DATASCI W266: Natural Language Processing with Deep Learning

Optimal Beam Width for English to Marathi Neural Machine Translation with Beam Search

Rohit Bakshi[1]

[1]UC Berkeley

December 5, 2021

Abstract

This paper reports the improvement in neural machine translation performance from English to Marathi with a sequence to sequence encoder-decoder model architecture with the implementation of beam search compared to a baseline model. We focus on varying beam widths to find an optimal beam width in terms of improvement to translations and considerations to execution times. As Marathi is a low resourced language, we use beam search as an improvement mechanism as a means to improve translation performance without augmenting the dataset. We measure performance using the BLEU metric and find that by implementing beam search to our model, we find translation performance improved by a difference of 14 points (BLEU) with a beam width of 35. We also find that lower beam widths also materially improve translation performance without drastic hits to translation times. We show that even for a low resourced language like Marathi, it is sensible to implement beam search to improve translation performance.

1 Introduction

In this paper, we present our experiments with beam width for Neural Machine Translation with beam search from English to Marathi. Neural machine translation (NMT) has shown widespread success in advancing the quality of machine translation [1]. NMT trains its parts end-to-end to maximize performance. It consists of two important components, the encoder and decoder. Both of these components are typically built on similar neural networks of different types, such as recurrent neural networks [1], convolutional neural networks [2], and more recently on transformer networks [2].

While NMTs are widespread in their use of translation of any languages and are agnostic to language, we focus on strictly the translation from English to Marathi in this paper. We are particularly interested in the translation from English to another language (Marathi) because if a person can understand both languages in context (in this case English and Marathi), they can easily translate from English to that language (Marathi). This is useful in scenarios where people want to display text in second language but use an English language keyboard.

Marathi is an arbitrary choice of a second language other than for personal reasons. Regarding background on the language, Marathi is an official language in India with over 100 million speakers worldwide [1]. It is a Subject-Object-Verb language as opposed to English which is a Subject-Verb-Object language [1].

Marathi is a low-resource language and quality parallel corpus are difficult to find [1]. Furthermore, machine translation is highly sensitive to the quality and size of data. The quality of translations will improve with augmented data, but even for low-resourced languages, we can make significant improvements without the use of an attention mechanism if we implement beam search. Beam search is a heuristic search algorithm that explores a graph by expanding the most promising nodes in a limited set to maintain tractability[1]. It is an improved version of greedy search.

In our experiments, we implement a NMT using an Encoder-Decoder model, and we show that implementing beam search for NMT from English to Marathi materially increases BLEU score than when not implementing beam search. When varying beam width for beam search, we see that there is a trade-off between improv-

[1] https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

ing BLEU scores and translation times when increasing beam width. The translation times also correspond to memory usage too, but we did not calculate the memory usage in our experiments.

We see that for low resourced languages like Marathi, we can introduce techniques to materially improve translation such as beam search without augmenting the dataset. We show that if such methods are used to improve translation, careful consideration must be given to optimally set hyperparameters to improve translation. In this case, it is extremely important to consider the optimal beam width in beam search to improve translation from English to Marathi.

2 Background

Neural Machine Translation has been studied at a substantial level. There has been research for the attentive fine tuning of transformers for low-resourced languages including Marathi [1]. However there is no implementation of beam search in their research. Similarly, neural machine translation has been studied on other Indian languages with a focus on languages other than Marathi, but still without beam search [3]. Regarding Marathi, there has been research done from Marathi to English for NMT with near perfect corpus and the use of transformers [2]. In that paper, the optimal beam search found was 5, but the translation was from Marathi to English, not vice versa. Other research with Machine Translation has been done from Marathi to English, but now with the use of beam search [4] [5]. Furthermore, statistical machine translation has also been explored with Indian languages, but not including Marathi [6].

3 Methods

3.1 Dataset

We use the dataset from the Tatoeba Project from the manythings.org website. The manythings.org website is a website for people studying English as a Second Language. The dataset consists of tab-delimited bilingual sentence pairs with English and Marathi. It consists of 43071 sentences with distinct Marathi translations.

The maximum length of a sequence in English is 34 words, and the maximum length of a sequence in Marathi is 37 words. The English vocabulary contains 5794 tokens after preprocessing, and the Marathi vocabulary contains 14037 tokens after preprocessing.

```
you cant understand - START_ तू समजू शकत नाहीस _END
```

Figure 1: English - Marathi sentence pairing after preprocessing

3.2 Data Preprocessing

Our data cleaning and data preprocessing is relatively simple. We perform the following steps:

- lowercase all characters (only materially applies to English)
- remove quotes from sentences
- remove all special characters from sentences
- remove all digits from English sentences and the equivalent Marathi digits from Marathi sentences
- strip all extra white space from the beginning and end of each sentence
- add start and end tokens to target sequences

A sample preprocessed sentence pairing is shown in Figure 1. We split the data into a 90/10 train and test split. We use a batch generator function to batch the data for simpler loading and experimenting later on. We use a batch size of 128.

3.3 Encoder - Decoder Model Architecture

We use a very simple Sequence to Sequence (seq2seq) model for our machine translation task. Specifically we use an Encoder - Decoder architecture. In this architecture, both the encoder and decoder are LSTM (long short-term memory) models [3] [2].

The encoder reads the input sequence and yields outputs and internal state information to propagate further. The outputs of the encoder are ignored, but the final internal state produced from the encoder contains information about the input sequence that will be fed into the decoder model.

The decoder model can be thought of as a language model that is fed the final internal states of the encoder model. The decoder model acts differently during training and inference for actual inputs. During the training phase, the decoder is set to the final internal states of the encoder and iteratively starts generating the next word starting with the original input sequence. We use a technique called teacher forcing to

[2] https://blog.paperspace.com/introduction-to-seq2seq-models/

train [3]. This is used for training models with recurrent connections from their outputs as is the case here. Teacher forcing means that instead of using inputs from previous steps, we use the original input from the actual sequence to feed into each step of the decoder. We use teacher forcing because the decoder training converges faster, and it leads to more accurate predictions during training.

During the inference phase, we cannot rely on teacher forcing because there is no ground truth, so we feed the decoder model's previous prediction into its next step. However, just like the training phase, in the inference phase, the decoder generates an output at each step given the entire input sequence through the final internal state of the encoder. For inference, the decoder model generates one word at a time iteratively. The initial input is set to the START token and we feed the predicted output for each step as the input for the next step. The decoding ends when the END token is reached.

During training, we use the Adam optimizer. The loss function is set to categorical crossentropy, and the metrics are set to accuracy. We use these values arbitrarily after researching about it briefly [1]. The Adam optimizer seems widespread in Deep Learning models and shows better results than simply using gradient descent[4]. We set the loss function to categorical crossentropy because it is a loss function that is used in multi-class classification tasks[5], which applies to predicting words in NMT here. Lastly, we set the metrics to accuracy to let the training file determine which metric should be used given certain parameters like output shape etc[6].

We train the model overnight for 50 epochs with a NVIDIA GTX 1080 and save the weights of the model to avoid training it again, it possible.

In the baseline model, during inference, we use a simple greedy algorithm to choose the token with the highest probability and move on to the next step with the updated states. We attempt to improve this by implementing beam search in our inference phase.

3.4 Implementing Beam Search

Beam search improves upon greedy search and yields a list of most likely output sequences at each step. Instead of choosing the most likely output sequence, beam search keeps the n most likely sequences at each step and expands upon these sequences again by repeatedly choosing the n most likely sequences until the END token is reached or the maximum length is reached [7]. n is a user defined hyperparameter known as beam width. Larger beam widths will yield better performance at the trade-off of speed and memory usage. We use beam search because keeping all candidates at all steps is intractable.

Our beam search function has an input list of probabilities of predicted tokens at a given step with beam width n. At each step, each candidate sequence is expanded with all possible next steps. We score each candidate step by multiplying the probabilities together. Since probabilities are small numbers, we use a natural logarithm to keep numbers manageable when calculating the score. We select the n most likely probabilities and prune all the other candidates. Regarding score, we actually optimize to minimize score to make it easy to sort the candidates for selection. We repeat this process until the end of the sequence or maximum length is reached. We return the top n sequences that beam search yielded.

3.5 BLEU Score

We use BLEU to evaluate machine translation in our model. To calculate BLEU score, we use the NLTK library. The BLEU score compares a sentence against a reference sentence and tells how well does the candidate sentence matched the reference[8]. It gives an output score between 0 and 1. A BLEU score of 1 means that the candidate sentence perfectly matches one of the reference sentences. With the BLEU function in the NLTK library, you can choose the number of words you want the model to match at once. In our experiments we used 1-gram (unigram) matching to simply see how well the translations stacked up word by word.

We calculate the BLEU score for the baseline greedy model by simply comparing the candidate to the reference. For the beam search results, we calculate the BLEU score for each of the candidates in the list of n beam width candidates and return the sentence with the top score.

[3] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c

[4] https://towardsdatascience.com/understanding-gradient-descent-and-adam-optimization-472ae8a78c10

[5] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[6] https://neptune.ai/blog/keras-metrics

[7] https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

[8] https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

3.6 Batch Experiments with Beam Width

We run two sets of the same experiments - one for the training data, one for the test data. In each experiment, we simultaneously test the greedy model and the beam search model. For the beam search model, we pass a value for beam width n to base the experiment on. For the experiments, we generate batches of data (batch size of 1) 100 times to decode with our decoder model. For each of the 100 sentences, we feed in the input sequence to both the baseline greedy model and the beam search model and output the best sentences of each model and compare the BLEU scores. Over the 100 sentences, we aggregate the difference in BLEU scores between the models and return the difference in BLEU score between the model as well as the time it takes to finish execution of all 100 sentences with the specified beam width n for that experiment. We run the experiments over again for values of n for (2, 3, 5, 7, 10, 15, 20, 35, 50, 100). We do not use beam width of 1 since that would just be equivalent to the greedy model.

3.7 Individual Assessment of Sentences

We also assess the results of both the greedy model and the beam search model on individual sentences to see the sentences yielded by both models and the actual difference in the output sentences and scores. We do this because during our batch experiments, we do not observe the intermediate results of the models. By individually assessing the sentences, we pinpoint the actual differences the models generate with regards to translating sentences and can subjectively mark the differences in the sentences the models yield and quality of the translations.

4 Results and Discussion

For our batch experiments we list our the results in the following tables. We provide two tables: one for training experiments and the other for test experiments. In each table, we show beam width, the BLEU score difference (increase) between the beam search model with given beam width and the execution time of the entire experiment in seconds.

Beam Width	Score Increase	Execution Time (s)
0	-	25.4178
2	5.186685	47.2349
3	7.096209	47.8501
5	9.741120	47.5811
7	10.121895	48.5193
10	11.387274	50.8125
15	11.972510	51.1363
20	12.673814	49.5731
35	14.020895	50.7741
50	15.211438	52.6561
100	16.82471	64.5568
0	-	25.0710
2	5.562557	51.1552
3	8.138747	52.1786
5	10.43592	56.6499
7	11.48919	55.9219
10	12.43389	57.2988
15	13.29763	57.1477
20	14.21019	56.7730
35	15.77790	57.4543
50	16.37300	60.5855
100	17.70241	74.4446

Training and Test Experiments

As we can see, there is a material BLEU score increase with our beam search model compared to the greedy baseline even with a low beam width of $n = 3$. In both training and test experiments, there is between a 0.07 - 0.08 BLEU score increase per sentence with our beam width equal to 3, confirming that beam search does improve performance of NMT from English to Marathi. In a low resource language like Marathi, we can see that implementing beam search is very helpful in increasing performance without augmenting data. For both training and test experiments, we see a steady increase in BLEU score difference from $n = 2$ to 20 and further on but the law of diminishing returns applies and the increases are marginal once n becomes very large, showing that there is a limit to how much beam search can improve NMT between English to Marathi with no other change.

Furthermore, we see that when we increase beam width, the BLEU score difference increases and the execution times increase mostly as well too, confirming that there is a trade-off between model performance and translation times. We can see that just by implementing beam search, we approximately double the execution time it takes. Furthermore, we see that execution times slowly rise as beam width increases from $n = 2$ to 35. Within these beam widths, the execution times differ by between 3-5 seconds for the training and test experiments. But once $n > 35$, the execution times increase more drastically with

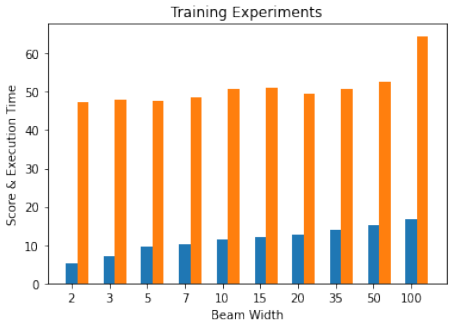

Figure 2: English - Marathi Training Experiments

Figure 3: English - Marathi Test Experiments

$n = 100$ taking approximately 30% longer than $n = 20$.

When we plot these values in bar charts (Figure 2 and 3), we can confirm that the scores marginally increase as beam width increases, and once beam width is quite large, the execution times start to dramatically increase around $n = 50$.

Given the trade-off between performance of the beam search model and greedy model and execution times of the beam search model and given that beam search can be used to improve the performance of NMT from English to Marathi, a beam width $n = 35$ seems ideal in terms of improving translation performance compared to the baseline since the execution time is similar to that of the times with models of much lower beam widths, but the performance is materially improved.

Of course, memory considerations are not taken into account here, so it might not be prudent to have even $n = 35$ as a beam width if memory has to be conserved. If memory must be taken into consideration, the results show that beam search does materially improve the translations from English to Marathi without augmenting the data, so it makes sense to implement beam search with a workable beam width parameter because the trade-off with execution times is probably sensible to improve translation performance with a minor hit to execution times.

```
Input English sentence: what are you two doing
Actual Marathi Translation:  तुम्ही दोघं काय करत आहात
Beam Predicted Marathi Translation: तुम्ही दोघं काय करत आहात
Predicted Marathi Translation:  तू दोघं काय करत आहेस
Beam Score:             1.0
Baseline Score:         0.6
```

Figure 4: Training Example 1

```
Input English sentence: i dont want to forget anything
Actual Marathi Translation:  मला काहीही विसरायचं नाहीये
Beam Predicted Marathi Translation: मला काहीही विसरायचं नाहीये
Predicted Marathi Translation:  मला काहीही खायचं नाहीये
Beam Score:             1.0
Baseline Score:         0.75
```

Figure 5: Training Example 2

4.1 Quality of Translations Analysis: Individual Sentences

When we look at individual sentences, we can see how beam search improves translations. In Figure 4, we can see from the training example that the BLEU score increased from 0.6 to a perfect 1.0. When we analyze this improvement, we see that beam search finds more logical verb conjugations with regards to formality and plurality. While the greedy solution might make sense as broken Marathi, the beam search solution improves the grammar of the sentence by correctly acknowledging the rules of Marathi (what are you two (guys) doing? vs what are you (singular and informal) two doing (singular)?).

In another training example (Figure 5) we can see that beam search correctly finds the mistranslated word that the greedy solution missed (I don't want to forget anything vs I don't want to eat anything).

In a test example (Figure 6), we see that while beam search improves the BLEU score compared to greedy search, both are technically grammatically correct in Marathi and mean the same thing (difference in formality). While beam search improves the BLEU score, this improvement is not as necessary since the baseline translation is correct albeit in a more informal manner.

This same case is echoed in another test example (Figure 7) where beam search improves the BLEU by changing the formality of the translation which matches the test label. In this case

```
Input English sentence: we fixed that
Actual Marathi Translation:  आपण ते दुरुस्त केलं
Beam Predicted Marathi Translation: आपण ते दुरुस्त केलं
Predicted Marathi Translation:  आम्ही ते दुरुस्त केलं
Beam Score:             1.0
Baseline Score:         0.75
```

Figure 6: Test Example 1

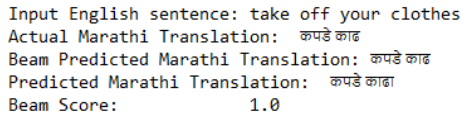

Figure 7: Test Example 2

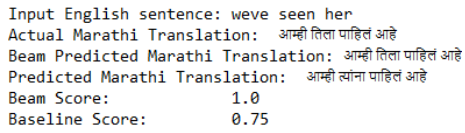

Figure 8: Test Example 3

also, the greedy translation is workable.

However, beam search is shown to be very useful in improving performance when we consider another test example (Figure 8) where beam search accurately improves the translation compared to the baseline model (We've seen her vs We've seen them).

Based on analysing these individual sentences, we can see that beam search does materially improve the translations from English to Marathi compared to the greedy baseline model. The improvements occur when the baseline model incorrectly translates a word or does not make correct grammatical sense in terms of Marathi rules. However, while beam search does make improvements to the baseline, some of the improvements should not be as highly regarded as the BLEU score improvement suggests because in some cases, the baseline translation is workable, but small differences compared to the test label exists such as the sense of formality.

Overall, when looking at NMT from English to Marathi, beam search does improve the translations in a meaningful way. In some cases, it definitely provides clearer and more accurate translations that the baseline model ignores. This is quite remarkable since the underlying seq2seq NMT encoder and decoder (training) model is the same, and the underlying dataset is the same too. As beam width increases, there are more opportunities for better translations but as we see in the previous part, the trade-off between beam search performance and execution time should definitely be taken into consideration. As long as memory usage is considered, a beam width of 35 is ideal to improve BLEU performance in NMT from English to Marathi. If memory usage at this beam width is problematic, it still makes sense to lower beam width to increase translation performance to accommodate for memory usage.

Conclusion

Getting accurate translations in a low resource language such as Marathi is difficult without finding better datasets. However, with regards to neural machine translation from English to Marathi using a simple seq2seq encoder-decoder model architecture, we find that implementing beam search materially improves the BLEU score and quality of translations when compared to a greedy baseline model even though execution times increase as well. With the given hardware, we find that a beam width of 35 is optimal in improving translation performance without having a drastic hit to execution times as well. If a beam width of 35 proves to be too arduous for memory constraints, it is still sensible to use lower beam widths to improve translation performance from English to Marathi.

References

[1] Karthik Puranik, Karthik Hande, Ruba Priyadharshini, Thenmozi Durairaj, Anbukkarasi Sampath, Kingston Pal Thamburaj, and Bharathi Raja Chakravarthi. Attentive fine-tuning of transformers for translation of low-resourced languages. 2021.

[2] Swapnil Ashok Jadhav. Marathi to english neural machine translation with near perfect corpus and transformers. 2020.

[3] Karthik Revanuru, Kaushik Turlapaty, and Shrisha Rao. Neural machine translation of indian languages. 2017.

[4] Nilesh Shirsath, Aniruddha Velankar, Ranjeet Patil, and Shilpa Shinde. Various approaches of machine translation for marathi to english language. 2021.

[5] Krushna Belerao and V.S. Wadne. Machine translation using open nlp and rules based system "english to marathi translator". 2014.

[6] Raj Nath Patel, Prakash Pimpale, and Sasikumar Mukundan. Machine translation in indian languages: Challenges and resolution. 2018.