Report On

AYNLP: Advanced Yet Simple NLP Toolkit (Library)

Submitted in partial fulfillment of the requirements of the Course Project for Natural Language Processing in Semester VII of Fourth Year Artificial Intelligence & Data Science Engineering

By

Rahul Rammilan Singh (Roll No. 53)

Ankit Vinod Bari (Roll No. 61)

Yash Ravindra Kerkar (Roll No. 63)

Under the guidance

Mr. Raunak Joshi

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(A.Y. 2025-26)

Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

CERTIFICATE

This is to certify that the project entitled "AYNLP: Advanced Yet Simple NLP Toolkit (Library)" is a bonafide work of Rahul Rammilan SinghAnkit Bari (Roll No. 61), Yash Kerkar (Roll No. 63)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VII of Fourth Year Artificial Intelligence and Data Science engineering.

Guide
Mr. Raunak Joshi

Abstract

Natural Language Processing (NLP) is a key part of Artificial Intelligence that helps computers understand and process human language. In this project, we have developed AYNLP (Advanced Yet Simple NLP), a lightweight and easy-to-use Python library for performing basic NLP tasks. AYNLP provides features like tokenization, stopword removal, stemming, lemmatization, part-of-speech tagging, named entity recognition, and sentiment analysis.

The library is designed to be modular and beginner-friendly, making it suitable for educational purposes, research projects, and open-source contributions. By combining multiple NLP functionalities into a single package, AYNLP allows users to quickly analyze and understand text data without dealing with complex setups. This project demonstrates how core NLP techniques can be integrated into a simple and accessible toolkit for students and developers alike.

Table of Contents

Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that focuses on enabling computers to understand, interpret, and generate human language. NLP is used in many real-world applications such as chatbots, virtual assistants, sentiment analysis, and text summarization.

For beginners and students, working with NLP can sometimes be complicated because it requires multiple tools and libraries to perform different tasks. To simplify this, we have developed AYNLP (Advanced Yet Simple NLP), a Python library that combines essential NLP tasks into a single, easy-to-use package.

AYNLP provides functionalities such as tokenization, stopword removal, stemming, lemmatization, part-of-speech tagging, named entity recognition, and sentiment analysis. Users only need to input text, and the library returns results for all these tasks in a structured and readable format.

This project aims to create a lightweight, modular, and beginner-friendly NLP toolkit that can be used for educational purposes, research, and small projects. It also demonstrates how combining multiple NLP techniques into a single package can save time, simplify workflows, and make learning NLP easier.

Additionally, AYNLP is designed to be extensible, meaning that new functionalities can be added easily in the future. This makes it not only a learning tool but also a foundation for more advanced NLP projects. It encourages students and developers to experiment with text processing and understand how different NLP tasks work together.

Finally, this project highlights the importance of creating user-friendly software in AI and NLP. By providing clear outputs and integrating multiple NLP tasks into one function, AYNLP reduces the technical barrier for newcomers. This approach can help students, researchers, and hobbyists explore NLP without worrying about installing and configuring multiple libraries separately.

Problem Statement

Natural Language Processing (NLP) is widely used in modern applications like chatbots, sentiment analysis, and automated text processing. However, for beginners and students, working with NLP can be difficult. This is because performing even basic NLP tasks often requires multiple libraries, complex installations, and knowledge of different APIs.

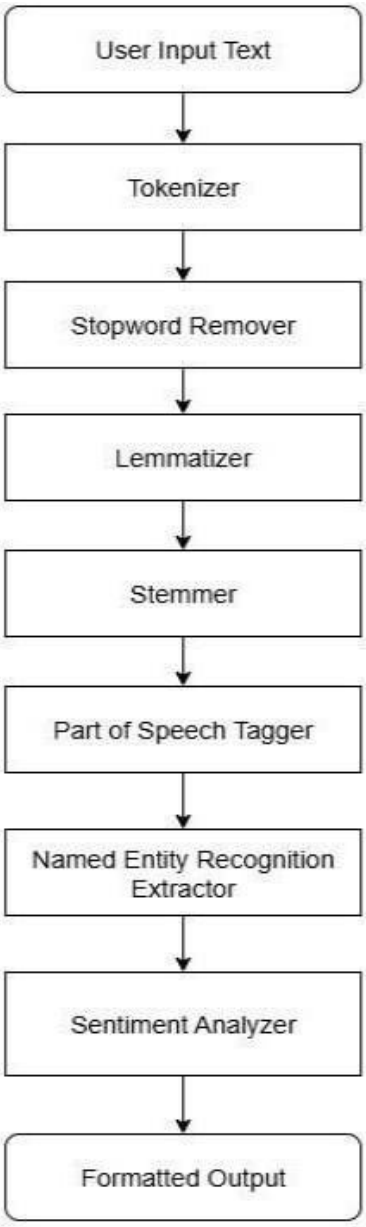
Currently, there is no single, lightweight toolkit that combines all essential NLP tasks—such as tokenization, stopword removal, stemming, lemmatization, part-of-speech tagging, named entity recognition, and sentiment analysis—into a simple, beginner-friendly package. As a result, students and beginners spend a lot of time setting up environments and learning how to use multiple libraries, which slows down learning and experimentation.

The main problem is to create a unified, easy-to-use NLP toolkit that allows users to perform multiple text-processing tasks with minimal setup. Such a toolkit should simplify the learning process, provide clear and structured outputs, and reduce dependency on multiple external libraries.

Our project, AYNLP, addresses this problem by providing a lightweight, modular Python library that integrates core NLP functionalities in one package. It is designed for educational purposes, small projects, and research, making NLP accessible and easier for beginners to understand and use.

3 Proposed System

3.1 Block Diagram



Description

1. **User Input Text**: The user provides a text string that needs to be analyzed.
2. **Tokenizer**: The text is split into individual tokens (words, punctuation, etc.).
3. **Stopword Remover**: Common words like “the”, “is”, “and” are removed to focus on meaningful words.
4. **Stemmer/Lemmatizer**: Words are reduced to their base or root form to standardize the text.
5. **POS Tagger**: Each word is tagged with its grammatical role (noun, verb, adjective, etc.).
6. **NER Extractor**: Named entities like people, locations, or organizations are identified.
7. **Sentiment Analyzer**: The sentiment of the text (positive, negative, neutral) is determined.
8. **Formatted Output**: The final result is displayed in a readable format, like a table.

Working:

The user interacts with the library through a simple function call (e.g., aynlp.analyze(text)). Internally, the library processes the text sequentially through each module. Each component performs its NLP task and passes the processed data to the next component. Finally, all results are compiled into a structured output, making it easy for users to understand the analysis.

3.2 Module Description

1. Tokenizer

- This module splits the input text into individual words or tokens.
- Example: "I am a good boy" → [I, am, a, good, boy]
- It is the first step in almost all NLP pipelines because further analysis works on tokens.

2. Stopword Remover /Filtered Tokens

- Removes common, less meaningful words (stopwords) like "I", "am", "a".
- Example: [I, am, a, good, boy] → [good, boy]
- Helps focus on words that carry more semantic meaning.

3. Lemmatizer

- Converts words to their base or dictionary form (lemma).
- Example: "was" → "be", "yesterdays" → "yesterday"
- Useful for reducing variations of a word to a single form.

4. Stemmer

- Reduces words to their root/stem form by removing suffixes.
- Example: "running" → "run", "fishing" → "fish"
- Helps in text normalization and indexing.

5. POS Tagger (Part-of-Speech Tagger)

- Assigns grammatical categories (noun, verb, adjective, etc.) to each token.
- Example: "I/PRP, am/VBP, a/DT, good/JJ, boy/NN"
- Enables understanding of sentence structure and syntax.

6. Named Entity Recognizer (NER)

- Identifies proper nouns like people, organizations, places, dates, etc.
- Example: "Apple" → recognized as a company/entity
- Helps extract structured information from unstructured text.

7. Sentiment Analyzer

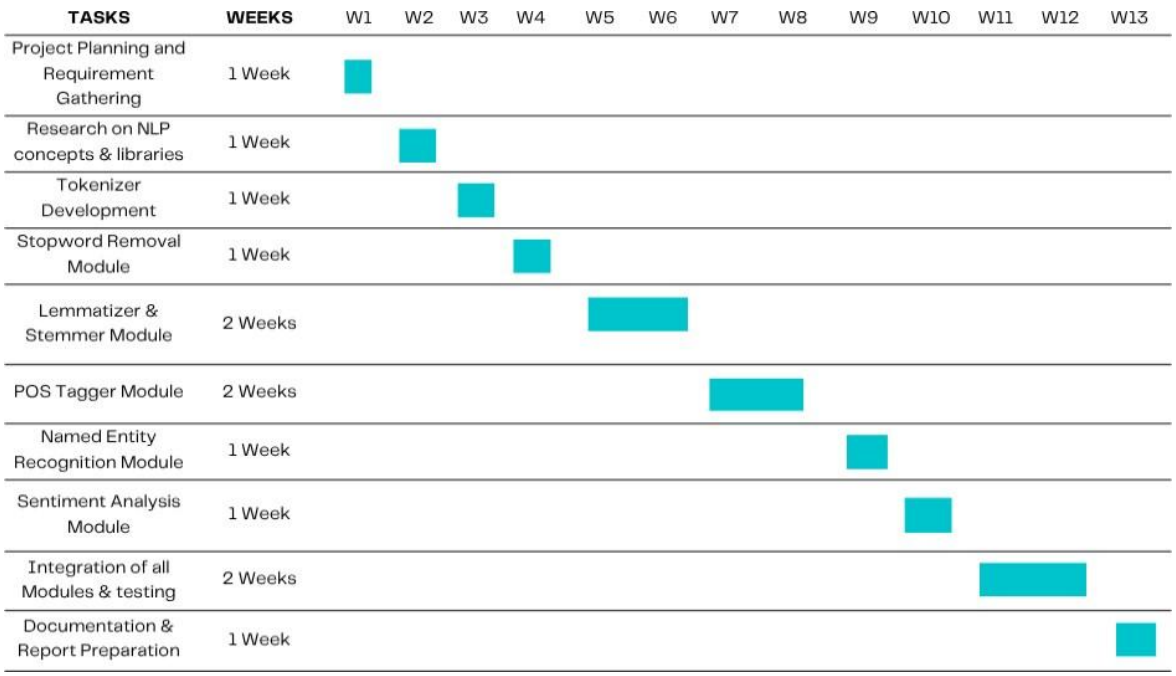
- Determines the sentiment or emotion expressed in the text.
- Example: "I am happy" → Positive, "I am sad" → Negative
- Useful for opinion mining, social media analysis, and customer feedback.

8. Output Formatter / Display Table

- Combines results from all modules into a readable table format with emojis and labels.
- Makes it easy for users to interpret results quickly.

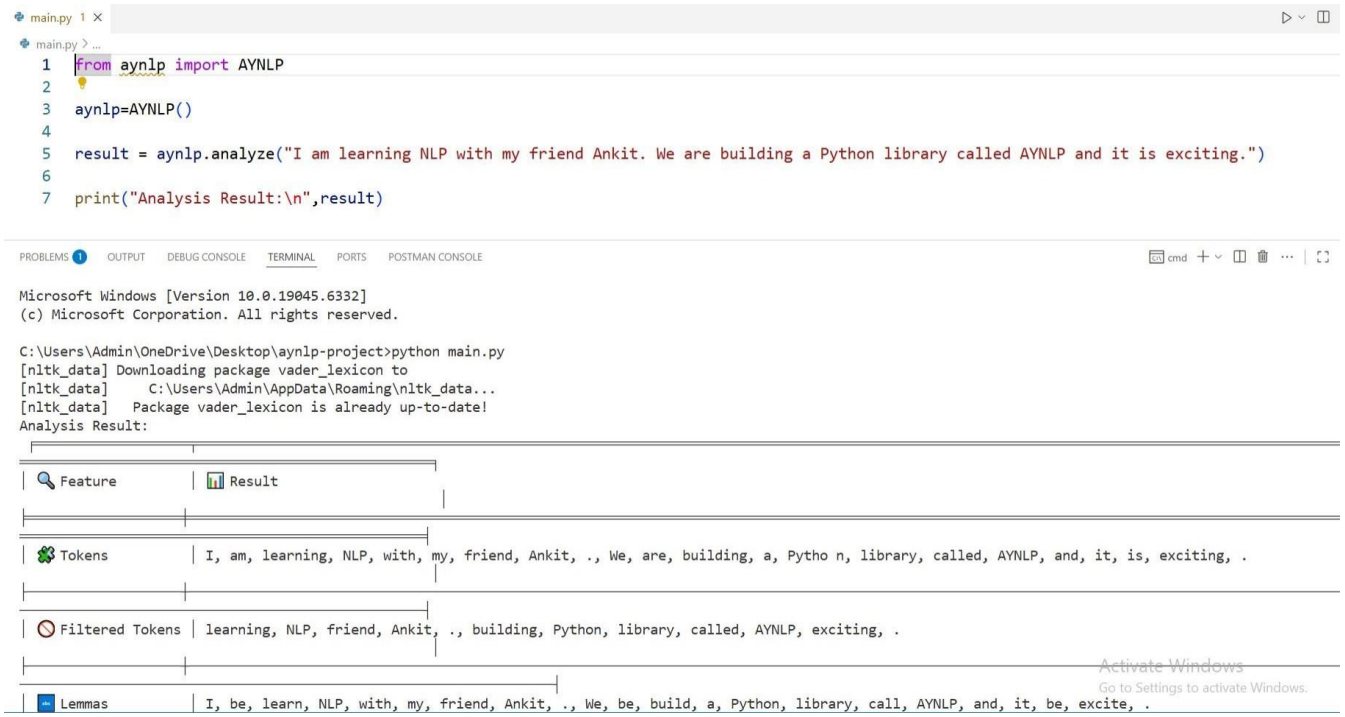
Implementation Plan Details

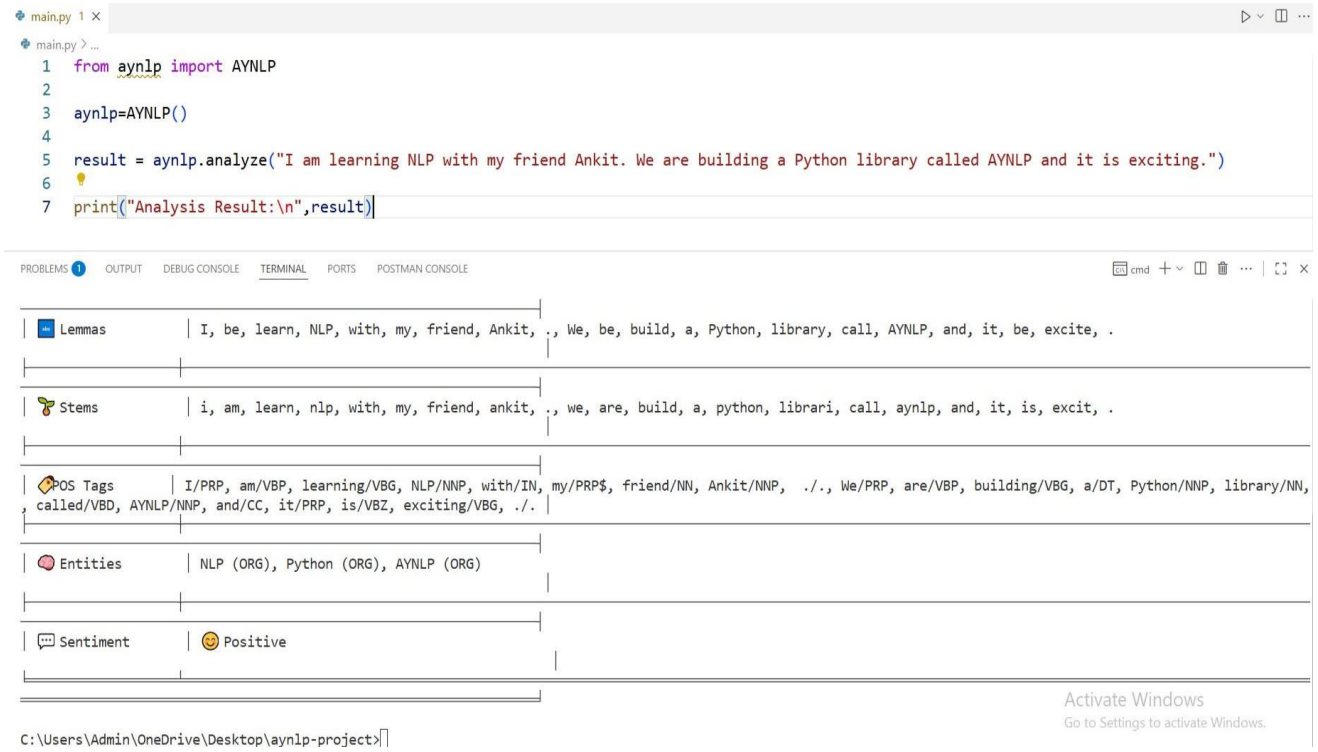
4.1 Gantt Chart

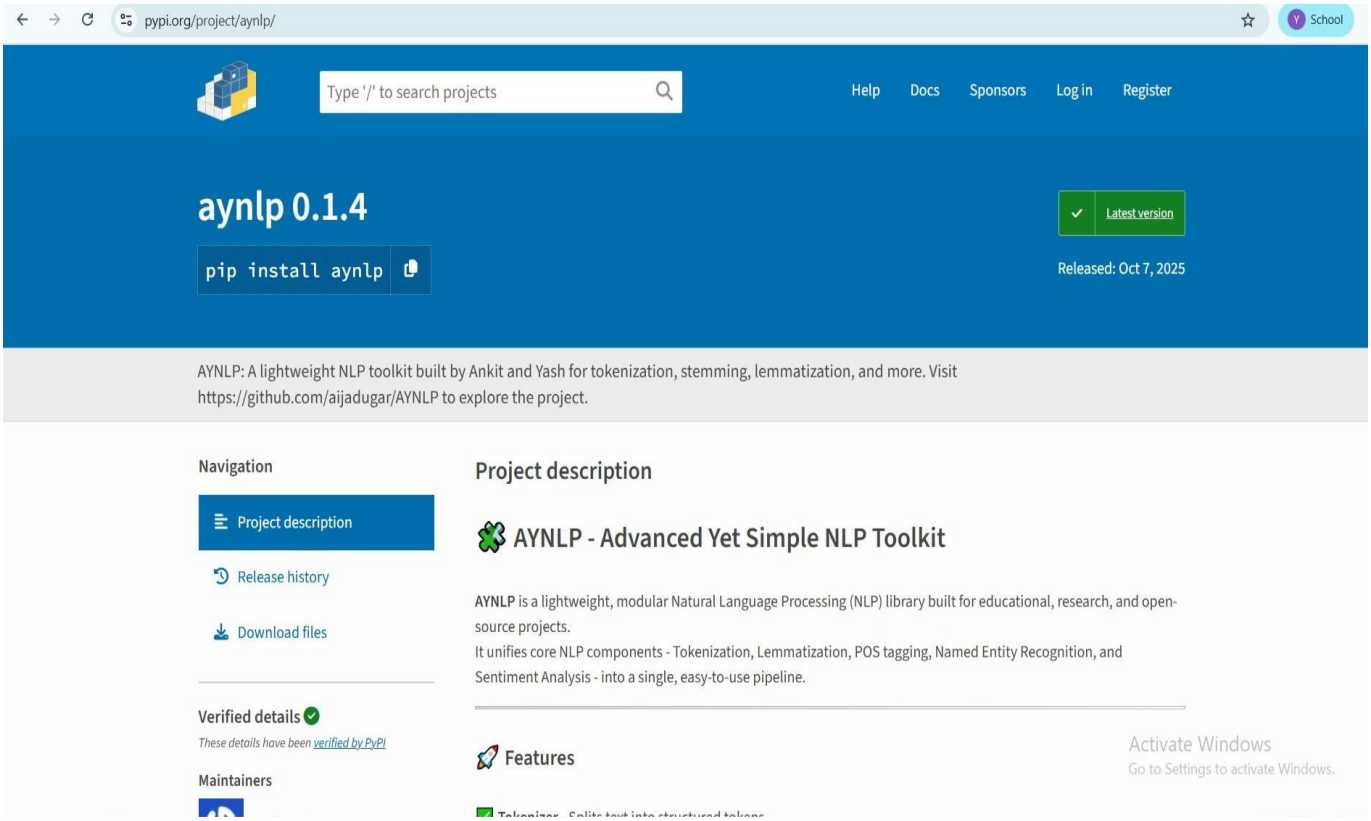


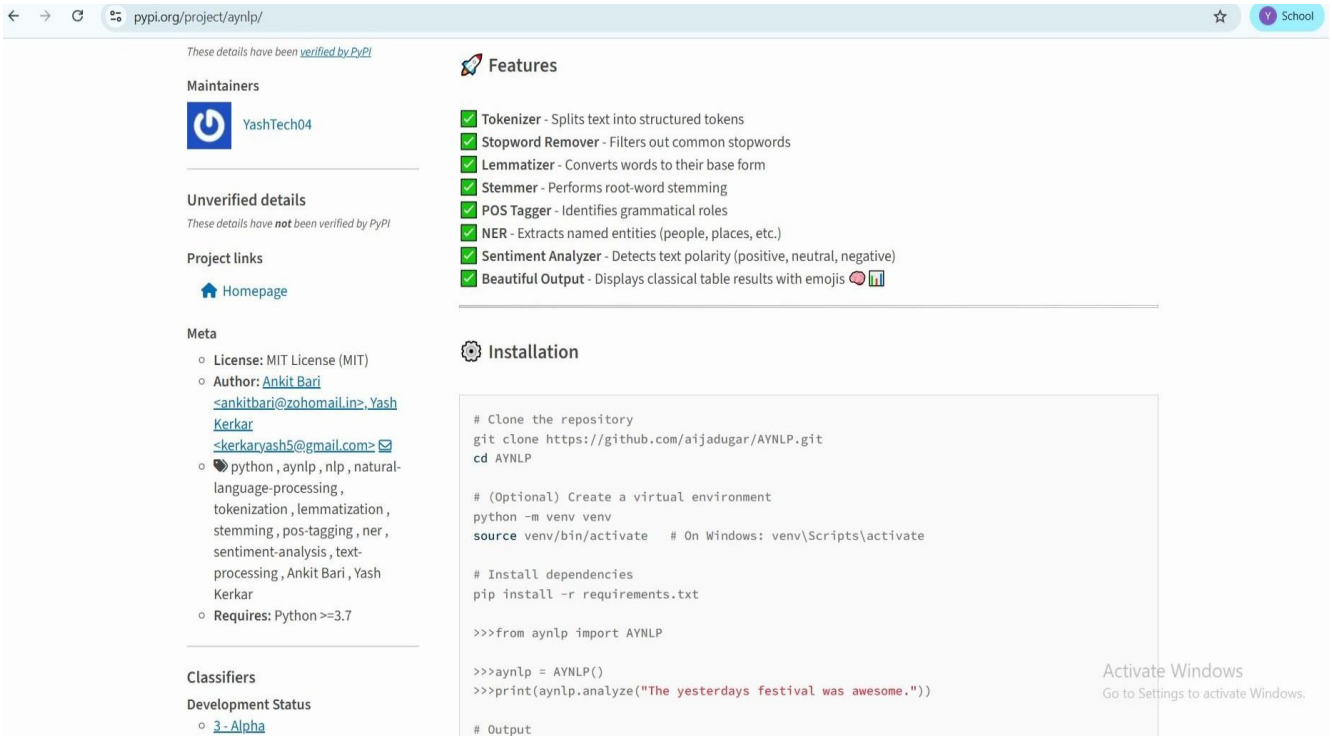
Implementation Result & Analysis

5.1 Screenshots






pypi.org/project/aynlp/
School
Type '/' to search projects
Help
Docs
Sponsors
Log in
Register
aynlp 0.1.4
pip install aynlp
Latest version
Released: Oct 7, 2025
AYNLP: A lightweight NLP toolkit built by Ankit and Yash for tokenization, stemming, lemmatization, and more. Visit https://github.com/aijadugar/AYNLP to explore the project.
Navigation
Project description
Release history
Download files
Verified details
These details have been verified by PyPI
Maintainers
Project description
AYNLP - Advanced Yet Simple NLP Toolkit
AYNLP is a lightweight, modular Natural Language Processing (NLP) library built for educational, research, and open-source projects.
It unifies core NLP components - Tokenization, Lemmatization, POS tagging, Named Entity Recognition, and Sentiment Analysis - into a single, easy-to-use pipeline.
Features
Activate Windows
Go to Settings to activate Windows.


pypi.org/project/aynlp/
School
These details have been verified by PyPI
Maintainers
YashTech04
Unverified details
These details have not been verified by PyPI
Project links
Homepage
Meta
License: MIT License (MIT)
Author: Ankit Bari <ankitbari@zohomail.in>, Yash Kerkar <kerkaryash5@gmail.com>
python , aynlp , nlp , natural-language-processing , tokenization , lemmatization , stemming , pos-tagging , ner , sentiment-analysis , text-processing , Ankit Bari , Yash Kerkar
Requires: Python >=3.7
Classifiers
Development Status
3 - Alpha
Features
Tokenizer - Splits text into structured tokens
Stopword Remover - Filters out common stopwords
Lemmatizer - Converts words to their base form
Stemmer - Performs root-word stemming
POS Tagger - Identifies grammatical roles
NER - Extracts named entities (people, places, etc.)
Sentiment Analyzer - Detects text polarity (positive, neutral, negative)
Beautiful Output - Displays classical table results with emojis
Installation
Clone the repository
git clone https://github.com/aijadugar/AYNLP.git
cd AYNLP
(Optional) Create a virtual environment
python -m venv venv
source venv/bin/activate # On Windows: venv\Scripts\activate
Install dependencies
pip install -r requirements.txt
>>>from aynlp import AYNLP
>>>aynlp = AYNLP()
>>>print(aynlp.analyze("The yesterdays festival was awesome."))
Output
Activate Windows
Go to Settings to activate Windows.

Financial Expense

6.1 Material Expenses

Item	Estimated Cost (INR)
Laptop/PC (already available)	₹0
Internet Charges	₹500
Total	₹500

6.2 Direct Expenses

Item	Estimated Cost (INR)
Software / Libraries (All free)	₹0
Printing / Documentation	₹300
Total	₹300

6.3 Indirect Expenses

Item	Estimated Cost (INR)
Electricity Charges for Computer Usage	₹300
Total	₹300

Conclusion

In this project, we developed AYNLP, a lightweight and easy-to-use Natural Language Processing toolkit. It combines essential NLP tasks like tokenization, lemmatization, stemming, POS tagging, named entity recognition, and sentiment analysis into a single library.

This toolkit is modular, efficient, and suitable for educational, research, and open-source purposes. Through this project, we learned how to structure a Python library, integrate multiple NLP components, and publish it on PyPI for public use.

Overall, AYNLP demonstrates that even complex NLP tasks can be simplified into a user-friendly interface, making it accessible for beginners and developers alike. This project has strengthened our understanding of NLP concepts and Python packaging, and it provides a foundation for future enhancements and contributions.

Code

```
from .tokenizer import Tokenizer
from .lemmatizer import Lemmatizer
from .stemmer import Stemmer
from .stopwords import StopwordRemover
from .pos_tagger import POSTagger
from .ner import NER
from .sentiment import SentimentAnalyzer
from tabulate import tabulate


class AYNLP:
    """Unified NLP pipeline with table output      """

    def __init__(self):
        self.tokenizer = Tokenizer()
        self.stopword_remover = StopwordRemover()
        self.lemmatizer = Lemmatizer()
        self.stemmer = Stemmer()
        self.pos_tagger = POSTagger()
        self.ner = NER()
        self.sentiment_analyzer = SentimentAnalyzer()

    def analyze(self, text):
        tokens = self.tokenizer.tokenize(text)
        filtered_tokens = self.stopword_remover.remove(tokens)
        pos_tags = self.pos_tagger.tag(tokens)
        lemmas = [self.lemmatizer.lemmatize(tok, tag) for tok, tag in pos_tags]
        stems = [self.stemmer.stem(tok) for tok in tokens]
        entities = self.ner.extract_entities(tokens)
        sentiment = self.sentiment_analyzer.analyze(text)
```

```
data = [
    ["☐ Tokens", ", ".join(tokens)],
    ["      Filtered Tokens", ", ".join(filtered_tokens)],
    ["      Lemmas", ", ".join(lemmas)],
    ["      Stems", ", ".join(stems)],
    ["      POS Tags", ", ".join([f"{w}/{t}" for w, t in pos_tags])],
    ["☐ Entities", ", ".join([f"{e} ({t})" for e, t in entities]) if entities else "—"],
    ["      Sentiment",
     "      Positive" if sentiment == "positive"
     else ("      Neutral" if sentiment == "neutral" else "      Negative")],
]

table = tabulate(data, headers=["      Feature", "      Result"], tablefmt="fancy_grid")
return table
```

References

[1] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. Sebastopol, CA: O'Reilly Media, 2009.

[2] Python Software Foundation, "Python Documentation," Python.org. [Online]. Available: https://docs.python.org/3/ .

[3] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008. [Online]. Available: https://www.cs.cornell.edu/home/llee/omsa/omsa.pdf .

[4] "AYNLP: Advanced Yet Simple NLP Toolkit," GitHub repository, 2025. [Online]. Available: https://github.com/aijadugar/AYNLP .

[5] "AYNLP Python Package," PyPI, 2025. [Online]. Available: https://pypi.org/project/aynlp/0.1.4/ .