Machine Translation using Transformers(As NLP Application for Hindi to Punjabi Language Translation)

Conversational AI : Natural Language ProcessingProject

Submitted by:

(102166004) Vimlendu Sharma(3CS11)

BE Third Year

Under the Mentorship

of**Dr. Jasmeet Singh**

Assistant Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

June 2024

Description of Machine Translation(NLP Application)

This project involves the Translation of Hindi to Punjabi Language using Machine Translation as an crucial application of Natural Language Processing using Transformers for this Language Translation Task.

Machine translation refers to the use of computer algorithms and techniques to automatically translate text or speech from one language to another.

The goal of machine translation is to produce accurate and fluent translations that convey the meaning of the original text or speech in the target language.

The foundation Transformer model was specifically designed for machine translation.

The task described in this project involves building a Machine Translation system using the Transformer architecture for translating text from Hindi to Punjabi.

The NLP application aims to perform machine translation from Hindi to Punjabi using a transformer-based model. The model is fine-tuned on a parallel corpus of Hindi and Punjabi texts to learn the translation patterns between these two languages.

Description of DataSet used for Machine Translation Task

For Hindi to Punjabi Language Translation:-

We require two parallel corpus of language text, one for Hindi language and other for Punjabi Language.

1. For Hindi Language:- We can download the Hindi Language Corpora from the Leipzig Wortschatz Website which contain around 30 thousand text sentences written in Hindi Language.
2. For Punjabi Language:- We can download the CC100 Punjabi language dataset which contain text messages in Punjabi language and we can get it from MetaText website.

Description about Transformer Model used

A multilingual version of the BART (Bidirectional and Auto-Regressive Transformers) architecture, the Transformer Model used in this machine translation task is '**Facebook/mbart-large-50-many-to-many-mmt**'. It is intended for sequence-to-sequence tasks such as machine translation. It is a member of the mBART (multilingual BART) family and was created by Facebook AI. With the help of this particular paradigm, many-to-many translation is possible between 50 differentlanguages.

Overview of mBART

The original BART model is expanded by mBART to support many languages. BART is a transformer-based denoising auto-encoder that combines aspects of GPT (autoregressive decoding) and BERT (bidirectional encoding). It is intended for jobs like text generation, summarisation, and translation that require the creation of text.

Key Features of 'Facebook/mbart-large-50-many-to-many-mmt'

1. **Multilingual Capability:-** supports translation between 50 languages, including less often used languages and widely spoken languages like English, Spanish,French, and Chinese.

2. **Pre-Training and Fine-Tuning:-** Following pre-training, the model is improved using certain translation tasks. It also provides fine-tuning on parallel corpora for many language pairs for 'Facebook/mbart-large-50-many-to-many-mmt'.

3. **Architecture:-** Encoder-Decoder Structure: A stack of encoders and decoders is used in the transformer architecture paradigm.

Attention Mechanism: Enables it to capture contextual dependencies within and between sentences by using cross-attention from the decoder to the encoder and self-attention in both the encoder and the decoder.

Positional Encodings: Since transformers are not designed to comprehend the sequence in which tokens are placed, this adds positional information to the input tokens.

4. **Tokenisation:-** uses a SentencePiece tokeniser(MarianTokenizer) that is compatible with all supported languages and has a common vocabulary. This guarantees consistent multilingual text processing and tokenisation.

```
1.3.0)
```

```
In [3]: from transformers import MarianMTModel,MarianTokenizer
```

```
In [4]: !pip install accelerate -U
        !pip install -U transformers accelerate
```

```
Requirement already satisfied: accelerate in /Users/anaconda3/lib/python3.11/site-packages (0.30.1)
Requirement already satisfied: numpy>=1.17 in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (1.2
4.3)
Requirement already satisfied: packaging>=20.0 in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (
23.1)
Requirement already satisfied: psutil in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (5.9.0)
Requirement already satisfied: pyyaml in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (6.0)
Requirement already satisfied: torch>=1.10.0 in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (2.
2.0)
Requirement already satisfied: huggingface-hub in /Users/anaconda3/lib/python3.11/site-packages (from accelerate) (
0.23.2)
Requirement already satisfied: safetensors>=0.3.1 in /Users/anaconda3/lib/python3.11/site-packages (from accelerat
e) (0.4.3)
Requirement already satisfied: filelock in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->accel
erate) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /Users/anaconda3/lib/python3.11/site-packages (from torc
h>=1.10.0->accelerate) (4.9.0)
Requirement already satisfied: sympy in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->accelera
te) (1.11.1)
Requirement already satisfied: networkx in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->accel
erate) (3.1)
Requirement already satisfied: jinja2 in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->acceler
ate) (3.1.2)
Requirement already satisfied: fsspec in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->acceler
ate) (2024.5.0)
Requirement already satisfied: requests in /Users/anaconda3/lib/python3.11/site-packages (from huggingface-hub->acc
elerate) (2.31.0)
Requirement already satisfied: tqdm>=4.42.1 in /Users/anaconda3/lib/python3.11/site-packages (from huggingface-hub-
>accelerate) (4.65.0)
```

Snapshots of Code(Jupyter Notebook)

```
In [7]: import pandas as pd
        from datasets import Dataset

        hindi_file = "hin_mixed_2019_30K-sentences.txt"
        punjabi_file = "pa.txt"

        with open(hindi_file, 'r', encoding='utf-8') as f:
            hindi_lines = f.readlines()

        with open(punjabi_file, 'r', encoding='utf-8') as f:
            punjabi_lines = f.readlines()

        if len(hindi_lines) > len(punjabi_lines):
            hindi_lines = hindi_lines[:len(punjabi_lines)]
        elif len(punjabi_lines) > len(hindi_lines):
            punjabi_lines = punjabi_lines[:len(hindi_lines)]

        # Verify that both lists are now of equal length
        assert len(hindi_lines) == len(punjabi_lines), "The Hindi and Punjabi corpora must have the same number of lines."

        # Create a pandas dataframe
        data = {'hi': hindi_lines, 'pu': punjabi_lines}
        df = pd.DataFrame(data)

        print(f"Number of lines after equalization: {len(hindi_lines)}")

        dataset = Dataset.from_pandas(df)

        tokenizer.src_lang = "hi_IN"
        tokenizer.tgt_lang = "pa_IN"

        def preprocess_function(examples):
            inputs = [ex for ex in examples['hi']]
            targets = [ex for ex in examples['pu']]
            model_inputs = tokenizer(inputs, max_length=128, truncation=True, padding="max_length")
            with tokenizer.as_target_tokenizer():
                labels = tokenizer(targets, max_length=128, truncation=True, padding="max_length")
            model_inputs["labels"] = labels["input_ids"]
            return model_inputs
```

In [5]:
```
import transformers
import accelerate

print(transformers.__version__)
print(accelerate.__version__)
```

```
4.41.2
0.30.1
```

In [6]:
```
from transformers import MBartForConditionalGeneration, MBart50TokenizerFast

model_name = "facebook/mbart-large-50-many-to-many-mmt"
tokenizer = MBart50TokenizerFast.from_pretrained(model_name)
model = MBartForConditionalGeneration.from_pretrained(model_name)
```

```
    return model_inputs

tokenized_datasets = dataset.map(preprocess_function, batched=True)
```

```
Number of lines after equalization: 30000
```

```
/Users/anaconda3/lib/python3.11/site-packages/transformers/tokenization_utils_base.py:3946: UserWarning: `as_target
_tokenizer` is deprecated and will be removed in v5 of Transformers. You can tokenize your labels by using the argu
ment `text_target` of the regular `__call__` method (either in the same call as your input texts if you use the sam
e keyword arguments, or in a separate call.
  warnings.warn(
```

In [8]:
```
tokenized_datasets = tokenized_datasets.train_test_split(test_size=0.1)
train_dataset = tokenized_datasets["train"]
val_dataset = tokenized_datasets["test"]
```

In [9]:
```
source_lang='hi_IN'
target_lang='pa_IN'
prefix='translate Hindi to Punjabi:'
def preprocess_text(examples):
  inputs=[prefix+example[source_lang] for example in examples['translation']]
  targets=[example[target_lang] for example in examples['translation']]
  tokenized_inputs=tokenizer(inputs,max_length=128,truncation=True)
  with tokenizer.as_target_tokenizer():
        labels = tokenizer(targets, max_length=128, truncation=True)

  tokenized_inputs["labels"] = labels["input_ids"]
  return tokenized_inputs
```

In [12]:
```
from transformers import DataCollatorForSeq2Seq
datacollator=DataCollatorForSeq2Seq(tokenizer=tokenizer,model=model,return_tensors='tf')
```

jupyter Machine_Translation_using_Transfomers_from_hindi_to_punjabi_language Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import pandas as pd
!pip install datasets
!pip install evaluate
!pip install scarbleu
!pip install accelerate -U
```

```
ate) (3.1.2)
Requirement already satisfied: fsspec in /Users/anaconda3/lib/python3.11/site-packages (from torch>=1.10.0->acceler
ate) (2024.5.0)
Requirement already satisfied: requests in /Users/anaconda3/lib/python3.11/site-packages (from huggingface-hub->acc
elerate) (2.31.0)
Requirement already satisfied: tqdm>=4.42.1 in /Users/anaconda3/lib/python3.11/site-packages (from huggingface-hub-
>accelerate) (4.65.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/anaconda3/lib/python3.11/site-packages (from jinja2->torch
>=1.10.0->accelerate) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/anaconda3/lib/python3.11/site-packages (from requ
ests->huggingface-hub->accelerate) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/anaconda3/lib/python3.11/site-packages (from requests->huggin
gface-hub->accelerate) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/anaconda3/lib/python3.11/site-packages (from requests->
huggingface-hub->accelerate) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /Users/anaconda3/lib/python3.11/site-packages (from requests->
huggingface-hub->accelerate) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /Users/anaconda3/lib/python3.11/site-packages (from sympy->torch>=1.
10.0->accelerate) (1.3.0)
```

In [2]:
```
!pip install transformers datasets torch
```

```
Requirement already satisfied: transformers in /Users/anaconda3/lib/python3.11/site-packages (4.41.2)
Requirement already satisfied: datasets in /Users/anaconda3/lib/python3.11/site-packages (2.12.0)
Requirement already satisfied: torch in /Users/anaconda3/lib/python3.11/site-packages (2.2.0)
Requirement already satisfied: filelock in /Users/anaconda3/lib/python3.11/site-packages (from transformers) (3.13.
1)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /Users/anaconda3/lib/python3.11/site-packages (from
transformers) (0.23.2)
```