**COMP 6751 Natural Language Analysis**

**Project Report 1**

Student name: Md Sakib Ullah Sourav

ID: 40264066

*Expectations of originality:*

*I, Md Sakib Ullah Sourav (student id 40264066), certify that this submission is my original work and meets the Faculty’s Expectations of Originality.*

*Date: September 22, 2023*

Contents

[1. Project Goal 3](#_Toc146212594)

[2. Modules of the Project 4](#_Toc146212595)

[2.1 Tokenizer 4](#_Toc146212596)

[2.2 Sentence Splitting (SS) 5](#_Toc146212597)

[2.3 POS Tagging 5](#_Toc146212598)

[2.4 Gazetteer 5](#_Toc146212599)

[2.5 Named Entity Recognition (NER) 6](#_Toc146212600)

[2.6 Measured Entity Detection (MET) 6](#_Toc146212601)

[3. Limitations 7](#_Toc146212602)

# Project Goal

In this project, we will explore various modules and functionalities of the Natural Language Toolkit (NLTK) to delve into the exciting world of natural language processing. The reference data that we work on will be NLTK version of Reuter's text training/267. Our journey will encompass the following key components:

Tokenizer: The tokenizer module allows us to break down text into individual words or tokens, which serves as the fundamental building block for most NLP tasks.

Sentence Splitting: Sentence splitting enables us to segment text into distinct sentences, a critical step for many NLP applications, including text summarization and sentiment analysis.

POS Tagging: Part-of-Speech (POS) tagging involves labeling each word in a sentence with its grammatical category (e.g., noun, verb, adjective). This information is crucial for understanding the structure and meaning of text.

Gazetteer: Gazetteer annotation involves identifying and annotating specific entities, such as country names, currencies, and units, based on predefined lists (gazetteers).

Named Entity Recognition (NER): NER is a more advanced technique that goes beyond simple gazetteer annotation. It involves identifying and classifying named entities in text, such as persons, organizations, locations, dates, and more.

Measured Entity Detection (MED): While NER identifies and classifies entities, measured entity recognition (MED) is a specific task whose primary objective is to locate and extract numeric or quantitative information from unstructured text data in text.

Throughout this project, we will dive into these NLTK modules to perform various tasks, gaining a deeper understanding of how they can be applied to real-world natural language processing challenges.

# Modules of the Project

The given passage in the of Reuter's text training/267 is as below-

INDONESIA UNLIKELY TO IMPORT PHILIPPINES COPRA

Indonesia is unlikely to import copra from the Philippines in 1987 after importing 30,000 tonnes in 1986, the U.S. Embassy's annual agriculture report said. The report said the 31 pct devaluation of the Indonesian rupiah, an increase in import duties on copra and increases in the price of Philippines copra have reduced the margin between prices in the two countries. Indonesia's copra production is forecast at 1.32 mln tonnes in calendar 1987, up from 1.30 mln tonnes in 1986.

## **Tokenizer**

Tokenizers break down raw text into smaller units, such as words or phrases, making text analysis more manageable.

At first, we introduced tokenizer in NLTK to our IDE to get the small chunks from our given text.

tokens = nltk.word\_tokenize(document\_content)

However, it could not tokenize some words properly. One of the primary challenges is handling languages with complex grammatical rules and word formations. Tokenizers must also address punctuation marks, contractions, and hyphenated words correctly.

To encounter the challenges, we implemented below regular expression tokenizer

|  |  |
| --- | --- |
| **Operator** | **Explanation** |
| (?:) | Non-capturing group |
| (?:[A-Z]\.)+ | abbreviations, e.g. U.S.A. |
| |\d+(?:,\d{3})\*(?:\.\d+)? | Numbers with optional commas and decimals |
| \w+(?:-\w+)\* | words with optional internal hyphens and apostrophes |
| |[.,;!?()] | Common punctuation |
| \$?\d+(?:\.\d+)?%? | currency and percentages |
| |(?:[...]) | ellipsis … |
| |[“”‘’"'] | Quotation marks |
| |[\[\]{}:<>] | Brackets and colons |
| |(?:\S) | Any other character (non-space) |
| |[—–-] | Dashes and hyphens |
| |[/] | Slashes |

## **Sentence Splitting (SS)**

One major challenge in SS is dealing with ambiguous sentence-ending punctuation marks, such as periods within abbreviations or ellipses, which may be mistaken for sentence boundaries. Sentences can also span multiple lines or paragraphs, making it difficult to determine where a sentence ends.

sentences = nltk.sent\_tokenize(document\_content)

nltk.sent\_tokenize returns a sentence-tokenized copy of the given passage

Here in this script, we used a function called “enumerate” which is a built-in Python function that allows you to iterate over an iteral (e.g., a list or string) while keeping track of the current index or position of each item.

## **POS Tagging**

First of all, we import the needed functions along with the below one which uses the Penn Treebank POS tags.

pos\_tags = pos\_tag(words)

Eventually, this function typically returns a list of tuples those represent the grammatical roles of words, such as nouns (NN), verbs (VB), adjectives (JJ), and others.

## **Gazetteer**

Gazetteer annotation involves annotating specific named entities or terms, such as country names, currencies, and units of measurement, in a text.

# Define gazetteers

countries = ["INDONESIA", "PHILIPPINES"]

currencies = ["rupiah", "pct"]

units = ["tonnes", "mln"]

# Create regular expression patterns for each category

country\_pattern = r'\b(?:' + '|'.join(re.escape(country) for country in countries) + r')\b'

currency\_pattern = r'\b(?:' + '|'.join(re.escape(currency) for currency in currencies) + r')\b'

unit\_pattern = r'\b(?:' + '|'.join(re.escape(unit) for unit in units) + r')\b'

# Define chunking rules using regular expressions

chunk\_grammar = r"""

    COUNTRY: {<NNP>{1,}}

    CURRENCY: {<NN><NN><NN><NN>?}

    UNIT: {<NN><NN><NN><NN>?}

"""

# Create a chunk parser with the defined grammar

chunk\_parser = nltk.RegexpParser(chunk\_grammar)

# Tag the words with part-of-speech tags

pos\_tags = nltk.pos\_tag(words)

# Parse the tagged words using the chunk parser

chunks = chunk\_parser.parse(pos\_tags)

We had to create regular expressions in code to specify country, currency and unit. Then parsing the words using chunk parser. Interruption in data, contextual identification and domain adaptation is important to get the desired output in terms of gazetteer annotation using NLTK.

## **Named Entity Recognition (NER)**

Identifying named entities can be challenging when a word can have multiple interpretations. For example, "Washington" could refer to a city or a person's name. Here we adopt the below command to perform the NER on our given passage.

ner\_tags = ne\_chunk(nltk.pos\_tag(words))

The main four categories in NER are person (PER), organization (ORG), Geo-political entity (GPE), and GSP which refers a specific place or entity.

## **Measured Entity Detection (MET)**

By using regular expression and domain specific tagging, such as list of words and symbols, MET can be effectively extracted. Here we used the below expressions to get the best results:

(\d{1,3}(,\d{3})\*(\.\d+)?\s\*(mln|pct|tonnes)

I printed the value and corresponding units in the end to show the METs.

# Limitations

1. In the NER, the output of the word “U. S. Embassy” should be Organization but our model splits it into two different entity

Entity: U.S., Label: GPE

Entity: Embassy, Label: ORGANIZATION