word match refers to that either Manya is looking for a partner or Manya is looking for a match. (Cricket or other match)

Syntactic Ambiguity

- Syntactic Ambiguity exists in the presence of two or more possible meanings within the sentence.

Example:

- I saw the girl with the binocular.
- In the above example, did I have the binoculars? Or did the girl have the binoculars?

Referential Ambiguity

- Referential Ambiguity exists when you are referring to something using the pronoun.

Example: Kiran went to Sunita. She said, "I am hungry."

- In the above sentence, you do not know that who is hungry, either Kiran or Sunita.

Parse Tree

A parser is a program, that accepts as input a sequence of words in a natural language and breaks them up into parts (nouns, verbs, and their attributes), to be managed by other programming.

- Parsing can be defined as the act of analyzing the grammaticality an utterance according to some specific grammar.
- Parsing is the process to check, that a particular sequence of words in a sentence correspond to a language defined by its grammar.
- Parsing means show how we can get from the start symbol of the grammar to the sequence of words using the production rules.
- The output of a parser is *a Parse tree*.

Parse Tree is a way of representing the output of a parser.

- Each phrasal constituent found during parsing becomes a branch node of the parse tree;
- the words of the sentence become the leaves of the parse tree;
- there can be more than one parse tree for a single sentence;

Parsing

To parse a sentence, it is necessary to find a way in which the sentence could have been generated from the start symbol. There two ways to do : One, Top-Down Parsing and the other, Bottom–UP Parsing.

- **Top-Down Parsing**

  Begin with the start symbol and apply the grammar rules forward until the symbols at the terminals of the tree corresponds to the components of the sentence being parsed.

- **Bottom–UP Parsing**

  Begin with the sentence to be parsed and apply the grammar rules backward until a single tree whose terminals are the wards of the sentence and whose top node is the start symbol has been produced.

Modeling a Sentence using Phase Structure

Every sentence consists of an internal structure which could be modeled with the phrase structure.

Algorithm : Steps

- Apply rules on an proposition
- The base proposition would be :

  S (the root, ie the sentence).
- The first production rule would be :

  (NP = noun phrase, VP = verb phrase)

  S -> (NP, VP)
- Apply rules for the 'branches'

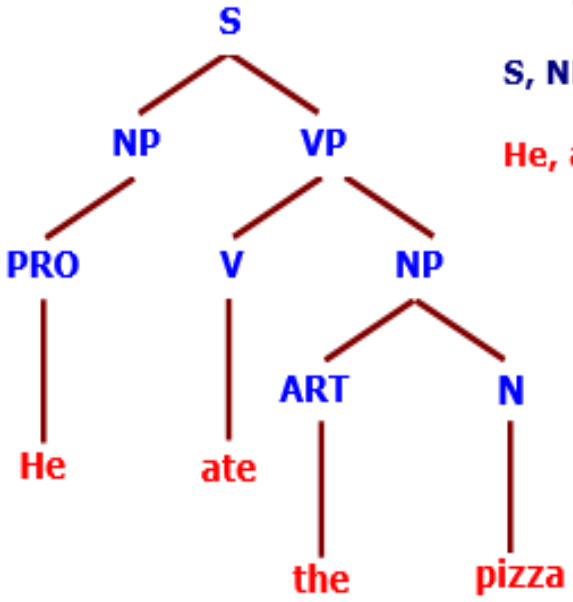
  NP -> noun VP -> verb, NP
- The verb and noun have terminal nodes which could be any word in the lexicon for the appropriate category.
- The end is a tree with the words as terminal nodes, which is referred as the sentence.

Example : Parse tree

- sentence "He ate the pizza",
- apply the grammar with rules

  S -> NP VP, NP -> PRO, NP -> ART N, VP -> V NP,
- the lexicon structure is

  ("ate" V) ("he" PRO) ("pizza" N) ("the" ART)
- The parse tree is



PRO, V, ART, N - lexical non-terminals

S, NP, VP - phrasal non-terminal

He, ate, the, pizza - words or terminal

Tokenization

Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.

Why is Tokenization required in NLP?

- Before processing a natural language, we need to identify the *words* that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). **This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.**
- Let's take an example. Consider the below string:

"This is a cat."

What do you think will happen after we perform tokenization on this string?

['This', 'is', 'a', cat'].

- There are numerous uses of doing this. We can use this tokenized form to:
- Count the number of words in the text
- Count the frequency of the word, that is, the number of times a particular word is present

Bag of Words Model

Bag of Words (BoW) **model** is a simple algorithm used in **Natural Language Processing**. In BoW **model** a sentence or a document is considered as a '**Bag**' containing **words**. It will take into account the **words** and their frequency of occurrence in the sentence or the document disregarding semantic relationship in the sentences.

Example of the Bag-of-Words Model

Step 1: Collect Data

- Below is a snippet of the first few lines of text from the book "A Tale of Two Cities" by Charles Dickens, taken from Project Gutenberg.
- *It was the best of times,*
  it was the worst of times,
  it was the age of wisdom,
  it was the age of foolishness,
- For this small example, let's treat each line as a separate "document" and the 4 lines as our entire corpus of documents.

Step 2: Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary.

- The unique words here (ignoring case and punctuation) are:

“it”
“was”
“the”
“best”
“of”
“times”
“worst”
“age”
“wisdom”
“foolishness”

- That is a vocabulary of 10 words from a corpus containing 24 words.

Step 3: Create Document Vectors

- The next step is to score the words in each document.
- The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model.
- Because we know the vocabulary has 10 words, we can use a fixed-length document representation of 10, with one position in the vector to score each word.
- The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.
- Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document ("*It was the best of times*") and convert it into a binary vector.

- The scoring of the document would look as follows:

“it” = 1

“was” = 1

“the” = 1

“best” = 1

“of” = 1

“times” = 1

“worst” = 0

“age” = 0

“wisdom” = 0

“foolishness” = 0

As a binary vector, this would look as follows:

```
1 [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

The other three documents would look as follows:

```
1 "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
2 "it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
3 "it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
```

- All ordering of the words is nominally discarded and we have a consistent way of extracting features from any document in our corpus, ready for use in modeling.
- New documents that overlap with the vocabulary of known words, but may contain words outside of the vocabulary, can still be encoded, where only the occurrence of known words are scored and unknown words are ignored.
- You can see how this might naturally scale to large vocabularies and larger documents.

Managing Vocabulary

- As the vocabulary size increases, so does the vector representation of documents.
- In the previous example, the length of the document vector is equal to the number of known words.

There are simple text cleaning techniques that can be used as a first step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don’t contain much information, called stop words, like “a,” “of,” etc.

Fixing misspelled words.

- Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.
- A more sophisticated approach is to create a vocabulary of grouped words. This both changes the scope of the vocabulary and allows the bag-of-words to capture a little bit more meaning from the document.
- In this approach, each word or token is called a “gram”. Creating a vocabulary of two-word pairs is, in turn, called a bigram model. Again, only the bigrams that appear in the corpus are modeled, not all possible bigrams.

Scoring Words

- Once a vocabulary has been chosen, the occurrence of words in example documents needs to be scored.

Some additional simple scoring methods include:

- **Counts**. Count the number of times each word appears in a document.
- **Frequencies**. Calculate the frequency that each word appears in a document out of all the words in the document.

- A **Spell checker** is one of the basic tools required for language processing.
- Spell checking involves identifying words and non words and also suggesting the possible alternatives for its correction.

- used in
  - Word processing
  - Character or text recognition
  - Speech recognition and generation.

- Most available spell checkers focus on processing isolated words and do not take into account the context.

- “Henry **sar** on the box”
- “Henry **at** on the box”

Spelling Errors

- Three cause of error are:
- **Insertion:** Insertion of extra letter while typing. E.g. maximum typed as maxiimum.
- **Deletion**: A case of a letter missing or not typed in a word. E.g. netwrk instead of network.
- **Substitution:** Typing of a letter in place of the correct one. E.g. intellugence.

- Spelling errors may be classified into following types:
- Typographic errors:
  - Cause due to mistakes committed while typing.
  - E.g. netwrk in place of network.
- Orthographic errors:
  - Result due to a lack of comprehension of the concerned language on part of user.
  - E.g. arithmetic, wellcome, accomodation.
- Phonetic errors:
  - result due to poor cognition on part of listner.
  - E.g. the word rough could be spelt as ruff.
  - Listen as lisen, piece as peace or peas, reed as read.

Spell checking techniques

- Spell checking techniques can be broadly classified into three categories-
- (a) Non-Word error detection: This process involves the detection of misspelled words or non-words.
  - E.g. the word sopar is a non-word ; its correct form is super or sober.
  - The most commonly used techniques to detect such errors are the **N-gram analysis** and **Dictionary look-up.**

N-Gram Analysis:

- Make use of probabilities of occurrence of N-grams in a large corpus of text to decide on the error in the word.
- N-gram to be sequence of letters rather than words.
- Try to predict next letter rather than next words.
- Used in text (handwritten or printed) recognition

- Dictionary look-up involves the use of an efficient dictionary lookup coupled with pattern-matching algorithm (such as hashing technique, Finite state automata), dictionary portioning schemes and morphological processing methods.

(b) Isolated-word error correction:

- Focus on the correction of an isolated non-words by finding its nearest and meaningful word and make an attempt to rectify the error.
- It thus transform the word “soper” into super.
- Isolated word correction may be looked upon as a combination of three sub-problems-Error detection, candidate (correct word) generation, and ranking of correct candidates.

Minimum Edit distance technique:

- Wanger[1974] define the minimum edit distance between the misspelled word and the possible correct candidate as the minimum number of edit operations needed to transform the misspelled word to the correct candidate.
- Edit operation-insertion, deletion, and substitution of a single character.
- The minimum number of such operations required to affect the transformation is commonly known as *Levenshtein distance.*

(c) Context dependent error detection and correction:

- In addition to detect errors, try to find whether the corrected word fits in to context of the sentence.
- More complex to implement.
- “Peace comes from within”, “Piece comes from within” ; first word in both sentence is a correct word.
- This involves correction of real-word errors or those that result in another valid error.

Soundex Algorithm

- Can be effectively used as a simple phonetic based spell checkers.
- Algorithm:
  - Remove all punctuation marks and capitalize the letters in the word.
  - Retain the first letter of the word.
  - Remove nay occurrence of letters- A,E,I,O,U,H,W,Y apart from very first letter.
  - Replace the letter other than first as shown in table.

Letters	Substitute with integer
B, F, P, V	1
C, G, J, K, S, X, Z ,Q	2
D, T	3
L	4
M, N	5
R	6

–If two or more adjacent letters , non separated by vowels, have the same numeric value retain only one of them.

–Return the 1st four character; pad with zeros if there are less than four.

Soundex code for some words:

Word	Soundex code
Grate, Great	G360
Network, network	N362
Henry, Henary	H560
Torn	T650
Worn	W650
Horn	H650

Used to measure similarity of two words