Greedy and Beam Search Approach in Neural Machine Translation (NMT)

Maria Khelli - 13520115[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520115@std.stei.itb.ac.id

***Abstract*—Neural Machine Translation is a machine translation implemented using a neural network. The most common neural network used is the sequence-to-sequence model. The model will receive a sequence as an input, and pass it to the encoder, then the encoder will produce a fixed-length vector that represents all the words in the data. The vector will be fed to the decoder to produce an output sequence. This output is usually generated using the greedy method—taking the most probable word in every time step. The drawback of the greedy method is that its search space is very narrow (only consider one path). On the other hand, there is another searching algorithm called beam search which considers more paths than greedy. This paper compared two methods to see which one gives a better overall result. The author found that, overall, greedy method is preferable since the result are similar to the beam search method, but with lower computational time.**

Keywords—neural network, machine translation, greedy, beam search

I. Introduction

A neural network is a learning-based agent artificial intelligence built from a series of algorithms to recognize patterns and relationships in a dataset. One of the applications of neural networks is neural machine translation (NMT). It is a more recent approach to machine translation. Unlike statistical machine translation (SMT), NMT uses the trained neural network to predict the translated output [1].

The Neural Machine Translation method has also changed Google Translate's Phrase-Based Machine Translation (PBMT). PBMT breaks a sentence into words or phrases and translates them individually [2]. In contrast, a neural network considers the entire sentence in its translation process.

The most common neural network architecture in machine translation problems is encoder-decoder architecture [1]. This architecture is also known as the sequence-to-sequence model because the length of input and output is not fixed—hence, sequence, not vector. This type of network differs from a regular neural network which produces a fixed number of outputs.

Although the sequence-to-sequence model seems to not involve fixed-length vectors, this encoder-decoder pair is built from the conjoined sequence-to-vector model and vector-to-sequence model. Namely, the encoder that maps a sequence to a fixed-length vector and the decoder that receives the encoder's vector and maps it into a sequence. Both models will be trained together to minimize its loss and maximize the conditional probability of a given input [3].

This paper emphasized on the decoder's side after it received the encoder's context vector. In each time step, the decoder will produce a fixed-length vector of word probabilities given the vectors as well as its previous predicted words. The general approach in choosing the next probable word is using a greedy algorithm, that is, taking the word with the highest probability.

However, one drawback of the greedy method is that its search space is very narrow. Consequently, making it does not consider other paths which may give a better result. Therefore, another method should be considered. The author used a breadth-first beam search which considers the n-number of words—called beamwidth—in every depth layer (in this context: time step) [4].

For performance measurement, the author has done semantics analysis, to see qualitatively, which method output a translation that makes more sense. The trade-off in computation time would also be considered as a quantitative measurement.

II. Methodology

In this section, we will discuss the implemented neural network as well as the algorithm in the model's inference which are compared together (greedy and beam search).

A. Recurrent Neural Network (RNN)

The encoder and decoder architecture used were based on a recurrent neural network (RNN). A recurrent neural network is a neural network that consists of hidden state $\boldsymbol{h}$ and optional output $\boldsymbol{y}$ which is produced from a variable-length sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_t)$. At each time step t, the hidden state is updated by

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, x_t) \quad (1)$$

where f is a non-linear activation function [3]. A simple recurrent neural network will use the tanh activation function with the equation of

$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t) \quad (2)$$

where $\boldsymbol{W}$ is the weight.

However, this standard RNN would suffer from a vanishing gradient problem, making it forget its early information (shortterm memory). Therefore, the writer used a gated recurrent unit (GRU) which is an improved version of a simple RNN.

The difference between GRU and standard RNN is that GRU uses an update gate and reset gate. These are the vectors that will decide what information should be passed to the next unit [5]. They are able to retain early information better than a simple RNN because they will erase the information that is irrelevant to the predictions.

Mathematically, there are four important equations in a single unit of the gated recurrent unit. The first one is the update gate

$$\boldsymbol{z}_t = \boldsymbol{\sigma}\left(\boldsymbol{W}^{(z)}\boldsymbol{x}_t + \boldsymbol{U}^{(z)}h_{t-1}\right) \tag{3}$$

which helps the model determine the information it needs to carry [5]. The second one is the reset gate

$$\boldsymbol{r}_t = \boldsymbol{\sigma}\left(\boldsymbol{W}^{(r)}x_t + \boldsymbol{U}^{(r)}\boldsymbol{h}_{t-1}\right) \tag{4}$$

that is used by the model to decide what information to forget [5]. The difference between Equations 3 and 4 is only in the weights. The third one is

$$\boldsymbol{h}'_t = \tanh(\boldsymbol{W}x_t + \boldsymbol{r}_t \odot \boldsymbol{U}\boldsymbol{h}_{t-1}) \tag{5}$$

this will compute current memory content and the symbol $\odot$ is a Hadamard-product (element-wise multiplication). Lastly, the fourth equation will decide what is the output of the current hidden state (memory).

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}'_t \tag{6}$$

The output of each time step t in an RNN unit is the conditional distribution $p(x_t|x_{t-1}, \dots, x_1)$. In other words, the previous output will affect the next output. By combining these probabilities, we can compute the probability of sequence $\mathbf{x}$ using

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \dots, x_1) \tag{7}$$

which can be used to predict a new sequence.

B. Encoder-Decoder

The encoder on the implemented neural network consisted of only one layer of GRU.

On the other side, the decoder used is an attention decoder which is also using one layer of GRU unit, but with the addition of considering the attention vector. This allows the decoder to focus on a different part of the encoder's outputs for every time step. Each time the model generates a word, it searches for a position where the most relevant information is concentrated [1]. That way, the context vector does not need to encode the entire sentence [6].

Visually, the encoder layers stack is

Embedding
GRU

Fig 2.1 Encoder layers stack

and the decoder layers stack is

Embedding
Attention
Dropout
ReLU
GRU
SoftMax

Fig 2.2 Decoder layers stack

Both encoder and decoder had a hidden size of 512. In addition to that, the writer used stochastic gradient descent as optimizer and negative likelihood loss to compute the model's loss.

C. Training Method

Since the inferences were compared between greedy and beam search, the training method was not taken from either of them to avoid weight bias. Hence, the training method used is teacher forcing. Teacher forcing is a strategy to train a recurrent neural network that uses the correct input (ground truth) instead of the prior model's prediction [7].

D. Greedy

Greedy is an algorithm that solves the problem step by step so that in every step the program chooses the best solution without considering the path ahead with the hopes that it will converge to the global optimum [8]. There are six components to define a greedy algorithm.

1. Candidate set: contains the candidates that are chosen in each step.
2. Solution set: contains the chosen candidates.
3. Solution function: determines whether the current set has already achieved the solution.
4. Selection function: criteria or rules to choose in each step based on the greedy characteristic.
5. Feasibility function: check whether the chosen candidate is feasible or not.
6. Objective function: the result we hope to see (maximum or minimum).

The mapped elements from greedy components to this paper's problem—choosing the next probable word—is

1. Candidate set: the set of all vocabulary in the trained data.
2. Solution set: the set of words in the translated sentence.
3. Solution function: checks if the last chosen word is the "<EOS>" token.
4. Selection function: picks the word with the highest probability.

5. Feasibility function: checks whether the current solution set plus chosen candidate has exceeded maximum length or not.
6. Objective function: maximize Equation 7.

E. Beam Search

There are two kinds of beam search: best-first beam search and breadth-first search. Best-first beam search runs just like a best-first search, but when the queue grows beyond a predetermined size limit, only n highest quality nodes are retained. On the other hand, breadth-first beam search works like breadth-first search, but only a fixed number of nodes are kept in each depth layer [4].

In this writing, we used the second type of beam search. From this point on, beam search will refer to breadth-first beam search. The beam search algorithm works such that

a. From the root node, pick n highest probable words where n is the predetermined beam width or beam size. For each word, create a new decoder node.
b. For each n decoder node created, consider only n highest probable words for the next output. This will create new n^2 child nodes to put in a node list.

   Note that the original algorithm considers all words, then filters them. A heuristic modification is needed for a computational reason.
c. After producing n^2 child nodes, the algorithm is going to filter n nodes with the highest objective function. This way, only n nodes remain in the node list.
d. Repeat steps b and c until maximum depth or maximum sentence length is reached. If the algorithm encounters an end-of-sentence (EOS) token, then it is put into an answer list.
e. From the answer list, choose the highest node with a maximum objective function.

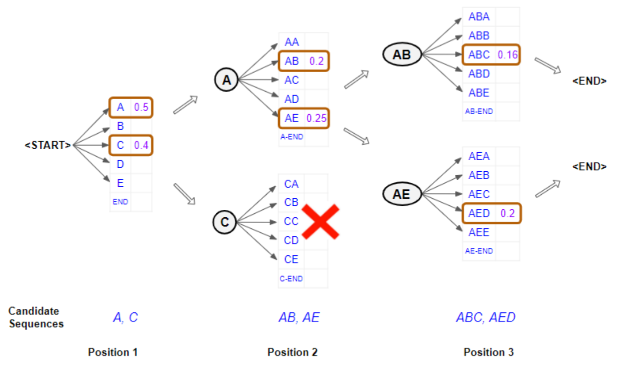

Fig 2.3 Beam search illustration

There are several things to note when calculating beam search node priority. The first one is the metrics to measure the highest probable word. Instead of considering probability which ranged from [0, 1], we use log-likelihood that ranged from $(-\infty, 0]$. Accordingly, the most probable word is the highest value of log-likelihood (near zero).

The second one is the objective function. To decide the best sequence, we use the sum of log probability, defined as

$$F = \arg\max \sum_{y=1}^{T_y} \log P(y_t \mid x, y_1, \ldots, y_{t-1}) \quad (8)$$

where T_y is the length of the sequence (or depth of a node). We also did use normalization, that is, dividing the objective function by T_y to try avoiding short sentence bias. In addition to that, we also use the EOS token penalty. It was added to the objective function when the EOS token is met. EOS token penalty is defined as

$$p = -\gamma \frac{(input\ length)}{T_y} \quad (9)$$

Gamma symbol γ is an arbitrary number that is chosen. For this paper, the γ was divided into piecewise function

$$\gamma = \begin{cases} 7.5 \leq t \leq 9, & r > 1.1 \\ 5 \leq t \leq 6, r \leq 1.1 \end{cases} \quad (10)$$

where t is the random float number between the range and r is the ratio of input length to the length of the sequence.

The rough pseudocode of beam search will be shown in the next paragraph.

```
function BeamSearch(n: beam_width, encoder_output)
→ sequence of answers
{ Returns a node answer (can be tracked to ancestors) that
have maximum objective function }
Algorithm
n ← beam width
parents ← {}; children ← {}
answers ← {}
root ← create new beam node
while parents not empty do
    for each node in parents
        if node depth < max length and word != EOS do
            log_prob ← decoder(inputs)
            n_highest ← take_highest(log_prob, n)
            for each proba in n_highest
                create new node for proba
                put it in children
            endfor
        else
            put node in answers
        endif
    endfor
    parents ← take_highest(children, n)
    children ← {}
endwhile
return max(answers)
```

III. Experiments

A. Data Descriptions

In this paper, the data used is the translation from English to Indonesian taken from Tatoeba [10]. We also limited the data to reduce potential bad results because of the neural network. Those limitations are

1. Sentence lengths in either English or Indonesian were limited to 15.
2. The English translation taken should only start with the word *i, you, we, they, he, she,* and *it.*

B. Data Preprocessing

Since the computer can not process the alphabet directly, we transformed the sentences into numbers first. We represented each word as a one-hot vector, a zero vector except the index of the represented word was valued as one.

To uniform the data, sentences were lower-cased and regular expression was used. Regex was used to remove non-alphabetic characters and to make space on both sides of punctuations. For example, the sentence

He's still young.

was transformed to

he s still young .

C. Hyperparameters

As had been said before, both encoder and decoder had a hidden size of 512. It would be a control variable and would not be tweaked. For the hyperparameters, the writer only adjusted the training iterations.

The purpose of tweaking training iterations is to see whether beam search will give good performance in small iterations or not. Since beam search is more computationally expensive compared to greedy, it is reasonable to trade the inference computation for training computation depending on the overall purpose of the use case. Training iterations that were tested are 5,000; 10,000; 25,000; 50,000; and 100,000.

IV. Results and Discussions

A. Training Iterations

The result of training iterations is tabulated as

Iterations	Training loss	Duration
5,000	3.4451	1m 27s
10,000	2.6111	2m 59s
25,000	1.3333	7m 7s
50,000	0.5103	14m 26s
100,000	0.1457	29m 1s

Table 4.1 Training iteration and loss data

Note that the training is done with a GPU. Computation time using a CPU might differ. In the next section, we compared the performance by doing top-down analysis from the highest training iterations.

B. 100,000 Training Iterations

Firstly, we will compare beam search with width of 3 to greedy. Overall, greedy really did a good job because it predicted most of the translation correctly. From 30 sample of randomly chosen sentence, it only mistranslates 3 sentences, whereas for beam search, it mistranslate about 20 sentences. Several cases of mistranslate are

1. *I could see the happiness in her eyes.*

Correct translation: "Aku bisa melihat kebahagiaan di matanya."

The beam search translated it into, "Aku bisa di matanya." This is semantically incorrect because the translated output has different meaning than the original sentence. In English, it is roughly translated into, "I can in her eyes."

Conversely, greedy method correctly translated it.

2. *I met my teacher on the way to the station.*

Correct translation: "Saya bertemu dengan guru saya dalam perjalanan ke stasiun."

Beam search translated it into, "Saya bertemu dengan guru." In English, it is roughly translated into, "I met a teacher." This translation made more sense compared to the first example, but it is not 100-percent semantically correct because it had lost some of its information.

In the contrary, greedy method translation is semantically closer to the correct translation. It translates the sentence into, "Saya bertemu dengan guru saya." In English, "I met my teacher." Compared to beam search, this translation retains more information ("my teacher", not only "teacher").

However, both had lost the "on the way to station" information.

3. *I had to make a list of things i needed to do.*

Correct translation: "Aku harus membuat daftar tentang hal apa saja yang perlu aku lakukan."

The beam search algorithm translated it into, "Aku membuat daftar tentang aku tidak pernah membuat." This translation is semantically incorrect. In English, "I made a list of things I never made."

Accordingly, greedy algorithm translated it into, "Aku membuat daftar tentang hal lain yang perlu aku lakukan," which translated back into "I made a list of other things I needed to do."

Although both translations are not the same as the correct one, greedy algorithm's translation retains all the meaning in the sentence, except the "must" part.

The author had also tried beam width of 5, 20, and 40, but the algorithm suffers from short sentence bias even though normalization and EOS penalty had been applied.

C. 50,000 Training Iterations

Similar to previous section, this part will elaborate the performance of greedy and beam search with width of 3. Overall, from 30 samples of sentences, greedy got two-third correct while beam search got similar performance as in 100,000 training iterations. Interestingly, there are sentences that mistakenly translated (word by word), but essentially means the same thing. For instance,

1. *Is Tom well?*

Correct translation: “Apa tom sehat?”

The beam search translated it into, “Apa tom baik baik baik saja ?” This is semantically similar to the correct translation, except that it repeats the word *baik* three times. In English, “baik-baik saja” is equivalent to the word *well*.

However, greedy method had still better performance because it got the translation correct word by word.

2. *You are still green.*

Correct translation: “Kamu masih bau kencur.”

In this case, both beam search and greedy translated it into, “Kamu masih muda,” which semantically means the same as the correct translation. The word “bau kencur” is a metaphor for “muda” or *young*.

In this attempt, the author had also tried beam width of 5, 20, and 40, but the algorithm suffers from the same bias as previous section.

D. 25,000 Training Iterations

The specification of beam search and greedy is still the same as previous section. For overall performance, greedy and beam search had approximately the same performance. Although, beam search might do slightly better. Some examples for translated sentence are

1. *I need more information.*

Correct translation: “Aku butuh informasi lagi.”

Beam search translated the sentence into, “Saya butuh informasi,” which essentially has the same meaning as the correct translation, except without emphasize on the word *more*.

On the other side, greedy algorithm translated this sentence into, “Saya butuh informasi. Dia lagi.” This means the algorithm did not stop when it has found a period. The model was not trained enough to have high probability of EOS token after a period—in this paper, the data only consist of one sentence.

2. *We saw Jane swimming across the river*

Correct translation: “Kami melihat jane berenang menyeberangi sungai.”

The beam search algorithm translated the sentence into, “Kami berhasil sungai.” This is semantically incorrect. But interestingly, it behaved similar to the first example. It has a tendency to produce shorter sentence.

In the contrary, greedy method translation is, “Kami melihat berenang sungai sungai sungai,” which is also semantically incorrect. However, compared to beam search, greedy has a tendency to produce repeating word. Therefore, outputting longer sentence.

The changing of beam size to 5, 10, or 20 was not making the performance of the beam search better, but it no longer suffers badly from short sentence bias likewise in the previous examples.

E. 10,000 Training Iterations

The last section that will be elaborated is 10,000 iterations because at this point, the bad result is coming towards the models, not the inference algorithm. Both beam search and greedy performed poorly with this hyperparameter. Nevertheless, the writer noticed that greedy search tends to repeat the same word over and over again while beam search avoids this. One example would be

1. *I wanna visit paris New York and Tokyo someday.*

Correct translation: “Saya ingin mengunjungi paris new york dan tokyo suatu hari nanti.”

The beam search translated it into, “Saya akan ke tokyo dalam bahasa inggris dan menjadi seorang diri.” Although the meaning is different, this sentence made more sense than greedy translation result which is “Saya akan membuat membuat dan menjadi makan siang dan mary dan menjadi bahasa inggris dan,” and it repeats too many *dan* word.

Similar to previous result, changing beam size did not give much impact to the performance.

V. Conclusions

To conclude, beam search algorithm does not give significant better result than greedy. Overall, considering the computation time and performance trade-off, one might prefer greedy rather than beam-search method. With high beam width, beam search can be very computationally expensive compared to greedy algorithm.

Theoritically, beam search considers more paths than greedy. Hence, choosing an appropriate weighing function or objective function should be done in order to have a better performance. This is needed to make sure that we are not filtering out high-quality nodes. In addition to that, it should avoid some biases, such as a length bias that the beam search suffered in this paper.

Further research can be done with another searching method such as hill climbing, A*, or simulated annealing. Furthermore, other encoder-decoder architecture or other training methods (e.g., training with beam search) should be tested to give beam-search algorithm the benefit of the doubt. The weight from this paper’s model might not be suitable for the beam-search algorithm. Hence, it did not align with the theory.

Acknowledgment

The author would like to gratitude for the help provided by Dr. Masayu Leylia Khodra, S.T, M.T., and Dr. Nur Ulfa Maulidevi, S.T., M. Sc. as Algorithm Strategies Instructor, Dr. Rinaldi Munir, M. T. as Algorithm Strategies Coordinator, and Christian Hadiwinoto, Ph.D. for giving me the idea of this paper. The author would also like to thank friends and family who support the author throughout the study at Institut Teknologi Bandung.

Repository

A code implementation in this paper can be accessed here.

References

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate.

[2] Le, Q. V., & Schuster, M. (2016, September 27). *Google AI.* Retrieved from Google AI Blog: https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

[3] Cho, et al. (2014). Learning Phrase Representations using RNN Encoder–Decoder.

[4] Wilt, C., Thayer, J., & Ruml, W. (2010). A Comparison of Greedy Search Algorithms. *Association for the Advancement of Artificial Intelligence.*

[5] Kostadinov, S. (2017, December 6). *Understanding GRU Networks*. Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[6] Robertson, S. (n.d.). *PyTorch*. Retrieved from https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html#the-seq2seq-model

[7] Brownlee, J. (2021, April 21). *What is Teacher Forcing in Neural Networks?*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/

[8] Doshi, K. (2021, April 2). *Foundations of NLP Explained Visually: Beam Search, How It Works*. Retrieved from Towards Data Science: https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24

[9] Munir, R. (2022). *Informatika ITB*. Retrieved from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[10] Tatoeba. (n.d.). Retrieved from Tatoeba: https://tatoeba.org/en/downloads

Declaration

I hereby declare that this research paper is my own writing, not an adaptation, translation, or plagiarism.

Bandung, 20 Mei 2022

[illegible] KHELLI

Maria Khelli
13520115