VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

NGUYEN HUU THANG – 518H0567
NGUYEN DINH VIET HOANG - 522H0120

MIDTERM REPORT
INTRODUCTION TO NATURAL LANGUAGE PROCESSING

HO CHI MINH CITY, YEAR 2024

VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

NGUYEN HUU THANG – 518H0567
NGUYEN DINH VIET HOANG - 522H0120

MIDTERM REPORT
INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Advised by

Assoc. Prof. Dr. LE ANH CUONG

HO CHI MINH CITY, YEAR 2024

ACKNOWLEDGEMENT

We sincerely thank Assoc. Prof. Dr. Le Anh Cuong for teaching us the Introduction to Natural Language Processing course with great enthusiasm. We want to express our deep appreciation for the dedication and professional knowledge that you shared with us. Through your classes, we gained a better understanding of the fundamental aspects of the Introduction to Natural Language Processing, thanks to your detailed explanations and practical applications. You helped us grasp the knowledge and apply it effectively. Finally, we extend our heartfelt gratitude to Assoc. Prof. Dr. Le Anh Cuong for your commitment and invaluable support throughout our learning journey in this course. The skills and knowledge we acquired will continue to impact our future development. We sincerely thank you and wish your health, success, and happiness.

Ho Chi Minh City, October 28, 2024

Authors:

Nguyen Huu Thang

Nguyen Dinh Viet Hoang

DECLARATION OF AUTHORSHIP

We hereby declare that this thesis was carried out by ourselves under the guidance and supervision of Assoc. Prof. Dr. Le Anh Cuong; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by our own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

We will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

Ho Chi Minh City, October 28, 2024

Authors:

Nguyen Huu Thang

Nguyen Dinh Viet Hoang

ABSTRACT

Named Entity Recognition (NER) is a core task in Natural Language Processing (NLP) aimed at identifying and classifying entities such as people, organizations, locations, and dates in text. This project explores two approaches to building an NER model for English and Vietnamese sentences: a classification model using BERT and a Long Short-Term Memory (LSTM) model. The classification approach leverages BERT, a transformer-based pretrained model known for its strong context understanding and high accuracy in NER. In this approach, words are labeled with BIO tagging, and BERT is fine-tuned for sequence classification to predict the entity type of each word in a sentence. The LSTM approach applies recurrent neural networks, particularly effective for sequence-based tasks like NER, to learn contextual dependencies within sentences. Additionally, FastText word embeddings are used to improve the input representations for both models. FastText provides rich, subword-aware embeddings by incorporating character n-grams, thus capturing morphological details crucial for morphologically complex languages such as Vietnamese. This study evaluates the performance of these models using metrics such as precision, recall, and F1-score, assessing the effectiveness of FastText embeddings as input for the BERT-based classification model and LSTM model.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
NER	Named Entity Recognition
Bi-LSTM	Bi-directional Long-Short Term Memory
NLP	Natural Language Processing
NSP	Next Sentence Prediction

QUESTION 1: BUILD A NAME ENTITY RECOGNITION (NER) MODEL FOR ENGLISH OR VIETNAMESE SENTENCES USING CLASSIFICATION MODEL APPROACH AND LONG SHORT TERM MEMORY (LSTM) MODEL APPROACH.

1.1 Classification model approach

1.1.1 What is Named Entity Recognition (NER)?

+ NER is a task in Natural Language Processing (NLP) aimed at identifying and classifying entities in text, such as names of people, organizations, locations, dates, etc.

+ The input is a sequence of words (a sentence), and the output consists of labels for each word, identifying its entity type (for example, B-ORG for the first word of an organization, I-ORG for subsequent words of the organization, and O for words not part of any entity).

1.1.2 Classification Approach for NER

To build the NER model, we can use sequence classification models with the following steps:

Step 1: Preprocessing and Labeling the Data:

+ Labeling the data: Each word in a sequence is labeled using the standard BIO tagging scheme:

- B-X: Begin of an entity of type X.
- I-X: Inside an entity of type X.
- O: Outside any entity.

+ Example: In the sentence "Apple Inc. is located in Cupertino," the labels are:

Apple B-ORG

Inc. I-ORG

is O

located O

in O

Cupertino B-LOC

+ The data is then split into training and testing sets.

Step 2: Using a Pretrained Language Model (like BERT):

+ BERT (Bidirectional Encoder Representations from Transformers): A popular model for classification tasks, particularly for NER.

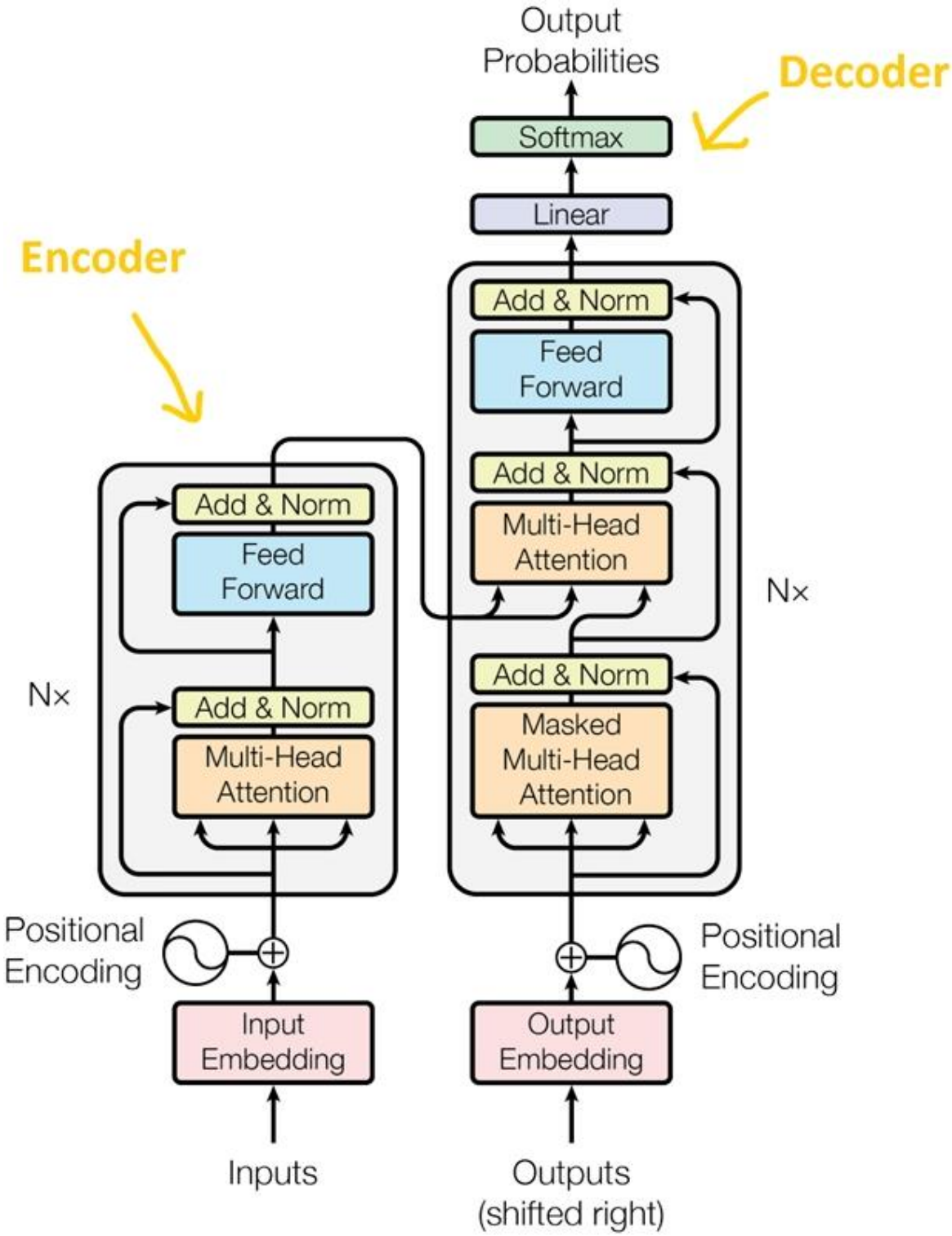


Figure 1.1.a: Transformer Model

+ Explanation for Figure 1.1.a:

- Positional encoding: Since the Transformer does not have recurrent or convolutional layers, it does not inherently know the order of input tokens. Therefore, there needs to be a way for the model to know this information, which is the task of positional encoding. After the embedding layers, which produce token embeddings, we add positional encoding vectors representing the position of each word in the sentence.
- Normalization Layer: In the diagram's architecture, the "Add & Norm" layer refers to the normalization layer. This layer simply normalizes the output of the multi-head attention, improving convergence efficiency.
- Residual Connection: The residual connection is a simple concept of adding the input of a block to its output. This connection allows stacking multiple layers in the network. In the diagram, the residual connection is used after the FFN (Feed-Forward Network) and attention blocks. In the "Add" part of "Add & Norm," it represents the residual connection.
- Feed-Forward Block: This is a basic block where, after performing computations in the attention block at each layer, the next block is the FFN. You can understand that the attention mechanism helps gather information from the input tokens, and the FFN processes that information.

+ BERT is a deep learning model based on Transformer architecture, pretrained in a bidirectional manner to predict a word within its context. It's widely used for NER due to its strong context-understanding abilities.

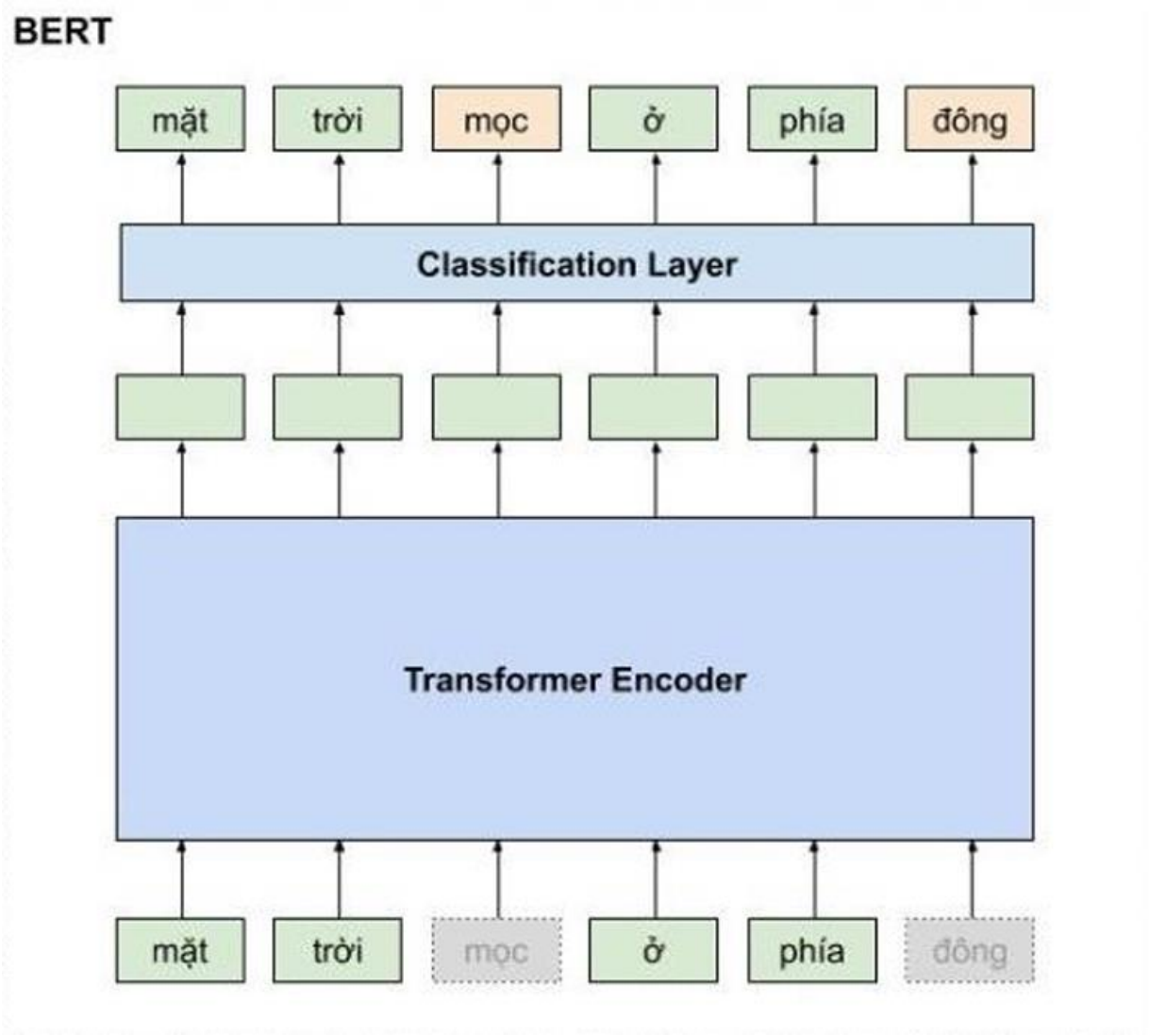


Figure 1.1.b: An example model of BERT

+ Tokenization: Using BERT's tokenizer, we convert the words in a sentence into vectors.

+ Fine-tuning: BERT is fine-tuned by adding a classification layer on top so the model can predict entity labels for each token.

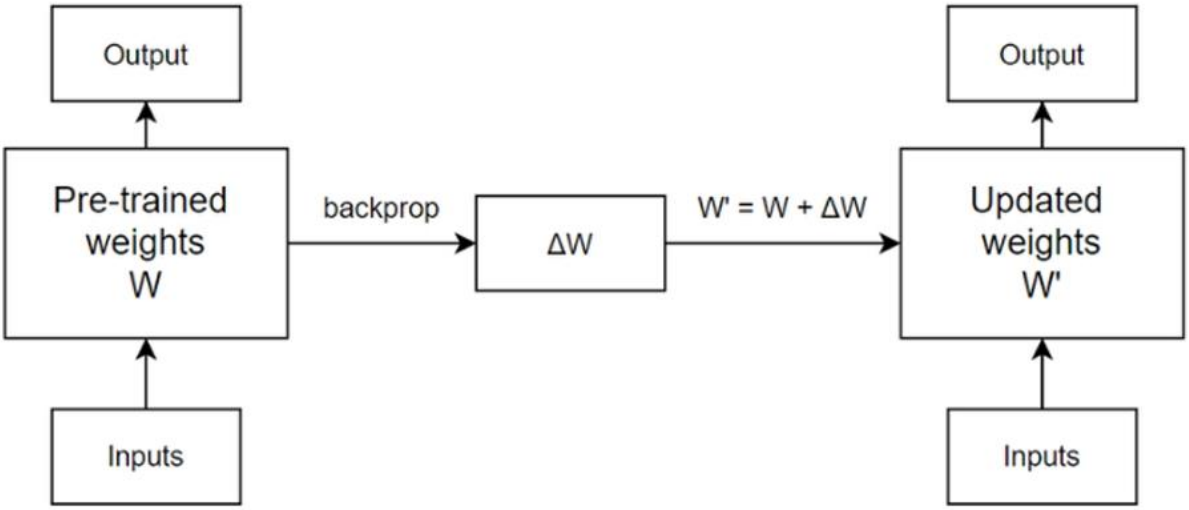


Figure 1.1.c: The process of updating one layer of the model in fine-tuning

Note for the figure 1.1.c: After pretraining for full model fine-tuning, the Transformer-based Encoder model can be fine-tuned on a specific task. In this process, a small amount of labeled data is used to adjust the model's weights. The model's output is compared to the ground truth labels, and a loss function such as Cross-Entropy is used to compute the error and update the network weights using backpropagation. Also, the pretrained weights W of the model will be transformed into updated weights W' based on the weight changes ΔW obtained from the backpropagation process. In the next iteration, W' is updated again with a different ΔW.

Step 3: Training the Model:

+ Training with a classification approach: The model is trained on the training set by optimizing a loss function (typically cross-entropy loss).

+ During training, the model learns to predict the label for each word in the sequence based on its context.

Step 4: Making Predictions and Post-processing:

+ After the model makes predictions, we convert label IDs back to the original BIO format for interpretability.

+ The labels are evaluated based on precision, recall, and F1-score to assess model quality.

1.1.3 Advantages and Disadvantages of Classification Approach

+ Advantages:

- Modern classification models like BERT achieve high accuracy thanks to a deep understanding of context.
- Suitable for complex languages, such as Vietnamese, where context can change a word's meaning.

+ Disadvantages:

- Requires substantial computational resources and long training times.
- Needs large labeled datasets for the model to perform well.

1.1.4 Overall Example

After processing, a data sequence might look like this:

Input: ["Apple", "Inc.", "is", "located", "in", "Cupertino"]

Labels: ["B-ORG", "I-ORG", "O", "O", "O", "B-LOC"]

⇨ After training, the model can predict labels for each word in a new sentence.

1.1.5 Build our program

Similar to the example mentioned above, I have also built a Named Entity Recognition (NER) model for English or Vietnamese sentences using the classification model method in the ClassificationModel.ipynb.

1.2 Long Short Term Memory (LSTM) model approach

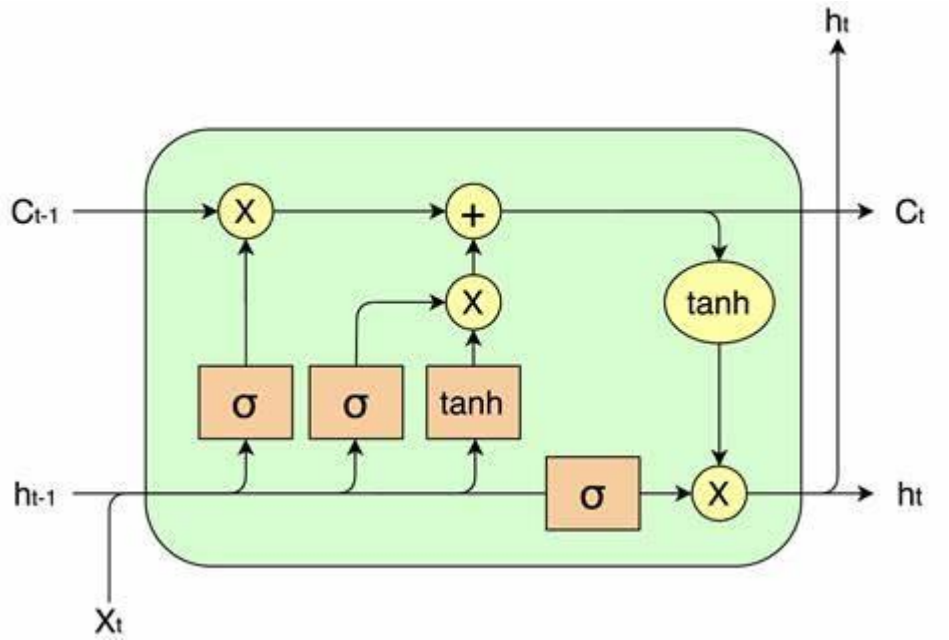
1.2.1 Definition

LSTM stand for Long short-term memory is an improved version of recurrent neural network (RNN) that can hold information for an extended period of time.

The LSTM architecture involves the memory cell which is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell.

- The input gate controls what information is added to the memory cell.
- The forget gate controls what information is removed from the memory cell.
- The output gate controls what information is output from the memory cell.

This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.



1.2.2 LSTM for NER

To train a pre-build LSTM model for NER, we can follow the step below:

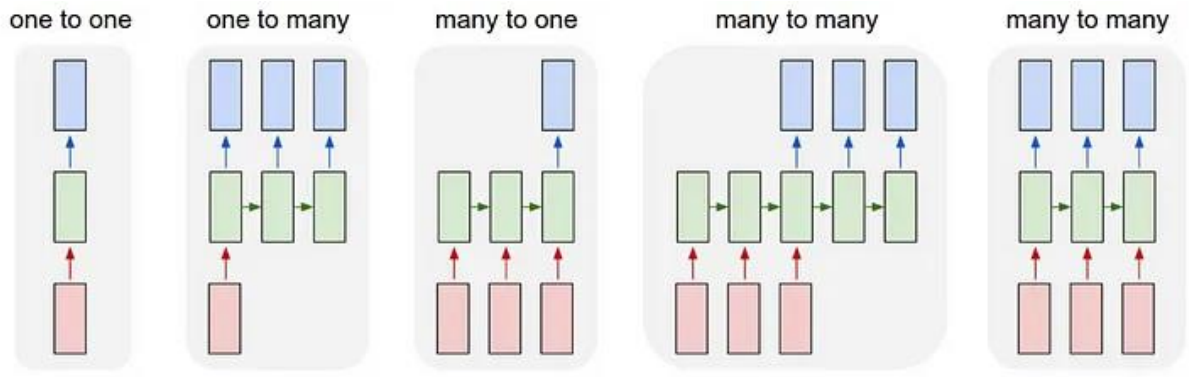
Step 1: Data processing:

+ Importing data: The data is feature engineered corpus annotated with IOB and POS tags.

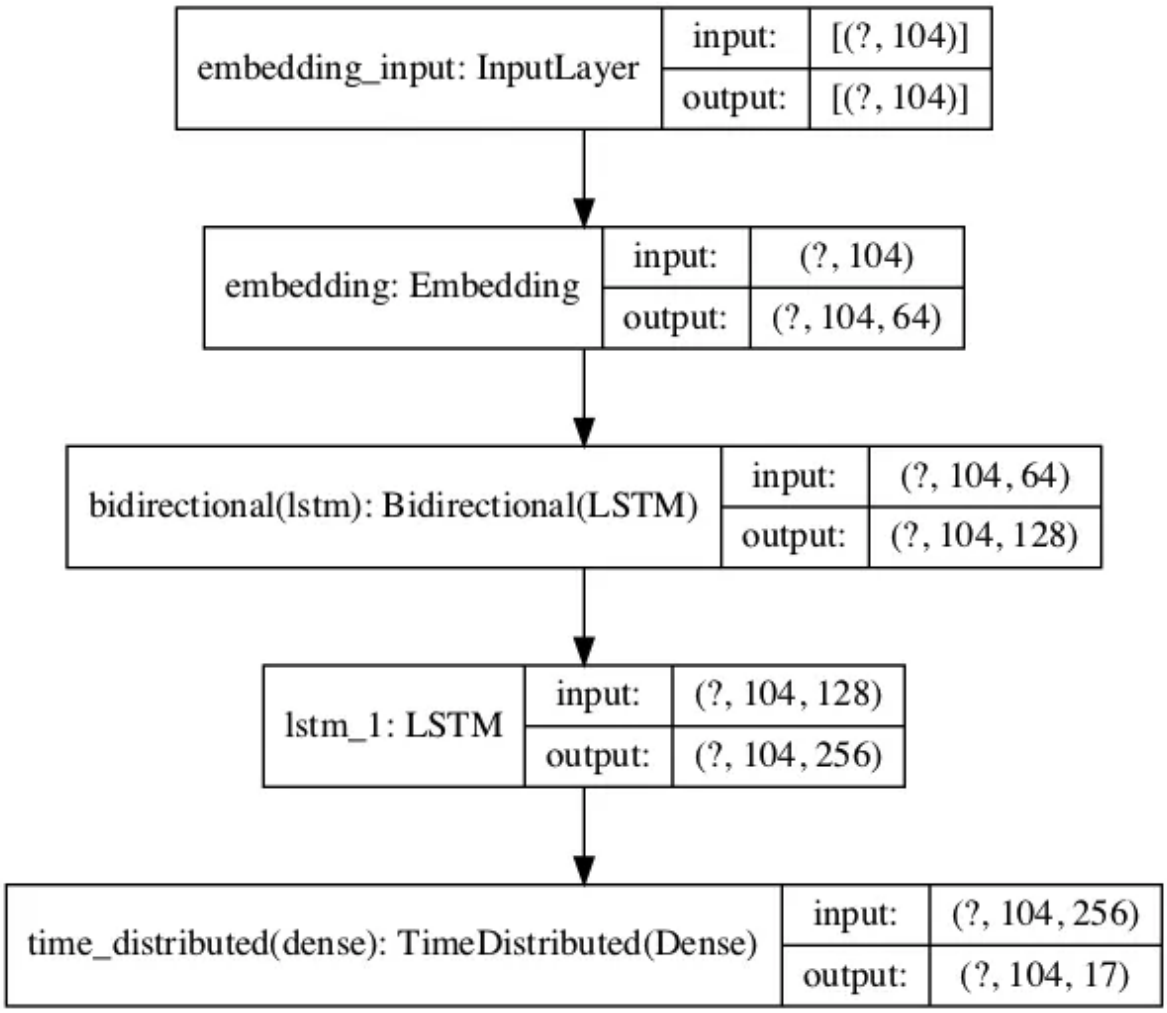
+ Extract mapping for data: In order for the model to work it will required 2 set:
    - <word> that corresponding with <wordID>
    - <tag> that corresponding with <tagID>
+ Padding data, create training and testing set:

Step 2: Building model:

Neural network models work with graphical structure. Therefore, we will first need to design the architecture and set input and out dimensions for every layer. RNNs are capable of handling different input and output combinations. We will use “many to many” architectures for this task. Refer to the last architecture in the image given below.



In this architecture, we are primarily working with three layers (embedding, bi-lstm, lstm layers) and TimeDistributed Dense layer, to output the result. We will discuss the layers in detail in the below sections.



- Layer 1 — Embedding layer: We will specify the maximum length (104) of the padded sequences. After the network is trained, the embedding layer will transform each token into a vector of n dimensions. We have chosen the n dimensions to be (64).
- Layer 2 — Bidirectional LSTM: Bidirectional LSTM takes a recurrent layer (e.g. the first LSTM layer) as an argument. This layer takes the output from the previous embedding layer (104, 64).

- Layer 3 — LSTM Layer: An LSTM network is a recurrent neural network that has LSTM cell blocks in place of our standard neural network layers. These cells have various components called the input gate, forget gate, and output gate.
- Layer 4 — TimeDistributed Layer: We are dealing with Many to Many RNN Architecture, where we expect output from every input sequence. Here is an example, in the sequence (a1 →b1, a2 →b2…an →bn), a, and b are inputs and outputs of every sequence. The TimeDistributeDense layers allow Dense(fully-connected) operation across every output over every time-step. No using this layer will result in one final output.

Step 3: Training:

- Training with a LSTM approach: The model is trained on the training set
- During training, the model learns to predict the label for each word in the sequence based on its context.

Step 4: Evaluating model:

+ Evaluate model base on test data.

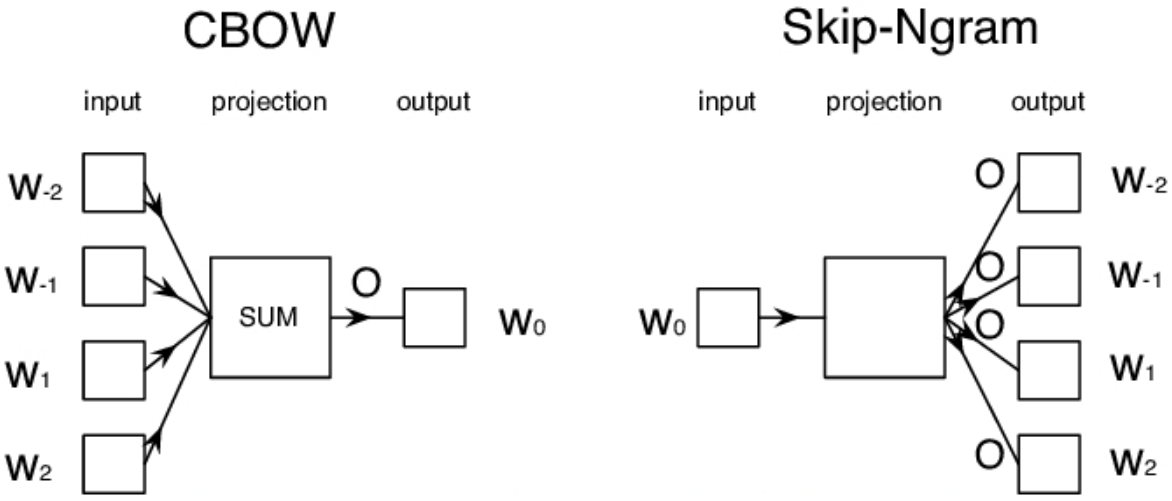
QUESTION 2: PRESENT THE FASTTEXT METHODS FOR WORD REPRESENTATION USING VECTORS (WORD2VEC AND WORD EMBEDDINGS) AND APPLY THIS REPRESENTATION TO PROBLEM 1, I.E., INPUT FOR EACH WORD SHOULD BE AN EMBEDDING VECTOR USING THE FASTTEXT METHOD FOR BOTH THE LSTM AND CLASSIFICATION APPROACHES.

2.1 Present the Fasttext methods for word representation using vectors (Word2Vec and Word Embeddings)

2.1.1 Word2Vec

Word2Vec is a predictive embedding model. There are two main Word2Vec architectures used to represent distributed word representations:

1. **Continuous Bag-of-Words (CBOW)** – The order of the context words does not affect the prediction (assuming a bag-of-words approach) (according to Aaron (Ari) Bornstein on Towards Data Science). In the continuous skip-gram architecture, the model uses the current word to predict a surrounding range of context words.
2. **Continuous Skip-Gram** focuses on nearby context words. Each context vector is weighted and compared independently from CBOW.



CBOW is faster, while skip-gram is slower but performs better for infrequent words.

2.1.2 FastText

FastText is a word representation method developed by Facebook's AI Research (FAIR) lab. FastText builds on Word2Vec by learning vector representations for each word and the n-grams found within each word. The values of these representations are then averaged into a vector at each training step. Although it adds more computation to training, it allows word embeddings to encode subword information. FastText vectors have been shown to be more accurate than Word2Vec vectors by various measures. This approach allows FastText to:

1. Handle **out-of-vocabulary (OOV)** words by representing them as the sum of their subword embeddings.
2. Capture **morphological features** of words, making it especially useful for morphologically rich languages like Vietnamese.

FastText embeddings are pretrained, so we can use them directly without further training. Each word will be mapped to a fixed-dimensional vector that can be used as input to a neural network for classification.

2.2 Apply this representation to problem 1, i.e., input for each word should be an embedding vector using the Fasttext method for both the LSTM and classification approaches.

2.2.1 Classification model approach

In file FastTextEmbeddingsForClassificationModel.ipynb.

2.2.2 LSTM model approach

In file LSTMModel.ipynb.

REFERENCES

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* https://aclanthology.org/N19-1423/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems.* https://arxiv.org/abs/1706.03762

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.* https://arxiv.org/abs/1301.3781

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics.* https://aclanthology.org/E17-2068/

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics.* https://aclanthology.org/Q17-1010/

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural Architectures for Named Entity Recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* https://aclanthology.org/N16-1030/

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pre-training Approach. *arXiv preprint arXiv:1907.11692*. https://arxiv.org/abs/1907.11692

Facebook AI Research. (2019). FastText Documentation: Word Representations. FastText. https://fasttext.cc/docs/en/word-representations.html

Hugging Face. (2024). BERT for Named Entity Recognition. Hugging Face Transformers Documentation. https://huggingface.co/docs/transformers/model_doc/bert