CS7.401: Introduction to NLP | Assignment 3

Instructor: Manish Shrivastava

Deadline: April 5, 2022 | 23:55

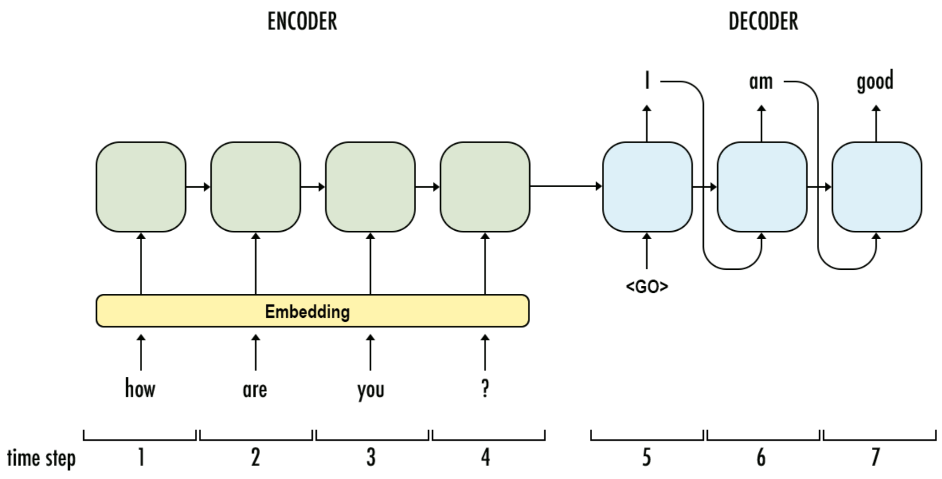
General Instructions

1. You should implement the assignment in Python.
2. Ensure that the submitted assignment is your original work. Please do not copy any part from any source, including your friends, seniors, and/or the internet. If any such attempt is caught, serious actions, including an F grade in the course, are possible.
3. A single .zip file needs to be uploaded to the Moodle submission portal.
4. Your grade will depend on the correctness of answers and output. Due consideration will also be given to the clarity and details of your answers and the legibility and structure of your code.
5. Please start early to meet the deadline. Late submissions won't be evaluated.

1 About

1.1 Sequence-to-Sequence Learning

A Sequence-to-Sequence model is a particular Neural Architecture used to solve complex language problems like Question-Answering, Machine Translation and Summarisation. It typically consists of two models: an Encoder (to understand and represent language) and a Decoder (to decode said representation as desired output).



Example of seq2seq model

An Encoder stores the context of the language in a "context vector". This context vector contains a summarised representation of all the input text. The Decoder's initial states are initialised as the final state of the Encoder (the context vector). Using the given context, the Decoder starts giving an output, which it uses to generate more output in the future. Since a Decoder's initial vector is the context vector of the Encoder, it intuitively means that the Decoder is trained to generate output sequence based on the information encoded by the Encoder.

1.2 Transfer Learning

Transfer Learning is a method in Machine Learning in which a model stores the knowledge gained by solving one problem and applies it to another different but related problem. It has recently dominated the field of NLP since it is expensive to train models on massive datasets repeatedly. Instead, it is preferred to train the model on general datasets using self-supervised learning and then use the information gained on other tasks. Such models are called pre-trained models. Transfer Learning is also practical for cases where it is challenging to procure large datasets for the problem since you now need fewer data samples to get meaningful results.

Although the main model has learnt language representation on general tasks, it is fruitful to gear the model towards the desired downstream task. This process is called fine-tuning the model, which requires significantly fewer data points than training a model from scratch for the same task.

2 Neural Language Model

You have been given two subsets of monolingual corpora for English (Europarl) and French (News Crawl). They have train, dev and test splits. Although you'll be training two separate models on both the corpora, the explicit evaluation for the language modeling task will be done on only the English language model, while the trained weights of the language model on French will be directly used in the subsequent translation task.

1. Create a neural language model using a recurrent architecture such as LSTMs for both the language corpora provided. Use the train split for training the model parameters, dev split for hyperparameter optimization as well saving the best model checkpoint and test split for a final check on the performance.
2. Perplexity computation:
   (a) Calculate the perplexity score for each sentence of the train split of the **English corpus only** using the trained model and also get the averaged perplexity score on it.
   (b) Report the perplexity score on the test sentences as well, in the same manner as above.

3 Machine Translation

You have been given a separate parallel corpus of English and French sentences (a subset of the TED Talks Corpus) for training on the translation task. In the parallel corpus, the *i*th sentence in the French corpus is the translation of the *i*th sentence in the English corpus. Your task is to design a Sequence-to-Sequence Machine Translation model to translate from English to French using LSTMs on this corpus.
The corpus contains train, dev and test splits which are to be used in a similar way as that for the Neural Language Model part.

1. Create a Sequence-to-Sequence model in the following ways:
   (a) MT-1: Both - Encoder and Decoder are trained on the parallel corpus from scratch.
   (b) MT-2: Use weights from the LSTM model trained on the language modeling task on English for the Encoder, and weights from the model trained on the French task for the Decoder. Then, using these pretrained weights as the starting point, finetune the Encoder and Decoder on the parallel corpus for translation.
2. Evaluation using BLEU scores:
   (a) Calculate BLEU score for each sentence of the parallel corpus for each of the above models on the train set and also get the corpus-level BLEU score for both the MT models on the train corpus.
   (b) Report the BLEU scores for all the sentences in the test set in a similar manner above.
3. Compare and analyse the behaviour of the two models and put your analysis and any visualisations in a report.

You may try other recurrent neural architectures like vanilla-RNN or GRU based on your choice of modeling - there's no compulsion on using LSTMs. Specify any such choices you make in the README.

4 Corpus

The following corpora have been given to you for training:

1. **Europarl Corpus**
   A subset taken from the English version of the Europarl v7 dataset. This is a monolingual English corpus to be used for training the neural language model for English. It contains the files for train, dev and test splits as below:
   (a) train.europarl: 20000 lines
   (b) dev.europarl: 500 lines
   (c) test.europarl: 1000 lines
2. **News Crawl Corpus**
   A subset from the News Crawl 2013 corpus for French language. This is a monolingual French corpus for training the neural language model for French. It contains the files for train, dev and test splits as below:
   (a) train.news: 20000 lines
   (b) dev.news: 500 lines
   (c) test.news: 1000 lines
3. **TED Talks Corpus**
   A subset of the dataset for the English-French translation task in IWSLT 2016.
   It contains the files for train, dev and test splits as below:
   (a) train.[en|fr]: 30000 lines each
   (b) dev.[en|fr]: 887 lines each
   (c) test.[en|fr]: 1305 lines each

   Here, the extensions .en and .fr signify the English and French part of the parallel corpus, respectively.

Please download the corpus files from this link.

5 Submission Format

Zip the following into one file and submit to the Moodle course portal. Filename should be <roll number>_<assignment3>.zip, e.g.: 2021xxxxxx_assignment3.zip:

1. **Source Code:**
   Note that the files mentioned here need to be present as a part of the submission, but your logic in code could be divided into a number of different other additional files - just specify the same in README, and submit the complete, working code in submission.
   (a) `language_model.py`: Runs the language model given the following:

```
$ python3 language_model.py <path to model>
```

   On running the file, the expected output is a prompt, which asks for a sentence and provides the probability of that sentence. Therefore, an example would be:

```
$ python3 language_model.py ../models/trained_en_lm.pth
input sentence: I am a man.
0.89972021
```

(b) machine_translation.py: Runs the MT model given the following:

```
$ python3 machine_translation.py <path to model>
```

Input for this part will be the model path to either MT1 or MT2. On running the file, the expected output is a prompt, which asks for a sentence and provides the translated sentence in target language. Therefore, an example would be:

```
$ python3 machine_translation.py ../models/trained_mt2.pth
input sentence: I am a man.
je suis un homme.
```

2. **Output Files:**
These are to be submitted only for the train and test splits. Use the dev split for obtaining the best model parameters over training epochs and other hyperparameter optimization - we don't expect to have output files on the dev split.

   (a) For the LM, submit a text file for the train & test splits with the average perplexity and the perplexity score of each sentence in the following format:

```
avg_perplexity
sentence_1 <tab> perplexity
sentence_2 <tab> perplexity
..
```

   As mentioned earlier, these files need to be present only for the language model trained on English (Europarl) data.

   (b) For each MT model, submit a text file with BLEU score for each translated sentence in the following format:

```
corpus_BLEU
translated_sentence_1 <tab> sentence_BLEU
translated_sentence_2 <tab> sentence_BLEU
..
```

   Here, `corpus_BLEU` stands for the corpus-level BLEU score over all the sentence pairs in the corpus, while `sentence_BLEU` represents the individual sentence-pair-wise BLEU score. It is important to note that the corpus-level BLEU score is to be computed separately, and is not the same as the simple average of sentence-pair-wise BLEU scores.

   In total, you will have 6 text files:

   i <roll-number>_LM_train.txt
   ii <roll-number>_LM_test.txt
   iii <roll-number>_MT1_train.txt
   iv <roll-number>_MT1_test.txt
   v <roll-number>_MT2_train.txt
   vi <roll-number>_MT2_test.txt

3. **Report** containing the average perplexity scores of LM and corpus-level BLEU scores of MT models along with your analysis of the results in a PDF.

4. **README** on how to execute the files, how to get perplexity of sentences along with any other information.

6 Grading

Evaluation will be individual and will be based on your viva, report and submitted code review. In the slot you are expected to walk us through your code, explain your experiments, and report. You will be graded based on the correctness of your code, accuracy of your results and quality of the code. Other factors such as inclusion of a proper README file in the code submission, crisp walk-through of the code, etc. will also play a role.

Marks Distribution (out of 100):

1. Neural Language Model: **30** Marks
2. Machine Translation: **60** Marks
   (a) Seq-2-Seq: **25** Marks
   (b) Seq-2-Seq with Fine-Tuning: **35** Marks
3. Quality, efficiency & presentation: **10** Marks

7 FAQs

1. Do I need to train on the whole corpus?
   Yes, the size of the training corpus is 40k for the Language Model tasks (combined) and 30k for Machine Translation - so compute should not be an issue. Ensure that you use GPU for training with an appropriate batch size so as to make the process faster.
2. Can I submit my code in Jupyter Notebook?
   No, the final submission should be a Python script. You may work using Jupyter Notebooks, but make sure to convert them to `.py` files before submitting.
3. Do I need to code everything from scratch?
   No. You are free to use neural layers from deep learning frameworks like PyTorch / Keras etc. Do not use direct implementation of a model though!
4. My model takes too long for train/test. What should I do?
   Ensure you are using a GPU environment with proper batch sizes for training. For computing answer given an input, ensure you are storing the weights and not calculating them while running the script.
5. Am I expected to implement the BLEU metric?
   No, that's not necessary - there are several implementations of BLEU readily available, and you can use any of them. A simple-to-use one could be the one available in the NLTK library. Same goes for the perplexity score calculation, but then you've probably already learnt to implement it in the first assignment.

8 Resources

1. Sequence to Sequence Learning with Neural Network
2. Understanding LSTM Networks
3. Mechanics of Seq2Seq Model
4. Fine-Tuning Representation Model
5. Stanford Slides on Seq2Seq Model
6. NLP From Scratch in PyTorch