

Vietnamese Sentiment Analysis

Capstone Project

Group 3

Phạm Văn Cường - 20194421

Nguyễn Việt Hoàng - 20194434

Hoàng Văn Khánh - 20194440

Phan Đức Thắng - 20194452

Phạm Việt Thành - 20194454

Introduction

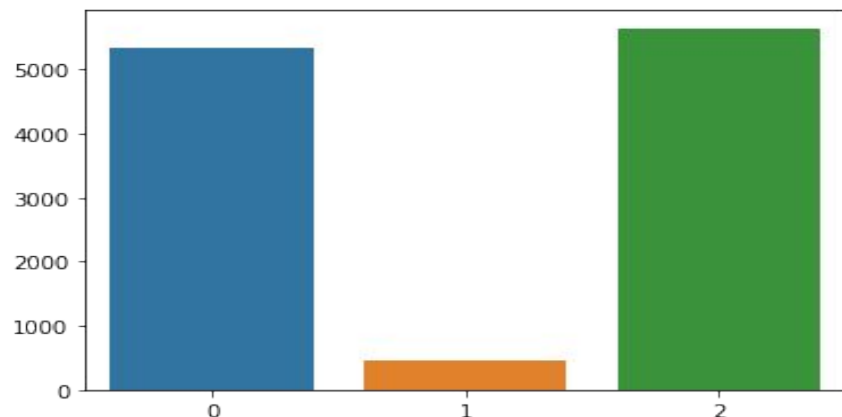
In this project, we will build sentiment analysis systems with a Vietnamese Dataset. The dataset is called UIT-VSFC, which is released by Ho Chi Minh City University of Technology. The data is built from Vietnamese students' feedback on various topics.

For the given dataset, we perform experiments with several neural networks architectures for sentiment analysis. The chosen models are as follow:

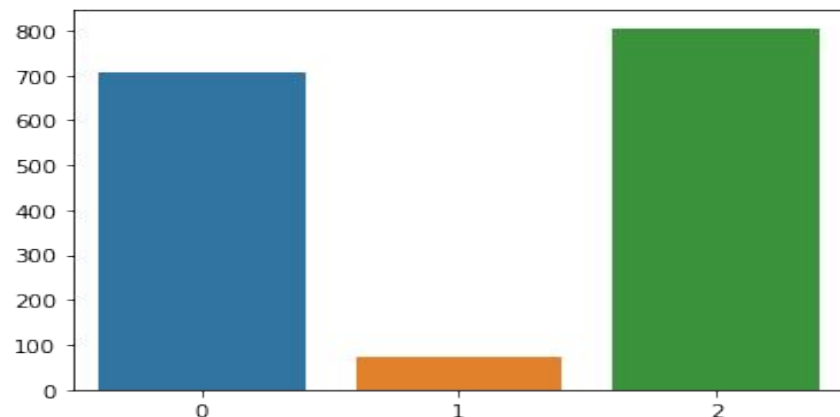
- RNN (Recurrent Neural Networks).
- LSTM (Long-Short Term Memory Networks).
- Transformers.
- BERT (Bidirectional Encoder Representations from Transformers).

With each model, we tried various configurations to see if the current solutions can be improved.

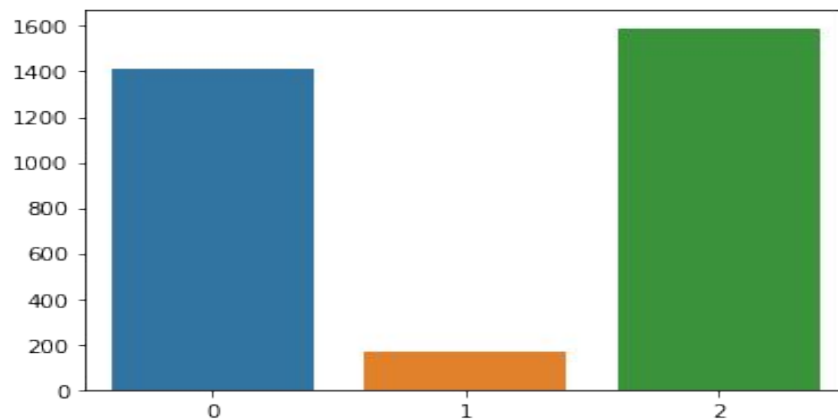
Data Overview: Imbalanced Dataset



Number of samples per class (train set)

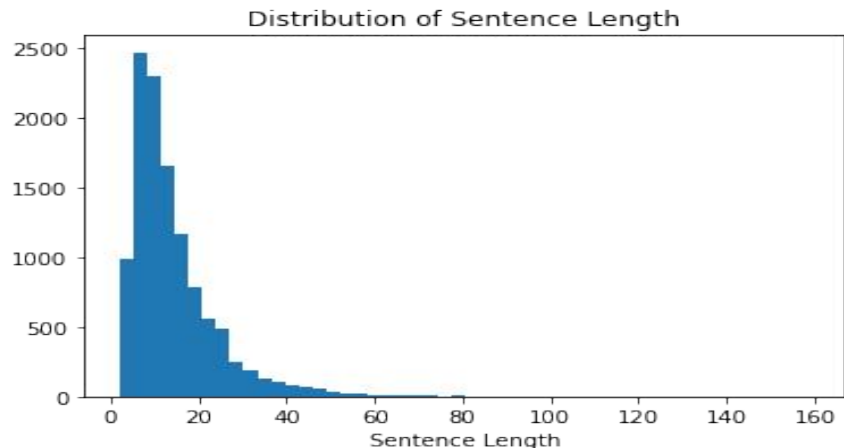


Number of samples per class (dev set)

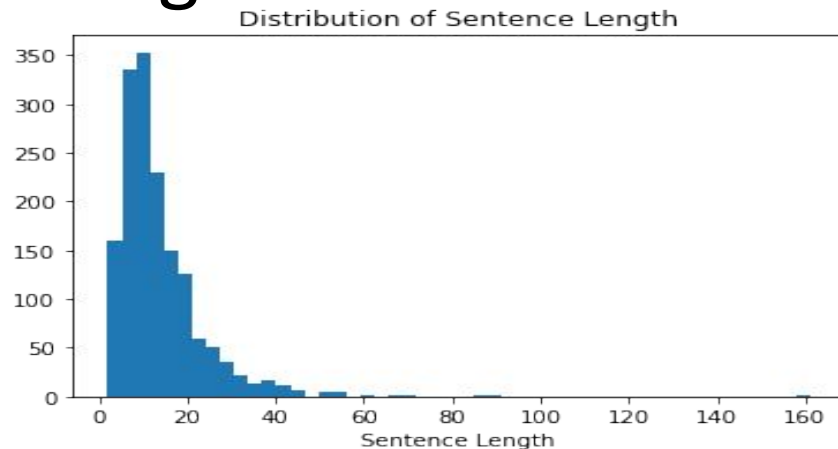


Number of samples per class (test set)

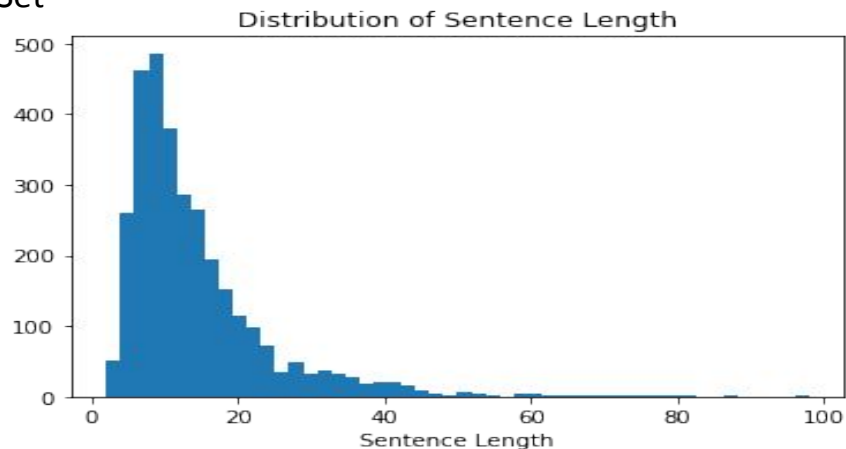
Data Overview: Sentence Length Distribution



Train Set



Dev Set



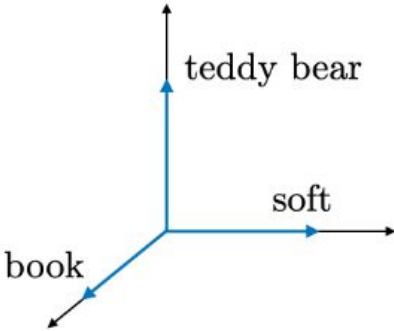
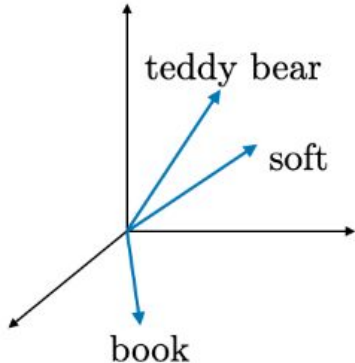
Test Set

Preprocessing

We use these two main ways of representing data.

We added some special tokens:

- “<name>”: human name
- “<unknown>”: words in dev set and test set that don't exist in training set.
- “<pad>”: used for padding.
- Punctuations (“!”, “?”, “.”,...) are not removed.

1-hot representation	Word embedding
	
<ul style="list-style-type: none">• Noted o_w• Naive approach, no similarity information	<ul style="list-style-type: none">• Noted e_w• Takes into account words similarity

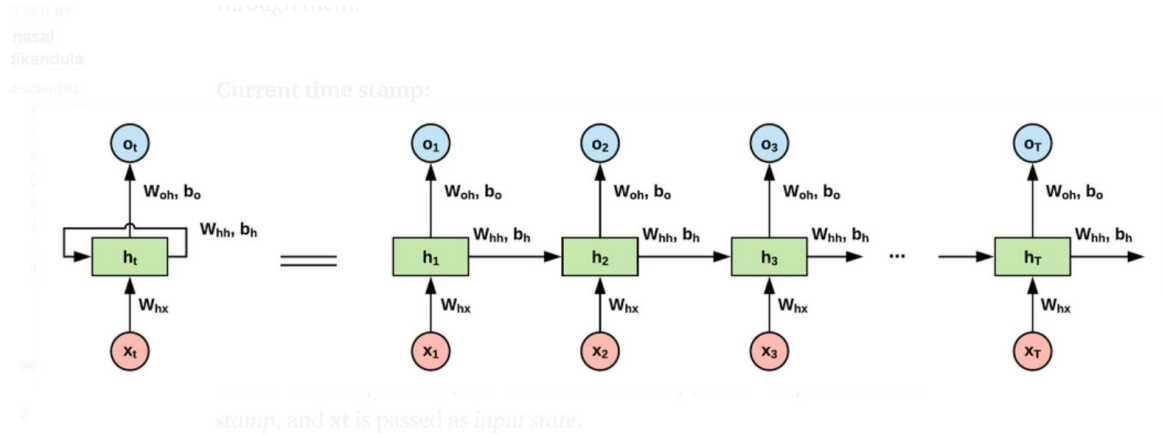
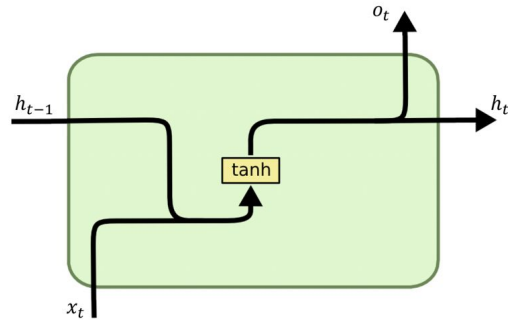
Preprocessing - Word segmentation

- Words in Vietnamese can contain 1 or more syllables. For instance, a 6-syllable sentence “Tôi là một nghiên cứu viên” forms 4 words sentence “Tôi là một nghiên_cứu_viên”.
- Because of this, applying word representations without any pre-processing can cause a degradation in the model performance.
- In this project, we experiment with applying a word segmenter to the text before training to see if there is any performance improvement. RDRSegmenter from VnCoreNLP is the chosen word segmentation method.

Recurrent Neural Network (RNN)

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

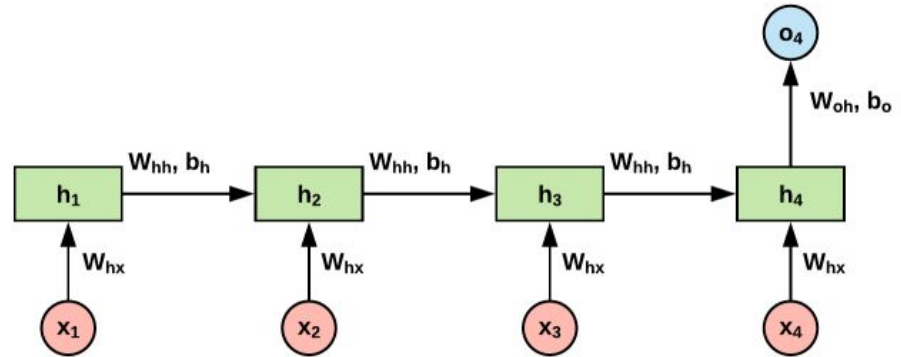
$$o_t = g(W_{oh}h_t + b_o)$$



- x_t : is the input data at timestep t . More specifically, the t^{th} word in the sentence in our case.
- h_t : is the hidden state at timestep t .
- o_t : is the output at timestep t .
- W_{hx} , W_{hh} , W_{oh} , b_h , and b_o : are shared weights and biases as described in Fig 4.1.1.
- f and g : are activation functions.

Recurrent Neural Network (RNN)

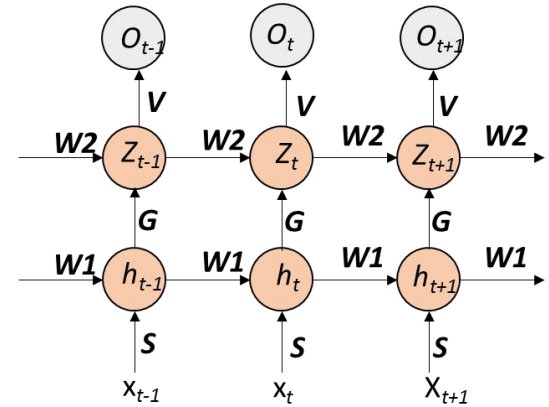
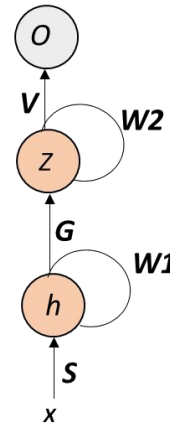
For our problem: We use the many-to-one RNN architecture. Only the last output is used to predict the label.



Recurrent Neural Network (RNN)

More layers can be added to increase the expressiveness of the model.

We use 3 layers RNN model.



a) 2-layer Recurrent Neural Network (RNN)

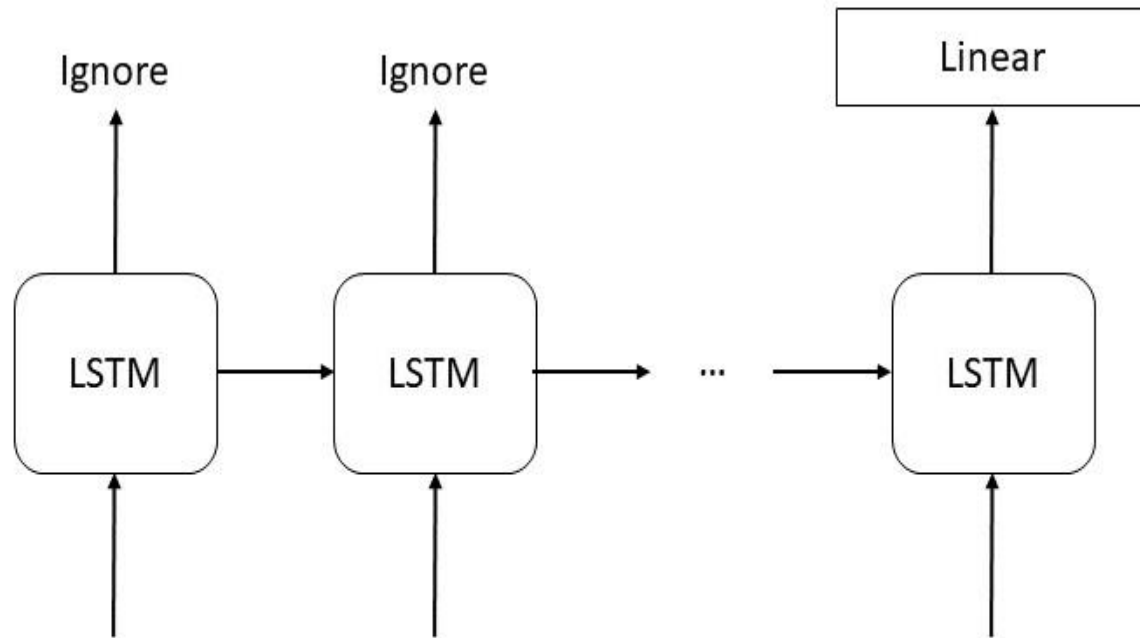
b) Unfolded 2-layer Recurrent Neural Network (RNN)

RNN performance

No	Model	Hyper params tuning	Acc	F1 Score			
				Negative	Neutral	Positive	Weighted Avg
1	1 layer, one-hot	No	88.31	88.94	42.03	91.66	88.16
2	1 layer, word embedding	No	88.82	89.84	43.10	91.09	88.36
3	3 layers, one-hot	Yes	90.59	91.77	46.51	93.08	90.35
4	3 layers, word-embedding	Yes	90.84	92.03	42.52	93.60	90.54
5	3 layers, one-hot, word segmentation	Yes	90.52	91.49	39.25	93.08	89.89
6	3 layers, word-embedding, word segmentation	Yes	91.09	92.46	41.12	93.18	90.46

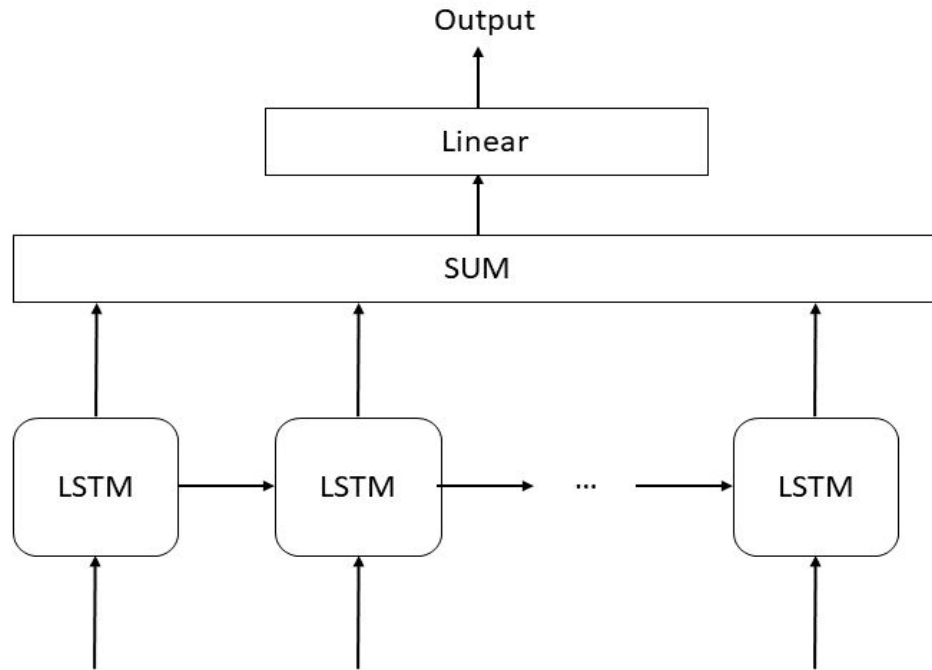
RNN performance on dev set with different configurations

LSTM



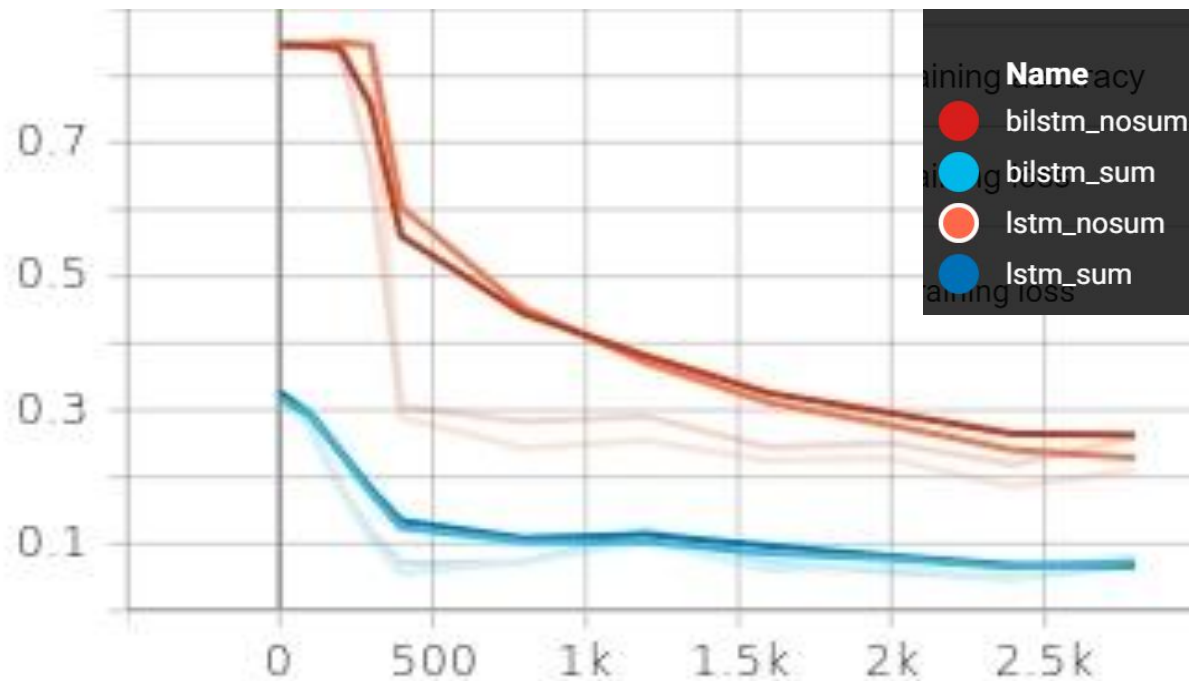
The initial implementation is the classic many-to-one model. However in this approach, this approach does not effectively utilize all informations of all words

LSTM



Add all the output of LSTM cells to use the information from all the words in the sentence

Sum vs No Sum



Training Loss

Models converge faster if use the sum of all outputs

Sum vs No Sum: Dev Results

Models	Accuracy	F1 Score			
		Negative	Neutral	Positive	Weighted Avg
1-directional, no sum output	90.84	92.32	43.55	93.15	90.49
1-directional, sum output	89.32	91.67	48.89	91.84	89.79
Bidirectional, no sum output	89.83	91.55	0.00	92.31	87.71
Bidirectional, sum output	89.96	92.20	52.97	92.30	90.44

Other hyperparameters: Dev Results

- Non bidirectional
- Sequence Length: 40
- Embedding Layer: 500

	Accuracy	F1 Score			
		Negative	Neutral	Positive	Weighted Avg
No word segmentation	90.21	90.96	53.52	92.76	90.15
Word segmentation	90.27	91.91	46.67	92.92	90.33

TRANSFORMER

- **Encoder-decoder structure**

- **Scaled dot-product Attention**

- An attention function maps a query and a set of key-value pairs to an output.
- Let Q the matrix of queries, K the matrix of keys, V the matrix of values, the matrix of output is calculated as:

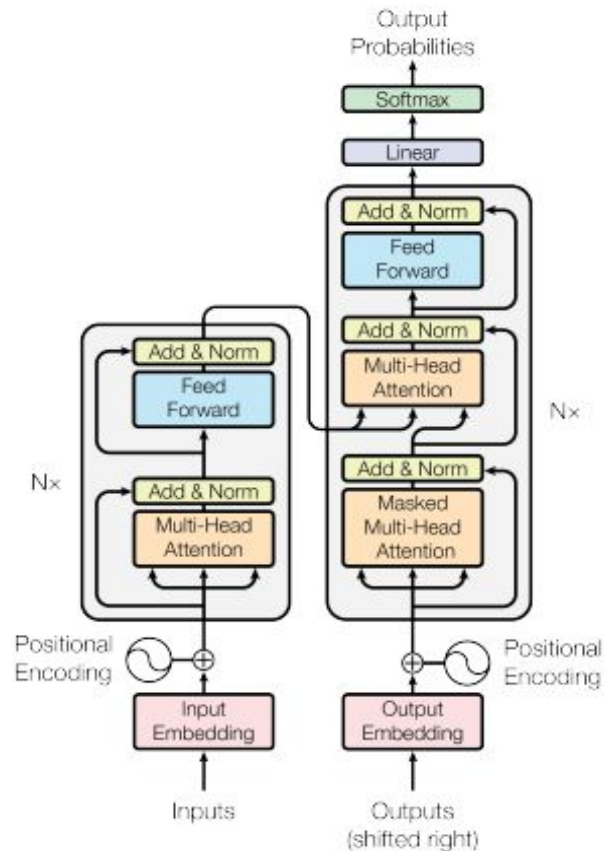
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **Multi-Head Attention**

- The multi-head attention includes several parallel attention layers (heads):

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$



Transformer by Vaswani et al. 2017

TRANSFORMER

- **Point-wise feed forward network**

- Each encoder/decoder layer contains a fully connected feed-forward network.
- Consists of two linear transformations with a ReLU activation in between.

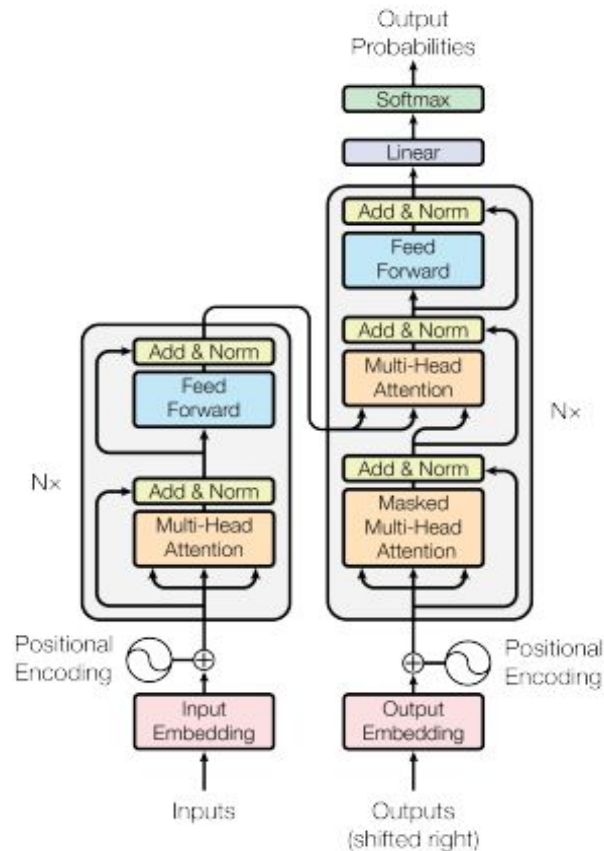
- **Embeddings**

- The input tokens and output tokens are converted into vectors of dimension $d_{\text{model}} = 512$.

- **Positional encoding**

- Add information about positions of the tokens in the sequence

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

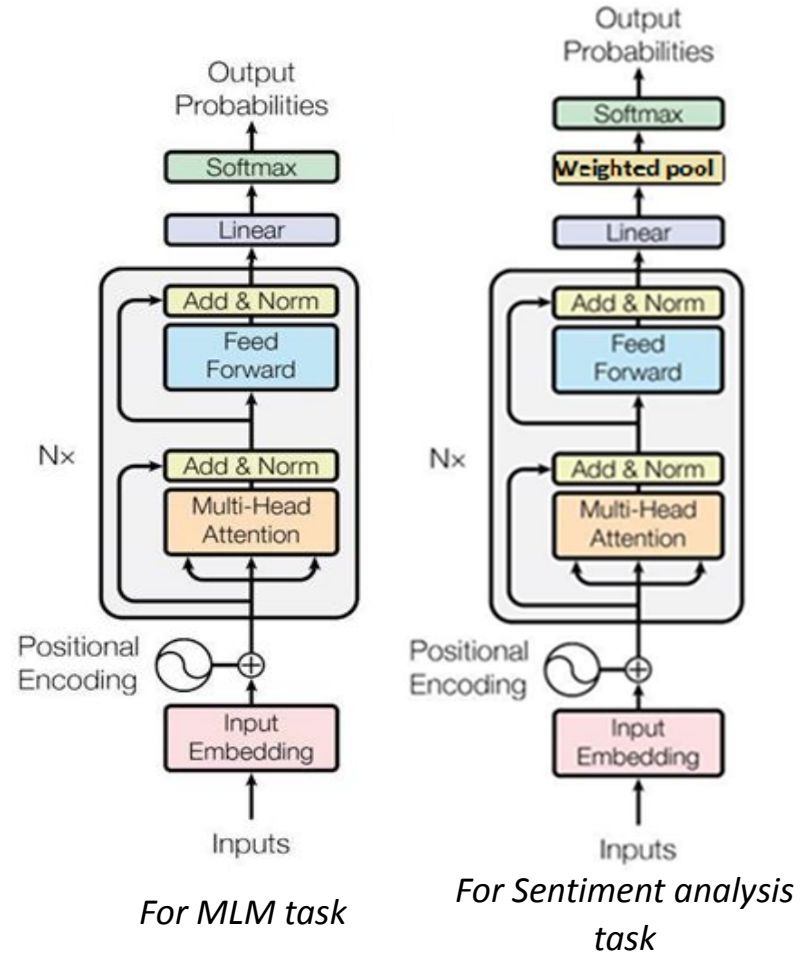


Transformer by Vaswani et al. 2017

TRANSFORMER

- Our model only contains encoder layers.
- Linear-transformation and softmax function are used to convert the encoder output to either:
 - predicted token probabilities for Masked Language Model task, or
 - predicted sentiment probabilities for sentiment analysis task.
- Hyperparameters (best value in **bold**):

Parameter	Values
Input/ output dimension	Number of tokens in train-data's longest sentence.
Embedding dimension	20; 40 ; 100; 200
Number of heads in multi-head attention	1; 2 ; 4
Number of encoding layers	1; 2 ; 4
Dimension of the feedforward network model in encoder layers	20; 50 ; 200; 400



TRANSFORMER

- **Preprocessing:**

- Word split vs word segmentation (e.g: “giáo viên” => [“giáo”, “viên”] or [“giáo_viên”])
- Tokenizer (e.g: given a dictionary: {“giáo”: 1, “viên”: 2} , then “giáo viên” => [1,2])
- Padding (e.g: given input dimension = 3, then “giáo viên” => [1,2,0])

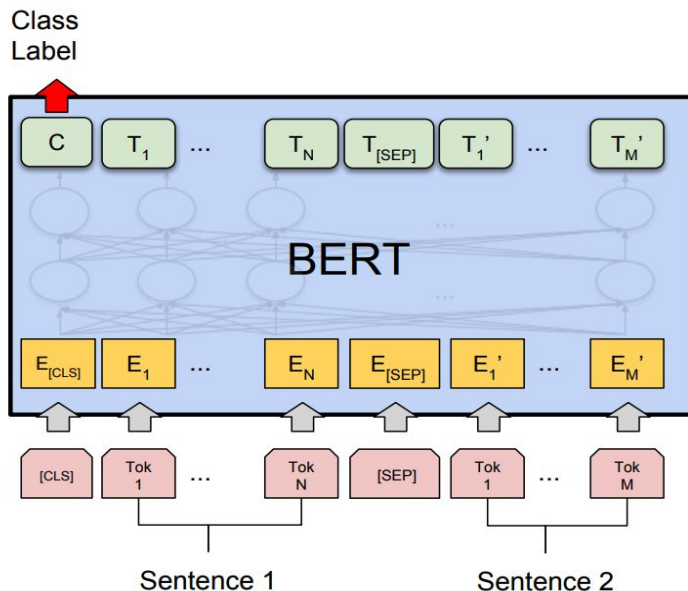
- **Pretrain: Masked Language Modeling**

- Give the model a sentence and optimizing the weights in order to output the same sentence on the other side.
- While the input and label sentence are essentially the same, the input sentence has some of its tokens masked.

- **Models performance:**

Model No.	Data preprocessing	Pretrain	Valid acc.	Valid F1	Test acc.	Test F1
1	Word split, tokenized	No	91.60	91.46	89.51	89.10
2		Yes	92.17	91.89	89.67	89.30
3	Word segmented, tokenized	Yes	92.23	92.04	89.89	89.44

BERT

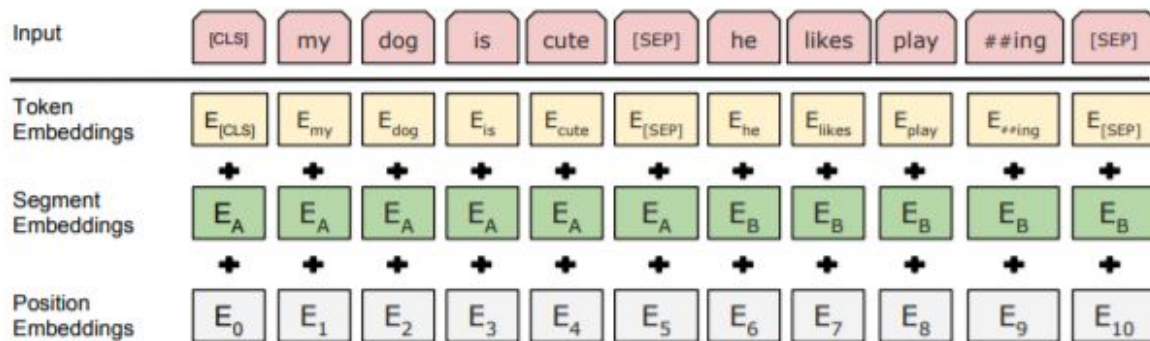


Architecture: Transformer encoders.

Model input: pair of sentences:

- A sentence here is a **segment of contiguous text sampled from the data rather than a true linguistic sentence.**
- A [SEP] token is placed at the end of each sentence.
- A [CLS] token is placed before the first sentence.

BERT



Token Embeddings. Wordpiece tokenizer.

Segment Embeddings. Binary numbers that tell which tokens belong to which sentences.

Positional Embeddings. Similar to Transformer's.

BERT - Training objectives

Masked language modeling

- A random sample of the tokens in the input is replaced with the *[MASK]* token.
- Use Cross-Entropy loss for predicting the masked tokens.

Next sentence prediction

- Predict if the 2 sentences is adjacent in the document.
- Use the hidden state output of the [CLS] token, pass it through a binary classification loss.

RoBERTa

Dynamic Masking

- Perform masking every time the model sees the sentence.

NSP Removal

- Remove the NSP training objective.

RoBERTa for sentiment analysis

- Append a linear layer on top with random initializations.

BERT - Results on dev set

Method		Dev set				
Pretrained model	Word Segmentation	Accuracy	F1			
			Negative	Neutral	Positive	Weighted Avg
From scratch	no	91.06	-	-	-	91.20
PhoBERT-base	no	94.57	96.48	60.53	96.06	94.64
PhoBERT-base	yes	95.51	96.85	68.22	96.48	95.36
PhoBERT-large	yes	95.83	96.86	67.19	97.24	95.66

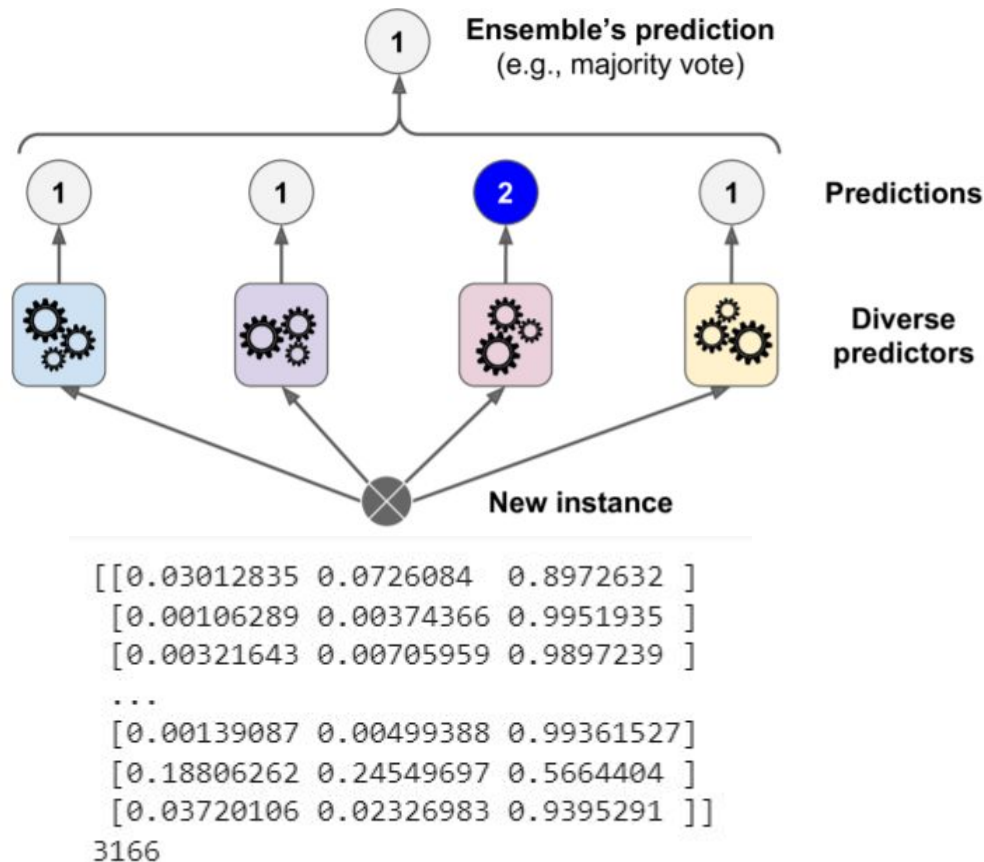
Final Results on Test Set

Model	Acc	F1 Score			
		Negative	Neutral	Positive	Weighted Avg
RNN	89.23	90.77	29.63	91.85	88.09
LSTM	89.01	90.94	42.81	91.70	88.78
Transformer	89.89	91.63	44.85	92.18	89.44
BERT	93.78	95.22	58.62	95.74	93.52

Ensemble learning

Idea: Combine the output of individual models into a single output of the ensemble.

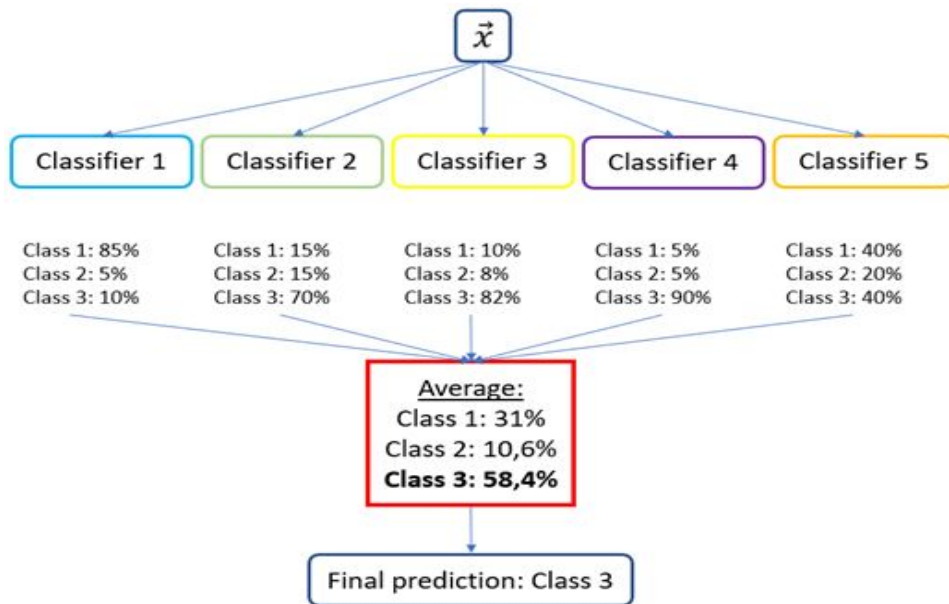
Hard voting: Since our models' accuracy is over 89% and Transformer is slightly higher than the others two models, we hope ensembling models will do a better result due to *The law of large numbers*. First, we $\text{argmax}(\text{axis}=1)$ the output and then counter the major voting.



Ensemble learning

Soft voting:

- This strategy predicts the class label based on the argmax of the sums of the predicted probability of the individual estimators that make up the ensemble.
- Soft voting can improve on hard voting because it takes into account more information; it uses each classifier's uncertainty in the final decision.



Ensemble performance on test set

Model	Word segmentation	Voting Strategy	Acc	F1 Score			
				Negative	Neutral	Positive	Weighted Avg
LSTM + RNN + Transformer	No	Soft voting	90.05	91.77	44.02	92.24	89.49
		Hard voting	89.73	91.43	41.86	92.08	89.14
	Yes	Soft voting	90.37	91.91	44.09	92.66	89.77
		Hard voting	90.24	91.92	42.52	92.53	89.62
LSTM + RNN	No	Soft voting	89.67	91.59	42.80	91.92	89.19
	Yes		90.05	92.01	41.35	92.35	89.51
RNN + Transformer	No	Soft voting	89.96	91.42	45.04	92.34	89.44
	Yes		90.11	91.69	38.37	92.67	89.37
LSTM + Transformer	No	Soft voting	90.15	91.90	46.98	92.38	89.77
	Yes		90.49	92.45	46.04	92.62	90.08

Further improvement on BERT

Model 1. Train without preprocessing the names, ie. keeping names as they are.

Model 2. Train with the data in which all names are removed.

Model 3. Train with the data in which all names are replaced with <name> tokens.

Model	Test set				
	Accuracy	F1 Score			
		Negative	Neutral	Positive	Weighted Avg
Model 1	93.52	95.24	59.73	95.14	93.32
Model 2	93.81	95.32	56.18	95.63	93.39
Model 3	94.00	95.72	61.33	95.52	93.82

BERT - Ensembling strategies

Model	Test set				
	Accuracy	F1 Score			
		Negative	Neutral	Positive	Weighted Avg
Model 3 (Previous best)	94.00	95.72	61.33	95.52	93.82
Model 1 + Model 2 + Model 3	94.13	95.63	61.32	95.73	93.87
Model 1 + Model 3	94.09	95.71	61.86	95.59	93.86
Model 2 + Model 3	94.28	95.88	61.65	95.70	93.99

Results comparison to other papers

Model	Test set			
	F1 Score			
	Negative	Neutral	Positive	Weighted Avg
Our best model (BERT ensemble)	95.88	61.65	95.70	93.99
Huy et al. VNU-HCM (2020) (BERT + CNN + BiLSTM + LSTM ensemble)	-	-	-	92.79
UIT-VSFC paper by Kiet et al. VNU-HCM (2018) (MaxEnt)	90.52	33.99	91.32	87.94

Conclusion

- Using word segmentation improves every models.
- By applying ensemble mechanism, the combination of 2 BERT configurations **achieves the highest accuracy and weighted F1 score.**

Future works

- Achieve equivalent or better results on other Vietnamese sentiment analysis benchmarks.
- Experiment with adding **in-domain pre-training data** for Transformer and BERT.

Thank you!

References

- [UIT-VSFC: Vietnamese Students' Feedback Corpus for Sentiment Analysis](#)
- [A Simple and Efficient Ensemble Classifier Combining Multiple Neural Network Models on Social Media Datasets in Vietnamese](#)
- [Recurrent Neural Networks cheatsheet - CS 230 - Deep Learning - Stanford](#)
- [Fundamentals of Recurrent Neural Network \(RNN\) and Long Short-Term Memory \(LSTM\) Network](#)
- <https://www.coursera.org/learn/nlp-sequence-models>
- [\[1706.03762\] Attention Is All You Need \(arxiv.org\)](#)
- [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)
- [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#)
- [PhoBERT: Pre-trained language models for Vietnamese](#)