CS F437 Generative AI

Assignment 1

Training and Evaluating Transformer Models for English to Hindi Translation

A Project Report

By

2021B3A73032H	**Mehul Kochar**
2022A7PS0002H	**Simran Sesha Rao**
2022A7PS0233U	**Vennela Vallabhaneni**

under the supervision of

Prof. N L Bhanu Murthy

Birla Institute of Technology and Science, Pilani

Hyderabad Campus

Introduction

While the rise of generative AI has been exponential over the past five years, its foundational subject, natural language processing, has been around for quite a while. In fact, one of the very first and most utilitarian use-cases of natural language processing was machine translation, with the first recorded project being the completely automatic translation of 60+ sentences in Russian to English, back in 1954.

Since then, machine translation has become much more sophisticated and involves many intermediary processes; today, a simple translation task would involve completely decoding the source text's semantics, including analyzing all the features of the text, while having already gathered in-depth knowledge of the grammar, syntax, colloquialisms of the source language and then encoding the deduced meaning into the target language appropriately.

To accomplish this, there have been two majorly implemented methods of machine translation; rule-based, and corpus-based. However, HMT and NMT are newer techniques that have been implemented and have yielded better results for machine translation tasks.

1. **RBMT**

   Rule-based Machine Translation, or knowledge-based translation, is the classical method of translation. It is a system that relies majorly on information derived from dictionaries and the grammars of the source and the target languages.

   Sub-approaches within the RBMT techniques include the direct machine translation approach, the interlingual machine translation approach, and the transfer-based machine translation approach.

2. **CBMT**

   Corpus-based Machine Translation, or data-driven translation, involves techniques that are able to complete their task without tackling the problem of knowledge acquisition while also being able to handle linguistic issues, cultural items and concepts.

   This is achieved by using large amounts of raw data in parallel corpora that contains text and their translations. CBMT approaches are further classified into SMT (Statistical Machine Translation) and EBMT (Example-based Machine Translation).

3. **HMT**
   Hybrid Machine Translation techniques make use of both the statistical and rule-based translation methodologies; in cases where translations use a rule-based approach, the output can be corrected using statistical information. On the other hand, rules can be used to pre-process the input data and post-process the output of a statistics-based translation system.

4. **NMT**
   Neural Machine Translation methods utilize NN models, which are capable of incorporating training data that can either be generic (includes data learned from translations done over time, can be used as a general translator tool) or custom (includes training data specifically fed to the model so as to specialize it for a certain purpose).

   In the recent past, NMT has achieved SOTA performance in large-scale translation tasks in multiple languages. One of its greatest strengths is the fact that it is very conceptually simple, requires minimal domain knowledge, and yet is very powerful. Additionally, since it does not maintain phase tables, it has a small memory footprint.

In this project, we aim to train and evaluate transformer models for English to Hindi translation. We will be utilizing neural machine translation (NMT) methods for this purpose. The tasks involved in this project are dataset preprocessing for NMT, training a transformer-based model from scratch, fine-tuning a pre-trained model on a given dataset, and evaluating and comparing the performances of the different approaches using standard machine translation metrics.

Methodology
The dataset used was the IITB English-Hindi dataset developed and published by the Centre for Indian Language Technology (CFILT), and the transformer chosen for training and fine-tuning was MarianMT.

Dataset Features
- English (source) and Hindi (target)
- 1.66M+ parallel sentence pairs
- Parquet format on Hugging Face

Data Splits
- Training Set: 50K sentence pairs (rows)
- Validation Set: 520 sentence pairs (rows)

- Test Set: 2.51K sentence pairs (rows).

Part 1: Fine-Tuning a Transformer Model

The fine-tuned model was developed using a supervised learning approach with PyTorch Lightning to enhance the translation accuracy of the Helsinki-NLP/opus-mt-en-hi model. The methodology involved the following steps:

1. **Dataset Selection and Preprocessing**
    - The **IITB English-Hindi parallel corpus** was chosen as the dataset for training and evaluation.
    - Text pairs were extracted, tokenized using the `AutoTokenizer` from Hugging Face, and mapped into sequences with a maximum length of 128 tokens.
    - The dataset consisted of **training, validation, and test sets**, the training set is randomly sampled 50,000 points in each epoch using a `RandomSubsetSampler`.

2. **Model Architecture and Training Process**
    - The `Helsinki-NLP/opus-mt-en-hi` model, a MarianMT-based sequence-to-sequence transformer, was fine-tuned using the training set.
    - A **custom PyTorch Lightning module** was implemented to handle forward passes, loss computation, and evaluation.
    - The **AdamW optimizer** was used with an initial learning rate of **1e-4**, and learning rate tuning was performed using `lr_find` from PyTorch Lightning.
    - The model was trained using **gradient accumulation** over a subset of the dataset, with a batch size of 16 and a maximum of **10 epochs**.
    - `EarlyStopping` was implemented to halt training if validation loss did not improve for 3 consecutive epochs.

3. **Evaluation Metrics and Testing**
    - After training, the fine-tuned model was evaluated using the **test set**.
    - The evaluation included BLEU, ROUGE, METEOR, and token-level accuracy scores to compare predicted translations against reference translations.

4. **Model Saving and Deployment**
    - The trained model and tokenizer were saved using `save_pretrained()` for future inference use.

    - The model was tested on a set of sample English sentences to verify translation quality.

Part 2: Using a Pre-Trained Transformer Model
The pre-trained model was evaluated without fine-tuning to establish a baseline performance. The methodology included:

1. **Model Selection**
    - The **Helsinki-NLP/opus-mt-en-hi** model was selected as the pre-trained translation model.
    - This model was already trained on large-scale multilingual parallel corpora but had not been specifically optimized for the IITB English-Hindi dataset.

2. **Dataset Preparation**
    - The same **IITB English-Hindi** dataset was used for evaluation without additional fine-tuning.
    - Sentences were tokenized using the `AutoTokenizer` to match the model's input format.

3. **Baseline Evaluation**
    - The pre-trained model was tested using the same **test set**.
    - BLEU, ROUGE, METEOR, and token-level accuracy scores were computed to measure its translation performance.

4. **Comparison with Fine-Tuned Model**
    - The pre-trained model's test metrics were compared against the fine-tuned model's performance.
    - This comparison highlighted the benefits of task-specific fine-tuning for translation quality improvements.

By following this methodology, the experiment effectively demonstrated how fine-tuning on a task-specific dataset can enhance the accuracy of machine translation models.

Results and Comparison

Metric	Fine-tuned Model	Pre-trained Model
BLEU Score	10.647638320922852	9.330301284790039
ROUGE-L Score	0.11288895034407599	0.11275715356595344
Meteor Score	0.3243692357594135	0.29531525292165794
Accuracy	0.08877011388540268	0.07673494517803192

Translation Quality Analysis

Training time: 4494 seconds

1. <u>BLEU Score</u>
   The Bilingual Evaluation Understudy (BLEU) score ranges from 0 to 1, with 0 meaning that there is no semantic overlap with the reference translations, and 1 meaning that there is a perfect overlap, i.e. a perfect translation.

Fine-tuned Model	Pre-trained Model
0.1065	0.0933

   The fine-tuned model's higher BLEU score indicates better n-gram precision and higher translation fluency.

2. <u>ROUGE-L Score</u>
   The Recall-Oriented Understudy for Gisting Evaluation (ROUGE-L focuses on the longest common subsequence) score ranges from 0 to 1, with 0 representing zero similarity, and 1 representing perfect similarity.

Fine-tuned Model	Pre-trained Model
0.1128	0.1127

There is a very small improvement in the fine-tuned model's ROUGE-L score, which suggests that it maintains slightly better recall for longer sequences.

3. <u>Meteor Score</u>
   The Metric for Evaluation of Translation with Explicit ORdering (METEOR) score ranges from 0 to 1, where a higher score signifies a better match between the generated text and the reference tex

Fine-tuned Model	**Pre-trained Model**
0.3243	0.2953

   The fine-tuned model has a noticeable improvement, meaning it has better synonym matching, stemming, and word order.

4. <u>Accuracy</u>
   Accuracy scores reflect the amount of similarity of the translated text and the high-quality reference translation. The score ranges from 0 to 1, with 1 being the most accurate translation.

Fine-tuned Model	**Pre-trained Model**
0.0888	0.0767

   Though low, the fine-tuned model outperforms the pretrained one, implying better correctness in predictions.

Overall Analysis

Fine-tuning improves translation quality across all metrics, with the most significant gains in BLEU and Meteor scores. This suggests that the fine-tuned model is more fluent and captures meaning better, making it preferable for translation tasks.

Comparison of Pre-Trained Model vs. Fine-Tuned Model

While training models from scratch is a tedious process, it allows for a large amount of flexibility with regards to the architecture design and customizability with regards to domain-specific requirements.

Additional benefits include potentially avoiding biases that could exist in publicly available pre-trained models and better results in the case of a large, high-quality dataset.

However, there are multiple reasons why training models from scratch isn't a preferred approach. This is a process that requires extreme amounts of computational power and storage, which makes it rather expensive in terms of resource utilization. Training a model also requires a large amount of data, a long amount of time, and a lot of perseverance (since SOTA performance is not easily achievable).

Therefore, fine-tuning existing models is one of the most commonly taken approaches, mostly due to its low computational costs, faster training and low data requirements while simultaneously producing high-quality feature representations due to pre-existing knowledge. And, the prowess of many pretrained models has been tested and optimized extensively, giving them more reliability as models.

However, pre-trained models usually have constraints on memory, and are not the right choice for use cases that require flexibility and customization. They are also very likely to have inherited biases from their time during training. This implies that they are not the best option for highly specialized domains without fine-tuning.

Conclusion
For quick results and generalized translations, using the pre-trained model directly is ideal. For higher accuracy and domain-specific applications, fine-tuning is the better choice.
Both approaches have their merits, and the choice depends on the specific requirements of the translation task. However, for real-world applications, fine-tuning proves to be the more efficient and suitable approach, especially in domain-specific use cases such as legal, healthcare, or technical translations. Fine-tuning allows the model to achieve better accuracy and fluency, making it a preferred choice where high-quality translations are critical. Despite its computational cost, the improvements in translation quality justify the investment in fine-tuning for professional and commercial applications.

Low-Resource Translation Challenges and Improvements

- Challenges
    - Training high-quality models for requires significant computational power, and this is especially the case if the languages are low-resource
    - Existing datasets may contain inconsistencies, errors, or biases that affect translation quality and accuracy
    - Some low-resource languages have complex grammar and morphology, making translation difficult or inaccurate

- Improvements
    - Curating better datasets for low-resource languages and standardizing multiple low-resource languages using common high-resource languages
    - Implementing data augmentation (practices like synthetic data generation, multilingual datasets for model training improvement, etc.)
    - Utilizing knowledge from high-resource languages (transfer learning) or incorporating human feedback for refined translations (active learning)
    - Leveraging methods like meta-learning and unsupervised learning (essentially low-resource specific architectures) to improve model adaptability.

References

1. https://www.scaler.com/topics/nlp/machine-translation-in-nlp/
2. https://medium.com/@varun.tyagi83/multilingual-translation-a-deep-dive-into-marianmt-and-langcodes-57e794034af3
3. https://huggingface.co/docs/transformers/en/model_doc/marian
4. https://github.com/marian-nmt/marian
5. https://huggingface.co/datasets/cfilt/iitb-english-hindi
6. https://ieeexplore.ieee.org/abstract/document/9066238/
7. https://ieeexplore.ieee.org/abstract/document/7732363/
8. https://www.jsrtjournal.com/index.php/JSRT/article/view/59