Experimental Study Proposal

1. Introduction and background:

With the development of technology, machine translation has been more and more crucial in breaking down language barriers and facilitating communication and understanding across different linguistic and cultural groups. As a key component in machine translation systems, the decoder is responsible for turning machine-generated translations into human-readable, high-quality text, which is essential for making machine translation a practical and valuable tool in various domains. In the current decoding algorithm, beam search is a popular one used in various natural language processing tasks with its particular advantage for generating sequences, such as translations, in that it strikes a balance between quality and computational efficiency. However, it also has some unignorable limitations. Firstly, a beam search decoder could lead to high computational cost: Using a larger beam width requires more computational resources, which can be a limitation in applications where speed is crucial. Besides, it always lacks optimality: It relies on a fixed beam width and may miss out on rare or less probable but better translation options. It doesn't guarantee the best possible translation. Those two crucial insufficiency can make it challenging to strike the right balance between quality and efficiency. **Therefore, one problem raised and urgently needed to solve is how to conduct an effective decoding algorithm to reduce the computational cost and improve the optimality.** This research aims to apply Monte Carlo algorithms to solve both suboptimal and computational cost problems in beam search. Undoubtedly, the investigation of this research is very critical because it could solve some of the problems in

the current solution and help to achieve a better and more usable decoding performance in machine translation tasks.

2. Methodology:

To solve the problem raised in this research, we will apply monte carlo algorithms to solve both suboptimal and computational cost problems in beam search. To be more specific, we will use the metropolis hasting algorithm to find the most likely or optimal sequence according to a probabilistic model, also decreasing the beam size to reduce the computational cost.

Software: Python 3

Algorithm:

1. Beam Search Decoder: Using greedy search to linearly solve the decoding problem by finding the current optimal translation.
2. Metropolis Hasting Algorithm: When we run beam search, we will only accept the best current case, but this means we probably will be stuck on the local optimal, so we will use the metropolis hasting algorithm to accept some bad cases to help us skip the local optimal and get the global optimal. Thus we may set our beam size to a smaller number to achieve the goal which only a large number can achieve in the base beam search decoder.

Hypothesis: The Metropolis Hasting Algorithm applied in the decoder would improve the overall performance by reducing the computational cost and improving the optimality.

Input: The French-to-English translation dataset provided in HW3

Output: The overall likelihood score for decoding translation.

Evaluation: The beam-search baseline is -1439, and in the hw, the advance-model can reach -1330, and in our class, the highest score is -1225. We first will run the baseline algorithm (beam search decoder) on the French-to-English translation dataset provided in HW3, and get the output with an evaluated score to be the comparison group. Then, we will implement a metropolis hasting algorithm to our baseline on the French-to-English translation dataset provided in HW3, and get the output with an evaluated score to be the experiment group. If the output is higher than -1225 (good), then it means that the metropolis algorithm can help us find a better decoding solution. If output is lower than -1330, then it means that the score we compute does not increase performance, we can try to find the reason behind it.

3. Challenges in Translation Reordering and Baseline Analysis:

In our previous coding in HW3, the score of the baseline algorithm was only about -1450. The main reason leading to this low score result is that the algorithm does not do the reordering process after the translation. Because French and English have different grammatical logic, then the sentences translated are not exactly correct. (Mohammad) The current method to solve this problem is applying a beam search algorithm to the model, but beam search has its own disadvantages. First of all, the cost of the algorithm is very high since the running time of the algorithm is too long. For example, in the HW3 we have to add another loop to the model to let it do the reordering process, but it makes the complexity of the beam search to O(n^3). If we want to input high k values and s values, our running time will increase in exponential order. Besides the first problem, because we are still applying a greedy search algorithm to do the translation, we cannot get the optimal solutions. Thus, it is

important to apply the Metropolis Hasting Algorithm to both solve the running time and optimality problems.

4. Experiment:

Modelling: The Metropolis-Hastings algorithm is a Markov chain Monte Carlo (MCMC) technique used in statistics and statistical physics to generate a series of random samples from a probability distribution that is difficult to sample directly. The Metropolis–Hastings algorithm can draw samples from any probability distribution with probability density P(x), provided that we know a function f(x) proportional to the density P and the values of f(x) can be calculated. (Robert) The Metropolis-Hastings method produces a series of sample values so that the distribution of values gets closer to the intended distribution as more sample values are created. The series of samples becomes a Markov chain when these sample values are generated repeatedly, with the distribution of the subsequent sample depending only on the current sample value.(Siddhartha) To be more precise, the algorithm uses the current sample value to select a candidate for the next sample value at each iteration. Subsequently, there is a chance that the candidate will be either approved or rejected. If approved, the candidate value will be utilized in the subsequent iteration, and the current value will be utilized in the subsequent iteration. In our model, unlike beam search algorithms, we will separately do the reordering process and translation process, so we could compress our complexity to O(n^2). Besides, we could find the optimal solution with the highest language model probability given a relatively optimal translation model probability.

The pseudocode of the algorithm:

```
def metropolis(translation):
    curr_prob = basic lm prob
    best_prob = basic lm_prob
    curr_state = translation
    best_state = translation
    for i in range(iteration time):
        random select 2 numbers from len(translation)
        swap the position and receive the new translation
        generate a new lm state and calculate the lm score with the new translation
        if new_prob > best_prob:
            best_prob = new_prob
            curr_prob = new_prob
            curr_state = new translation
            best_state = new translation
        elif new_prob <   best_prob:
            if log(uniform(0,1)) < new_prob - curr_prob:
                curr_prob = new_prob
                curr_state = new translation
            else:
                reject this new translation and swap the position back to the previous
translation
```

5. Result:

k is how many possible translations for a phrase during our decoding process

s is our beam size

Baseline (k=5, s=40):

Time	Score
10.77s	-1362

Metropolis Hasting Algorithm (k=5, s=40) Time Table:

Iterations	Time
100	1.28s
1000	2.32s
10000	13s

Metropolis Hasting Algorithm Accuracy Table:

Iterations\Parameter	k=5, s=40	k=3, s=3	k=10, s=10	k=40, s=40
100	-1330.09	-1327.65	-1328.13	-1329.68
1000	-1313.31	-1310.97	-1310.97	-1309.09
10000	-1301.86	-1301.65	-1298.66	-1301.35

The best result MCMC can achieve is -1298.66, which is not higher than the ITG constraint method. In order to improve the overall accuracy, we need to change our starter code first. This time, we only consider the highest translation probability without considering their language model probability. Thus, the first part of our code will generate the highest translation model probability then use MCMC algorithm. Here is the new result:

Metropolis Hasting Algorithm (k=5, s=40) Time Table:

Iterations	Time
100	1.28s
1000	2.32s
10000	13s

Since we only changed the way to calculate the score, there are no big changes in the time.

Metropolis Hasting Algorithm Accuracy Table:

Iterations\Parameter	k=5, s=40	k=3, s=3	k=10, s=10	k=400, s=400
100	-1315.27	-1312.31	-1313.13	-1310.62
1000	-1301.27	-1301.54	-1302.26	-1298.72
10000	-1286.25	-1285.31	-1285.57	-1281.49

If we find the global optimal solution for the translation model probability, it's not hard to find that the best we can achieve increased to -1281.49.

6. Discussion

Since if we increase our k and s to 400, we still cannot increase the overall probability very much. If we compare MCMC to our class's highest score, MCMC is still the lower one. The reason could be a lot of different aspects, but the main reason is the global optimal translation model probability + the global optimal language model probability does not mean the sum of them is the global optimal solution. MCMC also has its weaknesses. First, although the MCMC method is much faster and more accurate than the beam search, if the sentence is too long, for example, a sentence consisting of 10000 words, we still need a huge number of iterations to make it converge. Second, we cannot solve the translation problem if the input

language is not very logical. To be more specific, currently, we have a sentence consisting of word A, word B, and word C, and words A and C should be considered together, we cannot find the global optimal translation model probability if we do not do reordering during the decoding process. Thus, MCMC cannot be the main method in the current machine translation area compared to the neural translation model, but MCMC can be considered a good substitution for beam search.

Reference List

Robert, C. P. (n.d.). The Metropolis–Hastings algorithm. Retrieved from https://arxiv.org/pdf/1504.01896.pdf

Siddhartha, C. Understanding the Metropolis–Hastings algorithm. Retrieved from https://eml.berkeley.edu/reprints/misc/understanding.pdf

Mohammad, S. Reordering in statistical machine translation. Retrieved from https://core.ac.uk/download/pdf/30696125.pdf