MAIN REPORT (NLP)

Multilingual Assistant

University of Zululand
4 CPS 311 Advance Programming Techniques

Table of Contents

Section1

Summary Of Natural Language Processing.

Natural language processing (NLP) is the ability of a computer program to understand human language as it's spoken and written -- referred to as natural language. It's a component of artificial intelligence (AI). There are many definitions of NLP, but they all have one common meaning. This chapter delves into the fascinating world of NLP, equipping you with the skills to unlock the hidden meaning within text data. The text discusses various natural language communications, including text, voice, video, sign language, and braille, in multiple languages.

Now that you understand What is NLP, we can now summarize all the objective in the chapter in section 2 you will be encountering the examples of all the summarized objectives.

In this report we will be using python for Natural language processing, therefore it is mandatory to ensure that python is installed in your machine. Now that you have installed python let's talk about text blob which is an object-orientated NLP text-processing library built on NLTK (natural language tool kit) and pattern libraries. Text Blob simplifies task such as tokenization, part of speech tagging, sentiment analysis, translation, inflection, spelling checking, stemming, lemmatization, word frequencies, WordNet integration, stop word elimination, and n-grams. These capabilities are fundamental for more complex NLP tasks.

You have heard about Text-blob, what follow is for you to install it by executing specific command in your terminal. Also, you should install NLTK corpora in your machine which is used by Text-Blob [$ pip install -U textblob, $ python -m textblob.download_corpora]. The code is included in square brackets so that it will be easy to procced with the other step most of them will be in section 2 of the report now let's simplify and summarize all the concept that you need to understand.

The following are the task that you should consider in NLP.

Text preprocessing: This includes tasks such as tokenization, lowercasing, stop word removal, and stemming to prepare text for further analysis.

Part-of-speech tagging: Assigning a part-of-speech label to each word in a text, such as noun, verb, or adjective.

Named Entity Recognition (NER): Identifying and classifying proper names in a text, such as people, organizations, and locations.

Tokenization: refers to the process of converting input text into smaller units or tokens such as words and sub words.

Stemming: is a technique designed to enhance language understanding by focusing on core of words

lemmatization: is text preprocessing technique used to break a word down to its roots meaning to identify similarities.

Now because you know about the mandatory task in the NLP lets dive deep into NLP. Did you know that with textblob library you can detect the language used in the corpus and translate to a different language using google translator. By using detect_language and translate methods, text can be translated into different languages. Inter-language translation is becoming increasingly important with advancements in machine learning and AI, enabling real-time translation for travellers and communication with people who speak different languages. This will be part of our case study in the project which will be available in section 3 of this document.

Stemming and lemmatization are normalization operations used to prepare words for analysis. Stemming involves removing prefixes or suffixes to obtain a stem, which may or may not be a real word. Lemmatization considers the word's part of speech and meaning to produce a real word. Functions like stem and lemmatize are available for Word and Wordlist objects in TextBlob. These operations are essential for representing different forms of words as their root, enabling consistent analysis of words in text data.

Ensuring text is free of spelling errors is essential for natural language processing tasks. Spelling error detection help avoiding misinterpretations in your writing and ensuring that the intended message is conveyed accurately, they are usually integrated into word processing software and text editing tools. The textblob library provides a spellcheck method that returns possible corrections for misspelled words along with a confidence value.

Did you also know you can also create a dictionary. WordNet, a word database created by Princeton University, is integrated into TextBlob through NLTK's WordNet interface. TextBlob allows users to retrieve word definitions, synonyms, and antonyms from WordNet. The definitions property of a Word returns a list of definitions for that word in the WordNet database. Similarly, synonyms can be obtained using the synsets property, with each Synset representing a group of synonyms. Antonyms can also be retrieved using the antonyms method of a Lemma object. These functionalities enhance semantic analysis and text comprehension by providing additional linguistic context and relationships for words in text data.

Stop words, common words that do not provide useful information, were discussed and a list of English stop words from NLTK was shown. The NLTK library provides lists of stop words for various natural languages, which can be downloaded using the nltk.download('stopwords') function.

Different methods of visualizing word frequencies were discussed, including a bar chart and word cloud. The example showcased how to visualize the top 20 words in 'Romeo and Juliet' excluding stop words using TextBlob, NLTK, and pandas. The process involved loading the text, getting word frequencies, eliminating stop words, and sorting words by frequency for visualization. The chapter also introduced the use of pandas for data visualization capabilities based on Matplotlib.

We're talking about assessing how easy a piece of text is to read. For this, we use a library called Textatistic. This library is pretty cool because it uses well-known formulas like Flesch Reading Ease, Flesch-Kincaid, Gunning Fog, SMOG, and Dale-Chall to determine readability.

To get started with Textatistic, you first need to install it. Once that's done, you can use it to calculate various readability statistics. Let's say we want to analyse the text of Romeo and Juliet. First, we load the text file into a variable using the Pathlib library. Then, we create a Textatistic object with this text. Now comes the fun part! We use the "dict" method from Textatistic to get a dictionary full of interesting stats and readability scores. This includes things like the number of characters, words, sentences, syllables, and more. Each of these values in the dictionary corresponds to a specific property that you can access via the Textatistic property with the same name. So, in a nutshell, Textatistic helps us understand how readable a text is by providing a bunch of useful statistics and scores. Pretty neat, right?

Named Entity Recognition (NER) is a super useful part of Natural Language Processing (NLP). It helps us find and categorize things like dates, times, places, and organizations in a piece of text. To do this with the spaCy library, you first install spaCy, load up the language model, and create a spaCy Doc object that represents your text. Then, you can extract named entities using the ents property of the Doc object. These named entities are represented by Span objects, and they have text and label properties that tell you the type of entity, like DATE, PERSON, or ORG.

Another cool thing you can do with NLP is similarity detection. This is all about analyzing documents to see how alike they are. You might do this by comparing word frequencies or using machine learning techniques. SpaCy has features that let you compare the similarity of Doc objects, which represent different texts. You load the language model, create spaCy Doc objects for your two texts, and use the similarity method to calculate how similar the texts are. For example, you could compare Shakespeare's Romeo and Juliet with Christopher Marlowe's Edward the Second to see how similar their writing styles are.

In this part, we talked about how to analyse documents to see how similar they are. We used Shakespeare’s works like Romeo and Juliet, Hamlet, Macbeth, and King Lear as examples. We showed how to compare these documents for similarity using spaCy code. The similarity values we got showed a high similarity between Romeo and Juliet and the other Shakespearean tragedies.

With a solid understanding of Natural Language Processing (NLP), you’re well-equipped to delve deeper into the world of Artificial Intelligence (AI). NLP is indeed a crucial subfield of AI, and mastering it opens a myriad of possibilities. By combining different aspects of NLP, like Named Entity Recognition and Similarity Detection, you can create powerful models that can revolutionize how we interact with technology.

There are several ongoing projects in the field of Natural Language Processing (NLP) that are quite fascinating. For instance, one project is working on extracting important keywords from text using TF-IDF and Python’s Scikit-Learn library. Another project is focusing on sentiment analysis, which involves analysing text data to determine the sentiment behind it. This can be particularly useful in areas like product reviews. There’s also a project on creating conversational bots or chatbots that can understand and respond to human language, thereby improving user experience on websites and apps. Another interesting project is on topic identification, which involves identifying the main topics in a text document. This can be particularly useful in content recommendation systems. There’s also a project on creating a model that can read a long piece of text and generate a short summary, which can be used in news apps to provide quick summaries of long articles. Another project is working on building a model that can detect and correct grammatical errors in a text, which can be used in word processors and text editors. There’s also a project on building a model that can classify emails or messages as spam or not spam, which can be used in email services to filter out unwanted emails. Lastly, there’s a project on processing and classifying text data into different categories, which can be used in various applications like document classification, sentiment analysis, and more. These projects are industry-ready and real-life situation-based projects using NLP tools and technologies to drive business outcomes. Working on real-world NLP projects is the best way to develop NLP skills and turn user data into practical experiences.

Section2

Now let represent every learnt objective with code which show that we understand. For the proof that its work the program and its output will be displayed.

Let's perform natural language task which are as follows:

Tokenization:

```
import textblob
text = "NewYork is a beautiful city"    #load the text you want to analyze
blob = textblob.TextBlob(text)        #create a TextBlob object from the text
tokens = blob.tokens    #split the text into individual tokens using the tokens attribute
print(tokens)        #print the tokens for tokens

/------------------------------ Output---------------------------------------------------/
['NewYork', 'is', 'a', 'beautiful', 'city']
```

Part of Speech tagging:

```
import textblob
blob = textblob.TextBlob("The cat is sitting on the mat.") # creating a TextBlob object
Tags = blob.tags
print(Tags)    # printing each token with it part of speech.

/------------------------------ Output---------------------------------------------------/
[('The', 'DT'), ('cat', 'NN'), ('is', 'VBZ'), ('sitting', 'VBG'), ('on', 'IN'), ('the', 'DT'), ('mat', 'NN')]
```

Sentiments Analysis:

```
import textblob

# The input text is "I love python programming. It's so fun and exciting!"

blob = textblob.TextBlob("I love python programming. It so fun and exciting!")

print(blob.sentiment)


/------------------------------------------Output------------------------------------------/

Sentiment(polarity=0.39166666666666666, subjectivity=0.5333333333333333)
```

Detect a language and translate:

```
from textblob import TextBlob

from langdetect import detect

from googletrans import Translator

text = 'Today is a beautiful day. Tomorrow looks like bad weather.'

blob = TextBlob(text)

textblob = TextBlob("Today is a beautiful day. Tomorrow looks like bad weather.")

detected_lang = detect(text) #the text will be detected first to find which language is it
in

print(f"Detected language:{detected_lang}")


translator = Translator()

result = translator.translate(text, dest='es') #the text will be translated to spanish

print(result.text)


/----------------------------------Output--------------------------------------------/

Detected language: en

Translated text: Hoy es un hermoso dia. Mañana parece mal tiempo.
```

Stemming and lemmatization:

```
from textblob import Word

word = Word('strawberries') # we use the word 'strawberries'

print(word.stem())

print(word.lemmatize())


/-----------------------------------------Output------------------------------------------/

strawberry       #output for stemming

strawberry       #output for lemmatization
```

Spell checking and Correction capabilities!

```
#  import Textblob

From textblob import Textblob

gfg = TextBlob (“I amm goodd at spelling mistake”.)

#  using TextBlob.correct () method

gfg = gfg . correct ()

print (gfg)

/-----------------------------------------Output------------------------------------------/

I am good at spelling mistake.
```

Get word definitions, synonyms, and antonyms.

```
From textblob import Textblob

blob= textblob.Textblob('The incredible and impressive good girl is having a bad day today'.)

blob. tags

print(tags)

/-----------------------------------------Output------------------------------------------/

[('incredible/impressive';'synonyms');('good/bad';'antonyms')]



from textblob import word

incredible = word('incredible')

incredible.definition

print(incredible)

/-----------------------------------------Output------------------------------------------/

['something is so amazing, astonishing, or extraordinary that it's hard to believe or understand'.]
```

Remove stop words from text.

```
from textblob import TextBlob

from nltk.corpus import stopwords

import nltk

nltk.download('punkt')

nltk.download('stopwords')


blob="Mama, they tell me you were a dancer they tell me you had long beautiful legs to
carry your graceful body they tell me you were a dancer"

text=TextBlob(blob)


tokens=set(text.words)

print("Tokens: ",tokens)


stop=set(stopwords.words("english"))


print("Filtered Tokens: ",tokens-stop)

/----------------------------------------Output-----------------------------------------/

Text:  {'legs', 'to', 'a', 'carry', 'beautiful', 'were', 'Mama', 'long', 'you', 'graceful', 'body', 'tell',
'your', 'had', 'dancer', 'they', 'me'}

Filtered Text:  {'legs', 'carry', 'beautiful', 'graceful', 'Mama', 'long', 'body', 'tell', 'dancer'}
```

Create word clouds.

```
import nltk
from textblob import TextBlob
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
def remove_stopwords(text):
  blob = TextBlob(text)
  stop_words = set(stopwords.words('english'))
  words = [word.lower() for word in blob.words]
  filtered_words = [word for word in words if word not in stop_words]
  filtered_text = ' '.join(filtered_words)
  return filtered_text
def count_word_frequencies(text):
    blob = TextBlob(text)
    word_frequencies = blob.word_counts
    return word_frequencies
paragraph = "Mama, they tell me you were a dancer they tell me you had long beautiful legs to carry your graceful body they tell me you were a dancer Mama, they tell me you sang beautiful solos they tell me you closed your eyes always when the feeling of the songwas right, and lifted your face up to the sky they tell me you were an enchanting dance"
filtered_paragraph = remove_stopwords(paragraph)
word_frequencies = count_word_frequencies(filtered_paragraph)
‘’’print("Original paragraph:")
print(paragraph)
print("\nParagraph after removing stopwords:")
print(filtered_paragraph)’’’ # you can uncomment to check the other output.
print(word_frequencies)
/-----------------------------------------Output------------------------------------------/
defaultdict(<class 'int'>, {'mama': 2, 'tell': 6, 'dancer': 2, 'long': 1, 'beautiful': 2, 'legs': 1, 'carry': 1, 'graceful': 1, 'body': 1, 'sang': 1, 'solos': 1, 'closed': 1, 'eyes': 1, 'always': 1, 'feeling': 1, 'songwas': 1, 'right': 1, 'lifted': 1, 'face': 1, 'sky': 1, 'enchanting': 1, 'dance': 1})
```

Determine text readability with Textatistic. And Spacy

```
from textatistic import TextBlob

#Read in text to analyze

text = "New york is a beautiful city."

#create a textblob object from the text

blob = TextBlob(text)

#print out the readability metrics

print(blob.readability)


import spacy

#create an English language

nlp = spacy.load ('en_core_web_sm')

#Define a text sample

sentence = "New York is a beautiful city."

#use spaCy's text parsing

parsed_text = nlp(sentence)

#print out the text and the type of each token

for token in parsed_text:

   print(token.text,token.lemma_,token)


/------------------------------------------Output------------------------------------------/


New New New

York York York

is be is

a a a

beautiful beautiful beautiful

city city city
```

Section 3

Case study

Title: Multi-linguicism Virtual Assistance for South African Languages with Python

This project aims to develop a Speech-to- text and text-to-Speech translator for different languages including some South African languages using Python. The goal is to solve issues like language barriers and miscommunications by making technology more accessible and inclusive for local communities. This would help in developing a text-to-speech translator that understands and generates speech in local South African languages, thereby reducing language barriers and improving communication within the community. This approach not only promotes linguistic diversity but also ensures that technology is inclusive and accessible to everyone, regardless of their language.

Challenges include the need to handle variations in accent and dialect, changes in meaning due to cultural or contextual differences, and a shortage of data from used libraries. Additionally, time constraints can pose a significant challenge in developing and refining the models. Ethical handling of data remains a crucial concern, especially when dealing with local languages and communities. These challenges highlight the complexity of developing a text-to-speech translator tailored for local communities in South Africa.

The project's success could help preserve and promote local languages and dialects, and the approach could potentially be extended to other underrepresented languages globally. Python libraries that could be used in this project include NLTK for natural language processing, googletrans,python translator, pyglet, speech recognition and pyttsx3 or gTTS for text-to-speech conversion.

Demo Code

speech_to_text.py

```
import speech_recognition as audio


class SpeechToText:
    def __init__(self, language='zu-ZA'):
        self.recogn = audio.Recognizer()
        self.language = language


    def listen_and_recognize(self):
        with audio.Microphone() as source:
            print("Talk")
            audio_text = self.recogn.listen(source)
            print("/--------Analyzing ----------/")
            try:
              print("Your words")
              return self.recogn.recognize_google(audio_text, language=self.language)
            except audio.UnknownValueError:
              print("Please try again I could not recognise anything you've said")
            except audio.RequestError as e:
              print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

speech_translator.py

```
from translate import Translator
from googletrans import Translator as GoogleTranslator


class SpeechTranslator:
  def __init__(self, text):
    self.text = text


  def speech_translator(self, text):
    translator = Translator(to_lang=self.language)
    try:
      translation = translator.translate(text)
      return translation
    except Exception as e:
      print(f"The text wasn't clear enough{e}")


  def google_text_translation(self, text):
    translator = GoogleTranslator()
    return translator.translate(text).text
```

text_to_speech.py

```
import pyttsx3


class TextToSpeech:
    def __init__(self, rate=110, voice_id=1):
        self.speech = pyttsx3.init()
        self.speech.setProperty('rate', rate)
        self.speech.setProperty('voice', self.speech.getProperty('voices')[voice_id].id)


    def say(self, corpus):
        self.speech.say(corpus)
        self.speech.runAndWait()
```

main.py

```
from speech_to_text import SpeechToText
from text_to_speech import TextToSpeech
from speech_translator import Translator
from googletrans import Translator as GoogleTranslator


class SpeechTranslator:
    def __init__(self, language='zulu'): # insert your from languange
        self.language = language
        self.stt = SpeechToText()
        self.tts = TextToSpeech()
        self.translator = Translator(to_lang=self.language)
        self.google_translator = GoogleTranslator()


    def translate_speech(self):
        while True:
            text = self.stt.listen_and_recognize()
        if not text:  # Assuming speech_to_text() returns None or an empty string when the speaker
is done
          break
        try:
          translation = self.google_translator.translate(text, dest=self.language).text
          print(f"Original Text: {text}")
          print(f"Translated Text: {translation}")
          self.tts.say(translation)
        except Exception as e:
          print(f"Error: {e}")
          print(f"No translation found for:{text}")


# Create an instance of the SpeechTranslator class
translator = SpeechTranslator(language='english')  # replace 'english' with your preferred
language to language
# Use the translate_speech method
translator.translate_speech()
```

```
Demo output.
Talk
/--------Analyzing ----------/
Your words
Original Text: sawubona bhuti omdala ngabe uyaphila
Translated Text: Hi old bro is you alive
```

NB: You can hear speech when you run the code.

Section 4

1.Which are the African Perspectives that can be incorporated into (replace this with a phrase from your selected application title)? Hints: Can you rewrite the hint to suit your selected application?

- There are various important tactics involved in integrating African perspectives into a multilingual virtual assistant. First of all, it entails utilizing native tongues like Swahili, Xhosa, and Zulu in order to support linguistic diversity and inclusivity in technology. Second, it necessitates knowledge of the cultural background, which includes expressions, proverbs, and idioms specific to various African communities. Thirdly, the assistant ought to be built with a focus on incorporating local expertise, offering data that is pertinent to the African setting, including news, events, and culturally noteworthy facts. In addition, the assistant could encourage cultural awareness and interchange by promoting African music, literature, and art. Finally, by supplying knowledge on regional goods, services, and trade prospects, the assistant might help local companies and promote global trade and business. This strategy guarantees a more varied, inclusive, and culturally sensitive virtual assistant.

2.How would you replace Western Concepts with African thought or thinking in your selected application? Hints: Can you figure out how to do this on your own or do you prefer to consult other students especially postgraduate student?

- African thought has been influenced by western concept due to colonization and other historical factors so to end that, we need to have more African data meaning we must publish everything using our African languages. Replacing western concepts by incorporating African languages into various sectors like Education, Media and other educational departments and encouraging research writers to write using African languages. In that way our languages cannot dissipate or end. Using powerful NLP algorithm and models like the multilingual virtual assistant we can translate existing NLP models and algorithms to make African based one's after we that we can improve them.

3 What are the challenges faced by your group in determining the role African Thought can play in an application like your own*?* **Hints**: Lack of understanding of Data Science/AI process in your own languages, Ignorance of what Object Orientation represents in the African way of thinking

- First, we had a challenged of ascent which made our model to not function perfectly this is a problem because in African culture every word has a meaning and with an ascent the meaning can be altered easily this may result to sensitivity to the person/ issue being addressed.

- Due to time that we hand which was limited maybe we could have tried to train our model to accommodate African language using machine learning models unfortunately we couldn't.
- Translation to African language is still not that good this could be caused by shortage of date. This are one of the things that NLP expertise is working on let soon enough we will be able to control our device and machine with our own languages this could have good impact on IoT actual NLP as a whole is having a good impact on IoT.

4.Can you reason out African Kingdom practices that can be used to make your application specifically relevant to those who will use it? **Hint**: Find African practices that can be used to explain the relevance of at least one aspect of your application.

- In our case study that we wrote about an interaction between a doctor and a patient whereby they speak different languages. So, the multilingual virtual assistant acts as a third part to help the communication between the doctor and patient to be better reducing language barriers.
- It can also be used in universities where there are students and lecturers who speak different languages. It can help to reduce miscommunication between people.
- It can be used in business where different people from different countries are collaborating on a single project. The multilingual virtual assistant can assist them in making decisions.
- This could be a tool for anywhere where language barrier is present, and it can reduce cost of paying for a human translator anytime you are going somewhere.

5.What are the stumbling blocks in the way of using African Community Practices and Beliefs to fully indigenize your application? **Hint**: Current books do not use such approaches, research findings on this matter may be limited, African communities do not see any value in

- The main stumbling block we had was Limited documentation, resources and data on African community practices and beliefs, requiring extensive research and community engagement. Other that may course this were copy writes this resulted to data shortage.