ADL HW2 Report
B04705026 資管四 林彥廷

- ## Q1: ELMo Model

  1. ### Training Corpus Processing Pipeline
     - Corpus Preprocessing
       - Randomly choose ⅙ of raw corpus.
       - Tokenize sentence by spliting spaces.
       - Lowercasing all characters.
       - Add <BOS> to the front and <EOS> to the end of sentences.
       - Randomly partition into training and validation set
       - Split sentence longer than 64 words into sub-sentences
       - Padding and Reversing for backward LM are done in PyTorch Dataset
     - Vocabulary
       - Word frequency threshold: 3
       - Only consider extended ASCII character (0-255)
       - Other non-ASCII character are mapped to "OOV_Char"
  2. ### ELMo Architecture and Implementation Details

Embedding	Char	Mean-pooling word-wise to match word embedding Char Embed Dim = 512	Concatenation on embedding dimension
	Word	Word Embed Dim = 300	GloVe
Bi-LSTMP x #LSTM Layers	LSTM	Dim = 2048	Forward and backward weights are not shared and passed separately
	Linear	Linear(512) + Relu	
	Layer Norm	*To stabilize training*	
	Dropout		
Skip-Connection	Previous Char Embedding is added with first LSTMP output		
Adaptive Softmax	Cutoffs = [1*unit, 3*unit, 5*unit] *unit = 2% vocab_size		Weights shared by forward, backward

Mask	Mask out [PAD] losses

Layer normalization was optional in the paper, but I think it was reasonable to adopt normalization ***for stabilizing training process and relatively consistent distribution when ELMo used as an feature extractor.***

Forward pass:
Word Embed. -> Bi-LSTMP -> Bi-LSTMP -> Softmax -> Calculated Loss with Masking
Char Embed. --^ ---------(add)^

3. Hyperparameters of ELMo

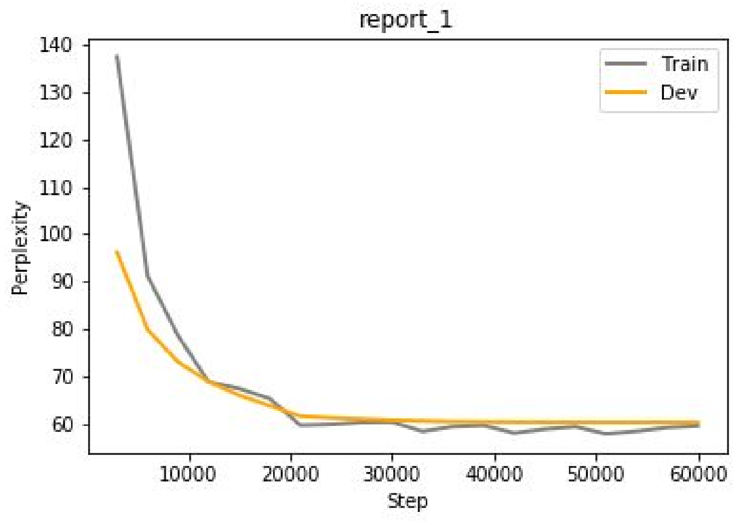
- # LSTM Layers: 2
- LSTM Hidden Dimension: 2048
- LSTM Projection Dimension: 512
- Dropout Rate: 0
- NO skip-connection

Optimizer	Learning Rate	Batch Size	Gradient Clipping Threshold
Adam (Amsgrad)	1e-4	200	L2 norm = 1

4. Training vs Validation Perplexity

Model converged at perplexity=60 even continuing training for a long time.

No obvious sign of overfitting (I think) in part because of low vocab. frequency threshold and the nature of Language Model.



5. Performance of BCN w/ or w/o ELMo

Performance difference (relative 6.03% improvement on validation set) of using contextualized embedding was quite significant.

Embedding \ Accuracy	Validation	Public LB	Perplexity
w/ ELMo	0.4641	0.4624	60.356972
w/o ELMo	0.4377	0.4226	x

Q2: Different Settings of ELMo

Base model: Q1 with 60k steps and no skip-connection / Perplexity was calculated on val. set of LM corpus.

1. Different numbers of training steps:

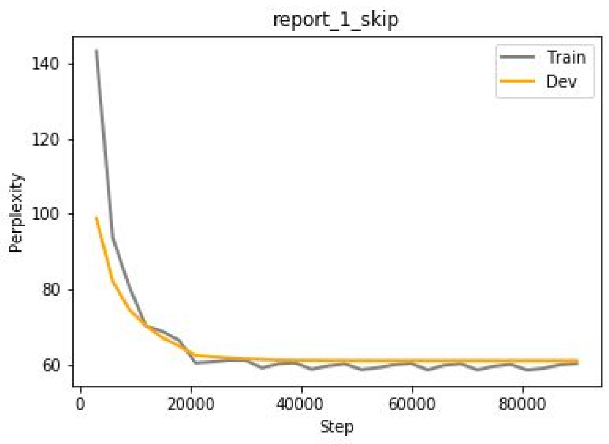
Model at 60k steps outperformed on both language model and downstream task, because there was no sign of overfitting at steps 60k.

Step \ Accuracy	Validation	Public LB	Perplexity
60000	0.4641	0.4624	60.356972
30000	0.4495	0.4533	60.709637

2. Different hyperparameters - Skip-connection

ELMo with *skip-connection which stabilize backprop path* converged **slightly quicker** than that without the trick, and however the performance was not better but **worse**. The main reason was in the ***random-initialized*** char embedding which offered little information for downstream LM task. (Training history of skip-connection model listed below)

Skip Connect \ Accuracy	Validation	Public LB	Perplexity
with**OUT** Skip-connection	0.4641	0.4624	60.356972
WITH Skip-connection	0.4523	0.4488	61.111698



Q2 Conclusion:

“ELMo with better LM performance benefits downstream task”

- ## Q3: Strong Baseline Model

1. ### Model Input

   Word-level contextualized embedding extracted from output of last Transformer encoder in pre-trained Bert (large-cased version).

   In the view on Bert, the input was sub*word-level tokens (WordPiece) together with “positional embedding”* and fed into following layers.

2. ### Model Architecture

   Bert was used as an feature extractor and connected to fully connected layers.

Layer	Note	Output Dimension
Tokens		
Pre-trained BERT	**Fine-tuned** when training	Seq_len x BERT_dim
Mean Pooling	Pooling on **sequential** dimension	BERT_dim
Fully Connected	Linear + relu + batchnorm + dropout	FC_dim=256
Fully Connected	Linear + relu + batchnorm + dropout	FC_dim=64
Fully Connected	Linear	FC_dim=5

3. Hyperparameters

Optimizer	Learning Rate	Batch Size	Weight Decay (L2 penalty)	Dropout Rate	Max Grad. Norm
Adam (Amsgrad)	1e-5	16	1e-4	0.1	1

- ## Q4: Best model

1. Model Input

Word-level contextualized embedding from Bert large cased. Details identical to Q3.

2. Model Architecture

Nearly identical to Q3 strong baseline model except ***concatenating last "NUM BERT LAYER" transformer encoder outputs.***

Layer	Note	Output Dimension
Tokens		
Pre-trained BERT	Fine-tuned when training	Num_bert_layer x Seq_len x BERT_dim
Concatenation		Seq_len x **(BERT_dim x Num_bert_layer)**
Mean Pooling	Pooling on sequential dimension	(BERT_dim x Num_bert_layer)
Fully Connected	Linear + relu + batchnorm + dropout	FC_dim=256
Fully Connected	Linear + relu + batchnorm + dropout	FC_dim=64
Fully Connected	Linear	FC_dim=5
Logits		

3. ### Hyperparameters

Add learning rate warm-up scheduler

Learning Rate Warmup	Warmup Epoch	Initial LR	Target LR
	5	1e-5	5e-5

Optimizer	Batch Size	Weight Decay (L2 penalty)	Dropout Rate	Max Grad. Norm
Adam (Amsgrad)	16	1e-4	0.3	1

4. ### Why Better?

Higher dropout rate also prevented the model from overfitting but the model was harder to train and demanded more time on hyper-parameters tuning.

Without learning rate warmup, model with dropout rate higher than 0.2 was so hard to converge. So learning warm-up do improve the model, and the training process was more smooth.

Concatenation of last four layer slightly outperformed sum(mean) of last four layer which was consistent with the result in this post.

- ## Q5: Different Input Embeddings **In English*

1. ### Char Embedding

    - Pros:
        - Small vocabulary size -> Small embedding model size
        - Few unseen character
        - Support word embedding on word-level OOV
    - Cons:
        - No semantic meaning
        - Hard to evaluate without using downstream tasks

2. Word Embedding
    - Pros:
        - Word is smallest unit with practical meanings
        - Embedding can be evaluated on similarity or other linguistic relationship
    - Cons:
        - Large Vocabulary Size -> Large embedding model size
        - Have to deal with Unseen words(OOV)
3. Byte Pair Encoding
    - Pros:
        - Sub-word level to handle OOV
        - Vocab size can be adjusts depends on users' tradeoffs
    - Cons:
        - BPE merge high frequency neighboring sub-words, which assumes some relationship between these sub-words.
          But this assumption might not be suitable for all kinds of corpus.
        - Need to consider consistency across different languages when using BPE. (eg. Machine Translation)

Char embedding, BPE, and word embedding view words hierarchically from low to high. BPE seems to be a good balanced points regarding tradeoffs between low-level morphological information and rich semantic meaning.

Q6: BERT

1. Describe the tasks that BERT used for pre-training. What benefits do they have comparing to language model pretraining used in ELMo?
    - Masked Language Model

      Randomly mask 15% of tokens of input sequence.
      Corresponding output of last encoder block is used to predict the actual token.

      To align pre-training and fine-tuning, 80% masked token is replaced with [MASK], 10% with random token and the remaining is left unchanged.

- Next Sentence Prediction

  A binary classification task where inputs are two sequence and output is the probability that whether the second sequence directly follows the first sequence in original corpus.

  True examples can easily extracted, and false examples are randomly chosen in corpus. True false examples ratio is 1:1.

- Benefits over ELMo
  - Language Model Improvement

    In Bi-LM, tokens in the second (or later) LSTM layers know the information of its own since *the input from opposite-directional pass has seen the exact same tokens*. Then, the conditioning context is in fact conditioning on its own, which is trivial for the model and **hurts the generalization capacity**.

    MaskLM, on contrary, is conditioned on whole sentence **except itself**. The encoders in the model have no idea of whether to predict, so **token-wise contextual information** was reserved.

  - Semantic Understanding

    BiLM in ELMo is really far from tasks requiring semantic understanding like NLU, NLI or MRC.

    Next sentence prediction forces the model to understand the sentence, which is totally different context in language model predicting next words.

2. Describes in detail how you would formulate the problem in HW1 and apply BERT on it?

   In HW1 the example consists of partial conversation and response candidates.

   I would formulate the problem as binary classification, and apply to BERT sentence prediction pre-training task.
   For each response candidates, partial conversation is concatenate as the

first sentence, and one candidate as the second sentence.
Finally, sort the probability of each response to get the top 10 response.