COMP 6751 Natural Language Analysis

Project Report 1 (Demo)

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

Table of Contents

1. Inputs and Outputs

(To access to my entire code, please visit this link-
https://colab.research.google.com/drive/1obYRxvJcsOsN87-FLOJPXx_p7_su8Ion?usp=sharing

And, let me know if the link is not working!)

First of all, we have to import necessary libraries, packages and corpus "reuters" by he below codes:

```
import pandas
import nltk
nltk.download('all') #run this line during the first time

# nltk.download()

# from nltk.book import
from nltk.corpus import reuters


# List the file IDs available in the Reuters corpus
file_ids = reuters.fileids()

# print(file_ids)
# Access and print the content of a specific document (e.g., 'test/14826')
document_id = 'training/267'
document_content = reuters.raw(document_id)
print(document_content)
```

Then we will get the below **output** in return-

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.

1.1 Tokenizer

The input of the codes for tokenizer part is-

```
from nltk import word_tokenize
# Tokenize the text
tokens = nltk.word_tokenize(document_content)

# Print the tokens
```

```
print(tokens)
from nltk import word_tokenize
# Tokenize the text
tokens = nltk.word_tokenize(document_content)

# Print the tokens
print(tokens)
```

The corresponding output-

['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', '.']

But here, some words are not properly tokenized.

1.1.1 Modification

So I modified the part like below with regular expression-

```
from nltk.tokenize import RegexpTokenizer

# Define a regular expression pattern
pattern = r'''(?x)          # Set flag to allow verbose regex
    (?:[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
    |(?:[...])              # Ellipsis
    |[“”‘’"']               # Quotation marks
    |[\[\]{}:<>]            # Brackets and colons
    |[——-]                  # Dashes and hyphens
    |[/]                    # Slashes
    |(?:\S)                 # Any other character (non-space)
    \$?\d+(?:\.\d+)?%?      # CURRENCY AND PERCENTAGE
'''

# Create the tokenizer using the defined pattern
tokenizer = RegexpTokenizer(pattern)
```

```
# Tokenize the text using the tokenizer
tokens = tokenizer.tokenize(document_content)

# Print the tokens
print(tokens)
```

Eventually we get the below output-

['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', '.']

1.2 Sentence Splitting (SS)

The input code for SS is-

```
# Use NLTK's sent_tokenize to split the passage into sentences
from nltk import sent_tokenize
sentences = sent_tokenize(document_content)

# Print the sentences
for i, sentence in enumerate(sentences):
    print(f"Sentence {i + 1}: {sentence}")
```

And the corresponding output we get

```
Sentence 1: 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.
Sentence 2: 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.
Sentence 3: Indonesia's copra production is forecast at 1.32 mln tonnes
  in calendar 1987, up from 1.30 mln tonnes in 1986.
```

1.2.1 Interesting Approaches

The SS done above could not separate the header phrase from the first sentence. So, I did further go for two approaches where one approach was **satisfactory** while another one was not as it removed the first line along with the header phrase.

Approach 1:

Input:

```
remaining_sentences = sentences[1: ]

# Now you can work with the remaining sentences as needed
for sentence in remaining_sentences:
    print(sentences)
```

Output:

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

Approach 2:

Input:

```
# Define the header phrase to exclude (assuming it's the first sentence)
header_phrase = sentences[0]

# Split the paragraph into sentences excluding the header phrase
paragraph_sentences = [sentence for sentence in sentences if sentence !=
header_phrase]

# Print the remaining sentences
for sentence in paragraph_sentences:
    print(sentence.strip())
```

Output:

```
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.
```

1.3 POS Tagging

The input of POS tagging code is-

```
from nltk import pos_tag

# Tokenize the passage into words
words = word_tokenize(document_content)

# Perform POS tagging
pos_tags = pos_tag(words)

# Print the POS tags
for word, pos_tag in pos_tags:
    print(f"Word: {word}, POS Tag: {pos_tag}")
```

The corresponding output is-

```
Word: INDONESIA, POS Tag: NNP
Word: UNLIKELY, POS Tag: NNP
Word: TO, POS Tag: NNP
Word: IMPORT, POS Tag: NNP
Word: PHILIPPINES, POS Tag: NNP
Word: COPRA, POS Tag: NNP
Word: Indonesia, POS Tag: NNP
Word: is, POS Tag: VBZ
Word: unlikely, POS Tag: JJ
Word: to, POS Tag: TO
Word: import, POS Tag: VB
Word: copra, POS Tag: NN
Word: from, POS Tag: IN
Word: the, POS Tag: DT
Word: Philippines, POS Tag: NNPS
Word: in, POS Tag: IN
Word: 1987, POS Tag: CD
Word: after, POS Tag: IN
Word: importing, POS Tag: VBG
Word: 30,000, POS Tag: CD
Word: tonnes, POS Tag: NNS
Word: in, POS Tag: IN
Word: 1986, POS Tag: CD
Word: ,, POS Tag: ,
Word: the, POS Tag: DT
Word: U.S., POS Tag: NNP
Word: Embassy, POS Tag: NNP
Word: 's, POS Tag: POS
Word: annual, POS Tag: JJ
Word: agriculture, POS Tag: NN
Word: report, POS Tag: NN
Word: said, POS Tag: VBD
```

Word: ., POS Tag: .
Word: The, POS Tag: DT
Word: report, POS Tag: NN
Word: said, POS Tag: VBD
Word: the, POS Tag: DT
Word: 31, POS Tag: CD
Word: pct, POS Tag: JJ
Word: devaluation, POS Tag: NN
Word: of, POS Tag: IN
Word: the, POS Tag: DT
Word: Indonesian, POS Tag: NNP
Word: rupiah, POS Tag: NN
Word: ,, POS Tag: ,
Word: an, POS Tag: DT
Word: increase, POS Tag: NN
Word: in, POS Tag: IN
Word: import, POS Tag: JJ
Word: duties, POS Tag: NNS
Word: on, POS Tag: IN
Word: copra, POS Tag: NN
Word: and, POS Tag: CC
Word: increases, POS Tag: NNS
Word: in, POS Tag: IN
Word: the, POS Tag: DT
Word: price, POS Tag: NN
Word: of, POS Tag: IN
Word: Philippines, POS Tag: NNPS
Word: copra, POS Tag: NNS
Word: have, POS Tag: VBP
Word: reduced, POS Tag: VBN
Word: the, POS Tag: DT
Word: margin, POS Tag: NN
Word: between, POS Tag: IN
Word: prices, POS Tag: NNS
Word: in, POS Tag: IN
Word: the, POS Tag: DT
Word: two, POS Tag: CD
Word: countries, POS Tag: NNS
Word: ., POS Tag: .
Word: Indonesia, POS Tag: NNP
Word: 's, POS Tag: POS
Word: copra, POS Tag: NN
Word: production, POS Tag: NN
Word: is, POS Tag: VBZ
Word: forecast, POS Tag: VBN
Word: at, POS Tag: IN
Word: 1.32, POS Tag: CD
Word: mln, POS Tag: NN
Word: tonnes, POS Tag: NNS
Word: in, POS Tag: IN
Word: calendar, POS Tag: NN
Word: 1987, POS Tag: CD
Word: ,, POS Tag: ,
Word: up, POS Tag: RB

```
Word: from, POS Tag: IN
Word: 1.30, POS Tag: CD
Word: mln, POS Tag: NN
Word: tonnes, POS Tag: NNS
Word: in, POS Tag: IN
Word: 1986, POS Tag: CD
Word: ., POS Tag: .
```

1.4 Gazetteer

The input part of this module is as below-

```
import re


# Tokenize the text into words
words = nltk.word_tokenize(document_content)

# 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)

# Function to annotate chunks
def annotate_chunks(tree, pattern, annotation):
    for subtree in tree.subtrees():
        if re.search(pattern, subtree.leaves()[0][0], re.IGNORECASE):
            subtree.set_label(annotation)

# Annotate chunks with gazetteer information
annotate_chunks(chunks, country_pattern, "COUNTRY")
annotate_chunks(chunks, currency_pattern, "CURRENCY")
annotate_chunks(chunks, unit_pattern, "UNIT")

# Print the annotated chunks
for subtree in chunks.subtrees():
    if subtree.label():
        print(subtree)
```

And the corresponding output is-

```
(COUNTRY
  (COUNTRY
    INDONESIA/NNP
    UNLIKELY/NNP
    TO/NNP
    IMPORT/NNP
    PHILIPPINES/NNP
    COPRA/NNP
    Indonesia/NNP)
  is/VBZ
  unlikely/JJ
  to/TO
  import/VB
  copra/NN
  from/IN
  the/DT
  Philippines/NNPS
  in/IN
  1987/CD
  after/IN
  importing/VBG
  30,000/CD
  tonnes/NNS
  in/IN
  1986/CD
  ,/,
  the/DT
  (COUNTRY U.S./NNP Embassy/NNP)
  's/POS
  annual/JJ
```

agriculture/NN
report/NN
said/VBD
./.
The/DT
report/NN
said/VBD
the/DT
31/CD
pct/JJ
devaluation/NN
of/IN
the/DT
(COUNTRY Indonesian/NNP)
rupiah/NN
,/,
an/DT
increase/NN
in/IN
import/JJ
duties/NNS
on/IN
copra/NN
and/CC
increases/NNS
in/IN
the/DT
price/NN
of/IN
Philippines/NNPS
copra/NNS
have/VBP
reduced/VBN
the/DT
margin/NN
between/IN
prices/NNS
in/IN
the/DT
two/CD
countries/NNS
./.
(COUNTRY Indonesia/NNP)
's/POS
copra/NN
production/NN
is/VBZ
forecast/VBN
at/IN
1.32/CD
mln/NN
tonnes/NNS
in/IN
calendar/NN

```
  1987/CD
  ,/,
  up/RB
  from/IN
  1.30/CD
  mln/NN
  tonnes/NNS
  in/IN
  1986/CD
  ./.)
(COUNTRY
  INDONESIA/NNP
  UNLIKELY/NNP
  TO/NNP
  IMPORT/NNP
  PHILIPPINES/NNP
  COPRA/NNP
  Indonesia/NNP)
(COUNTRY U.S./NNP Embassy/NNP)
(COUNTRY Indonesian/NNP)
(COUNTRY Indonesia/NNP)
```

1.5 Named Entity Recognition (NER)

The input coding of this part is

```
from nltk import ne_chunk


# Tokenize the text into words
words = word_tokenize(document_content)

# Perform NER
ner_tags = ne_chunk(nltk.pos_tag(words))

# Print the named entities
for subtree in ner_tags:
    if isinstance(subtree, nltk.Tree):
        entity = " ".join([word for word, tag in subtree.leaves()])
        label = subtree.label()
        print(f"Entity: {entity}, Label: {label}")
```

And the output is

```
Entity: INDONESIA, Label: GPE
Entity: UNLIKELY, Label: ORGANIZATION
```

```
Entity: IMPORT, Label: ORGANIZATION
Entity: Indonesia, Label: GPE
Entity: Philippines, Label: ORGANIZATION
Entity: U.S., Label: GPE
Entity: Embassy, Label: ORGANIZATION
Entity: Indonesian, Label: GPE
Entity: Philippines, Label: GSP
Entity: Indonesia, Label: GPE
```

1.6 Measured Entity Detection (MET)

The input coding part for MET is

```
import re

# Use regular expressions to find measurements with commas and group them
measurements =
re.findall(r'(\d{1,3}(,\d{3})*(\.\d+)?\s*(mln|pct|tonnes))',
document_content)

# Print the measurements found
for measurement in measurements:
    value, _, _, unit = measurement
    print(f"Value: {value}, Unit: {unit}")
```

And output is

```
Value: 30,000 tonnes, Unit: tonnes
Value: 31 pct, Unit: pct
Value: 1.32 mln, Unit: mln
Value: 1.30 mln, Unit: mln
```

2. Discussion

Doing this project was a happy learning journey. I would like to express my sincere gratitude to the course instructor Dr. Sabine Bergler. POD sessions were extremely helpful. And the coursemates were so kind while exchanging the ideas and understanding. Look forward to learn more through the next projects and lectures coming ahead in this course.