5/12/2024   
Bahria University   
Department of Computer Science   
   
Natural Language Processing   
CSC 441   
   
   
Project   
Final Deliverable   
   
   
   
   
   
   
   
   
   
   
   
Hannaan Ahmed Jadran 01-134211-106   
Muhammad Ali Momin 01-134211-051   
BS-CS-7B   
   
   
   
   
   
   
   
Submitted to   
Dr Arif ur Rahman   
Head of Department / Professor

Table of Contents   
Abstract ...................................................................................................................................... 2   
1.   
Introduction ......................................................................................................................... 2   
Application Diagram: ............................................................................................................... 3   
2.   
Methodology ........................................................................................................................ 3   
2.1.   
Data Collection ........................................................................................................... 3   
2.2.   
Preprocessing ............................................................................................................. 4   
Preprocessing steps   
............................................................................................................. 4   
2.3.   
Feature Extraction ........................................................................................................ 5   
Technique used:   
.................................................................................................................... 5   
Implementation: .................................................................................................................. 6   
2.4.   
Model Training and Evaluation ....................................................................................... 6   
2.4.1.   
Data Splitting: ...................................................................................................... 6   
2.4.2.   
Model Training ..................................................................................................... 7   
2.4.3.   
Hyperparameter Tuning ........................................................................................ 7   
2.4.4.   
Model Evaluation: ................................................................................................ 7   
2.4.5.   
Final Model Selection ........................................................................................... 7   
2.4.6.   
Test Set Evaluation: ............................................................................................... 7   
2.5.   
Application or Deployment ........................................................................................... 8   
References ................................................................................................................................. 9   
   
   
   
Table of Figures:   
Figure 1: Flow of Operations ............................................................................................................ 3   
Figure 2: Preprocessing Steps ....................................................................................................... 5   
Figure 3: Feature Extraction ............................................................................................................ 6   
Figure 4: Model Training and Evaluation .......................................................................................... 8   
Figure 5: Webpage in it's infancy .................................................................................................... 9

Multilingual Named Entity Extraction System   
Abstract   
Named Entity Recognition (NER) is a crucial aspect of Natural Language Processing (NLP)   
that involves identifying and classifying key information (entities) within a body of text. This   
report outlines a system designed to extract named entities from PDF documents, translate   
them if necessary, and save the processed text in a Word document with highlighted entities.   
This system leverages SpaCy for NER, PyMuPDF for PDF handling, and Google Translate for   
language translation.   
   
1. Introduction   
Named entity recognition (NER) is a fundamental task in natural language processing (NLP) that   
involves identifying and classifying named entities, such as persons, organizations,   
locations, and other relevant entities, within unstructured text data. NER has numerous   
applications, including information extraction, question answering, and knowledge base   
construction.   
In multilingual environments, NER becomes more challenging due to the need to handle texts   
in different languages. This project focuses on developing an NLP system that can extract   
named entities from documents written in Urdu or English. Urdu, being a widely spoken   
language in South Asia, presents unique challenges due to its complex script and linguistic   
properties.   
The proposed system employs a two-step approach for Urdu documents. First, the Urdu text is   
translated into English using machine translation techniques. This step is necessary   
because most state-of-the-art NER models are primarily trained on English data. Once the text   
is in English, a pretrained NER model is applied to identify, and extract named entities from the   
translated text. After the named entities have been extracted from the English text, they are converted   
back into their original Urdu script. This step ensures that the output is consistent with the input   
document's   
language,   
providing   
a   
seamless user experience for Urdu   
speakers. For English documents, the   
system directly applies the NER model to   
the input text, bypassing the translation   
step.   
This project aims to provide a robust and   
efficient solution for named entity extraction   
in both Urdu and English documents,   
enabling   
applications   
such   
as   
information retrieval, text mining, and   
knowledge management in multilingual

environment.   
   
2. Methodology   
   
The methodology section provides an in-depth explanation of each step involved in developing and   
deploying the NER application. This includes the design of the system architecture, data processing, and   
the implementation of key functionalities.   
   
2.1. Preprocessing   
The preprocessing stage serves as a pivotal preparatory step, laying the foundation for   
subsequent text analysis tasks. Its primary goal is to refine raw text data, ensuring its   
compatibility with downstream processes like named entity recognition (NER) and other   
natural language processing (NLP) tasks.   
Preprocessing steps   
   
1. Convert to Plain Text:   
   
Initially, the text undergoes a transformation to strip it of any extraneous elements,   
ensuring it exists in a clean, plain text format devoid of formatting artifacts, such as leading   
or trailing whitespace, or any other non-textual elements that could obfuscate   
analysis.   
   
2. Check Language:   
   
Following the conversion to plain text, an important consideration is determining the   
language of the text. This step is pivotal, especially for multilingual applications, as it allows   
for the application of language-specific preprocessing techniques tailored to the   
linguistic nuances of the text. For instance, distinct preprocessing strategies might be   
employed for Urdu and English text due to differences in syntax, vocabulary, and structure.   
   
3. Preprocess and Clean Data:   
o Once the language is identified, the text is subjected to a series of cleaning   
operations aimed at refining its quality and enhancing its suitability for analysis. This   
multifaceted process encompasses several key operations:   
o Lowercasing: The text is uniformly converted to lowercase to facilitate   
standardization and mitigate issues arising from case discrepancies during   
subsequent analysis.   
o Punctuation and Special Character Removal: Extraneous elements such as   
punctuation marks and special characters are systematically stripped from the text.   
This serves to declutter the data and eliminate non-semantic elements that could   
potentially interfere with analysis.   
o Tokenization: The text is segmented into individual tokens or words, a   
foundational step for subsequent analysis tasks. Tokenization serves to break

down the text into its constituent units, enabling granular analysis at the word level.   
o Stop Word Removal: Stop words, commonly occurring words that contribute   
little to the overall semantic content of the text, are pruned from the data. This helps   
streamline analysis by focusing on substantive content while discarding   
superfluous linguistic elements.   
   
   
   
   
2.2. Feature Extraction   
Figure 2: Preprocessing Steps   
The objective of feature extraction is to transform the preprocessed text data into a   
structured format that can be utilized as input for subsequent analysis tasks. By extracting   
relevant features, we aim to capture the essential information encoded within the text,   
enabling effective analysis and interpretation.   
Technique used:   
   
1. TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization   
   
TF-IDF is a widely used technique for feature extraction in text analysis. It calculates the   
importance of a term within a document relative to a corpus of documents. The TF-IDF   
value increases proportionally to the number of times a term appears in the document   
but is offset by the frequency of the term in the corpus, which helps to adjust for the fact   
that some terms are more common overall.

Implementation:   
   
•   
Implementation of feature extraction techniques typically involves utilizing libraries or   
tools that provide implementations of these techniques. For example:   
o Scikit-learn provides a TFIDF-Vectorizer class for TF-IDF vectorization.   
o Domain-specific feature engineering approaches may require custom   
implementation based on the specific requirements of the analysis task.   
•   
The preprocessed text data serves as input to these feature extraction techniques, and   
the resulting feature representations are used for subsequent analysis tasks such as   
classification, clustering, or information retrieval.   
   
Figure 3: Feature Extraction   
   
   
Application development   
   
System Architecture   
The system is built using a modular approach, where each module is responsible for a specific   
task. The main components include the Flask web application, the NER module, and the   
document processing module. The Flask application serves as the interface between the user and   
the backend processing logic.

Data Ingestion   
The first step in the process is data ingestion. The user uploads a text file through the web   
interface. The Flask application handles this file upload, ensuring that the file is correctly received   
and stored temporarily for processing.   
   
   
Data Cleaning   
Once the file is uploaded, the text content is extracted and passed to the data cleaning module.   
This step involves removing unwanted characters, such as control characters or non-ASCII   
characters, that may interfere with the processing. The text is then standardized to ensure   
consistency in the subsequent analysis.

Named Entity Recognition   
The core of the application is the NER module. This module uses pre-trained machine learning   
models to identify and classify named entities within the text. The models are trained on large   
corpora of labeled data, allowing them to recognize patterns and classify entities with high   
accuracy. The NER process involves tokenizing the text, applying the model, and extracting the   
entities along with their corresponding types.

Post-processing   
After extracting the named entities, the text undergoes post-processing to format the output   
appropriately. This includes reassembling the text with highlighted or annotated named entities,   
which makes it easier for the user to identify the relevant information.   
Document Generation   
The final processed text is then used to generate a Word document. The python-docx library is   
utilized for this purpose. The module creates a new document, adds the processed text, and   
ensures that the formatting is preserved. The document is saved and made available for the user   
to download.   
   
Error Handling and Validation   
Throughout the process, the application incorporates robust error handling and validation   
mechanisms. This ensures that any issues encountered during file upload, text processing, or   
document generation are properly managed and communicated to the user.

Dependencies   
   
1. Flask   
Role: Web Framework Description: Flask is a lightweight WSGI web application framework in   
Python. It is designed to make getting started quick and easy, with the ability to scale up to   
complex applications.   
2. NLTK (Natural Language Toolkit)   
Role: Natural Language Processing Description: NLTK is a leading platform for building Python   
programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora   
and lexical resources, such as WordNet, along with a suite of text processing libraries.   
3. re (Regular Expressions)   
Role: Text Processing Description: The re module provides regular expression matching   
operations similar to those found in Perl. It is used for text searching and manipulation.   
Installation: Included in Python standard library.   
4. python-docx   
Role: Document Generation Description: python-docx is a Python library for creating and   
updating Microsoft Word (.docx) files. It allows for complex document generation and formatting.

Application and Deployment   
The application is deployed on a web server, allowing users to access it via a web browser. The   
Flask application is configured to run on the server, handling incoming requests and serving the   
user interface. The deployment process involves setting up the server environment, installing   
necessary dependencies, and configuring the web server to forward requests to the Flask   
application.   
The deployment ensures that the application is accessible to users, with a focus on scalability and   
reliability. The server setup includes load balancing and failover mechanisms to handle multiple   
user requests simultaneously and ensure continuous availability.   
   
   
Links:   
Github: https://github.com/gibran404/NLP   
Linkedin: https://www.linkedin.com/feed/update/urn:li:activity:7198031288887324673/   
   
References   
Beautiful Soup Documentation - https://www.crummy.com/software/BeautifulSoup/bs4/doc/   
Scikit-learn Documentation - https://scikit-learn.org/stable/   
Eftekhari, A. (2020). Machine learning for named entity recognition: A study on dataset selection   
and model confidence.   
Mahmood, A., Hussain, F., Shafiq, H. M., C Siddiqui, E. (2010, October). Machine translation   
between English and Urdu. In 2010 International Conference on Arabic and Multilingual   
Information Processing (pp. 105-114). IEEE.   
Leaman, R., Islamaj Doğan, R., C Lu, Z. (2020). DNorm: disease name normalization with   
pairwise vector space mappings. Journal of the American Medical Informatics Association,   
27(Supplement\_1), 49-55.   
Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., C Dyer, C. (2016). Neural   
architectures for named entity recognition.

Riza Batista-Navarro, R. T., Rak, R., C Ananiadou, S. (2015, July). Analysis of live multi- domain   
machine translation for multilingual named entity recognition. In Proceedings of the 11th   
International Conference on Computational Semantics (pp. 68-76).