POS TAGGER AND SPELL CHECKER WITH STOPWORDS IDENTIFICATION USING NLP

DESIGN PROJECT REPORT

Submitted by

DHANUSH.V (Register No. 811720243011)

BHARATH.G(Register No. 811720243009)

BATRICK JOSHWA G.N (Register No. 811720243008)

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JULY 2023

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this design project report titled **"POS TAGGER AND SPELL CHECKER WITH STOPWORDS IDENTIFICATION USING NLP"** is the Bonafide work of **DHANUSH.V (811720243011), BHARATH.G (811720243009), BATRICK JOSHWA.G.N (811720243008),** who carried out the project under my supervision of **Mr. P.B. ARAVIND PRASAD B.E,M.Tech., (ASSISTANT PROFESSOR)**. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr. P.B. ARAVIND PRASAD B.E,M.Tech.,
ASSISTANT PROFESSOR
Department of AI&DS
K.Ramakrishnan College of Technology (Autonomous)
Samayapuram – 621 112

SIGNATURE

DR.T.AVUDAIAPPAN M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
Department of AI&DS
K.Ramakrishnan College of Technology (Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"POS TAGGER AND SPELL CHECKER WITH STOPWORDS IDENTIFICATION USING NLP"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

DHANUSH.V
(811720243011)

BHARATH. G
(811720243009)

BATRICK JOSHWA. G.N
(811720243008)

Place:Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** for providing his encourage pursuing this project.

I express my deep and sincere gratitude to my project guide **Mr P.B. ARAVIND PRASAD B.E,M.Tech.,** Assistant Professor, Department of **ARTIFICIAL INTELLIGENCE,** for his incalculable suggestions, creativity, assistance, and patience which motivated me to carry out this project.

I render my sincere thanks to Course Coordinator **Mrs. E. SANTHOSHINI M.E.,** and other staff members for providing valuable information during the course. I wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Parts of Speech (POS) Tagging is a notable NLP topic that aims in assigning each word of a text the proper syntactic tag in its context of appearance. POS is a grammatical classification that commonly includes verbs, adjectives, adverbs, nouns, etc. DL and ML based POS taggers are being implemented as potential solutions to efficiently identify words in a given sentence across a paragraph.

Usually systems employ this tagging process for preprocessing the text. There are mainly two approaches, they are Rule and Stochastic Based Approach. My proposed approach use lateral entries in the dictionary along with their information. Here, it mainly depends on the frequency of occurrence of words in the dictionary and identifying the meaning, the context they are referred as well. While researchers use readily available stopwords lists that are derived from non-technical resources, the technical jargon of engineering fields contains their own highly frequent and uninformative words and there exists no standard stopwords list for technical language processing applications

Key Words:

POS, Stochastic, Rule Based, Hybrid Approach, Tokenization, Lemmatization, and Ambiguity Resolution.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLP	NATURAL LANGUAGE PROCESSING
POS	PARTS OF SPEECH
DL	DEEP LEARNING
SVM	SUPPORT VECTOR MACHINE
CNN	CONVOLUTION NEURAL NETWORK
LSTM	LONG SHORT-TERM MEMORY
NER	NAMED ENTITY RECOGNITION
NLTK	NATURAL LANGUAGE TOOLKIT
HMM	HIDDEN MARKOV MODELS
UML	UNIFIED MODELING LANGUAGE
DFD	DATA FLOW DIAGRAM

CHAPTER 1

INTRODUCTION

1.1 NATURAL LANGUAGE PROCESSING

NLP, which stands for Natural Language Processing, is a branch of artificial intelligence (AI) that focuses on the interaction between computers and human language. It involves developing algorithms and models to enable computers to understand, interpret, and generate natural language in a way that is meaningful and useful.

The main objective of NLP is to bridge the gap between human language and computer systems, enabling machines to process, analyze, and derive insights from text or speech data. NLP encompasses a wide range of tasks and applications that deal with language understanding and generation

NLP combines techniques from various fields, including linguistics, machine learning, and computational linguistics. It involves preprocessing steps like tokenization, stemming, and part-of-speech tagging, as well as advanced methods like deep learning, probabilistic models, and statistical analysis.

With the advancements in AI and the availability of large language models, NLP has seen significant progress in recent years. Language models like OpenAI's GPT-3 have demonstrated impressive language understanding and generation capabilities, pushing the boundaries of what NLP can achieve.

An algorithm is developed to process it. There are many different natural language processing algorithms, but two main types are commonly used:

Rules-based system. This system uses carefully designed linguistic rules. This approach was used early on in the development of natural language processing, and is still used.

Machine learning-based system. Machine learning algorithms use statistical methods. They learn to perform tasks based on training data they are fed, and adjust their methods as more data is processed. Using a combination of machine learning, deep learning, natural language processing algorithms hone their own rules through repeated processing and learn.

1.2 PARTS OF SPEECH

In Natural language processing (NLP), parts of speech (POS) refer to the grammatical categories or classes to which words in a sentence can be assigned based on their syntactic roles and behavior. POS tags are used to understand the structure and meaning of sentences and are crucial for various NLP tasks, including language understanding, machine translation, text generation, and information retrieval.

Here is a brief explanation of some common parts of speech:

1. **Noun (NN):** Nouns are words that represent people, places, things, or ideas. They can be concrete (e.g., "dog," "city") or abstract (e.g., "love," "happiness").

2. **Verb (VB):** Verbs are action words that express actions, events, or states of being. They describe what the subject of a sentence is doing or experiencing (e.g., "run," "eat," "is").

3. **Adjective (JJ)**: Adjectives modify or describe nouns by providing additional information about their qualities or attributes (e.g., "happy," "blue").

4. **Adverb (RB):** Adverbs modify verbs, adjectives, or other adverbs to indicate manner, time, place, frequency, or degree. They often answer questions such as "how," "when," "where," or "to what extent" (e.g., "quickly," "often").

5. **Pronoun (PRP):** Pronouns are words that can be used in place of nouns to avoid repetition. They refer to a previously mentioned or implied noun (e.g., "he," "she," "it").

6. **Preposition (IN):** Prepositions establish relationships between words in a sentence, typically indicating location, direction, time, or manner (e.g., "in," "on," "at").

7. **Conjunction (CC):** Conjunctions are used to connect words, phrases, or clauses. They can join words of the same type (e.g., "and," "or") or indicate relationships between different parts of a sentence (e.g., "but," "because").

8. **Interjection (UH):** Interjections are short exclamatory words or phrases that express strong emotions or sudden reactions (e.g., "wow," "ouch").

1.3 STOPWORDS

In Natural language processing (NLP), stopwords are commonly used words in a language that are considered to have little semantic meaning and are often removed from text during preprocessing. These words are typically function words or grammatical particles that occur frequently but do not carry significant information or contribute to the overall meaning of a sentence.

Examples of stopwords in English include articles ("a," "an," "the"), conjunctions ("and," "but," "or"), prepositions ("in," "on," "at"), pronouns ("he," "she," "it"), and other words like "is," "are," "was," "were," "to," "of," and so on. These words are usually necessary for constructing grammatically correct sentences but do not carry much semantic weight on their own.
Stopword removal is a common preprocessing step in NLP tasks such as text classification, information retrieval, and sentiment analysis.

Stopwords are common words in a language that are often removed from text during NLP preprocessing to eliminate noise and focus on more meaningful content words. Removing stopwords can enhance the accuracy and efficiency of various NLP tasks by reducing noise and improving the representation of text data. However, the specific list of stop words can vary depending on the application, domain, or language being analyzed. For example, certain domain-specific terms or commonly used words in a specific industry may have significance and should not be treated as stop words.

Common techniques for stop word removal involve using predefined stop word lists, which contain a set of commonly occurring words in a language. These lists are often provided by natural language processing libraries or can be customized based on the specific requirements of the task or domain.It's worth noting that stop word removal may not be necessary or beneficial for all text analysis tasks. In some cases, such as topic modeling or certain language understanding tasks, retaining stop words can provide valuable contextual information. The decision to include or exclude stop words should be based on the specific objectives and requirements of the analysis being performed.

CHAPTER 2
LITERATURE SURVEY

2.1 **TITLE :** Part of speech tagging: a systematic review of deep learning and Machine learning approaches

AUTHOR: Alebachew Chiche & Betselot Yitagesu

YEAR & PUBLICATION: 24 January 2022,

ALGORITHM USED: Viterbi Algorithm

ABSTRACT:

Natural language processing (NLP) tools have sparked a great deal of interest due to rapid improvements in information and communications technologies. As a result, many different NLP tools are being produced. However, there are many challenges for developing efficient and effective NLP tools that accurately process natural languages. One such tool is part of speech (POS) tagging, which tags a particular sentence or words in a paragraph by looking at the context of the sentence/words inside the paragraph. Despite enormous efforts by researchers, POS tagging still faces challenges in improving accuracy while reducing false-positive rates and in tagging unknown words. Furthermore, the presence of ambiguity when tagging terms with different contextual meanings inside a sentence cannot be overlooked. Recently, Deep learning (DL) and Machine learning (ML)-based POS taggers are being implemented as potential solutions to efficiently identify words in a given sentence across a paragraph. This article first clarifies the concept of part of speech POS tagging.

CONCLUSION : Loses important information by removing words that may be significant in a specific context.

2.2 **TITLE :** Professional chat application based on natural language processing

AUTHOR: S Karthick & R John Victor

YEAR & PUBLICATION: 4 June 2019

ALGORITHM USED: Brill-Tagger Algorithm

ABSTRACT:

There has been an emerging trend of a vast number of chat applications which are present in the recent years to help people to connect with each other across different mediums, like Hike, WhatsApp, Telegram, etc. The proposed network-based android chat application used for chatting purpose with remote clients or users connected to the internet, and it will not let the user send inappropriate messages. This paper proposes the mechanism of creating professional chat application that will not permit the user to send inappropriate or improper messages to the participants by incorporating base level implementation of natural language processing (NLP). Before sending the messages to the user, the typed message evaluated to find any inappropriate terms in the message that may include vulgar words, etc., using natural language processing. The user can build an own dictionary which contains vulgar or irrelevant terms.

CONCLUSION : The subjectivity of stop word list can affect the results of any downstream tasks.

2.3 **TITLE :** Speech recognition using Support Vector Machines

AUTHOR: Samir Rustamov & Kamil Aida-zade

YEAR & PUBLICATION: 16 June 2020

ALGORITHM USED: Support Vector Machines

ABSTRACT:

In this article we applied Support Vector Machines to acoustic model of Speech Recognition System based on MFCC and LPC features for Azerbaijani DataSet. This DataSet has been used for speech recognition by Multilayer Artificial Neural Network and achieved some results. The main goal of this work is applying SVM techniques to the Azerbaijan Speech Recognition System. The variety of results of SVM with different Kernel functions is analyzed in the training process. It is shown that SVM with radial basis and polynomial kernels give better recognition results that Multilayer Artificial Neural Network.Support Vector Machine or commonly called SVM is one method that can be used to process the classification of a data. SVM classifies data from 2 different classes with hyperplane. In this study, the system was built using SVM to develop Arabic Speech Recognition. In the development of the system, there are 2 kinds of speakers that have been tested that is dependent speakers and independent speakers. The results from this system is an accuracy of 85.32% for speaker dependent and 61.16% for independent speakers.

CONCLUSION : The possibility of losing important information by removing words that may be significant in a specific context.

2.4 **TITLE :** Part-of-speech labeling for Reuters database

AUTHOR: A.David & D.Morairu

YEAR & PUBLICATION: 21 February 2019

ALGORITHM USED: Deep Learning (convolutional neural networks)

ABSTRACT:

Even if the Vector Space Model used for document representation in information retrieval systems integrates a small quantity of knowledge it continues to be used due to its computational cost, speed execution and simplicity. We try to improve this document representation by adding some syntactic information such as the parts of speech. In this paper, we have evaluated three different tagging algorithms in order to select the most suitable tagger for using it to tag the Reuters dataset. In this work, we have evaluated the taggers using only five different parts of speech: noun, verb, adverb, adjective and others. We considered these particular tags being the most representative for describing the documents into these parts of speech space.Part of speech tagging is the basic step of identifying a token's functional role within a sentence and is the fundamental step in any NLP pipeline. Several methods and approaches have been employed for POS Tagging, with various levels of performance. These approaches can be broadly divided into Statistical approaches and Rule-based approaches. Almost all tagger performance results reported in the literature are at the word token level. Many state-of the-art methods report a high level of accuracy in POS tagging at the token level. For example, the Stanford POS Tagger [Klein, 2003], TnT POS Tagger [Brants, 2000] and SVMTool POS Tagger [Giménez, 2004] have reported their average token level accuracies at over 96% to measure the correctness of POStagging for the sentence as a whole.

CONCLUSION : Relevant stop word lists can be hard to find in some languages and so may not scale as more languages need to be processed.

2.5 **TITLE :** Part-of-Speech Tagging with Rule-Based Data Preprocessing and Transformer

AUTHOR: Hongwei Li & Hongyan Mao

YEAR & PUBLICATION: 24 December 2021

ALGORITHM USED: Baum-Welch Algorithm

ABSTRACT:

Part-of-Speech (POS) tagging is one of the most important tasks in the field of natural language processing (NLP). POS tagging for a word depends not only on the word itself but also on its position, its surrounding words, and their POS tags. POS tagging can be an upstream task for other NLP tasks, further improving their performance. Therefore, it is important to improve the accuracy of POS tagging. In POS tagging, bidirectional Long Short-Term Memory (Bi-LSTM) is commonly used and achieves good performance. However, Bi-LSTM is not as powerful as Transformer in leveraging contextual information, since Bi-LSTM simply concatenates the contextual information from left-to-right and right-to-left. In this study, we propose a novel approach for POS tagging to improve the accuracy. For each token, all possible POS tags are obtained without considering context, and then rules are applied to prune out these possible POS tags, which we call rule-based data preprocessing.

.

CONCLUSION : The need to maintain and update the stop word list as the language and domain evolve.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 H/W SYSTEM CONFIGURATION: -

- ➢ Processor - Intel core i5
- ➢ RAM - 8 GB (min)
- ➢ Hard Disk – 50 GB

3.2 S/W SYSTEM CONFIGURATION: -

- ➢ Operating System: Windows 7 and above
- ➢ Front End : **Visual Studio Code**
- ➢ Back End : **Jupyter Notebook**

3.3 SOFTWARE DESCRIPTION:

Our NLP software is an advanced and powerful tool designed to process and analyze natural language data. It leverages state-of-the-art machine learning algorithms and linguistic techniques to extract meaningful information and insights from text-based data, enabling a wide range of applications and use

- ➢ ***Text Parsing and Tokenization***: Our software can break down raw text into individual words, sentences, or other meaningful units, allowing for precise analysis and understanding.

- ➢ ***Part-of-Speech Tagging:*** It assigns grammatical tags to each word in a sentence, such as nouns, verbs, adjectives, etc. This feature enables syntactic analysis and language understanding.

- ➢ ***Named Entity Recognition (NER)***: Our NLP software can identify and classify named entities within text, such as names of people, organizations, locations, dates, and more. This aids in information extraction and entity-based analysis.

- ➢ ***Sentiment Analysis:*** It analyzes the sentiment expressed in text, determining whether the overall sentiment is positive, negative, or neutral. This feature is valuable for sentiment monitoring, brand reputation management, and customer feedback analysis.

3.4 LIBRARIES:

1) Gramformer:

Gramformer is a deep learning model specifically designed for text generation and correction. It builds upon the popular Transformer architecture and utilizes a combination of deep learning techniques to improve the quality and fluency of generated text.

- ***Text Correction:*** Gramformer focuses on correcting grammar, spelling, and punctuation errors in text. It employs advanced language models and context-aware algorithms to suggest and apply corrections, improving the overall accuracy and readability of the generated content.

- ***Sentence Rewriting:*** The model can rewrite sentences to enhance clarity, coherence, and style. It can rephrase awkward or ambiguous sentences, improve sentence structure, and refine word choices to make the text more engaging and natural.

- ***Paraphrasing:*** Gramformer can generate alternative versions of a given sentence while preserving its meaning. This feature is useful for content generation, data augmentation, and text enrichment tasks.

2) NLTK:

NLTK (Natural Language Toolkit) is a popular open-source library for natural language processing (NLP) tasks in Python. It provides a wide range of tools, resources, and algorithms for processing, analyzing, and manipulating human language data.

Key Features and Capabilities of NLTK:

- ***Tokenization:*** NLTK offers various tokenization methods to split text into individual words or sentences, including word tokenization, sentence tokenization, and regular expression-based tokenization.

- ***Part-of-Speech Tagging:*** It provides functions and pre-trained models for assigning grammatical tags to words in a sentence, such as nouns, verbs, adjectives, etc. This is useful for syntactic analysis and language understanding.

- ***Stemming and Lemmatization:*** NLTK includes algorithms for stemming and lemmatization, which reduce words to their base or root form. This helps in normalizing and standardizing text for analysis and retrieval purposes.

- ***Named Entity Recognition (NER):*** It offers pre-trained models and tools to identify and extract named entities from text, such as names of people, organizations, locations, dates, and more.

- ***Chunking and Parsing:*** It supports chunking and parsing techniques to identify and extract phrases or syntactic structures from sentences

3) SpaCy:

SpaCy is a popular and efficient Python library for natural language processing (NLP). It is designed to provide fast and accurate processing of large volumes of text data. SpaCy focuses on production-ready NLP tasks and offers a range of capabilities for various stages of text processing and analysis.

- ***Lemmatization and Word Vectors:*** SpaCy supports lemmatization, which reduces words to their base or dictionary form. It also provides word vectors, which are dense numerical representations of words that capture semantic similarity. These features are helpful for tasks like word sense disambiguation and similarity analysis.

- ***Text Classification:*** SpaCy supports text classification tasks, allowing you to train and evaluate models for tasks like sentiment analysis, document classification, and topic classification. It provides a flexible API for training custom classifiers and supports integration with deep learning frameworks like TensorFlow and PyTorch.

- ***Rule-based Matching:*** SpaCy includes a powerful rule-based matching engine that allows you to define patterns to find and extract textual patterns based on syntactic or lexical rules. It is useful for tasks like information extraction, entity linking, and pattern-based text analysis.

- ***Language Support:*** SpaCy supports multiple languages and provides pre-trained models for various languages, including English, German, French, Spanish, and more.

4) Gradio:

Gradio is designed to work with various machine learning libraries, including those used in NLP tasks, such as TensorFlow, PyTorch, and scikit-learn. While Gradio itself does not provide NLP-specific functionalities, you can utilize it to create UIs for NLP models or applications.Here's an example of how Gradio can be used with an NLP model:

(i) You have an NLP model, such as a sentiment analysis classifier, built using libraries like TensorFlow or PyTorch.

(ii) With Gradio, you can create a user-friendly UI for your sentiment analysis model. You can define input components, such as text boxes or dropdowns, where users can enter text for sentiment analysis.

(iii) Gradio allows you to specify how the model should process the user's input and display the results. For example, you can define a function that takes the user's input, passes it through the NLP model, and returns the sentiment analysis prediction.

(iv) Gradio automatically generates a UI based on your specifications, displaying the input components and the model's output. Users can interact with the UI by entering text and immediately seeing the sentiment analysis result.

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

In this identifying POS Tagging is much more difficult when compared to matching words to their independent parts of speech via a dictionary method.

It is quite different to sense that one word has two or more meanings based on their reference and the context they are being used. It is difficult to individually classify and assign the parts of speech for words manually.

Algorithm Used : SVM(Support Vector Machine)

4.1.1 SUPPORT VECTOR MACHINE :

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

4.1.2 HIDDEN MARKOV MODELS :

(HMMs) are statistical models used to describe and analyze sequential data. They are particularly effective in situations where the underlying system is assumed to be a Markov process with hidden states.

In an HMM, we have a sequence of observable events or symbols, and the goal is to infer the sequence of hidden states that generated these observations. The hidden states are not directly observable but influence the observed symbols. For example, in speech recognition, the observed symbols are the acoustic features of the speech signal, while the hidden states represent the phonemes or words being spoken.

The key components of an HMM are:

1. State Set: A finite set of hidden states, which represent the different states of the underlying process. Each state has an associated probability distribution over the observable symbols.

2. Transition Matrix: A matrix that describes the probabilities of transitioning from one hidden state to another. The entry (i, j) in the matrix represents the probability of transitioning from state i to state j.

3. Emission Matrix: A matrix that specifies the probabilities of emitting each observable symbol from each hidden state. The entry (i, j) in the matrix represents the probability of emitting symbol j given that the system is in state i.

The three fundamental problems associated with HMMs are:

1. Evaluation*:* Given a sequence of observations and an HMM model, we want to calculate the probability of observing that sequence. This can be efficiently solved using the forward algorithm or the backward algorithm.

2. Decoding: Given a sequence of observations and an HMM model, we want to find the most likely sequence of hidden states that generated the observations. The Viterbi algorithm is commonly used to solve this problem.

3. Learning: Given a set of observations and their corresponding hidden states, we want to estimate the parameters of the HMM model (i.e., the transition probabilities and emission probabilities). The Baum-Welch algorithm, also known as the forward-backward algorithm, is commonly used for parameter estimation. HMMs have been successfully applied in various fields, including speech recognition, natural language processing, bioinformatics, gesture recognition, and finance. They provide a powerful framework for modeling and analyzing sequential data with hidden dependencies.

Disadvantages Of Existing System

- ➢ Since The System Uses Pre-Trained Data It Can’t Able To Identify New Words.
- ➢ The Existing System Needs Frequent Updates And Maintain The Stop-words List That Requires Man-Made Effort To Do It.
- ➢ Not Able To Identify The Homogeneous Words ie)like Bass-Base.

4.2 PROPOSED SYSTEM

The objective of this paper is to increase automaticity and maintain high precision, while limiting the size of human made corpus. In our current work we approach the task of POS tagging as an optimization problem. Thereafter two new approaches based on the principles of single objective optimization and multi-objective optimization are proposed for POS tagging.

ALGORITHM USED

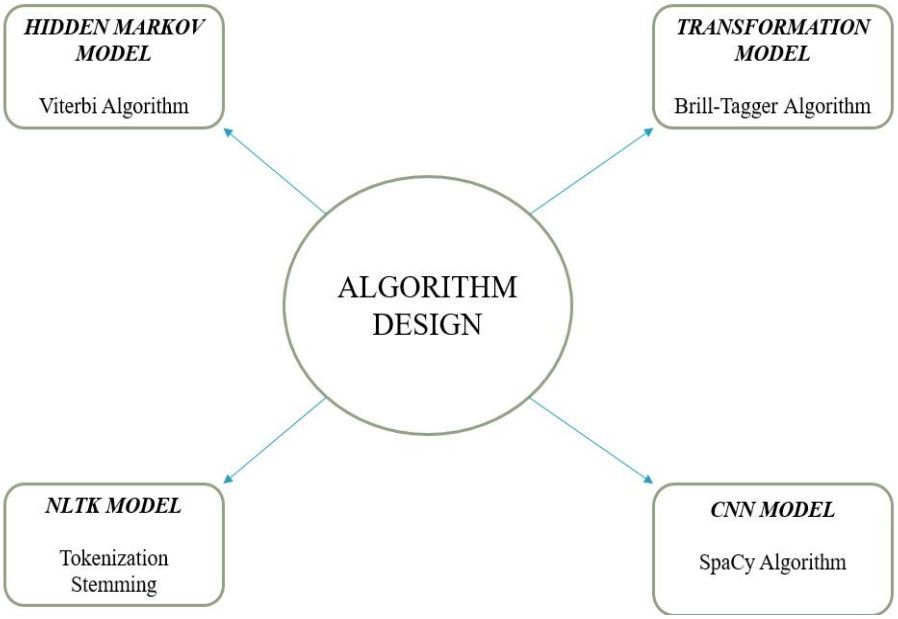

Fig : 4.2.1 Algorithm Structure

In this proposed system we are using an algorithms are

- ➢ Multinomial naive bayes
- ➢ Brill tagger algorithm
- ➢ Viterbi algorithm
- ➢ SpaCy algorithm

4.2.1 MULTINOMIAL NAIVE BAYES :

Multinomial Naive Bayes is a popular algorithm used for text classification tasks, particularly when dealing with features that represent discrete word counts or frequencies. It is based on the principle of Bayes' theorem and assumes independence between the features (words) in the input text. The algorithm is called "Multinomial" because it assumes a multinomial distribution for the features. Multinomial Naive Bayes assigns a probability to each possible class label based on the occurrence of features in the input text. It calculates the likelihood of a particular class label given the feature occurrences using the Bayes' theorem and assumes that the occurrences are conditionally independent. The algorithm estimates the probabilities using the training data, where the class labels and their corresponding feature occurrences are known. During training, it calculates the prior probability of each class label and the likelihood of each feature given a class label During prediction, the algorithm computes the probability of each class label for a given input text using the trained probabilities. The class label with the highest probability is assigned as the predicted class label for the input text. Multinomial Naive Bayes can handle large feature spaces efficiently and is particularly suited for text classification tasks such as sentiment analysis, spam detection, and document categorization. It performs well when the independence assumption holds reasonably well and when the feature frequencies are informative for the task at hand.

4.2.2 BRILL TAGGER ALGORITHM :

The Brill Tagger is a rule-based algorithm for part-of-speech tagging in natural language processing. It starts with an initial tagging of words in a sentence using a simple tagger. The algorithm then iteratively applies transformational rules to improve the tagging accuracy. Transformational rules specify how to change the tag of a word in a specific context. The algorithm uses a training corpus with manually annotated tags to learn effective transformations. In each iteration, the algorithm applies the rules to the training corpus and evaluates their performance. The best rule, which leads to the most improvement, is selected and applied. The process continues for multiple iterations until no further improvements are observed. The Brill Tagger algorithm learns from the training data and can adapt to specific domains or languages. It can capture complex patterns and linguistic regularities, improving tagging accuracy. The algorithm requires a training corpus with annotated tags and can be computationally expensive.

4.2.3 VITERBI ALGORITHM :

The Viterbi algorithm is a dynamic programming algorithm used to find the most likely sequence of hidden states in a hidden Markov model (HMM). It is named after Andrew Viterbi, who developed the algorithm in 1967.The Viterbi algorithm is commonly used in various fields, including speech recognition, natural language processing, bioinformatics, and more. The algorithm assumes an underlying HMM with a set of observable outputs (emissions) and a set of hidden states. Each state emits an output with a certain probability, and transitions between states occur with transition probabilities. The goal of the Viterbi algorithm is to find the most likely sequence of hidden states that generated a given sequence of observations.

The algorithm operates in a dynamic programming fashion, evaluating the probabilities of possible state sequences incrementally**.** It utilizes two main matrices: the Viterbi trellis and the backpointer matrix. The Viterbi trellis stores the probabilities of the most likely state sequences at each time step, considering all possible paths. The backpointer matrix keeps track of the optimal path leading to each state at each time step. The algorithm recursively calculates the probabilities in the Viterbi trellis and updates the backpointer matrix Once the entire sequence is processed, the algorithm backtracks through the backpointer matrix to find the most likely state sequence.

The Viterbi algorithm guarantees finding the globally optimal solution in terms of the most likely state sequence. The complexity of the algorithm is linear with respect to the number of states and quadratic with respect to the length of the observation sequence. Various optimizations and extensions have been proposed to improve the performance and address specific requirements of different applications.

4.2.4 SPACY ALGORITHM :

SpaCy is an open-source library for natural language processing (NLP) in Python. It provides a wide range of functionalities for tasks such as tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and more. While SpaCy utilizes multiple algorithms under the hood, Rule-based Matching: SpaCy allows you to define patterns using linguistic rules to perform efficient and customizable text matching.

It uses the Aho-Corasick algorithm, which constructs a finite state machine to efficiently search for multiple patterns in a text simultaneously. Tokenization: SpaCy employs a rule-based algorithm to split text into individual tokens. It handles cases like punctuation, contractions, hyphenated words, and more. The algorithm uses heuristics and specific rules to determine token boundaries. Part-of-Speech Tagging: SpaCy utilizes a statistical model trained on labeled data to assign part-of-speech tags to each token in a sentence.

The underlying algorithm is typically based on a combination of linear models, such as logistic regression or support vector machines, and neural network architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs). Named Entity Recognition (NER): SpaCy employs a statistical model to identify and classify named entities in text, such as person names, organizations, locations, dates, and more. The NER algorithm typically utilizes techniques like conditional random fields (CRF) or bidirectional LSTMs (Long Short-Term Memory networks) to perform sequence labeling. Dependency Parsing: SpaCy utilizes transition-based parsing algorithms to analyze the grammatical structure of a sentence and determine the dependencies between words. The specific algorithm used by SpaCy is based on the arc-eager parsing algorithm, which is a deterministic transition-based parsing algorithm.

Advantages Of Proposed System

- ➢ Here, We Uses POS Tagger As An Optimization Solution Like it's a Key Card To Overcome The Existing Problem.
- ➢ This System Has Updated Libraries So It Can Provide A Solution For New Words .
- ➢ The Major Advantage Of Using This System That It Can Able To Word Sense The Meaning Of The Context.

CHAPTER 5

ARCHITECTURAL DESIGN

5.1 ARCHITECTURE REPRESENTATION

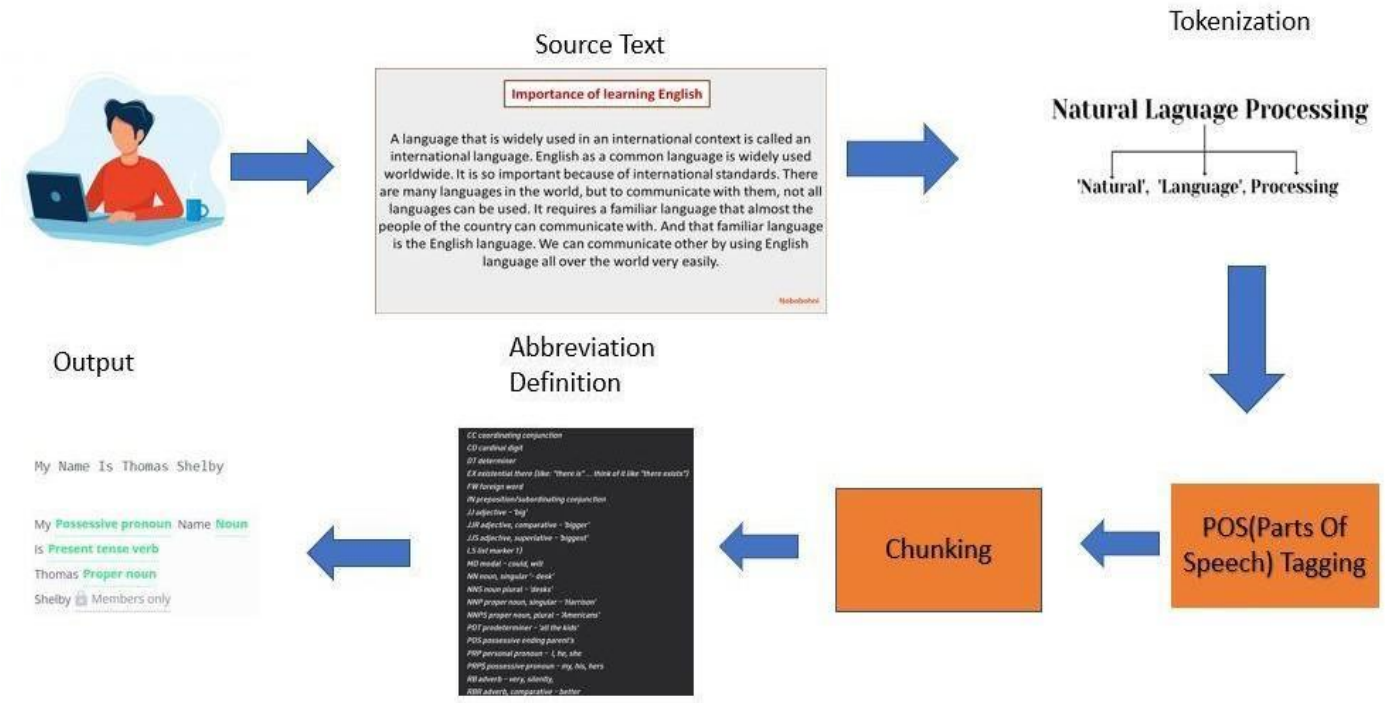

Fig: 5.1.1 Architecture Diagram

- ➢ This Architecture revolves around the user and how the programme works. The User first enter the text which they want to analyze it.
- ➢ Then the process involves to Tokenization, the whole text converted into words by removing stop words .
- ➢ After The text goes to POS Part Where It involves Grammatical Works.
- ➢ Chunking, in the context of natural language processing (NLP), is a process of identifying and grouping together certain parts of speech in a sentence to form meaningful syntactic units called chunks.
- ➢ It is a step beyond part-of-speech tagging and aims to extract higher-level structures or phrases from a sentence.

5.2 DATA FLOW DIAGRAM

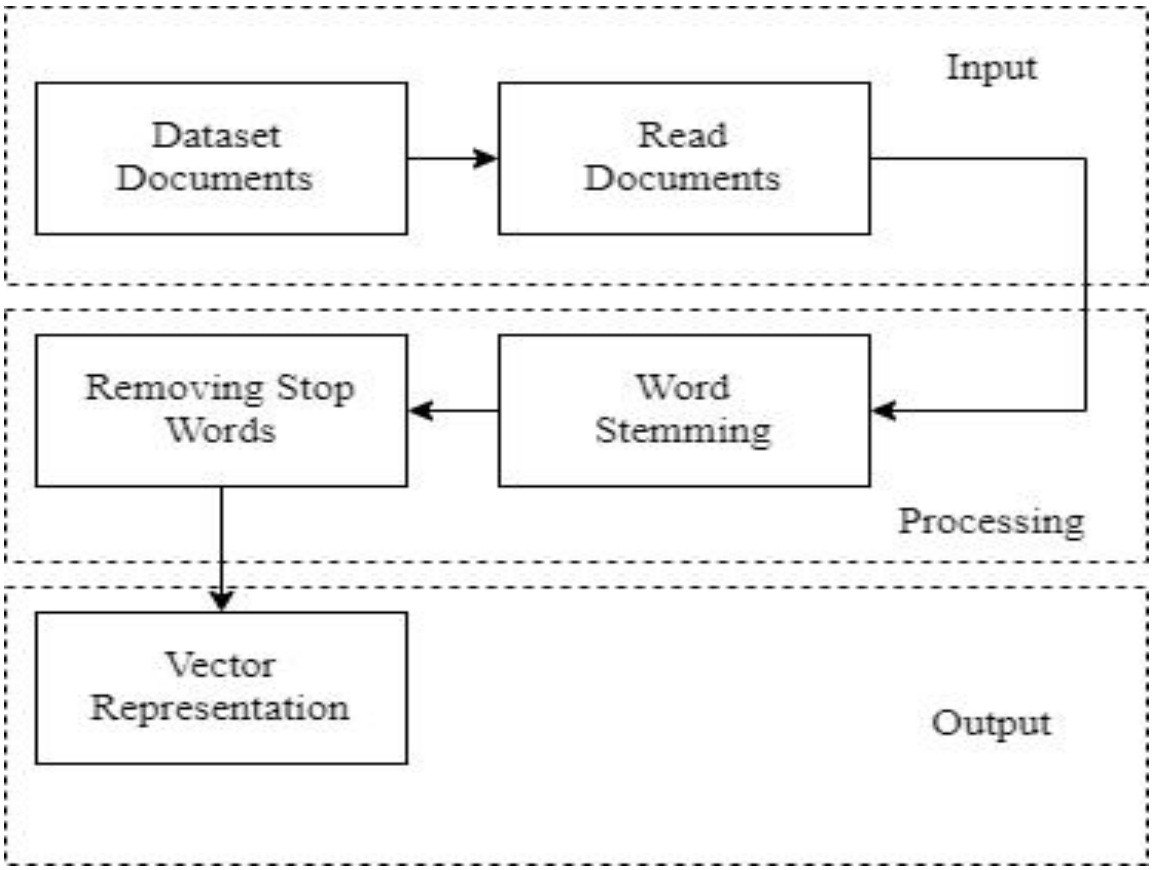

Fig : 5.2.1 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data within a system. In the case of a point of sale (POS) system, a DFD can illustrate how data moves through the system, including the use of stopwords in natural language processing (NLP). Here's an example of a DFD for a POS system that incorporates stopwords in NLP:

"Input Product Information" is a use case where the user provides product-related information, which may include product names, descriptions,

"Process Text (Stopword Removal and NLP)" is a use case where the POS system applies NLP techniques, including stopword removal, to preprocess and analyze the input text.

"POS System" represents the core functionality of the system, responsible for processing the input and generating the output.

"Output Results (Processed Text)" is a use case where the POS system provides the processed text, which may be devoid of stopwords and contain other relevant information derived from the NLP analysis.

5.3 UML DIAGRAM

5.3.1 USE CASE DIAGRAM

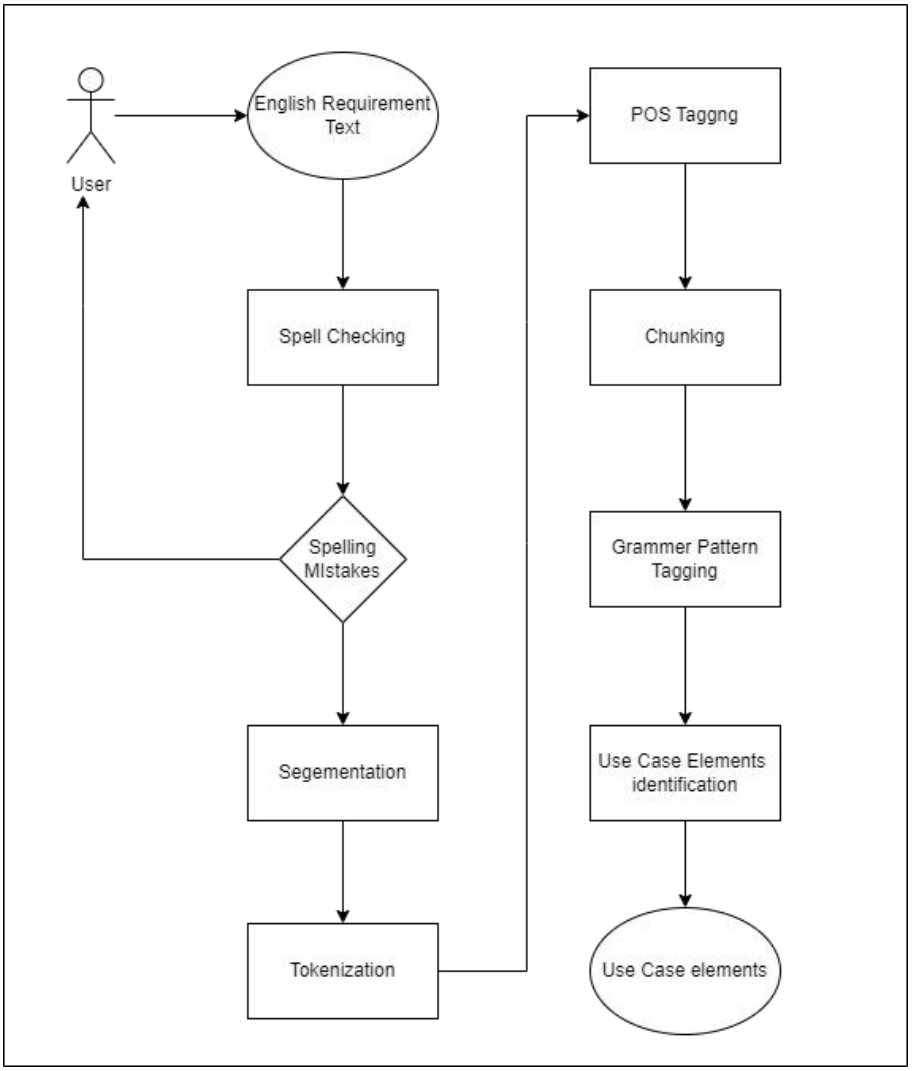

Fig : 5.3.1.1 Use Case Diagram

1. **Capture Input**: User interacts with the POS system, providing input such as product names or descriptions.

2. **Preprocessing*:*** The input data is preprocessed to remove unnecessary elements, such as punctuation and special characters. Stopwords, which are commonly used words that often do not carry significant meaning in NLP tasks, are identified and filtered out from the input data.

3. **Tokenization:** The preprocessed input is tokenized, meaning it is split into individual words or tokens. Each token is examined to determine if it is a stopword or contains any stopwords.

4. **Stopword Removal:** The tokens identified as stopwords are removed from the data flow, reducing noise and irrelevant information.

5.3.2 ACTIVITY DIAGRAM

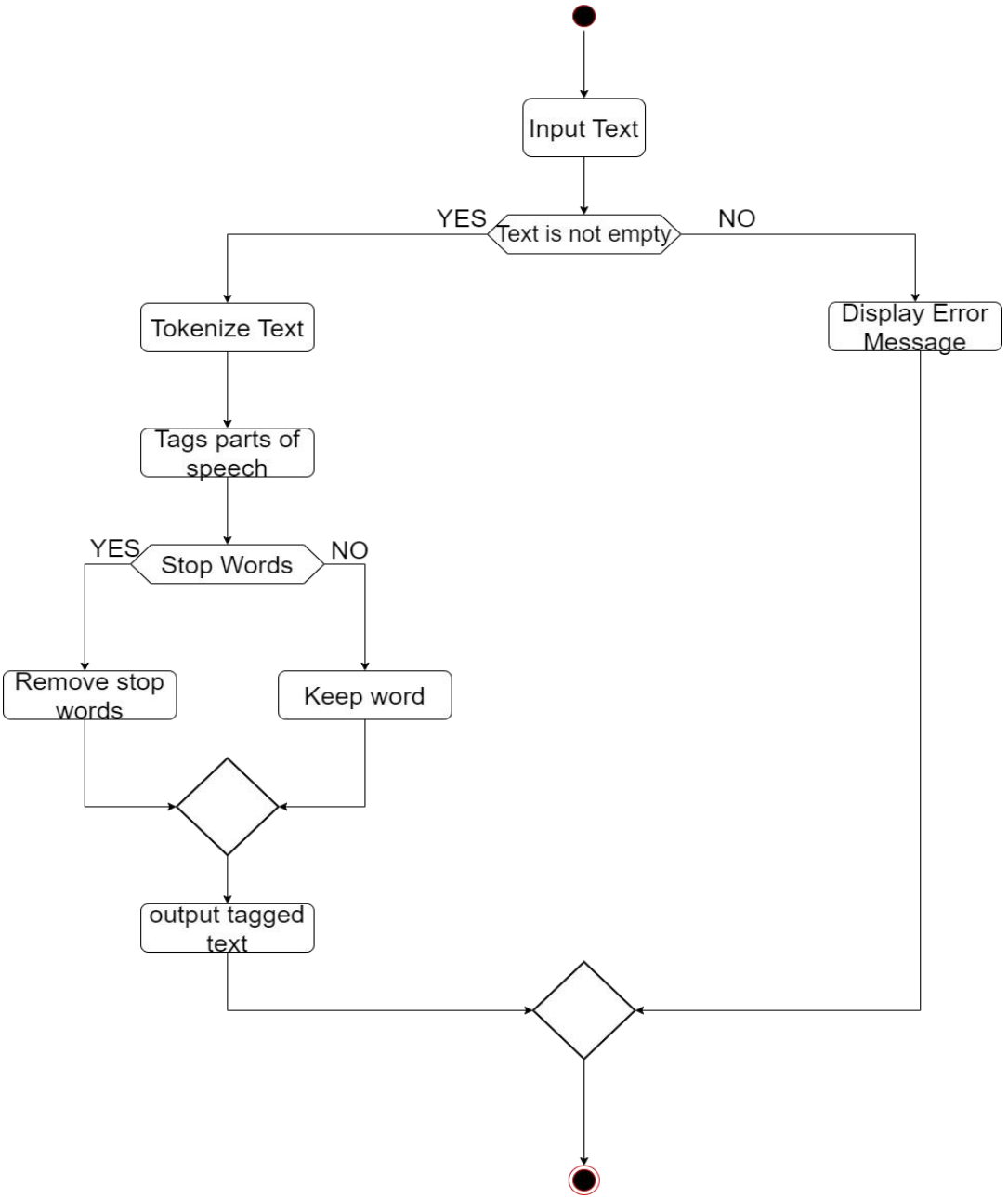

Fig : 5.3.2.1 Activity Diagram

1. The "User" initiates the process by providing input product information.
2. The "Input Product Information" activity represents the user inputting the product details, such as names or descriptions.
3. The "Preprocess Text" activity involves preprocessing the input text, which includes stopword removal and other NLP techniques.
4. The "Tokenization" activity breaks down the preprocessed text into individual tokens or words.
5. The "Stopword Removal" activity filters out the stopwords from the tokenized text.
6. The "POS Tagging" activity assigns part-of-speech labels to the remaining tokens (optional step).
7. The "Further Processing" activity represents any additional NLP tasks or analysis that may be performed using the processed text.
8. Finally, the "Output Results" activity delivers the processed text or the outcomes of the NLP processing to be utilized within the POS system, such as inventory management, sales analytics, or personalized recommendations.

5.3.3 SEQUENCE DIAGRAM

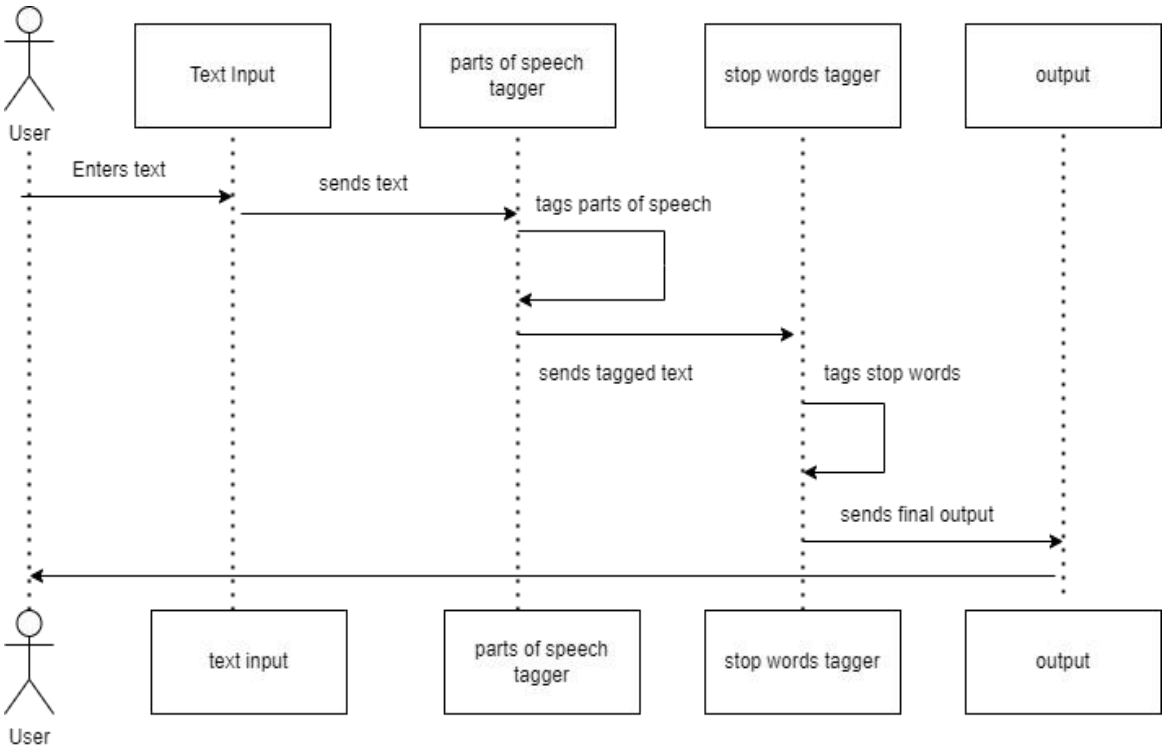

Fig : 5.3.3.1 Sequence Diagram

1. The user interacts with the POS system and provides input product information.
2. The POS system receives the input and initiates the NLP component.
3. The NLP component performs various NLP operations, such as preprocessing, tokenization, stopword removal, POS tagging (optional), and further processing.
4. The NLP component outputs the processed text or results.
5. The POS system stores the processed text or results in the database.
6. The POS system retrieves the stored results from the database.
7. The POS system provides the output results to the user.

5.3.4 STATE DIAGRAM

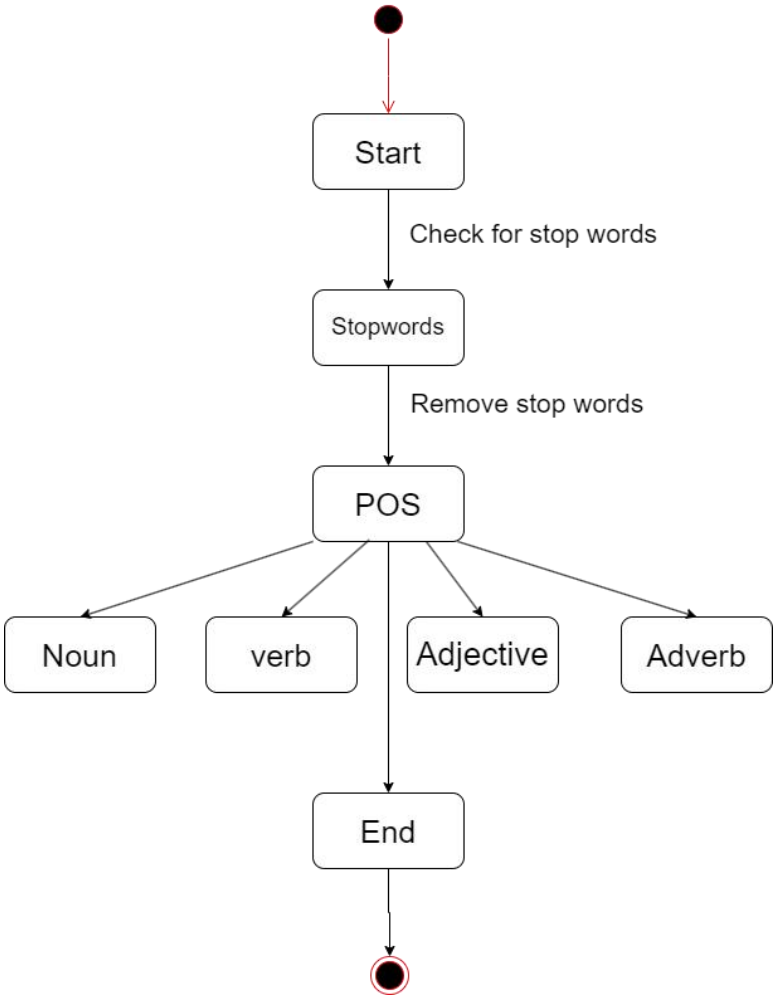

Fig : State Diagram

A state diagram, also known as a state machine diagram, represents the different states and transitions of an object or system. In the context of using stopwords in NLP within a POS system, a state diagram can illustrate the various states and transitions that occur during the processing of text. Here's a simple explanation of a state diagram for using stopwords in NLP:

The state diagram represents three main states: "Input", "Processing", and "Output".

Input State:

- ➢ This state represents the initial state where the system is ready to receive input.
- ➢ When the input is provided, the system transitions to the "Processing" state

Processing State:

- ➢ In this state, the system processes the input text using NLP techniques, including stopwords removal.
- ➢ The system may go through sub-states within the "Processing" state, such as "Preprocessing", "Tokenization", "Stopword Removal", and "POS Tagging" (optional), depending on the specific NLP steps involved.
- ➢ After completing the processing steps, the system transitions to the "Output" state.

Output State:

- ➢ This state represents the final state where the system provides the processed output or results.
- ➢ The output can be used for various purposes within the POS system, such as inventory management, sales analytics, or personalized recommendations.
- ➢ Once the output is delivered, the system transitions back to the "Input" state, ready to receive the next input.

CHAPTER 6

MODULE DESCRIPTION (SPLIT UP)

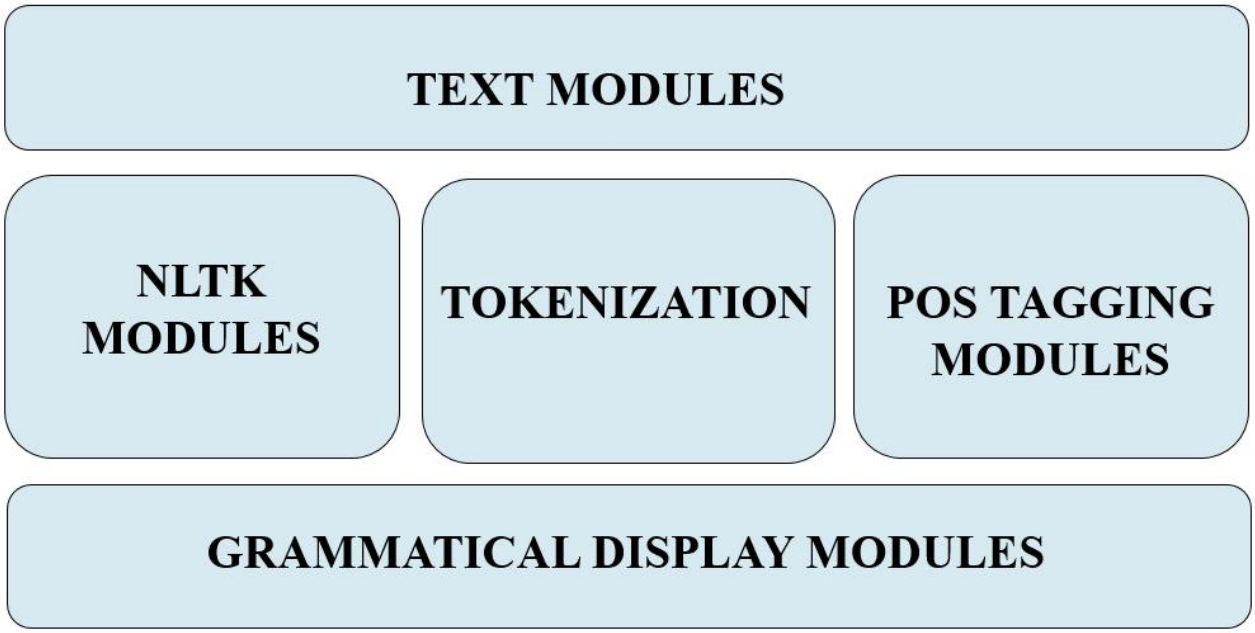

Fig : 6.1 Module Description (Split Up)

6.1 TEXT MODULES :

➢ **Word Embeddings**: Word embedding modules represent words or phrases as dense numerical vectors in a continuous vector space. They capture semantic relationships between words, enabling algorithms to understand the meaning and context of words in a text.

➢ **Language Modeling**: Language modeling modules are trained to predict the probability of the next word in a sequence given the previous words. They are used for tasks such as auto-completion, machine translation, and speech recognition.

➢ **spaCy** : spaCy is a popular Python library for natural language processing. It offers efficient parts of speech tagging and stop word removal capabilities. You can use the `pos_` attribute of spaCy's token objects for parts of speech tagging and the `is_stop` attribute for stop word identification.These libraries can be integrated into your code to perform parts of speech tagging and stop word removal on text data.

6.2 NATURAL LANGUAGE TOOLKIT :

➢ The Natural Language Toolkit (NLTK) is a popular Python library for natural language processing (NLP). It provides a wide range of functionalities and tools for working with human language data. NLTK includes various modules and resources for tasks like tokenization, stemming, lemmatization, parts of speech tagging, named entity recognition, sentiment analysis, and more.

➢ NLTK is designed to be beginner-friendly and provides a comprehensive set of documentation, tutorials, and examples to help users get started with NLP tasks. It also offers a collection of corpora, lexical resources, and pre-trained models that can be used for experimentation and research purposes.

➢ To work with NLTK, you need to install the library and download the required data and models using the NLTK downloader

6.3 TOKENIZATION :

6.3.1 Stemming and lemmatization:

NLTK offers modules for stemming and lemmatization, which help reduce words to their base or root form. The library includes stemmers like the Porter stemmer (PorterStemmer) and Lancaster stemmer (LancasterStemmer), as well as lemmatizers like the WordNet lemmatizer (WordNetLemmatizer).

Stemming: It is the process of reducing words to their base or root form by removing suffixes. It is a rule-based approach that chops off the ends of words based on predefined rules. The resulting form may not always be a valid word, but it helps in grouping together words with the same root.

For example:

Words: "running," "runs," "ran"

Stemmed forms: "run," "run," "run"

The popular stemming algorithms include Porter stemming algorithm, Snowball stemming algorithm (also known as the Porter2 algorithm), and Lancaster stemming algorithm.

Lemmatization: Lemmatization, on the other hand, aims to reduce words to their base form while ensuring that the resulting form is a valid word, known as the lemma. It involves analyzing the word's morphological features and applying linguistic rules to determine the lemma.

For example:

Words: "running," "runs," "ran"

Lemmatized forms: "run," "run," "run"

6.3.2 Named Entity Recognition (NER):

NLTK provides modules for NER, enabling you to identify and extract named entities from text. It includes a pre-trained NER chunker that can be used to label named entities like persons, organizations, locations, etc. NER can be performed using various approaches, including rule-based methods, statistical models, and machine learning techniques. Many NER systems leverage pre-trained models and annotated datasets to recognize and classify named entities accurately.

6.4 POS TAGGING :

➢ Part-of-speech (POS) tagging is a natural language processing (NLP) task that involves assigning grammatical categories or labels (such as noun, verb, adjective, etc.) to each word in a given sentence or text. POS tagging helps in understanding the syntactic structure and meaning of sentences.

➢ POS tags provide information about the word's role in the sentence, its grammatical properties, and its potential relationships with other words. For example:

- Noun (NN): dog, house, book
- Verb (VB): run, eat, sleep
- Adjective (JJ): big, happy, red
- Adverb (RB): quickly, very, often

➢ POS tagging can be performed using various techniques, including rule-based methods, statistical models, and machine learning algorithms. Machine learning-based approaches, such as Hidden Markov Models (HMMs) or deep learning models, have shown good performance in POS tagging.

➢ Different POS tagsets and taggers may have slight variations in the labels used. It's important to refer to the specific tagset and documentation of the library or tool you are using for accurate interpretation of the POS tags.

➢ POS tagging is a fundamental step in many NLP tasks, such as syntactic parsing, information extraction, sentiment analysis, and machine translation. It provides valuable insights into the structure and semantics of text, enabling more advanced analysis and understanding of natural language.

6.5 GRAMMATICAL DISPLAY MODULES :

➢ In POS tagging, grammatical displays refer to the labels or symbols used to represent the grammatical categories assigned to words in a sentence. These labels provide information about the part of speech and grammatical properties of each word.

➢ Different tagsets or taggers may use varying grammatical displays, but they generally follow a consistent convention to represent the different parts of speech and linguistic features. Here are some commonly used grammatical displays in POS tagging:

➢ Universal POS Tags: The Universal POS Tagset is a simplified and language-independent tagset that aims to provide a standard set of labels for parts of speech across different languages. Some common universal POS tags include:

- NOUN: noun
- VERB: verb
- ADJ: adjective
- ADV: adverb
- PRON: pronoun
- DET: determiner
- ADP: adposition (preposition or postposition)
- CONJ: conjunction
- NUM: numeral
- PUNCT: punctuation

Penn Treebank POS Tags: The Penn Treebank tagset is a widely used tagset for English, commonly employed in many NLP applications.

It provides a more detailed set of labels to represent different parts of speech and grammatical features.

Examples of Penn Treebank POS tags include:

- NN: singular noun
- NNS: plural noun
- VB: base form verb
- VBD: past tense verb
- JJ: adjective
- RB: adverb
- PRP: personal pronoun
- IN: preposition or subordinating conjunction
- DT: determiner
- CC: coordinating conjunction
- CD: cardinal number
- .: punctuation mark

➢ These are just a few examples, and there may be additional tags specific to certain tagsets or languages. It's essential to refer to the documentation or guidelines of the specific POS tagset or tagger you are using to understand the complete set of labels and their meanings.

➢ POS tagsets and taggers can vary depending on the language, corpus, or specific application. It's important to choose an appropriate tagset that aligns with your task and data, and to ensure consistency in the interpretation and usage of the tags within your NLP pipeline of application.

CHAPTER 7

TESTING

7.1 SYSTEM TESTING:

System testing in the context of parts of speech (POS) tagging and stop word tagging involves evaluating the performance and accuracy of the tagging system on a representative dataset. It aims to assess the system's ability to correctly assign POS tags and identify stop words in various contexts.

Here's a general outline of how you can perform system testing for POS tagging and stop word tagging:

1. Test Data Preparation: Gather a diverse and representative dataset that covers different text genres, domains, and sentence structures. The dataset should include sentences or texts with varying levels of complexity. Ensure that the dataset contains labeled POS tags and a list of expected stop words for evaluation purposes.

2. Evaluation Metrics: Define appropriate evaluation metrics to measure the performance of your POS tagging and stop word tagging system. Common evaluation metrics for POS tagging include accuracy, precision, recall, and F1 score. For stop word tagging, accuracy and F1 score can be used.

3. Test Execution: Apply your POS tagging and stop word tagging system to the test dataset. Obtain the predicted POS tags and stop word labels for each sentence or text in the dataset.

4. Error Analysis: Analyze the errors made by the system to gain insights into its strengths and weaknesses. Identify common error patterns, such as incorrect POS tags for specific word categories or missed stop words in certain contexts. This analysis can help you identify areas for improvement in your system or data.

5.Iterative Improvement: Refine and enhance your POS tagging and stop word tagging system based on the error analysis and evaluation results. This can involve adjusting the algorithms, incorporating additional training data, or modifying the feature engineering techniques.

7.2 TYPES OF TESTING:

7.2.1 UNIT TESTING:

Unit testing is an important practice in software development that involves testing individual components or units of code to ensure they function correctly. When it comes to parts of speech (POS) tagging and stop word tagging, unit testing can help verify the accuracy and reliability of the tagging algorithms and implementations. Here's an outline of how you can perform unit testing for these tasks:

1. Identify Test Cases: Begin by identifying different test cases that cover a range of scenarios. For POS tagging, consider sentences with various grammatical structures, verb tenses, noun forms, etc. For stop word tagging, include sentences with different stop words and variations in word order.

2. Define Expected Outputs: For each test case, determine the expected POS tags or stop word removal output based on your tagging algorithm's behavior and the specific rules or criteria you're using. These expected outputs will serve as the basis for comparison during testing.

3. Implement Test Functions: Write test functions using a testing framework or library of your choice, such as `unittest` in Python or any other suitable framework for your programming language.

4. Test POS Tagging: In each test function for POS tagging, provide input sentences and compare the actual POS tags generated by your tagging algorithm with the expected outputs. Assertions can be used to validate if the generated tags match the expected tags.

5. Test Stop Word Tagging: For stop word tagging, create test functions that pass sentences to the stop word removal algorithm. Verify that the algorithm correctly removes the stop words and returns the expected filtered output.

7. Update and Expand Tests: Regularly update and expand your test suite as you make modifications or improvements to your POS tagging and stop word tagging implementations. Add new test cases to cover additional scenarios, edge cases, or potential corner cases.

7.2.2 INTEGRATION TESTING:

- ➢ Integration testing for parts of speech and stop words tagging can be done by creating a set of test cases that cover the following:
- ➢ The correct identification of parts of speech for all words in the test data.
- ➢ The correct removal of stop words from the test data.
- ➢ The correct handling of words that are both stop words and parts of speech (for example, the word "the" is both a determiner and a stop word).
- ➢ The test cases should be created using a variety of different types of text, including news articles, blog posts, and social media posts. The test cases should also cover a variety of different grammatical structures.
- ➢ The integration tests can be automated using a variety of different tools, such as JUnit or PyUnit. The automated tests can be run on a regular basis to ensure that the parts of speech and stop words tagging algorithms are working correctly.
- ➢ Here are some examples of integration tests for parts of speech and stop words tagging:
- ➢ A test case that checks that the correct part of speech is assigned to the word "the" in the sentence "The dog ran across the street."
- ➢ A test case that checks that the word "the" is removed from the sentence "The dog ran across the street."
- ➢ A test case that checks that the correct part of speech is assigned to the word "ran" in the sentence "The dog ran across the street."

7.2.3 ACCEPTANCE TESTING:

- Acceptance testing for parts of speech (POS) and stop words tagging can be done by creating a set of test cases that cover a variety of different scenarios. For example, some test cases might include:
    - Testing the accuracy of POS tagging for a variety of different word types, such as nouns, verbs, adjectives, and adverbs.
    - Testing the accuracy of stop words tagging for a variety of different stop words.
    - Testing the handling of compound words and phrases.
    - Testing the handling of negation.
    - Testing the handling of ambiguity.
- The test cases should be created using a variety of different types of text, such as news articles, blog posts, and social media posts. This will help to ensure that the POS and stop words tagging algorithms are able to handle a variety of different writing styles.
- The results of the acceptance testing should be reviewed to identify any areas where the POS and stop words tagging algorithms are not performing as expected. These areas can then be targeted for improvement.
- Here are some additional tips for acceptance testing for POS and stop words tagging:
    - Use a variety of different datasets to test the algorithms.
    - Use a variety of different word types to test the accuracy of POS tagging.
    - Use a variety of different stop words to test the accuracy of stop words tagging.
    - Use a variety of different writing styles to test the algorithms.
    - Review the results of the acceptance testing carefully to identify any areas where the algorithms are not performing as expecte

CHAPTER 8

DEPLOYMENT

8.1 DEPLOYMENT

The deployment process for this project involves containerizing the application, setting up infrastructure and networking configurations, deploying databases and backend services, deploying the frontend application, implementing security measures, setting up monitoring and logging, establishing continuous integration and closely monitoring the application's performance.

8.2 SAMPLE OUTPUT

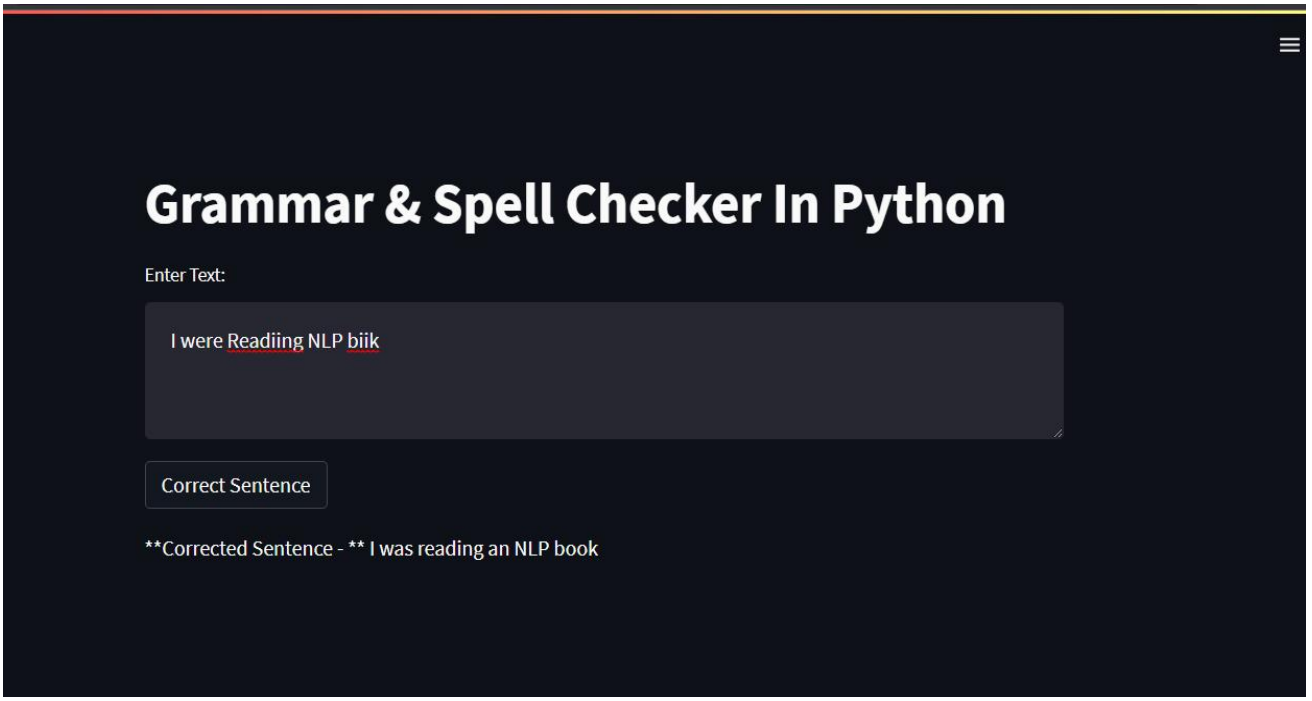

FIG : 8.2.1 Sample Output

7.5 RESULT

The result of this project using Python would be a powerful language processing tool that provides advanced grammar checking, spell checking, and writing enhancement features. It leverages Python's extensive libraries and natural language processing capabilities to analyze text, identify grammatical errors, suggest corrections, and offer writing suggestions to improve the overall clarity and quality of written content.

CHAPTER 9

CONCLUSION & FUTURE SCOPE

9.1 CONCLUSION:

This review paper presents a comprehensive assessment of the part of speech tagging approaches based on the deep learning (DL) and machine learning (ML) methods to provide interested and new researchers with up-to-date knowledge, recent researcher's inclinations, and advancement of the arena, a systematic approach is followed to prioritize and select important research articles in the field of artificial intelligence-based POS tagging. At the outset, the theoretical concept of NLP and POS tagging and its various POS tagging approaches are explained comprehensively based on the reviewed research articles. Based on this review, the recent development of research shows the use of deep learning (DL) oriented methodologies improves the efficiency and effectiveness of POS tagging in terms of accuracy and reduction in false-positive rate. Almost 68% of the proposed POS tagging solutions were deep learning (DL) based methods, with LSTM, RNN, and BiLSTM being the three topmost frequently used DL algorithms. The remaining 20% and 12% of proposed POS tagging models are machine learning (ML) and Hybrid approaches, respectively.

9.2 FUTURE SCOPE :

Parts of speech (POS) tagging and stop words tagging are fundamental tasks in natural language processing (NLP) that have seen significant advancements over the years. While these tasks have reached a high level of accuracy and performance, there are still several areas for future exploration and improvement.

Here are some potential future directions for POS and stop words tagging:

1. **Enhanced Contextual Understanding:**

Current POS tagging and stop words tagging models often treat words in isolation, without considering the broader context. Future research could focus on developing models that incorporate more contextual information, such as neighboring words or sentence-level context, to improve accuracy and handle ambiguous cases more effectively.

2. **Handling Ambiguity and Polysemy:**

Words in natural language can have multiple meanings and can serve different parts of speech depending on the context. Future work could explore techniques to better handle ambiguity and polysemy in POS tagging, leveraging semantic information or incorporating context-aware embeddings to disambiguate word meanings accurately.

3. **Multilingual and Cross-lingual Tagging:**

While POS tagging and stop words tagging have been extensively studied for English, there is a need for robust and accurate models in other languages. Future research can focus on developing multilingual or cross-lingual approaches that can handle various languages effectively, enabling broader applicability and supporting diverse NLP tasks.

4. **Fine-grained Tagsets:**

While existing POS tagsets provide a general categorization of words, there is room for more fine-grained tagsets that capture specific syntactic and semantic nuances. Fine-grained tagsets can enhance the expressive power of POS tagging and enable more detailed analysis and downstream applications.

5. **Domain-specific Tagging:**

Current POS and stop words taggers are trained on general-purpose datasets, which might not capture domain-specific language patterns accurately. improving the performance and relevance of tagging systems in specialized domains like medical, legal, or scientific texts.

6. **Active Learning and Semi-supervised Approaches:**

Collecting annotated data for POS and stop words tagging can be time-consuming and expensive. Future work can investigate active learning and semi-supervised learning approaches to make better use of limited labeled data and reduce the annotation effort, while still achieving high-quality tagging results.

7. **Real-time and Low-resource Settings:**

There is a growing need for POS and stop words tagging in real-time and low-resource settings, such as chatbots or resource-constrained devices. Future research can focus on developing lightweight and efficient models that can perform accurate tagging with low computational requirements.

These are just a few potential areas for future development and improvement in parts of speech and stop words tagging. Continued research and innovation in these areas will contribute to advancing the accuracy, efficiency, and applicability of these foundational NLP tasks.

APPENDIX

INSTALLATION:

```
! pip install gingerit
```

USAGE:

```
from gingerit.gingerit import GingerIt
text = 'The smelt of fliwers bring back memories.'
parser = GingerIt()
parser.parse(text)
```

PYTHON CODE:

```
import cloudscraper
API_KEY = "6ae0c3a0-afdc-4532-a810-82ded0054236"

class GingerIt:
    def __init__(self):
        self.url = URL
        self.api_key = API_KEY
        self.api_version = "2.0"
        self.lang = "US"

    def parse(self, text):
        session = cloudscraper.create_scraper()
        request = session.get(
            self.url,
            params={
                "lang": self.lang,
                "apiKey": self.api_key,
                "clientVersion": self.api_version,
                "text": text,
            },
            verify=True,
        )
        data = request.json()
        return self._process_data(text, data)
    @staticmethod
    def _change_char(original_text, from_position, to_position, change_with):
```

```
        return "{}{}{}".format(
            original_text[:from_position], change_with, original_text[to_position + 1 :]
        )
    def _process_data(self, text, data):
        result = text
        corrections = []

        for suggestion in reversed(data["Corrections"]):
            start = suggestion["From"]
            end = suggestion["To"]

            if suggestion["Suggestions"]:
                suggest = suggestion["Suggestions"][0]
                result = self._change_char(result, start, end, suggest["Text"])

                corrections.append(
                    {
                        "start": start,
                        "text": text[start : end + 1],
                        "correct": suggest.get("Text", None),
                        "definition": suggest.get("Definition", None),
                    }
                )
        return {"text": text, "result": result, "corrections": corrections}
```

MAIN INTERFACE :

```
from gingerit.gingerit import GingerIt
import streamlit as st
st.title('Grammar & Spell Checker In Python')
text = st.text_area("Enter Text:", value='', height=None, max_chars=None, key=None)
parser = GingerIt()
if st.button('Correct Sentence'):
    if text == '':
        st.write('Please enter text for checking')
    else:
        result_dict = parser.parse(text)
        st.markdown('**Corrected Sentence - **' + str(result_dict["result"]))
else: pass
```

REFERENCE

1. "A Fast and Accurate Part-of-Speech Tagger for Arabic" by Nizar Habash (2006)

2. "Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network" by Zhiheng Huang et al. (2015)

3. "Supervised Sequence Labelling with Recurrent Neural Networks" by Alex Graves (2012)

4. "TnT: A Statistical Part-of-Speech Tagger" by Thorsten Brants (2000)

5. "Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments" by Kevin Gimpel et al. (2011)

6. "A Maximum Entropy Approach to Part-of-Speech Tagging" by Adwait Ratnaparkhi (1996)

7. "Part-of-Speech Tagging for English Tweets" by Chris Dyer et al. (2013)

8. "Improving Part-of-Speech Tagging Accuracy for Medical Texts Using Distributional Semantics" by Pierre Zweigenbaum et al. (2014)

9. "Stop Words: A Survey of Research and Applications" by Sujata Ghatak et al. (2021)

10. "Improving Part-of-Speech Tagging for Clinical Texts with Recurrent Neural Networks and Word Embeddings" by Raul Rodriguez-Esteban (2016)

11. "An Empirical Investigation of Part-of-Speech Tagging for Arabic Social Media Texts" by Hassan Sajjad et al. (2013)

12. "A Simple but Tough-to-Beat Baseline for Sentence Splitting" by John S. Justeson and Slava M. Katz (1995)

13. "Adaptive Non-Iterative Part-of-Speech Tagging for Literary Old Icelandic" by Hrafn Loftsson and Eirikur Rögnvaldsson (2008)

14. "DNN Approaches to Large Vocabulary Conversational Speech Recognition" by George Saon et al. (2013)

15. "Using Very Large Corpora for Near-Real-Time Extraction of Linguistic Structure" by Joshua T. Goodman (1999)