# Summer of code - 2024

**Project Name:**
ParseCraft: Intelligent Resume Parsing Engine

**Project ID**: 122

**Mentors:**
*Mentor 1 Rishu Kuamr*
*Mentor 2 Ayush Jadia*
*Mentor 3 Vidyanand Kumar*

**Submission Deadline:**
*16/06/2024*

**Assignments**

## Assignment 1: Text Preprocessing

**Objective:** Understand the basics of text preprocessing.

**Tasks:**

1. **Case Normalisation:** Convert all text to lowercase.
2. **Handling HTML Tags & URIs:** Use Regex to remove HTMl tags and regex
3. **Removing punctuations:** Remove characters like ?, !, ., etc..
4. **Stop Words Removal:** Remove common stop words from the tokenized text.
5. **Chatwords Handlings:** Handle frequently used chat words by replacing it with actual sentence
6. **Emoji handling:**
7. **Tokenization:** Try different ways for tokenization learn from videos.
8. **Stemming and Lemmatization:** Implement both stemming and lemmatization on the text.

**Data set:**
[https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data](https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data)

**Note: Use only 10,000 rows from the above data set.**

**Resources:**

- NLTK or SpaCy libraries in Python.

- https://www.youtube.com/watch?v=6C0sLtw5ctc&list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX&index=3
- https://www.youtube.com/watch?v=IK9gx4q_vfI&list=PLeo1K3hjS3uuvuAXhYjV2IMEShq2UYSwX&index=3.

---

## **Assignment 2: Text Representation**(This assignment is the following of assignment-1)

**Objective:** Extract features from text.

**Problem 1: Find out the number of words in the entire corpus and also the total number of unique words (vocabulary) using just Python**

1. **Count Words:** Write a script to count the total number of words in the corpus.
2. **Unique Words:** Find the total number of unique words (vocabulary) in the corpus.

**Problem 2: Apply One-Hot Encoding**

1. **One-Hot Encoding:** Convert the preprocessed text into one-hot encoded vectors.

**Problem 3: Apply Bag of Words and find the vocabulary and the frequency of each word**

1. **Bag of Words:** Implement a Bag of Words model to convert text into numerical features.
2. **Vocabulary and Frequency:** Find the vocabulary and the frequency of each word in the corpus.

**Problem 4: Apply Bag of Bi-gram and Bag of Tri-gram and write down your observations about the dimensionality of the vocabulary**

1. **Bag of N-grams:** Implement Bag of Bi-gram and Trigram models.
2. **Observations:** Analyse and write down your observations about the dimensionality of the vocabulary for each model.

**Problem 5: Apply TF-IDF and find out the IDF scores of words, also find out the vocabulary**

1. **TF-IDF:** Implement TF-IDF (Term Frequency-Inverse Document Frequency) to represent text data.
2. **IDF Scores and Vocabulary:** Find the IDF scores of words and the vocabulary.

**Data set:**
**https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data**

**Note: Use only 10,000 rows from the above data set.**

**Resources:**

- Scikit-learn for One-Hot Encoding, Bag of Words, and TF-IDF.
- NLTK or SpaCy for text preprocessing.
- https://www.youtube.com/watch?v=vo6gQz5lYRI&list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX&index=4

---

## Assignment 3: Word2Vec Implementation

**Objective:** Understand and implement Word2Vec for word embeddings.

**Problem 1: Data Preparation**(Assignment-1)

1. **Dataset Selection:** Choose a text dataset suitable for training Word2Vec models.
2. **Data Cleaning:** Preprocess the text data (tokenization, stop word removal, etc.).

**Problem 2: Training Word2Vec Model**

1. **Word2Vec Basics:** Understand the concepts of Word2Vec, including Continuous Bag of Words (CBOW) and Skip-Gram models.
2. **Model Training:** Use the Gensim library to train a Word2Vec model on the preprocessed text data.
   - Train using both CBOW and Skip-Gram models.
   - Experiment with different parameters such as vector size, window size, and min_count.

**Problem 3: Exploring Word Embeddings**

1. **Vector Representation:** Extract the vector representation of words using the trained Word2Vec model.
2. **Similar Words:** Find and display the most similar words for a given set of words.
3. **Word Analogies:** Perform word analogy tasks (e.g., King - Man + Woman = Queen).

**Problem 4: Visualization of Word Embeddings**

1. **Dimensionality Reduction:** Use techniques like PCA or t-SNE to reduce the dimensions of word vectors.
2. **Plotting:** Visualize the word vectors in a 2D or 3D space using matplotlib or any other visualization library.

**Resources:**

- Gensim Documentation for Word2Vec implementation.
- https://www.youtube.com/watch?v=DDfLc5AHoJI&list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX&index=5
- Visualization libraries such as matplotlib and seaborn.

---

## Assignment 4: Text Classification

**Objective:** Build and evaluate text classification models.

**Problem 1: Data Preparation**(Assignment-1**)**

1. **Dataset Selection:** Choose a dataset suitable for text classification (e.g., sentiment analysis, spam detection).
2. **Data Cleaning:** Preprocess the text data (tokenization, stop word removal, etc.).

**Problem 2: Feature Extraction**(Assignment-2)

1. **Bag of Words:** Implement Bag of Words and transform the text data.
2. **TF-IDF:** Implement TF-IDF and transform the text data.
3. **Word Embeddings:** Use pre-trained word embeddings (e.g., Word2Vec, GloVe) to represent text data.

**Problem 3: Model Building**

1. **Baseline Model:** Build a baseline model using a simple algorithm (e.g., Logistic Regression).
2. **Advanced Models:** Build advanced models using algorithms like:
   - Naive Bayes
   - Support Vector Machines (SVM)
   - Random Forest
   - Gradient Boosting

**Problem 4: Model Evaluation**

1. **Metrics:** Evaluate the models using appropriate metrics (e.g., accuracy, precision, recall, F1-score).
2. **Confusion Matrix:** Generate and interpret confusion matrices for the models.
3. **Cross-Validation:** Perform cross-validation to ensure robustness of the models.

**Problem 5: Model Tuning**

1. **Hyperparameter Tuning:** Use techniques like Grid Search or Random Search to tune hyperparameters.
2. **Feature Selection:** Experiment with feature selection techniques to improve model performance.

### Problem 6: Model Comparison

1. **Performance Comparison:** Compare the performance of different models and document your findings.
2. **Final Model Selection:** Select the best-performing model and justify your choice.

**Resources:**

- Scikit-learn for various classification algorithms and evaluation metrics.
- NLTK or SpaCy for text preprocessing.
- https://www.youtube.com/watch?v=Qbd7U9F0QQ8&list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX&index=6

---

## Assignment 5: POS Tagging

**Objective:** Understand and implement Part-of-Speech (POS) tagging for text data.

### Problem 1: Data Preparation(Assignment-1)

1. **Dataset Selection:** Choose a text dataset suitable for POS tagging.
2. **Data Cleaning:** Preprocess the text data (tokenization, etc.).

### Problem 2: POS Tagging using NLTK

1. **POS Tagging Basics:** Understand the concept of POS tagging and its importance in NLP.
2. **NLTK POS Tagging:** Use the NLTK library to perform POS tagging on the text data.
3. **Explore Tags:** Extract and list all unique POS tags from the dataset.

### Problem 3: Custom POS Tagging

1. **Custom POS Tagger:** Build a simple custom POS tagger using a rule-based approach or a machine learning model.
2. **Training and Evaluation:** Train the custom POS tagger on a labeled dataset and evaluate its performance.

### Problem 4: Advanced POS Tagging using SpaCy

1. **SpaCy POS Tagging:** Use the SpaCy library to perform POS tagging on the text data.
2. **Comparison:** Compare the POS tagging results from NLTK and SpaCy.
3. **Accuracy Evaluation:** Evaluate the accuracy of SpaCy’s POS tagger on a test dataset.

### Problem 5: Application of POS Tagging

1. **Syntax-based Analysis:** Use POS tagging results for syntax-based text analysis, such as noun phrase extraction or verb identification.
2. **Feature Extraction:** Use POS tags as features for other NLP tasks like text classification or named entity recognition.

**Data:** Any one sentence that you want

**Resources:**

- NLTK Documentation for POS tagging.
- SpaCy Documentation for advanced POS tagging.
- https://www.youtube.com/watch?v=269IGagoJfs&list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX&index=7

---