40.614 Metaheuristic Optimization Project

Zhang Shaozuo 1003756

Abstract

Beam Search is a widely adopted decoding algorithm for Sequence to Sequence language models. This project compared various decoding algorithm including greedy search, beam search and a modified version of beam search.

Introduction

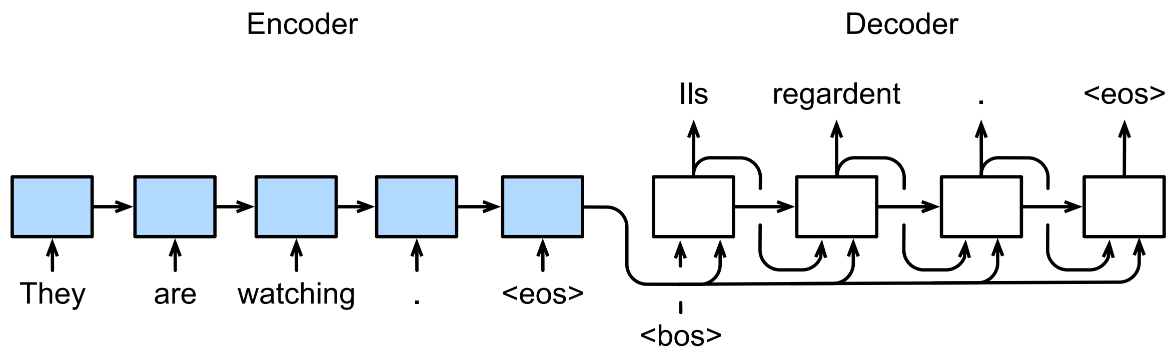

Fig. 1 RNN encoder-decoder. Encoder(blue), Decoder(blue)

Figure 1 shows a standard model architecture with Recurrent Neural Network(RNN) block to solve Neural Machine Translation(NMT) task. In encoding phase, the variable-length input is encoded into a state(in-between encoder and decoder), therefore the model parameters are trained. In decoding phase, the model generate the translated sequence token by token until maximum length or is reached. Beam search take places in decoding phase.

Optimization Problem Formulation

Decoding process could be challenging and costly when the target vocabulary is very large(10,000 more). In decoding phase, output sequence is generated token by token, where the current token is dependent on the previous token.

Let s denote maximum step numbers output sequence, m be the number of steps until , Vocab = $\{x_1, x_2, \dots, x_v\}$ be the set of target vocabulary, the problem could be formulated as

$\max \prod_{x_i, x_{i+1} \in Vocab}^{\min(s,m)} P(x_{i+1}|x_i)$ where $x_0 = \text{<bos>}, x_{last} = \text{<eos>}$

Solution Encoding

Solution = $[x'_0, x'_1, \dots, x'_{\min(s,m)}]$, where $x_i \in \text{Target Vocab Indices}$, in this case target vocabulary size is 21,062.

Search Operator

Since each token is dependent on previous token(s), update any $x_i \to x_{i'}$ will affect all the token after i. In other words, if $x_i \to x_{i'}$ token x_{i+1} probably will not be the optimal token (or even worse) at position $i+1$ due to the state change, chain reactions will take place for all the tokens after i, resulting in very bad output sequence.

Dataset and Preprocess

Training: Tab-delimited Bilingual English-French Sentence Pairs

The contains 167,130 pairs of sentences in both languages. One advantage of this dataset is that for some English sequence, there are multiple corresponding French Sequences. This allows the model to learn that there could be many ways to express the same meaning.

```
'''Examples'''
# [English]: You know that your English is good when people stop complimenting
you on how good your English is.
# [France]: Vous savez que votre anglais est bon lorsque les gens arrêtent de
vous en complimenter.

# [English]: Chinese officials say economic growth has dropped to a three-year
low because of the world economy.
# Les officiels chinois disent que la croissance économique est tombée à son plus
bas niveau depuis trois ans en raison de l'économie mondiale.

# [English]: Girls begin puberty around the ages of ten to eleven, and boys
around the ages of eleven to twelve.
# Les filles commencent leur puberté aux environs de dix ou onze ans et les
garçons autour de onze ou douze.

# [English]: I can't believe that you aren't at least willing to consider the
possibility of other alternatives.
# [France]: Je n'arrive pas à croire que vous ne soyez pas au moins disposé à
envisager d'autres possibilités.

# [English]: I can't believe that you aren't at least willing to consider the
possibility of other alternatives.
# [France]: Je n'arrive pas à croire que vous ne soyez pas au moins disposée à
envisager d'autres possibilités.

# [English]: I can't believe that you aren't at least willing to consider the
possibility of other alternatives.
# [France]:Je n'arrive pas à croire que vous ne soyez pas au moins disposés à
envisager d'autres possibilités.
```

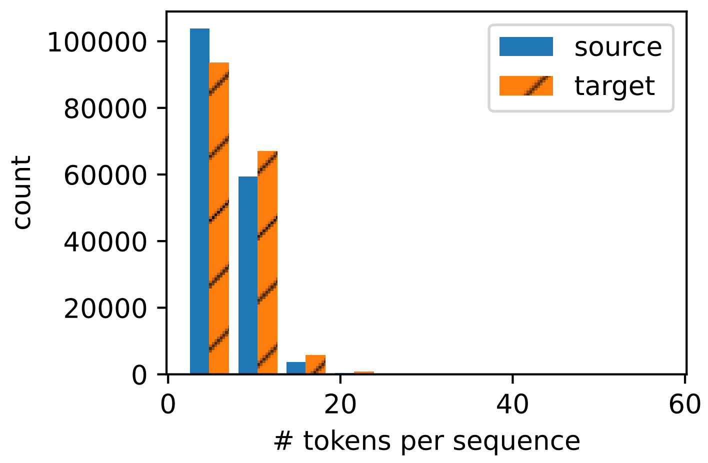

Fig2: Visualization of Tab-delimited Bilingual English-French Sentence Pairs

Average lengths are 7.2 words/sent, 7.6 words/sent for source and target respectively.

Validation: Multi30k

Seconding training dataset is Multi30k dataset, which contains less sentence pairs. Multi30k provides validation datasets that are used to evaluate the performance in decoding phase. To augment the dataset, Multi30k training set is incorporated in encoder training as well,

```
'''Description'''
# Training
# (en) 29000 sentences, 377534 words, 13.0 words/sent
# (fr) 29000 sentences, 409845 words, 14.1 words/sent
# Validation
# (en) 1014 sentences, 13308 words, 13.1 words/sent
# (fr) 1014 sentences, 14381 words, 14.2 words/sent
```

Data Preprocess

Data preprocess take places before training the model. Some methods adopted are

1. **Tokenization:** Tokenization is necessary as the model takes the input sequence token by token. Embedding vector is also performed on word level, not on sentence level

```
I am a student. => ['I','am','a','student','.']
```

2. **Variance-Length:** Set number of steps as k. Sequence longer than k will be truncated, shorter than k will be padded to k length.
3. **Build batch:** A group of sequence pairs form a batch, the model parameters is updated batch by batch

Training

Model Architecture:

```
EncoderDecoder(
  (encoder): Seq2SeqEncoder(
    (embedding): Embedding(12890, 32)
    (rnn): GRU(32, 32, num_layers=2, dropout=0.1)
  )
  (decoder): Seq2SeqDecoder(
    (embedding): Embedding(21062, 32)
    (rnn): GRU(64, 32, num_layers=2, dropout=0.1)
    (dense): Linear(in_features=32, out_features=21062, bias=True)
  )
)
```

12,890 is the size of source vocabulary, 21,062 is the target vocabulary

Experiment Setting:

Embedding size: 32. Length of the word vector

Number of Hidden layers: 32

Batch Size: 32

Number of Steps: 14, maximum length of output sequence

Number of epochs: 100

Loss Function: Masked SoftMax Cross Entropy Loss

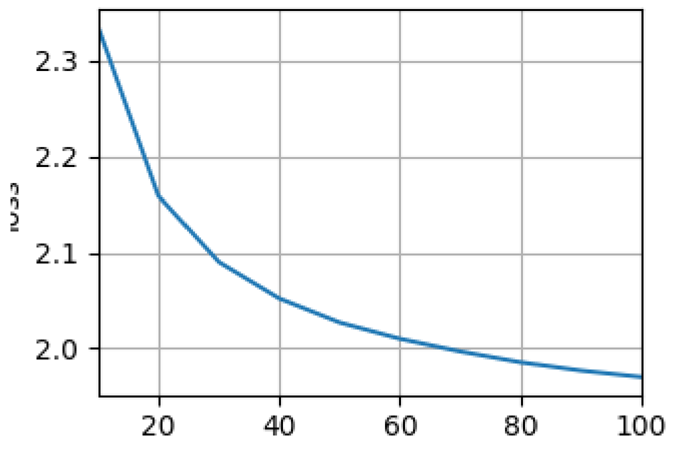

Fig 3: Training Process

Evaluation

Multi30k validation set is used for evaluation. Each validation English sequence and its predicted French Sequence adopts (Bilingual Evaluation Understudy)BLEU Score as the criteria. We obtain BLEU score 1.0 if the prediction and ground truth perfectly match, 0 if completely does not match.

Greedy Search

Given validation sample *src_sentence* and trained model *net*

1. **Encoding:** textual information into digital vectors

```
src_tokens = src_vocab[src_sentence.split(' ')] + [src_vocab['<eos>']]
```

2. **Initialization**: First generated token *dec_X*, initial hidden state *dec_state*
3. **Iteration**: Do iterations for *num_steps* times, in each loop

```
output_Y, dec_state = net.decoder(dec_X, dec_state)
output_token = argmax(output_Y)
# End loop if <eos> reached
output_seq.append(output_token)
```

In step 3, *dec_state* is constantly updated in each iteration, showing current predicted token is dependent on the previous tokens via *dec_state*. Tensor *output_Y* has 21,062 nodes(size of target vocabulary), indicating the weights of each token. For example, if *output_Y*=[-0.2, 3.3, 0.5] for vocabulary $[A, B, C]$, the output token of Greedy Search is token B.

Figure 4 shows the BLEU Score distribution on Multi30k validation set. The average score and variance are reported as 0.415, 0.0237 respectively.

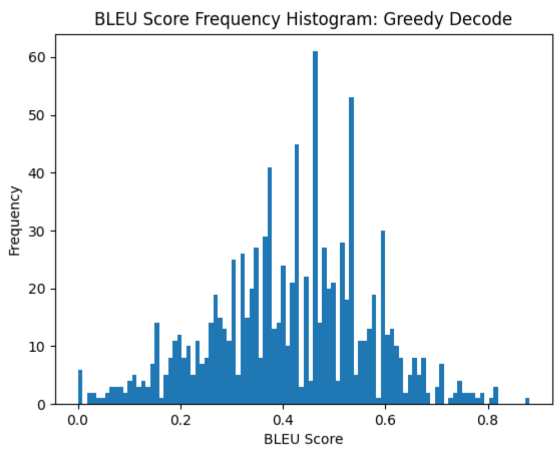

Fig 4: BLEU Score distribution: Greedy Decode

Beam Search

Given validation sample *src_sentence* and trained model *net*

1. **Encoding:** textual information into digital vectors

```
src_tokens = src_vocab[src_sentence.split(' ')] + [src_vocab['<eos>']]
```

2. **Initialization**: First generated tokens *dec_X_1, dec_X_2*, initial hidden state *dec_state*
3. **Dual Iteration**: Do iterations for *beam_size* times, in each loop:

   3.1. Do iterations for *num_steps*, in each loop:

   ```
   output_Y, dec_state = net.decoder(dec_X, dec_state)
   output_token = argmax(output_Y)
   # End loop if <eos> reached
   output_seq.append(output_token)
   ```

```
output_seqs.append(output_seq)
```

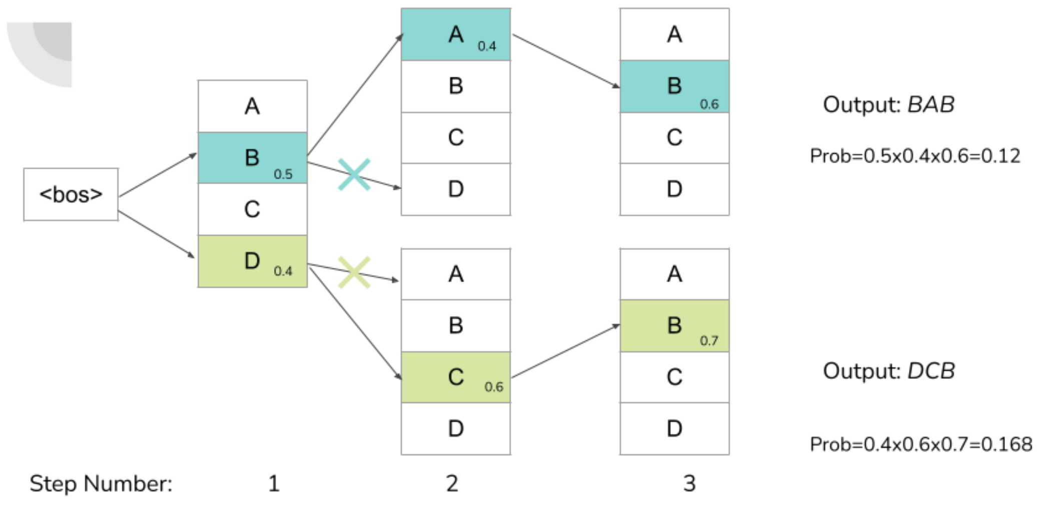

Fig 5: Beam Search illustration, Beam Size 2

Advantage: Greedy Search can not guarantee the output sequence has the highest probability. Fig 5 shows an example outperforming Greedy Search. In Step 1, Beam Search selects top 2 tokens as the candidates, so the search mechanism explores more possibilities. Even though the model has more confidence to generate token *B* than token *D* in Step 1, the green path ultimately generates more convincing output sequence. In this case, beam search outperforms Greedy Search.

It is expected that the performance benefits from larger beam size, at the price of more computational cost.

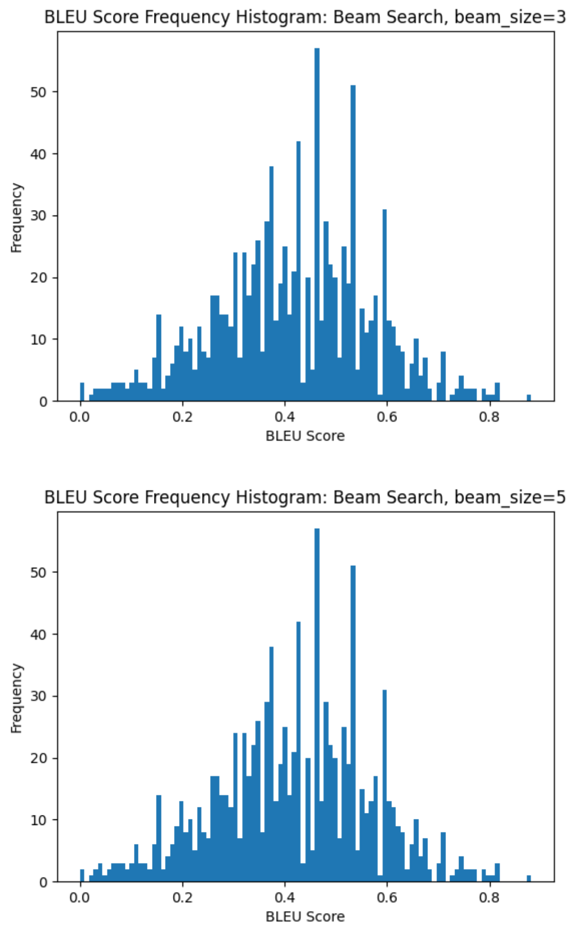

Fig 6: BLEU Score distribution: Beam Search

The BELU Score distributions are summarized as below:

Beam Size	Average BLEU Score	Variance
1	0.415	0.0237
2	0.417	0.0232
3	0.417	0.0230
5	0.418	0.0229
10	0.418	0.0228

Table 1: Statistics with varying Beam Size

Beam Search with Simulated Annealing

Beam Search, as observed, has limited improvement compared to Greedy Search. One possible reason is that although Beam Search explores more paths, the exploration takes place only at the first position. As shown in Figure 5, decoding process still generates the token with highest probability after Step 1.

One possible solution is to apply Simulated Annealing on Beam Search.

Given validation sample *src_sentence* and trained model *net*

1. **Encoding:** textual information into digital vectors

```
src_tokens = src_vocab[src_sentence.split(' ')] + [src_vocab['<eos>']]
```

2. **Initialization**: First generated tokens *dec_X_1, dec_X_2*, initial hidden state *dec_state*
3. **Dual Iteration**: Do iterations for *max_iter* times, in each loop:

```
optimal_sol = {'seq':[], score: Integer.MIN_VALUE}
```

3.1. Do iterations for *num_steps*, in each loop:

Focus one top-k indices out of output_Y, randomly sample one word from this subset

```
output_Y, dec_state = net.decoder(dec_X, dec_state)
Y5 = torch.topk(output_Y,5)
pred = random.choices(Y_idxs, weights=Yv_weights, k=1)[0].item()
# End loop if <eos> reached
seq_try.append(pred)
```

If the output sequence achieves higher score in optimal_sol, do update.

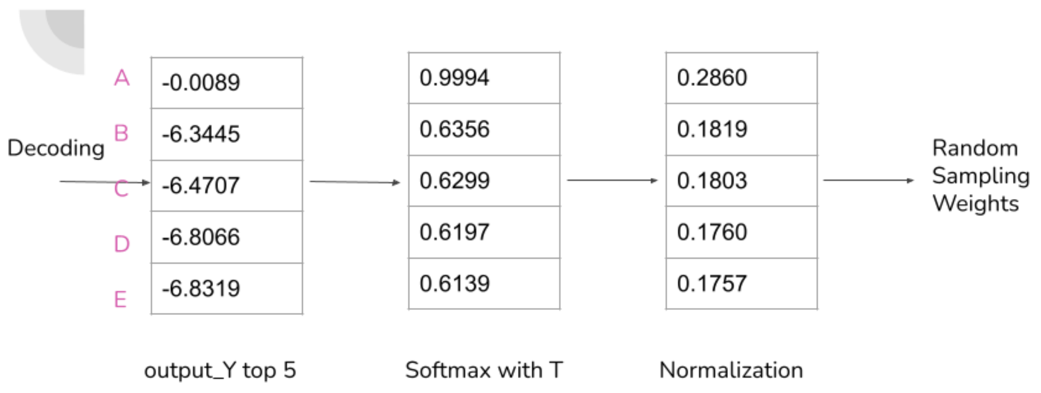

Fig 7: Illustration of Simulated Anneal temperature

Simulated Annealing is adopted in Random Sampling. In each iterations, the temperature decreases with generations going on token by token. It is expected that decoding process explores more possibilities at the beginning of generations, and behaves more constrained when the generations approaches the end.

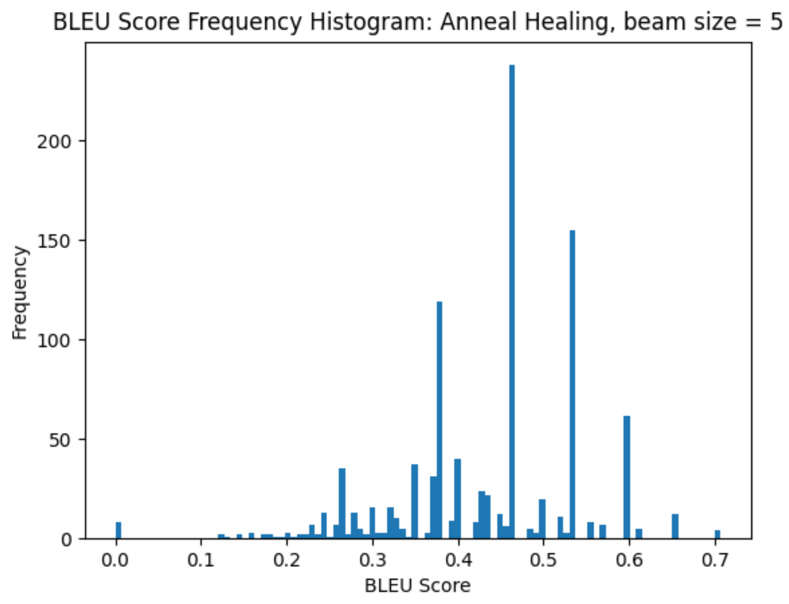

Fig 8: BLEU Score distribution: Simulated Annealing

The average score and variance are reported as 0.432, 0.0119 respectively.

References

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). *Bleu: A method for automatic evaluation of Machine Translation*. ACL Anthology. Retrieved April 21, 2022, from https://aclanthology.org/P02-1040/

Zhang, A. (2022, March 25). *modern recurrent neural networks*¶. 9. Modern Recurrent Neural Networks - Dive into Deep Learning 0.17.5 documentation. Retrieved April 21, 2022, from https://d2l.ai/chapter_recurrent-modern/index.html