**TeamName:BusyBees**

**Deep Learning 2/Section 1**

**Understanding notes:**

**Overview:**

* **Objective**: Fine-tune a language model to generate full blog articles from short summaries.
* **Dataset**: We will need a dataset of articles and their corresponding summaries. We can use a dataset from Kaggle (such as the Medium articles dataset).

**Step-by-Step Breakdown:**

1. **Data Collection**:
   * The dataset includes articles with titles, URLs, and other metadata. The key fields for this task are the **article content** and **summaries**.
2. **Extractive Summarization**:
   * **Goal**: Create summaries by extracting the most important sentences from each article (this is known as "extractive summarization").

**Steps to create an extractive summary**:

* + **Sentence Tokenization**: Break the article into individual sentences using sent\_tokenize() from NLTK.
  + **Text Preprocessing**: Clean up each sentence (convert to lowercase, remove punctuation).
  + **Stopword Removal**: Remove common words (e.g., "the", "is", "and") using NLTK’s list of stopwords.
  + **Word Embeddings**: Generate word embeddings using Word2Vec. These embeddings capture the meaning and relationships between words.
  + **Sentence Embeddings**: For each sentence, create an embedding by averaging the embeddings of all words in that sentence.
  + **Similarity Calculation**: Calculate the cosine similarity between pairs of sentence embeddings to find how similar the sentences are.
  + **Graph Representation**: Represent sentences as nodes in a graph, with cosine similarity scores as edges between them.
  + **PageRank Algorithm**: Use the PageRank algorithm to rank sentences based on their importance.
  + **Top Sentence Selection**: Select the most important sentences (based on PageRank scores) to include in the summary.
  + **Summary Generation**: Combine the selected sentences to form the extractive summary.

1. **Dataset Creation**:
   * Run this extractive summarization process on 1000 articles from the dataset. Store the article content and generated summaries in a new CSV file.
2. **Fine-Tuning the Language Model**:
   * **Input**: The summary of an article.
   * **Output**: The model generates the full content of the blog.
   * Fine-tune the model using this dataset of summaries (input) and full articles (output).

The idea is to train the model to generate blog content based on a few sentences summarizing the article, making it useful for content generation applications.