Project Title: NLP Named Entity Recognition (NER) Using BERT

Week 1: Introduction to NER and BERT

- By Hareetima Sonkar

Project Overview

This week, I focused on laying the foundation for the project titled **"NLP Named Entity Recognition (NER) Using BERT (Bidirectional Encoder Representation from Transformers)"**. The goal was to familiarize myself with the basics of NER and BERT, set up the required environment, and explore sample datasets.

Objectives for Week 1

1. Understand the basics of Named Entity Recognition (NER) and its applications.
2. Learn about the architecture and advantages of the BERT model for NLP tasks.
3. Set up the environment by installing necessary tools and libraries.
4. Explore sample datasets used for NER tasks (e.g., CoNLL-2003, OntoNotes).

Tasks Completed

1. Understanding Named Entity Recognition and Its Applications:

- **Named Entity Recognition (NER):**
  NER is an NLP technique to identify and classify entities in text into predefined categories like names, organizations, locations, dates, and more. It extracts structured information from unstructured text, enabling applications like text summarization, question answering, and knowledge graph construction.
  Steps in NER:
    - Detect entities in the text.
    - Classify entities into categories.
  Applications:
  NER is widely used in domains like question answering, information retrieval, and machine translation. It also enhances the precision of tasks like part-of-speech tagging and parsing.
    Ambiguity in NER
    - Classification can be ambiguous for computers. Examples:
    - *"England"* as an organization (*won the World Cup*) vs. a location (*happened in England*).
    - *"Washington"* as a location (*capital of the US*) vs. a person (*first US president*).
- Researched practical applications, including:
  - Information retrieval.
  - Customer feedback analysis.
  - Document classification.
  - Google search

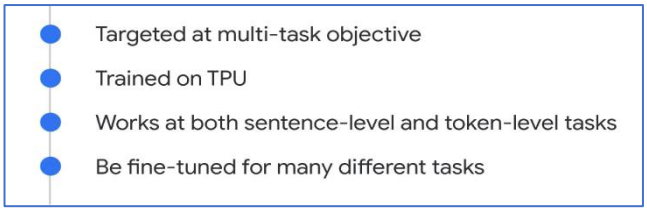
2. Learning About BERT Architecture:

- **Bidirectional Approach of BERT**

Traditional language models process text sequentially, either from left to right or right to left. This method limits the model's awareness to the immediate context preceding the target word. BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

Example: "The bank is situated on the _____ of the river."

In a unidirectional model, the understanding of the blank would heavily depend on the preceding words, and the model might struggle to discern whether "bank" refers to a financial institution or the side of the river.
BERT, being bidirectional, simultaneously considers both the left ("The bank is situated on the") and right context ("of the river"), enabling a more nuanced understanding. It comprehends that the missing word is likely related to the geographical location of the bank, demonstrating the contextual richness that the bidirectional approach brings.



BERT versions

	$BERT_{BASE}$	$BERT_{LARGE}$	Transformer
Layers	12	24	6
Feedforward networks (hidden units)	768	1024	512
Attention heads	12	16	8

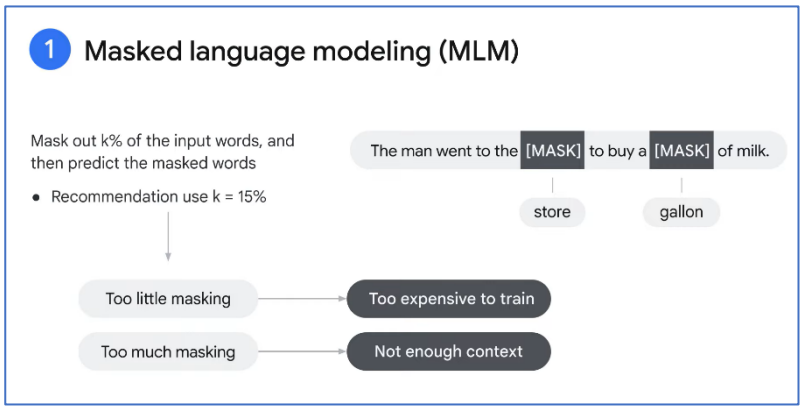
- **BERT addresses bidirectional approach with two innovative training strategies:**

1. **Masked Language Model (MLM)**
2. **Next Sentence Prediction (NSP)**

1. Masked Language Model (MLM)

In simple terms:

1. **Masking Words:** BERT hides about 15% of words in a sentence and replaces them with [MASK].
2. **Guessing Hidden Words:** BERT guesses the hidden words by understanding the context from surrounding words.
3. **How BERT Learns:**
   - BERT uses a special layer to predict the hidden words and checks how close its guesses are.
   - It outputs probabilities like, "I think this word is X, with Y% certainty."



4. **Focus on Hidden Words:** BERT prioritizes learning from the hidden words, which helps it better understand context and meaning.

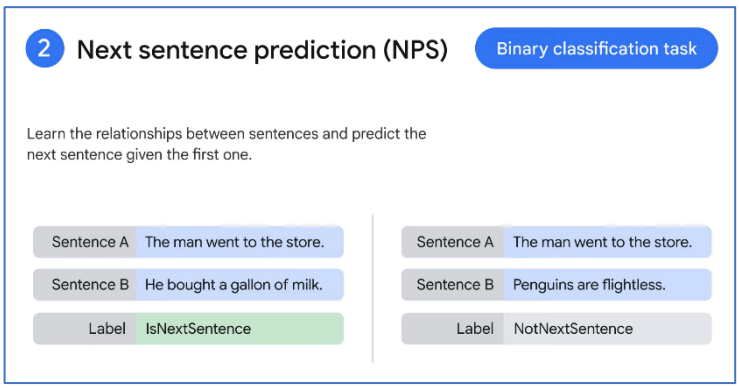
In technical terms:

1. **Classification Layer:** A layer on top of BERT predicts the masked words.
2. **Alignment with Vocabulary:** Predictions are transformed into vocabulary space using the embedding matrix.
3. **Probability Calculation:** SoftMax generates probabilities for all words in the vocabulary for each masked position.
4. **Loss Function:** Only predictions for masked words are penalized, ignoring non-masked words.
5. **Slower Convergence:** BERT trains slower because it focuses only on masked words, but this improves its ability to understand context.

2. Next Sentence Prediction (NSP)

In simple terms:

1. **What NSP Does:** BERT predicts whether the second sentence logically follows the first one.
2. **How It Works:**
   - BERT uses the output of the [CLS] token, processes it through a classification layer, and applies SoftMax to calculate the probability.
   - 50% of input pairs are connected (sequential sentences), and 50% are random pairs.



3. **Input Preparation:**
   - [CLS] is added at the start, and [SEP] separates sentences.
   - Sentence embeddings distinguish Sentence A from Sentence B.
   - Positional embeddings encode the token positions in the sequence.

4. **Joint Training:** BERT trains **Masked Language Model (MLM)** and **NSP** together to minimize a combined loss, improving its ability to understand both sentence context and relationships.

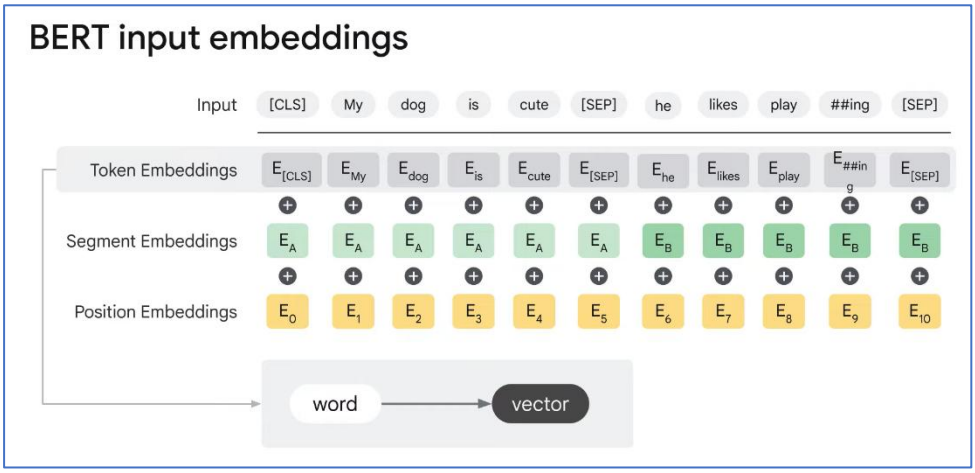
In technical terms:

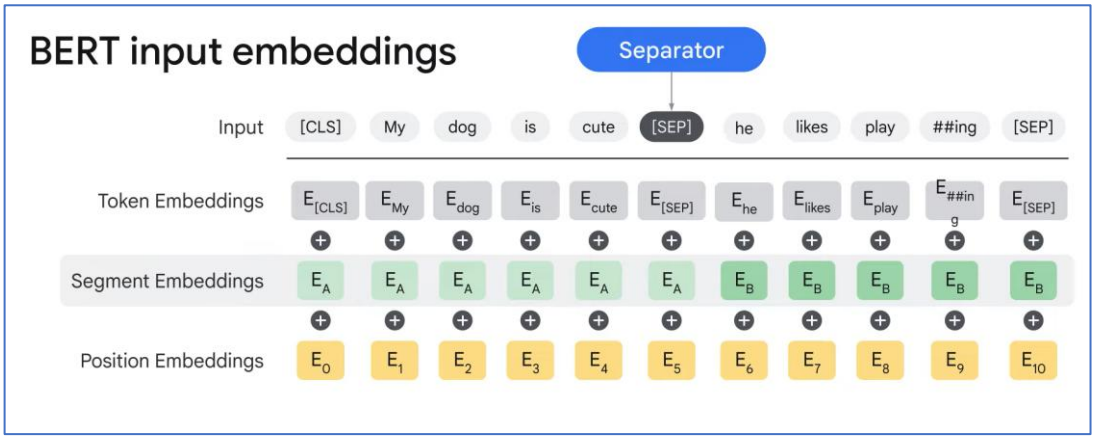
1. **Classification Layer:** The [CLS] token's output is transformed into a 2×1 vector to classify sentence pairs.
2. **SoftMax:** Converts predictions into probabilities of whether the second sentence follows the first.
3. **Balanced Training Data:** 50% of sentence pairs are sequential; the rest are random.
4. **Combined Loss:** BERT minimizes the loss for both **MLM** and **NSP**, enhancing its ability to understand both intra-sentence context and inter-sentence relationships.

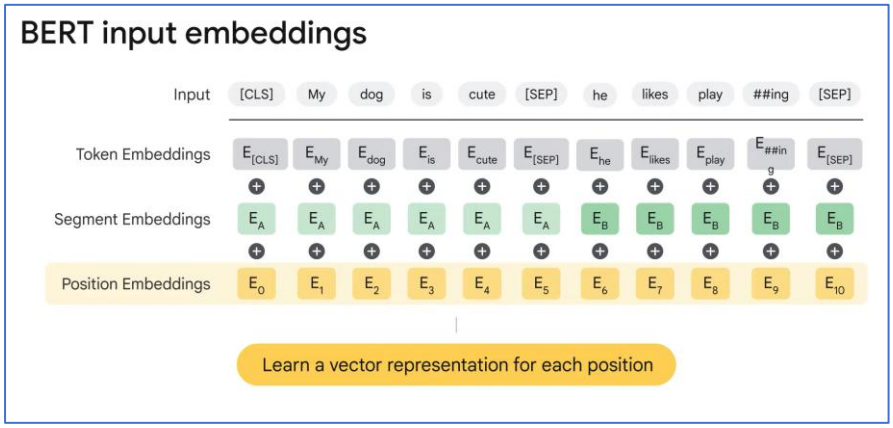
Input Representation

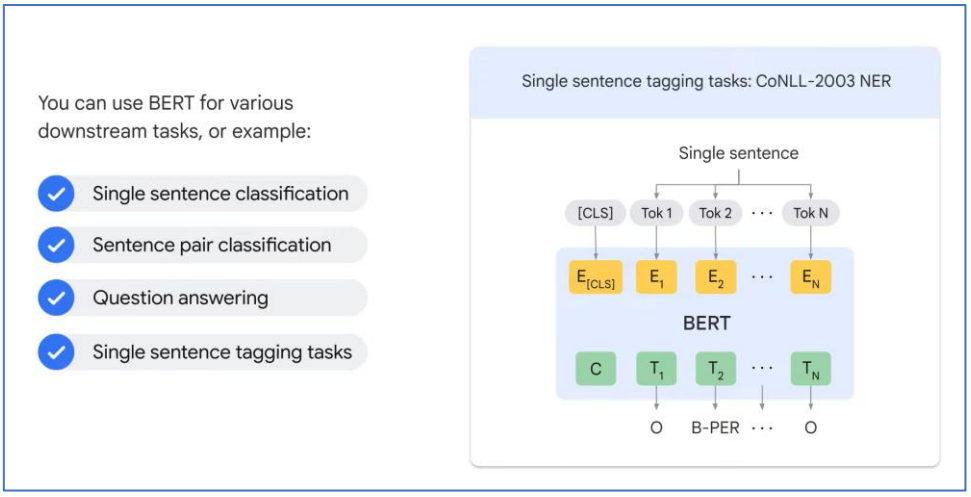
BERT processes text input into a format it can understand, combining:

1. **Token Embeddings:**
   - Each word in the input is split into subwords using a tokenizer (e.g., "playing" → ["play", "ing"]).
   - Each subword is mapped to a dense vector (embedding).
2. **Segment Embeddings:**
   - For tasks involving two sentences (e.g., Question Answering), BERT needs to know which tokens belong to which sentence. Segment embeddings differentiate Sentence A from Sentence B.
3. **Positional Embeddings:**
   - Transformers don't have a sense of word order. Positional embeddings are added to the token embeddings to encode the position of each word.









Environment Setup:

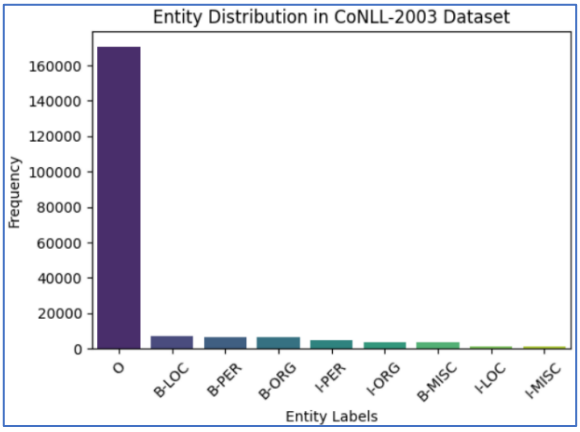
- Installed essential libraries:
  - Hugging Face Transformers
  - PyTorch
  - NumPy, Pandas
  - Matplotlib, Seaborn for visualization.
- Verified the installation by running a sample script to load a pre-trained BERT model.

Dataset Exploration:

- Examined the structure of NER datasets:
  CoNLL-2003: Contains labeled entities for locations, names, and more.

1) Entity Distribution

This bar chart displays the frequency of each entity type in the dataset.

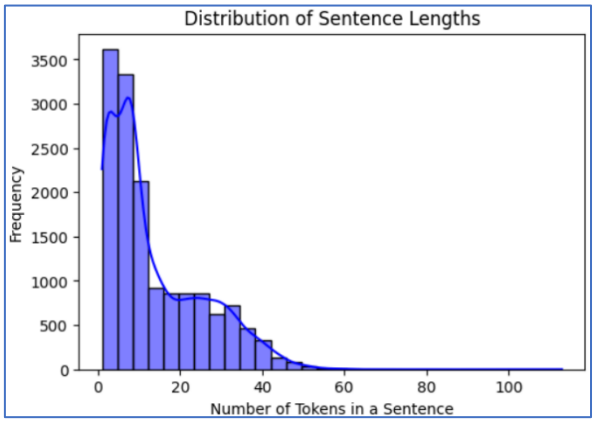


Key Insights:

- **Imbalanced Dataset:** If one entity type has significantly more tokens than others, the dataset is imbalanced. This can affect the model's performance, as it may favor predicting the more frequent entity.
- **Entity Coverage:** Helps understand which entity types are covered in the dataset. For example:
  - PER (Person)
  - LOC (Location)
  - ORG (Organization)
  - MISC (Miscellaneous)
- **Relevance to NER:** A balanced distribution is ideal for building a robust model. If certain entity types are underrepresented, consider techniques such as data augmentation.

2) Sentence Length Distribution

This histogram illustrates the distribution of sentence lengths (measured in the number of tokens) in the dataset.



Key Insights:

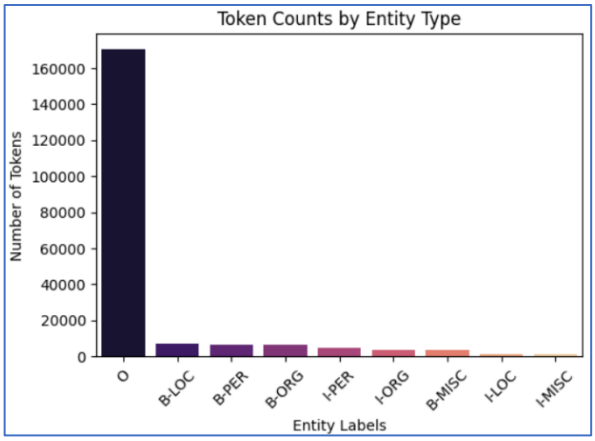
- **Dataset Complexity:** Longer sentences are harder to process because they require handling more dependencies and context. Shorter sentences are easier for models to handle.
- **Outliers:** If there are unusually long sentences, they might:
  - Indicate preprocessing issues (e.g., missing sentence splits).
  - Need truncation for compatibility with BERT's token limit (512 tokens).
- **Optimal Training Configurations:**
  - Short sentences (e.g., 10-20 tokens) allow for reduced sequence lengths during training.
  - Saves computation and memory while improving efficiency.

3) Token Counts by Entity Type

This bar chart shows the total number of tokens associated with each entity type in the dataset.

Key Insights:

- **Entity Complexity:** Certain entity types may span multiple tokens. For example:
  - ORG (Organization): Entities like "United Nations" or "Harvard University" require understanding token dependencies.
  - PER (Person): Often single tokens like "John" or "Alice."
- **Impact on Labeling Effort:** Entity types with higher token counts dominate the learning process during training, leading to potential bias.
- **Training Adjustments:** Consider:
  - Balancing loss weights for underrepresented entity types.
  - Augmenting data for rare entities.



Advantages of Using BERT for NLP

- Bidirectional Understanding: Considers both left and right context of words for better comprehension.
- Pre-Trained on Large Data: Trained on massive corpora (e.g., Wikipedia) for a strong language base.
- Transfer Learning: Fine-tuned for specific tasks with minimal labeled data.
- Handles Polysemy: Understands words with multiple meanings based on context.
- State-of-the-Art Performance: Excels in tasks like NER, text classification, and question answering.
- Long-Range Dependencies: Captures relationships between distant words effectively.
- Subword Tokenization: Breaks rare or unknown words into subwords for better handling of OOV (out-of-vocabulary) words.
- Multilingual Support: Supports 100+ languages via Multilingual BERT (mBERT).
- Wide Applications: Used for summarization, translation, information retrieval, and conversational AI.
- Parallelization: Faster training with parallel computations due to Transformer architecture.
- Easy to Use: Hugging Face's tools make implementation simple and accessible

Deliverables

1. **Summary Document:**
   - Created a detailed summary of NER basics, applications, and BERT's role in NLP.
   - Highlighted the differences between traditional and BERT-based approaches.
2. **Environment Setup:**
   - Set up a Python environment with all required libraries and tested their functionality.
3. **Dataset Exploration Notes:**
   - Documented key characteristics of CoNLL-2003 datasets.

Challenges Faced

1. Understanding the intricacies of the BERT architecture required revisiting transformer models and attention mechanisms.
2. Dataset formats and labeling schemes were confusing.

REFERENCE:

https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/
https://www.geeksforgeeks.org/named-entity-recognition/