COMP4332 / RMBI4310 Project 1 Group 8
Name: Lau Chun Wa, Chan Kwan Yu, Yeung Sze Man, Po Kit Man

Ordinary Models introduced in class

Models	**Data preparation**	**Hyper parameter tuning**	**Best Hyper parameter**	**Accuracy**
Logistic regression	Tfidf Vectorization	Tfidf: tokenize analyzer ngram_range norm stop_word lr:penalty C solver	tokenizer=tokenize analyzer=word ngram_range=(1,2) norm=l2 stop_word=english penalty=l2 C=1 solver=saga	0.5885
1-layer perceptron	Tokenize Stemming Filter stopword Uni,bi and tri gram	epochs learning rate batch size optimizers	epochs=2 batch size=20 optimizer=RMSprop learning rate=0.1	0.597800
Multi-layer perceptron	Tokenize Stemming Filter stopword n-gram Get_feats_dict Get_onehot_vector	optimizer learning rate dropping rate layer hidden size batch size epoch	batch size=8 dropping rate=0.01 epochs=5 hidden size: 100 layer: 2 Learning rate=0.01 optimizer: SGD	0.585100

From the above, we can see that the best parameter of the simple model is not enough to beat the baseline.

Model selection

Beyond the models we learned from the class, we have tried 2 extra models, BERT and XLNET, which are popular for NLP.

BERT

BERT is a bidirectional model which analyzes the relations between words regarding its sequences unlike unidirectional models. It applies two training strategies, Masked LM(MLM)[1] and Next Sentence Prediction(NSP)[2] to implement the bidirectional transformers, allowing us to investigate broader relations better than traditional directional-approach models.

XLNET

XLNet leverages the advantages of both, auto-regressive and auto-encoding methods for its pre-trained model which helps it to overcome pretrain-finetune discrepancy. XLNet implements Permutation Language Modeling (PLM)[3], two-stream self-attention[4] for target-aware representations and re-parameterization[5] with position to improve the performance, especially for the longer text sequence. XLNet thus overcomes the disadvantages of BERT and outperforms it.

Steps in using new mechanism

BERT		**XLNet**	
Bert Tokenizer	use pre- trained BERT tokenizer to tokenize the text	clean_text	Removal of the hashtags,hyperlinks and punctuation marks In the texts
adding special tokens	Adding of <sep> and <cls> token at the end of each sentence for indicating end of the sentence and classification	XLNet Tokenizer	use pre- trained XLNET tokenizer to tokenize the text
adding attention mask	indicates to the model which tokens should be attended to, and which should not	adding special tokens	Adding of <sep> and <cls> token at the end of each sentence for indicating end of the sentence and classification
Pedding to maximum sequence length	Adding of <pad> tokens to fixed length each sentence input to model	adding attention mask	indicates to the model which tokens should be attended to, and which should not
Bert pretrained Mode	Training of the pre-trained BERT model(bert-base-cased) with back propagate and updating weights	Pedding to maximum sequence length	Adding of <pad> tokens to fixed length each sentence input to model
		XLNET pretrained Mode	Training of the pre-trained XLNET model(xlnet-base-cased) with back propagate and updating weights

Evaluation

Models	**epoch**	**batch size**	**learning rate**	**Dropping Rate**	**max sequence length**	**optimizer**	**Accuracy**
BRET	4	16	2e-5	0.3	275	AdamW	0.656
BRET	4	32	2e-5	0.3	275	AdamW	0.648
BRET	6	48	2e-5	0.3	275	AdamW	0.640
XLNET	4	4	2e-5	0.1	512	AdamW	0.681
XLNET	4	4	3e-5	0.1	512	AdamW	0.667
XLNET	4	4	5e-5	0.1	512	AdamW	0.66

Conclusion

After tuning different parameters[6], the best model we select is the XLNET model with the highlighted parameters above. The accuracy rate for the validation set is 68.1%[7].

Appendix

BERT

Mechanism:

BERT refers to Bidirectional Encoder Representations from Transformers. BERT makes use of Transformer, which is an attention mechanism that learns contextual relations between words in a text. Unlike directional models, which read the text input sequentially, the transformer is considered as bidirectional as its encoder reads the entire sequence of words at once. The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

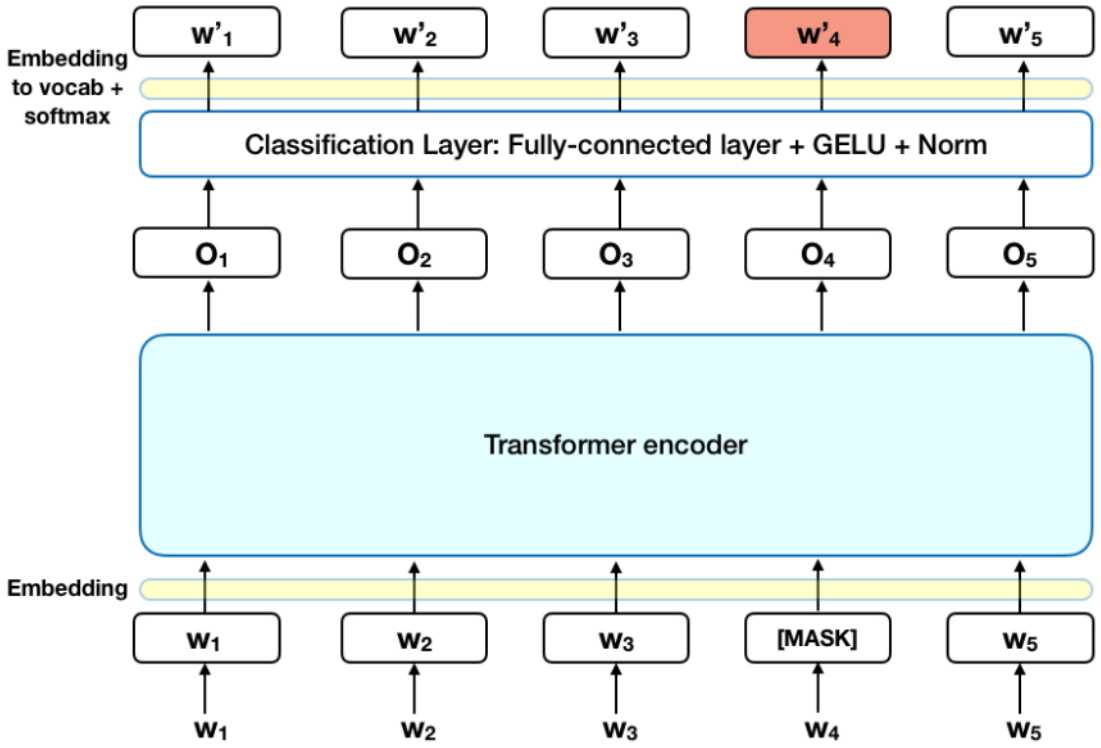
Advantages:

BERT makes use of two training strategies, Masked LM(MLM)[1] and Next Sentence Prediction(NSP)[2], to overcome the challenge brought by the directional approach, which inherently limits context learning.

1. **Masked LM (MLM)**

15% of the words in each word sequence are replaced by a [MASK] token before fitting these sequences into the BERT model. The model will then predict the original value of the masked words based on the context provided by the other non-masked words..

1. Adding a classification layer on top of the encoder output
2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
3. Calculating the probability of each word in the vocabulary with softmax

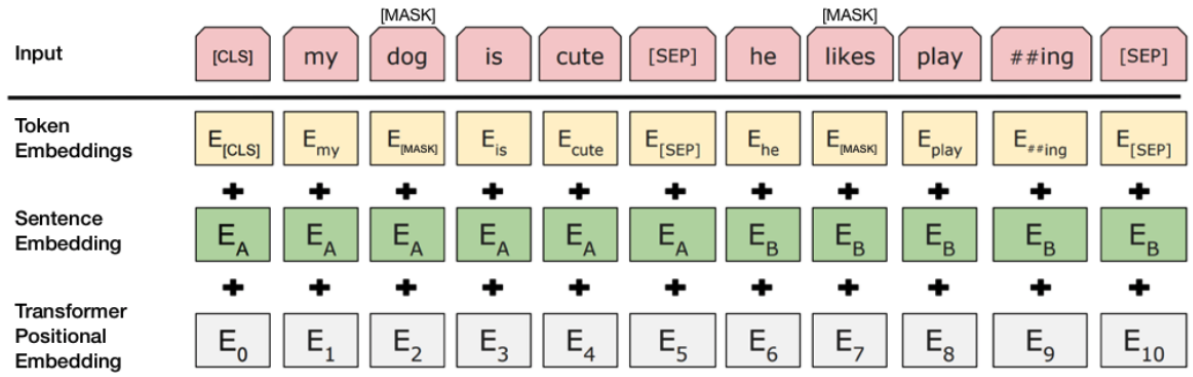


2. **Next Sentence Prediction (NSP)**

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other half of a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.



To predict if the second sentence is indeed connected to the first, the following steps are performed:

1. The entire input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
3. Calculating the probability of IsNextSequence with softmax.

XLNet

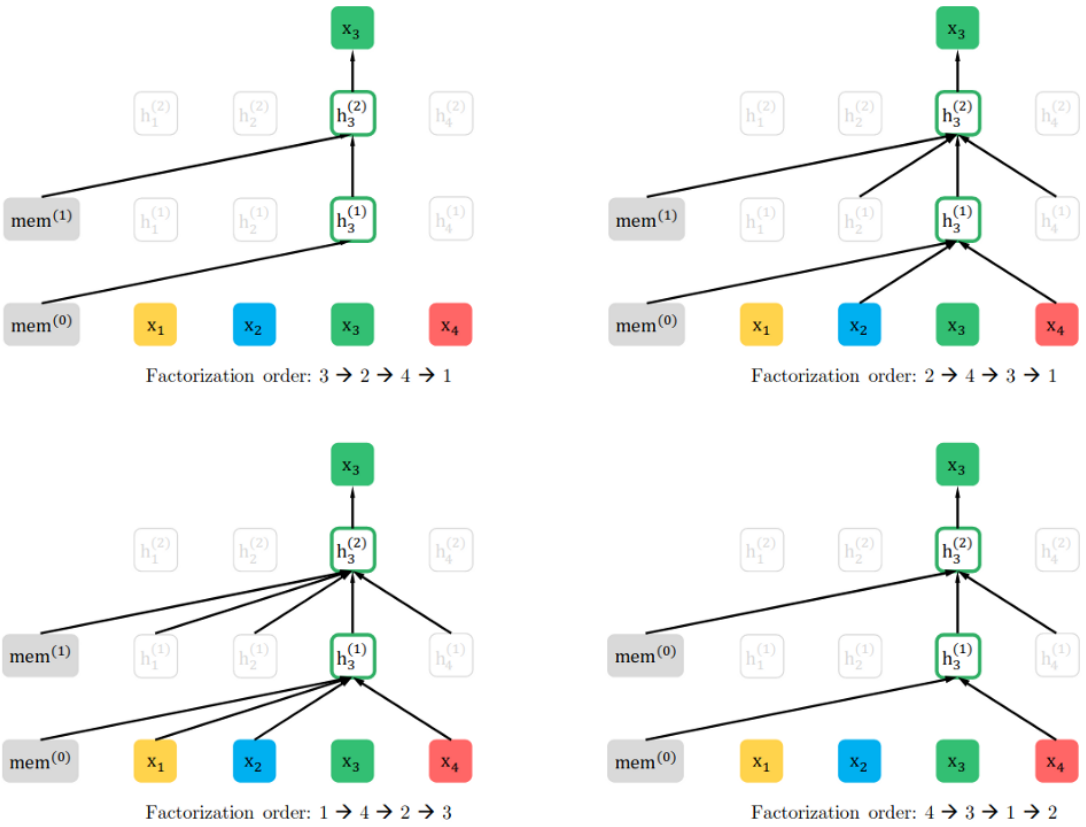
Mechanism:
XLnet is a generalized autoregressive model, which is opposed to BERT. In XLNet, the next token depends on all the previous tokens, which breaks away the assumption of token independence in BERT. Meanwhile, XLnet implements Permutation Language Modeling (PLM)[3], overcoming the problem of unidirectionality of the traditional autoregressive model.

Advantages:
PLM factorizes the joint probability P(x_t | x_(i < t)) and maximizes the log-likelihood over all possible permutations, capturing bidirectional context for each position. Unlike BERT, no Mask is needed, instead using two-stream self-attention[4] for target-aware representations. Although standard parameterization would be reduced to bag-of-word, XLNet tackles this issue by re-parameterization[5] with position. For the transformer, it only considers the hidden representation of the preceding tokens. It improves the performance, especially for the longer text sequence. XLNet thus overcomes the disadvantages of BERT and outperforms it.

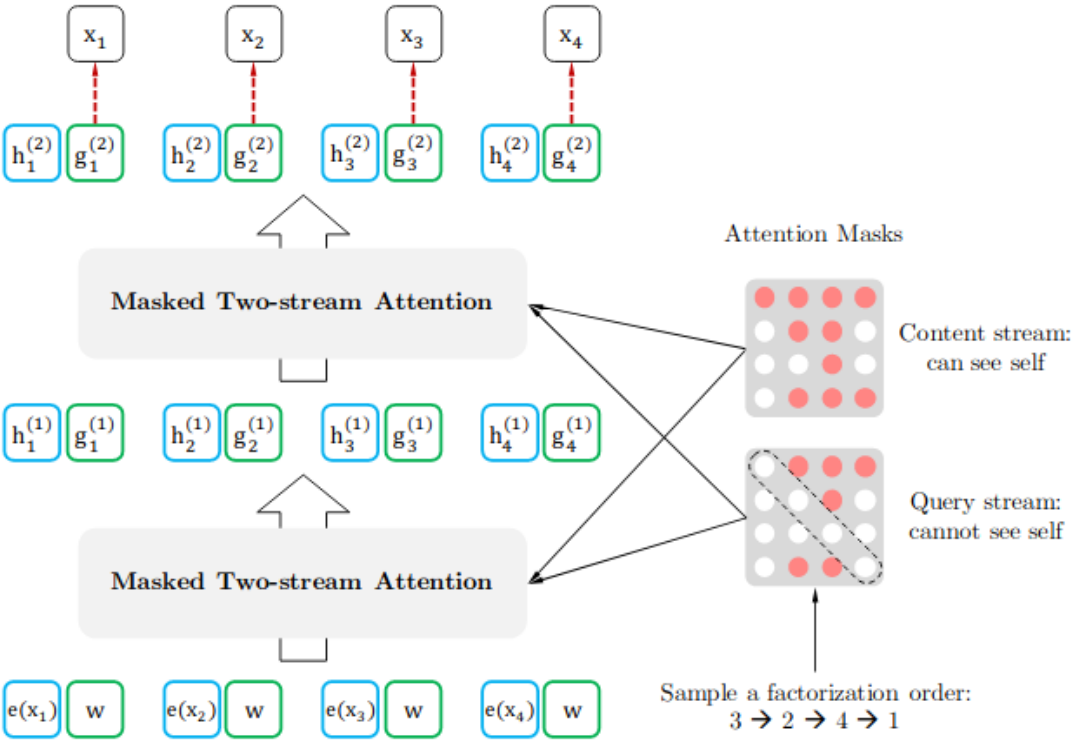
Implementation:
For the XLNet model, we will use the pre-trained model of “XLNet-Base cased” with “XLNetForSequenceClassification" from Huggingface transformer for this sentiment analysis. Since the maximum length of texts in the training dataset is 512, we will set the max_length = 512. According to the reference, the maximum batch size for XLNet-Base with sequence length of 512 is 8. With higher batch size, it is more computationally expensive. In order to fit into the hardware, we set the batch size = 4 and epoch = 4. For the learning rate, we train the model with 2e-5, 3e-5 and 5e-5.

3. **Permutation Language Modeling:**

- Bidirectional context mechanism
- Intuitively, if model parameters are shared across all factorization orders, in expectation, the model will learn to gather information from all positions on both sides.
- According to the graph below,
    1. Taking the order 3 -> 2 -> 4 -> 1, 3 happens to be the first token from the sequence. None of the other tokens contributes to its attention computation as they do not precede 3 in the current permutation.
    2. In the order 2 -> 4 -> 3 -> 1, 3 is preceded by 2 and 4, which contribute to its attention computation.
    3. Applying the same logic, for the order 1 -> 4 -> 2 -> 3 and 4 -> 3 -> 1 -> 2, the corresponding x_(i < t) contributes to the attention computation of x_t.



4. Two-Stream Self-Attention for Target-Aware Representations

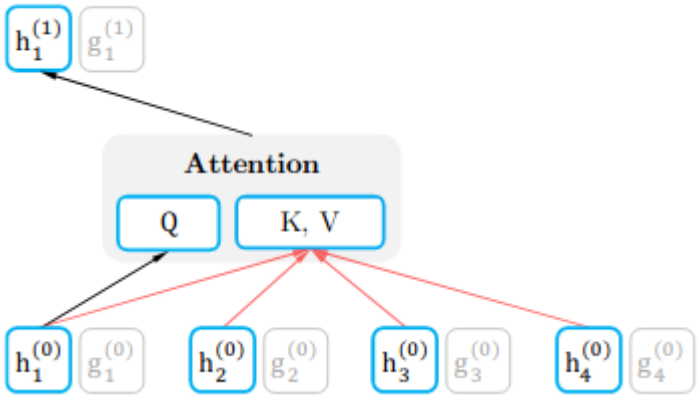


5. **Re-parameterization for the next token distribution to be target:**

- A modified representation g_θ is used, which additionally takes the target position z_t as the input. So, two hidden states are used instead of one:

$$\frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}$$

- The content representation, which is the same as the standard Transformer hidden state. This representation encodes both the context x_(z_(<t)) and the original token x_(z_t).

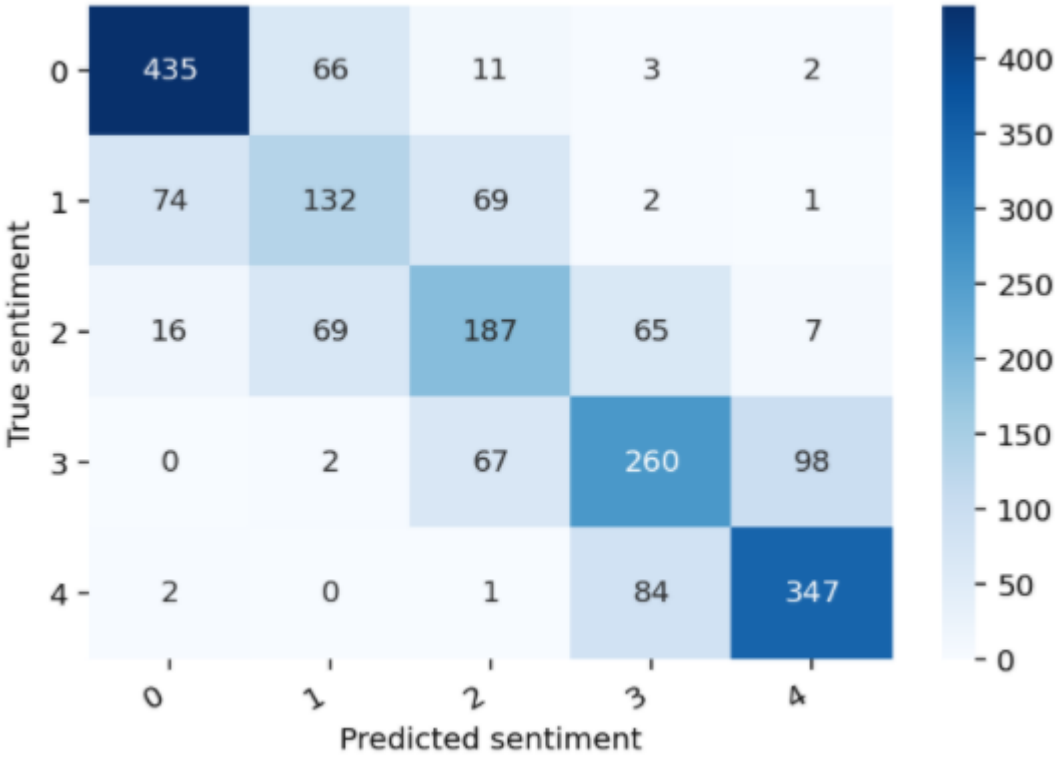


6. **Hyperparameter tuning reference**

System	Seq Length	Max Batch Size
`XLNet-Base`	64	120
...	128	56
...	256	24
...	512	8
`XLNet-Large`	64	16
...	128	8
...	256	2
...	512	1

7. Classification Report and Confusion Matrix of the selected model

	precision	recall	f1-score	support
0	0.83	0.84	0.83	517
1	0.49	0.47	0.48	278
2	0.56	0.54	0.55	344
3	0.63	0.61	0.62	427
4	0.76	0.80	0.78	434
accuracy			0.68	2000
macro avg	0.65	0.65	0.65	2000
weighted avg	0.68	0.68	0.68	2000



Reference

1. Rani Horev (2018). “BERT Explained: State of the art language model for NLP”. Retrieved from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
2. Rohan Jagtap (2020). “XLNet: Autoregressive Pre-Training for Language Understanding”. Retrieved from https://towardsdatascience.com/xlnet-autoregressive-pre-training-for-language-understanding-7ea4e0649710
3. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le (2019). “XLNet: Generalized Autoregressive Pretraining for Language Understanding". Retrieved from https://arxiv.org/abs/1906.08237v2
4. Aishwary V Srinivasan (2019). “XLNet explained in simple terms”. Retrieved from https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c
5. Ghag, S. (2020). “Using XLNet for Sentiment Classification - The Startup. Medium”. Retrieved from https://medium.com/swlh/using-xlnet-for-sentiment-classification-cfa948e65e85
6. “Sentiment Analysis with BERT and Transformers by Hugging Face using PyTorch and Python”. (2020, April 20). Curiousily. Retrieved from https://curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/