DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY

MSc. COMPUTER SCIENCE

IMMACULATE WANZA MUSYOKA

C241-01-2887/2021

KIKAMBA TO ENGLISH MACHINE TRANSLATION SYSTEM

2022

Abstract

Machine Translation (MT) is a subset of computational linguistics that deals with the use of computer software to automatically translate content from one human language to another. Advances in computing power, artificial intelligence and natural language processing have contributed to the use of MT technology. Though much has been done in developing MT systems for different languages all over the world, African languages are yet to be digitized. As a result, these languages still remain under-resourced.

This paper presents a neural MT system for Kikamba-English. Kikamba is an under-resourced language spoken in Kitui, Makueni, kwale and Machakos counties of Kenya. English language is spoken globally with different varieties such as American English and British English. There are very few efficient language tools and technologies that relate to the Kikamba. This has prompted the need to digitize Kikamba by developing a neural machine translation system.

Neural machine translation approach has proven to provide better results with language pairs as compared to statistical and rule-based machine translations. It uses deep neural networks to map between source and target languages. This project is done using OpenNMT which is a generic deep learning framework mainly specialized in sequence-to-sequence models covering a variety of tasks such as machine translation, summarization, image to text and speech recognition.

The copra is collected and preprocessed before it is passed to the OpenNMT framework for model training. During preprocessing, the sentences are tokenized in to words separated by space. The outcome of tokenization is passed to the framework inform of two parallel data sources for the source and target languages. Kikamba is used as the source language while English is used as the target language. The framework uses 2-layer LSTM with 500 hidden units on both the encoder and decoder. A parallel corpus of 3,500 sentences was used for training. The model was then tested using few sentences and a BLUE score of 10.67 was obtained. Neural machine translation requires large datasets for predictions to be accurate. These datasets are however not readly available hence the task is tedious and time consuming.

Table of Contents

Abstract 2
Table of Contents 3
Table of Figures 3
Introduction 4
Related Work 5
Methodology 7
Step 1: Data collection 8
Step 2: Prepare the data 8
Step 3: Train the model 10
Step 4: Translate 11
Model Evaluation 13
Conclusion 13
References 14

Table of Figures

Figure 1 Diagrammatic overview of OpenNMT-py code[8] 5
Figure 2:Configuration file 9
Figure 3: output of build vocab command 9
Figure 4: Configuration file with training parameters 10
Figure 5:Training on process 10
Figure 6: Start of the Training Process 10
Figure 7:End of the training Process 11
Figure 8: Translated Sentences 12
Figure 9:Blue score 13

Introduction

The availability of machine translation systems changes how society engages in communication practices [1] and attempts to minimize the communication gap among people from various linguistic backgrounds[2]. Language is a vital tool for communication that allows people to share ideas, thoughts and feelings. There are more than 2000 languages in Africa. Despite having all these languages, few linguistic resources for NLP research are built for African languages. The few existing resources are however not discoverable because they are either published in closed journals or remain undigitized[3]. Though African societies may not see the future of these languages, there preservation is useful in preserving culture. In addition, if machine translation is applied in these languages, it can be used in high risk setting such as legal and health [1].

Machine Translation has evolved over the years from rule-based to corpus-based approach [2]. Corpus-based approaches can be further classified into Statistical Phrase-based and Neural-based[4]. Both corpus-based approaches use a huge amount of aligned parallel text corpora in both the source and target languages. Statistical approaches rely upon Bayesian inference. They predict translation probabilities of phrase pairs in corresponding source target languages. The probability of a certain pair of phrases can be enhanced by increasing the size of the dataset. This approach however, suffers from the inability to achieve context information. Consequently, Neural machine translation offers reasonable translation accuracy in case of the context analysis and fluent translation. The approach has proven to provide better results with language pairs as compared to statistical and rule-based machine translations[2]. It has advantage of deep learning, which is very suitable for dealing with the high dimension, unlabeled and big data of natural language, therefore, its application is more general and reflects the power of big data and big data thinking [5].

Neural Machine Translation (NMT) is based on the Embed - Encode - Attend - Decode paradigm. The most commonly used architectures based on this paradigm are convolutional neural networks, recurrent neural networks (RNN) and self-attention/feed-forward network (SA/FFN)[6]. There are very few efficient language tools and technologies that relate to Kikamba [7]. This has prompted the need to digitize Kikamba by developing a neural machine translation system. This paper describes how Kikamba-English NMT system is trained using OpenNMT-py open-source toolkit. OpenNMT-py uses RNN architecture with 2-layer LSTM with 500 hidden units on both the encoder and decoder [8].

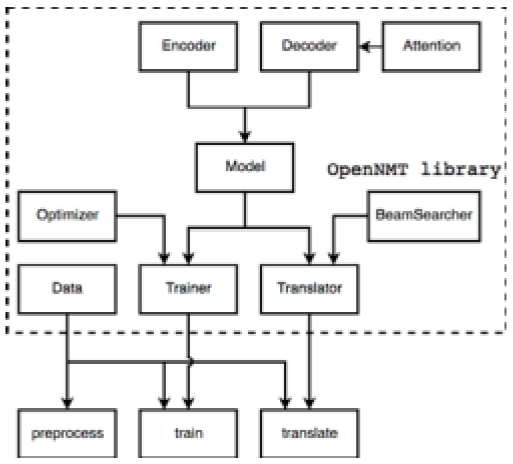


Figure 1 Diagrammatic overview of OpenNMT-py code[8]

This article examines (1) how large parallel corpora for Kikamba-English can be collected and made readily available, (2) how the collected corpora can be used to develop a NMT system for the language pair and (3) the metrics that can be used to evaluate such a system. The rest of the article is structured as follows: in section one, a state-of-art review of MT is presented, the methodology proposed for this research is then presented and subsequently, a discussion of the results is provided followed by conclusion.

Related Work

In regard to MT, neural machine translation is currently state-of-the-art and has dominated the sector for both commercial and academic research. Due to the usage of distributed representation of a language, which permits end-to-end training, it has proven to perform well for language pairs. The NMT framework can handle more than two languages, making it appropriate for multilingual machine translation, according to recent research[6]. The accuracy and inability of context information as limitations of statistical machine translation (MT) have been addressed by neural network technique [2].

In the context of translating from English to Hindi, Laskar et al., 2019, created a neural machine translation system. They used two systems in their research: NMT-1, which relied on the transformer models, and NMT-2, which depended on the Long Short-Term Memory (LSTM)-based attention model. They used Nematus to train the NMT system, and the results were BLEU scores of 23.25 and 12.23, respectively. During training, a dataset of parallel source-target sentence pairs in Hindi and English was used. A total of 14,92,827 instances made up of data sets from various existing sources were present in the parallel corpus. Compared to the earlier systems, the one that was developed performed better[2]. Despite this, it still has to be improved in cases where an unknown term is recognized, blank lines appear in the output, and the original text is translated differently.

A neural network-based deep learning technique for English to Urdu languages was proposed by [9]. They used a parallel corpus of around 30923 sentences which included content from news, and sentences which were frequently used in day-to-day life. The corpus contained 542810 English tokens and 540924 Urdu tokens. The system was trained using LSTM encoder and decoder architecture with an attention mechanism. After training the system was tested using 70:30 criteria and a BLUE score of 45.83 was obtained. This approach provided a good BLEU score as compared to existing approaches which used statistical machine translation approach.

Translation of English IMDb user movie reviews into Ser- bian was done in a low-resource scenario by[10]. To compare the two methods in that particular situation, they developed neural (NMT) machine system and phrase-based (PBMT) machine system using the publicly accessible clean out-of-domain news corpus. The two algorithms were then utilized to produce hypothetical Serbian movie reviews, resulting in more multilingual in-domain data. In the experiment, they created four English-to-Serbiann NMT models using OpenNMT and one PBMT English-to-Serbian system using Moses toolkit. A large movie review dataset that included 50 000 English-language reviews from IMDb users was used for the experiment. The models were evaluated using several metrics including BLUE score. All the four NMT models outperformed the PBMT model; the lowest performing model had a BLUE score of 11.6 as compared to the 10.8 score of the PBMT model.

Almost all the Indian languages originate from their ancestral language – Sanskrit [11][12]. Different researchers have done machine translation systems for the Sankrit language. A corpus-based translation system for Sanskrit to Hindi translation was proposed by [11]. A parallel corpus of around 145 million sentences was used to train the model and 12,698 was used to test. After evaluation using BLEU score the performance of the corpus-based MTS was 24% better than RBMT.

A neural machine translation system (NMT) that could efficiently translate Indic languages like Hindi and Gujarati was presented by [12]. This work was an improvement of their previous research work in which they worked on sequence-to- sequence model based machine translation system for Hindi language [2]. In the recent work they improved that model and applied for English-Gujarati language pair. The model was trained using a corpus of 65,000 parallel sentences. An average BLEU score of 59.73 on training corpus and 40.33 on test corpus from parallel English-Gujarati corpus having 65,000 sentences was achieved. The accuracy was improved by use of an Attention mechanism which overcame the limitation of a model coming across rare words which in turn decreases the accuracy dramatically.

Even though a lot has been done in neural machine translation, Kikamba language remain low resource language. There are a few resources that include bilingual sentences. Therefore, there are some difficulties to collect or crawl the parallel sentences. There are very few efficient language tools and technologies that relate to Kikamba [7]. Existing work use rule-based as well as statistical-based approach with the small amount of parallel corpus [13]. This has prompted the need to digitize Kikamba by developing a neural machine translation system.

Methodology

Neural machine translation (NMT) is data- driven approach to machine translation which embraces probabilistic framework. The goal of NMT is to estimate an unknown conditional distribution $P(y|x)$ given the dataset D, where x and y are random variables representing source input and target output, respectively [14]. NMT takes a conditional language modeling view of translation by modeling the probability of a target sentence $w1{:}T$ given a source sentence $x1{:}S$ as $p(w1{:}T|x)= \pi^{T}_{1}\, p(wt|w1{:}t-1,x;\, \theta)$ where the distribution is parameterized with θ [8]. In this work, OpenNMT an open-source toolkit for neural machine translation (NMT) by the Harvard NLP group and SYSTRAN, will be used. Below are the steps followed to create and test the NMT model:

Step 1: Data collection

To create and train the NMT model, we require a parallel corpus for Kikamba-English. The parallel corpus was obtained from the Bible since there are no other available sources. The content was drawn form the book of Genesis, Luke, Romans, Corinthians, Galatians, Ephesians, Philiphians, Collosians, Thessalonians, Timothy, Titus, Philomen and Hebrews [15]. Each English verse was taken as a sentence and was mapped to the corresponding verse in Kikamba thus creating two files; one with Kikamba verses and the other with corresponding English verses. Total of 3,500 sentences was collected which made up the parallel corpus.

Step 2: Prepare the data

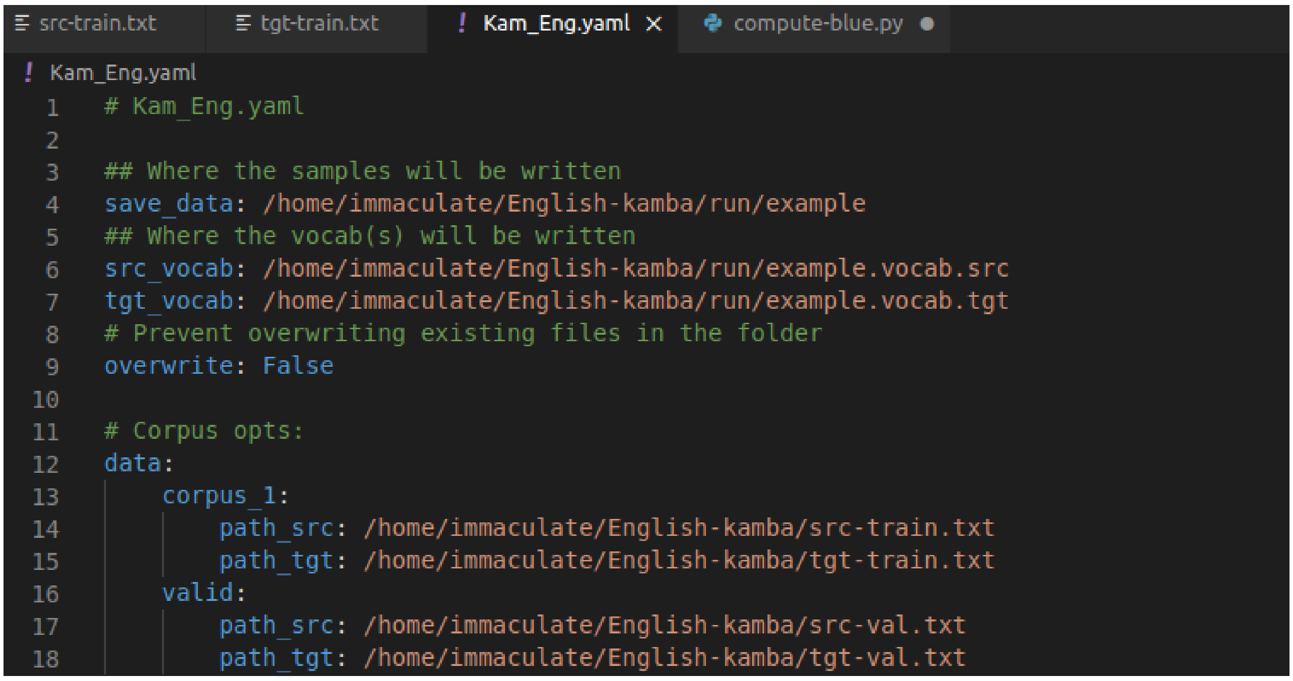
During pre-processing, the parallel corpus used to train the NMT model is adequately cleansed to eliminate noisy training samples. A vocabulary of the N most frequent words is then formed. The remaining words are then handled as unknown words and mapped to a single token denoted by "UNK." [6].

The data consisting of parallel source (src) and target (tgt) data containing one sentence per line is passed to a tokenizer to obtain tokens separated by a space: Kikamba is the source language while English is the target language. The corpus is the split into train and validation set. Validation files are used to evaluate the convergence of the training/ tune the model. It usually contains no more than 5k sentences and in this case, we have used 1k of the 3k sentences of the parallel corpus. The following files were obtained after the exercise.

- src-train.txt
- tgt-train.txt
- src-val.txt
- tgt-val.txt

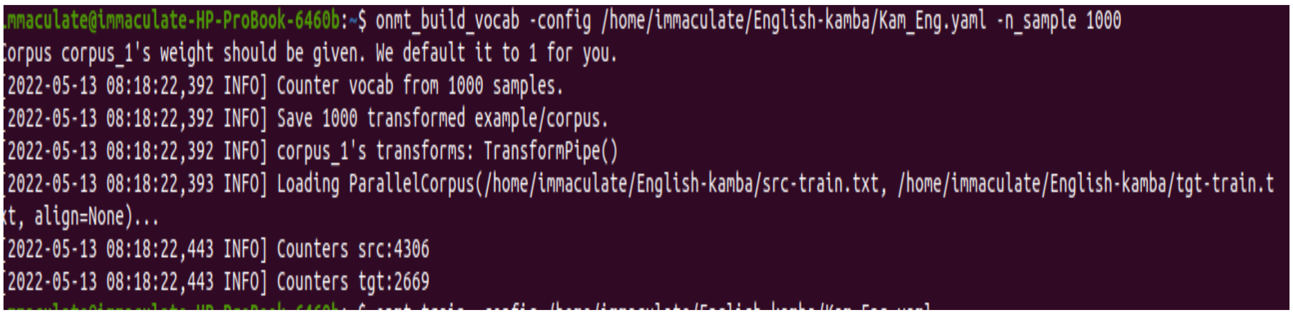
Create a Configuration file

A configuration file (**YAML configuration file)** is built to specify the data that will be used and where the vocabulary will be stored after it is generated. Below is the file[16]:



```yaml
# Kam_Eng.yaml

## Where the samples will be written
save_data: /home/immaculate/English-kamba/run/example
## Where the vocab(s) will be written
src_vocab: /home/immaculate/English-kamba/run/example.vocab.src
tgt_vocab: /home/immaculate/English-kamba/run/example.vocab.tgt
# Prevent overwriting existing files in the folder
overwrite: False

# Corpus opts:
data:
    corpus_1:
        path_src: /home/immaculate/English-kamba/src-train.txt
        path_tgt: /home/immaculate/English-kamba/tgt-train.txt
    valid:
        path_src: /home/immaculate/English-kamba/src-val.txt
        path_tgt: /home/immaculate/English-kamba/tgt-val.txt
```

Figure 2:Configuration file



```
mmaculate@immaculate-HP-ProBook-6460b:~$ onmt_build_vocab -config /home/immaculate/English-kamba/Kam_Eng.yaml -n_sample 1000
orpus corpus_1's weight should be given. We default it to 1 for you.
[2022-05-13 08:18:22,392 INFO] Counter vocab from 1000 samples.
[2022-05-13 08:18:22,392 INFO] Save 1000 transformed example/corpus.
[2022-05-13 08:18:22,392 INFO] corpus_1's transforms: TransformPipe()
[2022-05-13 08:18:22,393 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 08:18:22,443 INFO] Counters src:4306
[2022-05-13 08:18:22,443 INFO] Counters tgt:2669
```

Figure 3: output of build vocab command

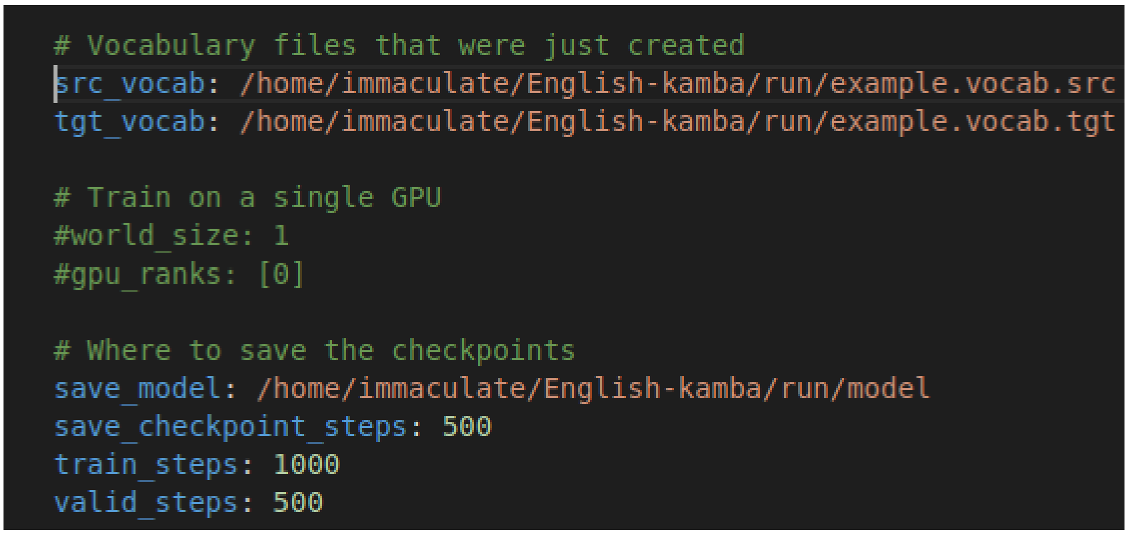
From this configuration, we can build the vocab(s), that will be necessary to train the model using the following command: -n_sample is required here – it represents the number of lines sampled from each corpus to build the vocab.

onmt_build_vocab -config /home/immaculate/English-kamba/Kam_Eng.yaml -n_sample 3500

Step 3: Train the model

To train a model, the following are **added to the YAML configuration file**:

- the vocabulary path(s) that will be used: can be that generated by onmt_build_vocab;
- training specific parameters.



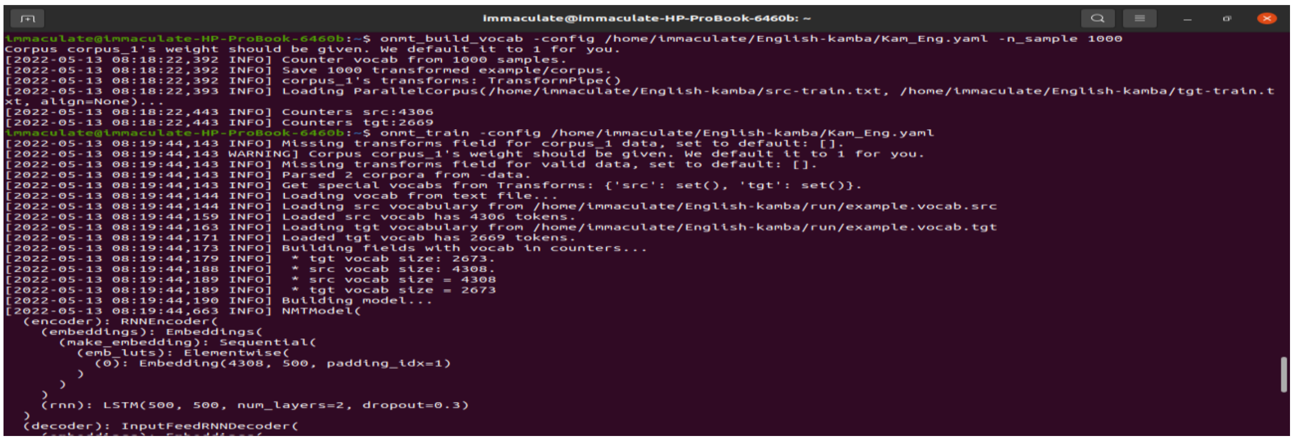
```
# Vocabulary files that were just created
src_vocab: /home/immaculate/English-kamba/run/example.vocab.src
tgt_vocab: /home/immaculate/English-kamba/run/example.vocab.tgt

# Train on a single GPU
#world_size: 1
#gpu_ranks: [0]

# Where to save the checkpoints
save_model: /home/immaculate/English-kamba/run/model
save_checkpoint_steps: 500
train_steps: 1000
valid_steps: 500
```

Figure 4: Configuration file with training parameters

The following command is the executed to create NMT model *model_step_1000.pt.*

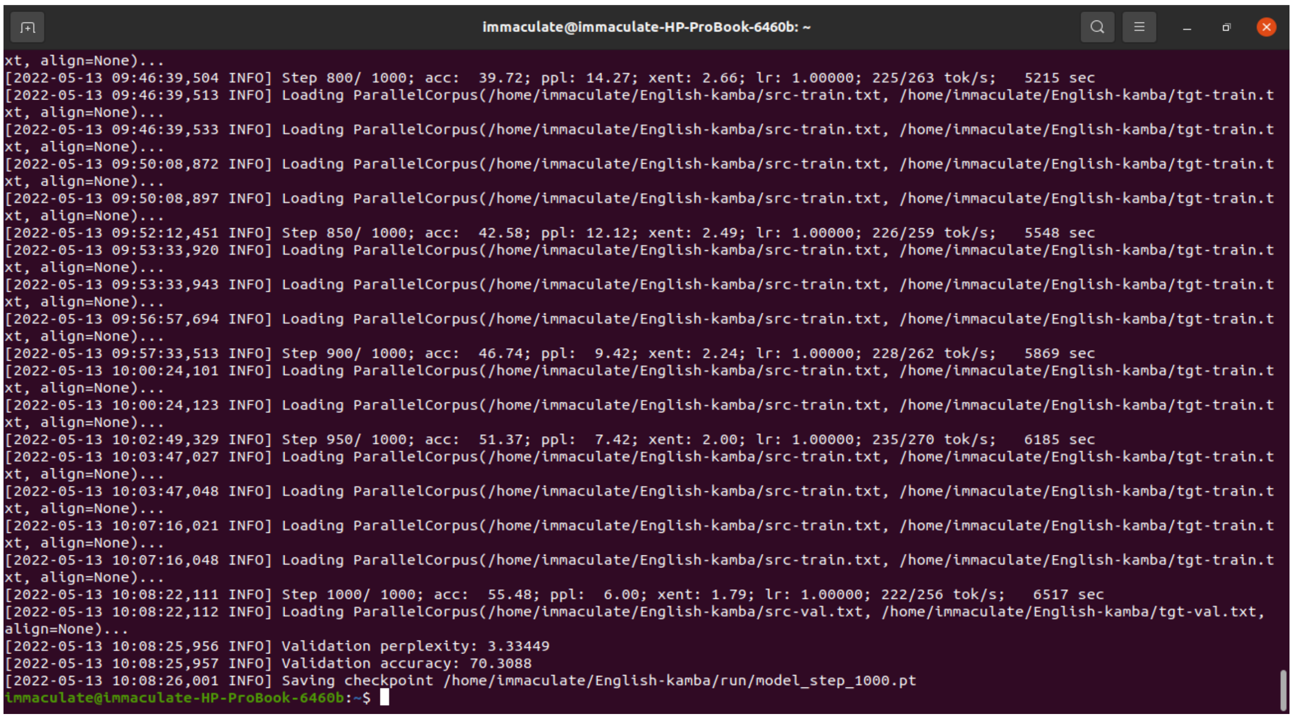
onmt_train -config /home/immaculate/English-kamba/Kam_Eng.yaml

The configuration will run the model consisting of a 2-layer LSTM with 500 hidden units on both the encoder and decoder. It will run on a CPU though it can run on a single GPU (world_size 1 & gpu_ranks [0]).



```
immaculate@immaculate-HP-ProBook-6460b:~$ onmt_build_vocab -config /home/immaculate/English-kamba/Kam_Eng.yaml -n_sample 1000
Corpus corpus_1's weight should be given. We default it to 1 for you.
[2022-05-13 08:18:22,392 INFO] Counter vocab from 1000 samples.
[2022-05-13 08:18:22,392 INFO] Save 1000 transformed example/corpus.
[2022-05-13 08:18:22,392 INFO] corpus_1's transforms: TransformPipe()
[2022-05-13 08:18:22,393 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 08:18:22,443 INFO] Counters src:4306
[2022-05-13 08:18:22,443 INFO] Counters tgt:2669
immaculate@immaculate-HP-ProBook-6460b:~$ onmt_train -config /home/immaculate/English-kamba/Kam_Eng.yaml
[2022-05-13 08:19:44,143 INFO] Missing transforms field for corpus_1 data, set to default: [].
[2022-05-13 08:19:44,143 WARNING] Corpus corpus_1's weight should be given. We default it to 1 for you.
[2022-05-13 08:19:44,143 INFO] Missing transforms field for valid data, set to default: [].
[2022-05-13 08:19:44,143 INFO] Parsed 2 corpora from -data.
[2022-05-13 08:19:44,143 INFO] Get special vocabs from Transforms: {'src': set(), 'tgt': set()}.
[2022-05-13 08:19:44,144 INFO] Loading vocab from text file...
[2022-05-13 08:19:44,144 INFO] Loading src vocabulary from /home/immaculate/English-kamba/run/example.vocab.src
[2022-05-13 08:19:44,159 INFO] Loaded src vocab has 4306 tokens.
[2022-05-13 08:19:44,163 INFO] Loading tgt vocabulary from /home/immaculate/English-kamba/run/example.vocab.tgt
[2022-05-13 08:19:44,171 INFO] Loaded tgt vocab has 2669 tokens.
[2022-05-13 08:19:44,173 INFO] Building fields with vocab in counters...
[2022-05-13 08:19:44,179 INFO]  * tgt vocab size: 2673.
[2022-05-13 08:19:44,188 INFO]  * src vocab size: 4308.
[2022-05-13 08:19:44,189 INFO]  * src vocab size = 4308
[2022-05-13 08:19:44,189 INFO]  * tgt vocab size = 2673
[2022-05-13 08:19:44,190 INFO] Building model...
[2022-05-13 08:19:44,663 INFO] NMTModel(
  (encoder): RNNEncoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(4308, 500, padding_idx=1)
        )
      )
    )
    (rnn): LSTM(500, 500, num_layers=2, dropout=0.3)
  )
  (decoder): InputFeedRNNDecoder(
```

Figure 6: Start of the Training Process



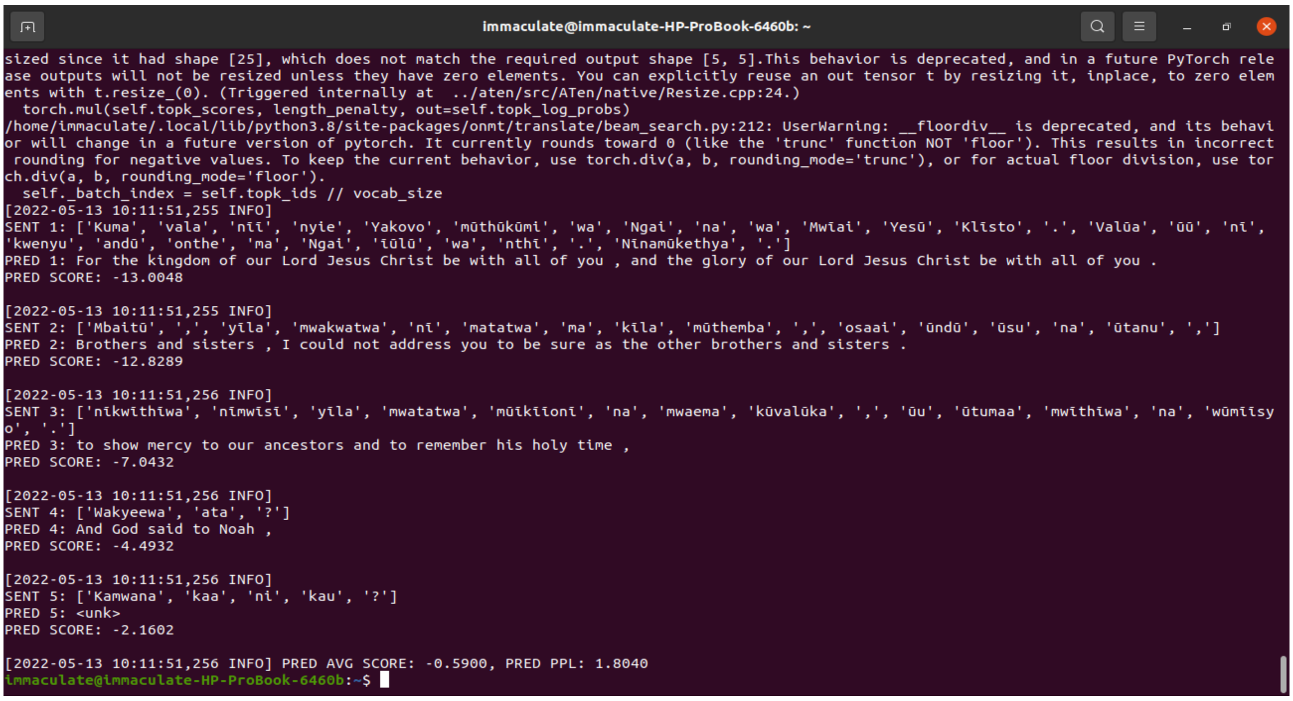
```
xt, align=None)...
[2022-05-13 09:46:39,504 INFO] Step 800/ 1000; acc:  39.72; ppl: 14.27; xent: 2.66; lr: 1.00000; 225/263 tok/s;   5215 sec
[2022-05-13 09:46:39,513 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:46:39,533 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:50:08,872 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:50:08,897 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:52:12,451 INFO] Step 850/ 1000; acc:  42.58; ppl: 12.12; xent: 2.49; lr: 1.00000; 226/259 tok/s;   5548 sec
[2022-05-13 09:53:33,920 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:53:33,943 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:56:57,694 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 09:57:33,513 INFO] Step 900/ 1000; acc:  46.74; ppl:  9.42; xent: 2.24; lr: 1.00000; 228/262 tok/s;   5869 sec
[2022-05-13 10:00:24,101 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:00:24,123 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:02:49,329 INFO] Step 950/ 1000; acc:  51.37; ppl:  7.42; xent: 2.00; lr: 1.00000; 235/270 tok/s;   6185 sec
[2022-05-13 10:03:47,027 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:03:47,048 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:07:16,021 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:07:16,048 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-train.txt, /home/immaculate/English-kamba/tgt-train.t
xt, align=None)...
[2022-05-13 10:08:22,111 INFO] Step 1000/ 1000; acc:  55.48; ppl:  6.00; xent: 1.79; lr: 1.00000; 222/256 tok/s;   6517 sec
[2022-05-13 10:08:22,112 INFO] Loading ParallelCorpus(/home/immaculate/English-kamba/src-val.txt, /home/immaculate/English-kamba/tgt-val.txt,
align=None)...
[2022-05-13 10:08:25,956 INFO] Validation perplexity: 3.33449
[2022-05-13 10:08:25,957 INFO] Validation accuracy: 70.3088
[2022-05-13 10:08:26,001 INFO] Saving checkpoint /home/immaculate/English-kamba/run/model_step_1000.pt
immaculate@immaculate-HP-ProBook-6460b:~$ 
```

Figure 7:End of the training Process

Step 4: Translate

Now we have a model which we can use to predict on new data. We do this by running beam search. The sentences to be translated are stored in src-test.txt file while the actual predictions are in tgt-test.txt file. The predicted output will be stored in English-Kamba/pred_1000.txt. The predictions are going to be quite terrible, as the dataset is small. The command below is used to obtain the predictions.

onmt_translate -model /home/immaculate/English-kamba/run/model_step_1000.pt -src /home/immaculate/English-kamba/src-test.txt -output /home/immaculate/English-kamba/pred_1000.txt -verbose

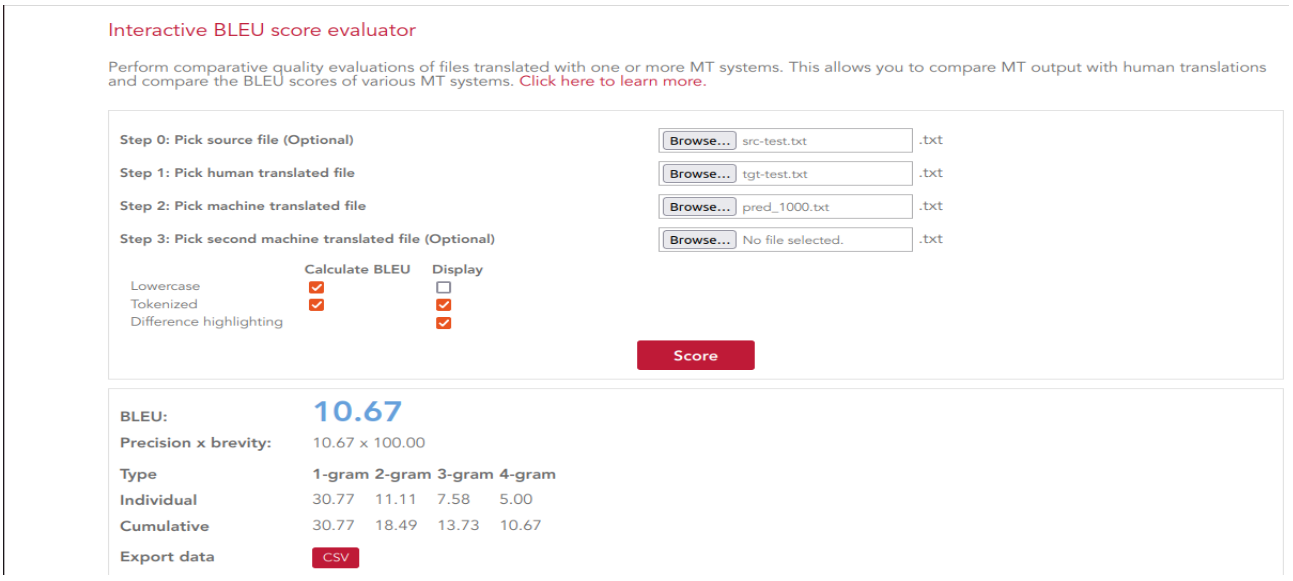


```
immaculate@immaculate-HP-ProBook-6460b: ~
sized since it had shape [25], which does not match the required output shape [5, 5].This behavior is deprecated, and in a future PyTorch rele
ase outputs will not be resized unless they have zero elements. You can explicitly reuse an out tensor t by resizing it, inplace, to zero elem
ents with t.resize_(0). (Triggered internally at  ../aten/src/ATen/native/Resize.cpp:24.)
  torch.mul(self.topk_scores, length_penalty, out=self.topk_log_probs)
/home/immaculate/.local/lib/python3.8/site-packages/onmt/translate/beam_search.py:212: UserWarning: __floordiv__ is deprecated, and its behavi
or will change in a future version of pytorch. It currently rounds toward 0 (like the 'trunc' function NOT 'floor'). This results in incorrect
 rounding for negative values. To keep the current behavior, use torch.div(a, b, rounding_mode='trunc'), or for actual floor division, use tor
ch.div(a, b, rounding_mode='floor').
  self._batch_index = self.topk_ids // vocab_size
[2022-05-13 10:11:51,255 INFO]
SENT 1: ['Kuma', 'vala', 'nĩĩ', 'nyie', 'Yakovo', 'mũthũkũmi', 'wa', 'Ngai', 'na', 'wa', 'Mwĩai', 'Yesũ', 'Klĩsto', '.', 'Valũa', 'ũũ', 'nĩ',
'kwenyu', 'andũ', 'onthe', 'ma', 'Ngai', 'ĩũlũ', 'wa', 'nthĩ', '.', 'Nĩnamũkethya', '.']
PRED 1: For the kingdom of our Lord Jesus Christ be with all of you , and the glory of our Lord Jesus Christ be with all of you .
PRED SCORE: -13.0048

[2022-05-13 10:11:51,255 INFO]
SENT 2: ['Mbaitũ', ',', 'yĩla', 'mwakwatwa', 'nĩ', 'matatwa', 'ma', 'kĩla', 'mũthemba', ',', 'osaai', 'ũndũ', 'ũsu', 'na', 'ũtanu', ',']
PRED 2: Brothers and sisters , I could not address you to be sure as the other brothers and sisters .
PRED SCORE: -12.8289

[2022-05-13 10:11:51,256 INFO]
SENT 3: ['nĩkwĩthĩwa', 'nĩmwĩsĩ', 'yĩla', 'mwatatwa', 'mũĩkĩĩonĩ', 'na', 'mwaema', 'kũvalũka', ',', 'ũu', 'ũtumaa', 'mwĩthĩwa', 'na', 'wũmĩĩsy
o', '.']
PRED 3: to show mercy to our ancestors and to remember his holy time ,
PRED SCORE: -7.0432

[2022-05-13 10:11:51,256 INFO]
SENT 4: ['Wakyeewa', 'ata', '?']
PRED 4: And God said to Noah ,
PRED SCORE: -4.4932

[2022-05-13 10:11:51,256 INFO]
SENT 5: ['Kamwana', 'kaa', 'nĩ', 'kau', '?']
PRED 5: <unk>
PRED SCORE: -2.1602

[2022-05-13 10:11:51,256 INFO] PRED AVG SCORE: -0.5900, PRED PPL: 1.8040
immaculate@immaculate-HP-ProBook-6460b:~$
```

Figure 8: Translated Sentences

Model Evaluation

After developing the model, the system was evaluated in order to check how good the translations are. The corpora used had 3,500 sentences as training data, 1000 sentences as validation data and 100 sentences as test data. An accuracy of -0.59 and perplexity of 1.8 on 100 sentences. A BLUE score of 10.6 was obtained.



Interactive BLEU score evaluator

Perform comparative quality evaluations of files translated with one or more MT systems. This allows you to compare MT output with human translations and compare the BLEU scores of various MT systems. Click here to learn more.

Step 0: Pick source file (Optional) — src-test.txt .txt
Step 1: Pick human translated file — tgt-test.txt .txt
Step 2: Pick machine translated file — pred_1000.txt .txt
Step 3: Pick second machine translated file (Optional) — No file selected. .txt

	Calculate BLEU	Display
Lowercase	✓	☐
Tokenized	✓	✓
Difference highlighting		✓

BLEU: 10.67

Precision x brevity: 10.67 x 100.00

Type	1-gram	2-gram	3-gram	4-gram
Individual	30.77	11.11	7.58	5.00
Cumulative	30.77	18.49	13.73	10.67

Export data: CSV

Figure 9:Blue score

Conclusion

The proposed Kikamba-English Translation system was completed successfully though the translations were not accurate. The low accuracy of the system is as a result of small dataset and the domain in which the it was collected. The entire dataset was limited to the religious domain which is not favorable for such a system. Lack digitized sources for the Kikamba Languages also contributed to its low performance. For future work, it is recommended to use large datasets from different domains in order to obtain better results.

References

[1] L. N. Vieira, M. O'Hagan, and C. O'Sullivan, "Understanding the societal impacts of machine translation: a critical review of the literature on medical and legal use cases," *Inf. Commun. Soc.*, vol. 24, no. 11, pp. 1515–1532, 2021, doi: 10.1080/1369118X.2020.1776370.

[2] S. R. Laskar, A. Dutta, P. Pakray, and S. Bandyopadhyay, "Neural machine translation: English to hindi," *2019 IEEE Conf. Inf. Commun. Technol. CICT 2019*, pp. 1–6, 2019, doi: 10.1109/CICT48419.2019.9066238.

[3] I. Orife *et al.*, "Masakhane -- Machine Translation For Africa," pp. 1–4, 2020, [Online]. Available: http://arxiv.org/abs/2003.11529.

[4] M. Singh, R. Kumar, and I. Chana, "Neural-Based Machine Translation System Outperforming Statistical Phrase-Based Machine Translation for Low-Resource Languages," *2019 12th Int. Conf. Contemp. Comput. IC3 2019*, pp. 1–7, 2019, doi: 10.1109/IC3.2019.8844915.

[5] Z. Zong and C. Hong, "On application of natural language processing in machine translation," *Proc. - 2018 3rd Int. Conf. Mech. Control Comput. Eng. ICMCCE 2018*, pp. 506–510, 2018, doi: 10.1109/ICMCCE.2018.00112.

[6] R. Dabre, C. Chu, and A. Kunchukuttan, "A Survey of Multilingual Neural Machine Translation," *ACM Comput. Surv.*, vol. 53, no. 5, 2020, doi: 10.1145/3406095.

[7] B. Kituku, W. Nganga, and L. Muchemi, "Towards Kikamba Computational Grammar," *J. Data Anal. Inf. Process.*, vol. 07, no. 04, pp. 250–275, 2019, doi: 10.4236/jdaip.2019.74015.

[8] G. Klein, Y. Kim, Y. Deng, V. Nguyen, J. Senellart, and A. M. Rush, "OpenNMT: Neural machine translation toolkit," *AMTA 2018 - 13th Conf. Assoc. Mach. Transl. Am. Proc.*, vol. 1, pp. 177–184, 2018.

[9] K. Deep, A. Kumar, and V. Goyal, "Machine Translation System Using Deep Learning for Punjabi to English," vol. 2022, pp. 865–878, 2021, doi: 10.1007/978-981-15-7533-4_69.

[10] P. Lohar, M. Popović, and A. Way, "Building English-to-Serbian Machine Translation System for IMDb Movie Reviews," no. August, pp. 105–113, 2019, doi: 10.18653/v1/w19-3715.

[11] M. Singh, R. Kumar, and I. Chana, "Corpus based Machine Translation System with Deep Neural Network for Sanskrit to Hindi Translation," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 2534–2544, 2020, doi: 10.1016/j.procs.2020.03.306.

[12] P. Shah and V. Bakrola, "Neural machine translation system of indic languages-an attention based approach," *2019 2nd Int. Conf. Adv. Comput. Commun. Paradig. ICACCP 2019*, 2019, doi: 10.1109/ICACCP.2019.8882969.

[13] peter W. Benson Kituku, Musumba George, "S peech & T heatre," vol. 4, no. 2, pp. 43–53,

2015.

[14] Z. Tan *et al.*, “Neural machine translation: A review of methods, resources, and tools,” *AI Open*, vol. 1, no. November 2020, pp. 5–21, 2020, doi: 10.1016/j.aiopen.2020.11.001.

[15] “BibleGateway.” https://www.biblegateway.com/.

[16] “OpenNMT-py,” [Online]. Available: https://opennmt.net/OpenNMT-py/quickstart.html.