COMP 6751 Natural Language Analysis

Project 1 Report 1

Student- Amrita Awasthi (40203422)

Table of Contents-

1. Introduction……………………………………………………………………………….2
2. Tokenization……………………………………………………………………………….2
3. Sentence Splitting………………………………………………………………………….2
4. POS Tagging……………………………………………………………………………….2
5. Gazetteer Annotation……………………………………………………………………...2
6. Named Entity Recognition………………………………………………………………..3
7. Measured Entity Recognition……………………………………………………………..4
8. Challenges………………………………………………………………………………...5
References…………………………………………………………………………………….5

1. Introduction-

The below report focuses on the work done on project 1 using NLTK that covers tokenisation, sentence splitting, POS tagging, gazetteer annotation (country, currency, units etc.), named entity recognition (NER), measured entity recognition.

2. Tokenisation-

Tokenization is a crucial step in natural language processing, which involves dividing paragraphs and sentences into smaller units to facilitate the assignment of meaning. The initial stage of the NLP process entails collecting data, typically a sentence, and then dissecting it into comprehensible components, namely words. Following is the function for performing tokenization.

```
def Tokenization(text):
    custom_tokenizer = nltk.RegexpTokenizer(r"\w+(?:[-']\w+)*|\d+(?:\.\d+)?|'[^']+'|\"[^\"]+\"|\S")
    tokens = custom_tokenizer.tokenize(text)
    return tokens
```

3. Sentence Splitting-

Sentence splitting, also known as sentence segmentation or sentence boundary detection, is the process of dividing a given text or paragraph into individual sentences. This task is crucial in natural language processing (NLP) and text analysis because it allows NLP systems to work with sentences as discrete units of meaning. Following is the function for performing sentence splitting.

```
def SentenceSplitting(text):
    custom_sent_tokenizer = nltk.tokenize.punkt.PunktSentenceTokenizer()
    sentences = custom_sent_tokenizer.tokenize(text)
    return sentences
```

4. POS Tagging-

Part-of-speech (POS) tagging involves the task of assigning words in a text their appropriate grammatical categories, aiding in the comprehension of a text's structure and meaning within the field of natural language processing (NLP). Following is the function for POS (parts of speech) tagging.

```
def POSTaagging(text):
    tokens = Tokenization(text)
    pos_tags = pos_tag(tokens)
    return pos_tags
```

5. Gazetteer Annotation-

A gazetteer consists of a set of lists containing names of entities such as cities, organizations, days of the week, etc. Following is the function for gazetteer annotation.

```
def GazetteerAnnotation(text):
    tokens = Tokenization(text)

    country_gazetteer = {
        'Canada', 'Denmark', 'England', 'France', 'Germany', 'Hungary', 'Indonesia', 'Philippines',
'JAPAN', 'U.S.', 'U.K.'
    }
    unit_gazetteer = {
        'kg', 'cm', 'm', 'ft', 'g', 'lb', 'tonnes',
    }
    year_gazetteer = {str(year) for year in range(1980, 2024)}

    crop_gazetteer = {
        'wheat', 'corn', 'rice', 'soybean', 'cotton', 'copra', 'Sunflowerseed',
    }

    gazetteer_names = {
        'country': country_gazetteer,
        'unit': unit_gazetteer,
        'year': year_gazetteer,
        'crop': crop_gazetteer
    }

    annotated_tokens = []

    for token in tokens:
        annotation = 'O'
        for gazetteer_name, gazetteer_list in gazetteer_names.items():
            if token in gazetteer_list:
                annotation = gazetteer_name
                break

        annotated_tokens.append((token, annotation))

    return annotated_tokens, gazetteer_names
```

6. Named Entity Recognition (NER)-

Named entity recognition (NER) is an essential technique in natural language processing (NLP) used to extract valuable information from textual data. This method revolves around the identification and classification of significant details within the text, commonly referred to as named entities. Named entities encompass pivotal elements found in the text, such as names of individuals, places, businesses, events, and products. Additionally, NER encompasses the identification of broader categories like concepts, subjects, dates, monetary values, and percentages. NER is often interchangeably called entity extraction, chunking, and identification. Following is the function for performing named entity recognition.

```
def ner_with_gazetteer_entities(text, gazetteers):
    ne_tree = ne_chunk(pos_tag(word_tokenize(text)), binary=True)

    annotated_entities = []

    for subtree in ne_tree:
        if type(subtree) == nltk.Tree:
            entity_name = " ".join([token for token, _ in subtree.leaves()])
            for gazetteer_name, gazetteer_list in gazetteers.items():
                if entity_name in gazetteer_list:
                    annotated_entities.append((entity_name, gazetteer_name))
                    break
        else:
            token, pos_tag_ = subtree
            if token.lower() in [item.lower() for item in gazetteers['year']]:
                annotated_entities.append((token, 'year'))
            elif token.lower() in [item.lower() for item in gazetteers['crop']]:
                annotated_entities.append((token, 'crop'))
            elif token.lower() in [item.lower() for item in gazetteers['country']]:
                annotated_entities.append((token, 'country'))
            elif token.lower() in [item.lower() for item in gazetteers['unit']]:
                annotated_entities.append((token, 'unit'))

    return set(annotated_entities)
```

7. Measured Entity Recognition-

Measured entity recognition is a technique that produces the entities that are measured like 5 kg, 2 million, e.t.c. Following function performs measured entity recognition.

```
def extract_measured_entities(text):
    units = ['cm', 'kg', 'Ecus', 'tonnes', 'pct', 'billion dollars', 'mln tonnes', 'mln stg', 'billion']
    unit_pattern = '|'.join(re.escape(unit) for unit in units)
    pattern = rf'(\d+(?:,\d{{3}})*(?:\.\d+)?\s*(?:{unit_pattern})\b)'

    matches = re.findall(pattern, text, re.IGNORECASE)

    return set(matches)
```

PreProcess Pipeline:-

```
def preprocess(document_id):
    text = reuters.raw(document_id)

    tokens = Tokenization(text)

    sentences = SentenceSplitting(text)

    pos_tags = POSTaagging(text)

    annotated_tokens, gazetteers = GazetteerAnnotation(text)

    ne_tree = ner_with_gazetteer_entities(text, gazetteers)

    ms_entity = extract_measured_entities(text)

    return sentences, tokens, pos_tags, annotated_tokens, ne_tree, ms_entity
```

The following code is the main function for running all the functions-

```
if __name__ == "__main__":
    document_id = 'training/9880'
    sentences, tokens, pos_tags, annotated_tokens, ne_tree, ms_entity = preprocess(document_id)

    print("Sentences:", sentences)
    print("\nTokens:", tokens)
    print("\nPOS Tags:", pos_tags)
    print("\nAnnotated Tokens:", annotated_tokens)
    print("\nNamed Entity Recognition (NER)  :", ne_tree)
    print("\nMeasured Entity Detection:", ms_entity)
```

8. Challenges-

1. In the cases where there are multiple ways to format a data like year(YYYY, YYYY/YY), the possibility of false positive arises.
2. In sentence splitting, if country names abbreviations are given, the mentioned code splits the sentence from country name that is not correct hence, resulting in incorrect result.

References

1. https://www.nltk.org/book/
2. Study material from professor.