Best way to learn NLP is to get your hands dirty, by actually doing it.

1. Your First Job in the Industry

You recently graduated and have been employed in the tech team of a mass-media conglomerate that publishes news and magazines in Hindi. The company now wants to expand its operations and wants to publish its articles and editorials in other languages including English. The CEO of the company came to know about the wonders of deep learning and he wants to deploy these technologies in the company. Since you are part of the tech team and more importantly you have done the CS779 course at IITK, the company wants you to develop a neural machine translation system that could automatically translate the articles/editorials written in Hindi to English. However, later someone else also approached the CEO and promised to develop a state of the art neural machine translation system. But since the CEO has faith in you and to be fair with all, he has organized a competition for developing a Hindi-English MT system.

From the course, you remember different types of neural models, e.g., RNN based models, sequence-to-sequence models, transformer models. You plan to use these to implement a MT system for translation from Hindi to English.

This competition requires you to implement Neural Machine Translation (NMT) models to translate from Hindi to English. One of the prominent model architectures used in NMT is encoder-decoder architecture. It consists of two components: an encoder and a decoder. The encoder takes the source language sentence (e.g., Hindi) as input and learns a representation for it, this representation is then fed to the decoder that then decodes (i.e., predicts) the target language sentence (e.g, English) word by word. There is flexibility with regard to the neural architecture of the encoder and the decoder.

In this competition, you have the freedom to choose any neural architecture for the encoder and the decoder. Similarly, for decoding the target language sentence, you have the freedom to choose any decoding strategy, e.g., greedy, beam search, etc.

2. Background Tutorials

To help you get started we are providing some background tutorials that you might find useful:

1. Tutorial on text pre-processing using Spacy, Regex, and IndicNLP. Clone it, to experiment with it (if you have any doubts please contact Shubham Nigam): Text Pre-Processing
2. Tutorial on implementing Seq2Seq models in PyTorch (if you have any doubts please contact Samik Some): https://github.com/lkulowski/LSTM_encoder_decoder
3. Tutorial on implementing Transformers: Clone it, to experiment with it (if you have any doubts please contact Samik Some): Transformers

---

NOTE: This is a dynamic document, it will keep getting updated as and when required

3. Dataset

We are only providing the train set and evaluation script. You would have to randomly split (e.g., 80-20 split) the train set into train and dev (validation) sets and use the dev set to evaluate your models internally. This will give you a good idea about different models that you would be experimenting with and help you in selecting the final model. Your final model will be evaluated using a separate test set that will not be provided to you during the training phase.

The use of external datasets is NOT allowed in this competition. Train dataset consists of 102,322 Hindi-English sentence pairs. The train set and evaluation scripts are provided at: Train set

The train dataset has been curated from the following publicly available sources: https://opus.nlpl.eu/ and http://www.opensubtitles.org/ . The datasets on these website have been proposed in the following papers: [3], [4], [5], [6]

4. Possible Ideas

a. Seq2Seq Architecture

There are various ways of implementing an NMT system. One possibility is to use a sequence to sequence (seq2seq) architecture [1, 2]. I will be covering seq2seq models in the coming lectures. Nevertheless, for the competition I would suggest that you make a head start by reading the following tutorial on NMT: https://arxiv.org/pdf/1703.01619.pdf (sections: 1, 2, 5, 6, 7 and 8 are interesting for the purpose of the competition, but of course feel free to read the entire tutorial). Please note that the above tutorial is only for reference, it is just one of hundreds of tutorials that are there on the internet, please feel free to refer to other tutorials as well.

For encoder/decoder you could use vanilla RNN but in practice these do not work well. More advanced RNN architectures like LSTM and GRUs work well in practice. For encoder decoder part any type of specialized RNN i.e., LSTM/GRU or Bi-LSTM/Bi-GRU can be used. You are free to use more advanced architectures like LSTM/GRU with Attention mechanism. You can also try out CNN based architectures. In fact, if you want your model to perform better than others, you would have to try out some advanced techniques. Same applies for decoding as well, you could try greedy decoding or beam search or any other decoding technique. I would recommend you read the above article and research papers on NMT to learn more about techniques that are out there. It is very likely that you would have to develop your own model architecture to get a superior performing model.

b. Transformers

NOTE: This is a dynamic document, it will keep getting updated as and when required

In recent times, Transformers based architectures [7] have shown SOTA performances on the majority of NLP tasks. Why not try it for the competition as well? In this case, both encoder and decoder are going to be a transformer-based architecture. You can select any of the transformer architecture of your choice or develop an enhanced version of original transformer [7]. But do remember transformers are resource hungry architectures!
I will also be covering Transformer models later in the course. For the head start, follow the tutorial 3 in section 2. Also, checkout these tutorials: https://mccormickml.com/tutorials/ . For the implementation, check out HuggingFace library (https://huggingface.co/transformers/) and PyTorch tutorial (https://pytorch.org/tutorials/beginner/transformer_tutorial.html).

c. What approach should I take?

There is no straight forward answer to it. You would have to play with different models to arrive at a final model. You could stick to seq2seq or could also try a hybrid approach where the encoder is a transformer and the decoder is a RNN-based model. This competition is about playing and experimentation with lots of models. Moreover, standard off-the-shelf models may not give good performance, you might have to tweak the existing seq2seq/transformer architecture to come up with a new architecture all together. While doing all this you need to keep in mind the resource constraints from google colab.

d. SOTA (State Of The Art) on Hindi-English Translation

To the best of our knowledge, the SOTA performance on Hindi-English NMT (https://www.cfilt.iitb.ac.in/~parallelcorp/iitb_en_hi_parallel/lrec2018_iitbparallel.pdf) is: BLEU score of 12.83 and METEOR score of 0.219. We provide this mainly to give you a general idea about what is typical range of metrics for Hi-En NMT. Also note that the above SOTA is on a dataset different from the one provided in the competition.

5. Implementation and Code

All model implementations will be done in PyTorch. In case you are new to PyTorch, check out the following tutorials: https://pytorch.org/tutorials/ . Note in your implementation you should use only the PyTorch library and not any other higher level NMT library built on top of PyTorch. You will be using Google Colab (https://colab.research.google.com/) for running models on GPUs. You are allowed to use the HuggingFace library for transformers.

Code needs to be very well documented. Document all the steps, model details, hyperparameters, etc. Please cite all the important references you have used during model development. You can refer to external tutorials and papers but you need to

---

NOTE: This is a dynamic document, it will keep getting updated as and when required

cite them properly in the code documentation. **But it should not happen that you simply copy the code from somewhere. If we found that you have copied, please expect an F grade for the entire course.**

6. CodaLab Leaderboard

Every week you would be evaluating your model on a separate validation set on the CodaLab (https://codalab.org/). You would get a maximum of 5 tries for the whole week and best of the tries would be taken as the final model for that week. Your model will be ranked on the leaderboard. This will give an idea about where your model stands with regard to others in the class. TAs will be contacting you regarding this on a regular basis. To begin with, you need to create an account on CodaLab (https://competitions.codalab.org/accounts/signup/), you are required to keep your username as "FirsLetterofNameInCaps_Rollno", e.g., if your name is Ashutosh Modi and roll number is 123456 then your username is "A_123456". On the leaderboard your username (e.g., A_123456) would appear and we would use this to score your performance. Note that performance on the dev set would not count towards final evaluation, this activity is only for improving your models. So this means no point in trying to cheat on the dev set by including it in your train set. But dev set evaluation is compulsory for all. Dev set will be released here: Dev set

Final evaluation on the test set will be done on CodaLab and we will have a leaderboard on the test set as well.

7. Evaluation

This competition has to be attempted INDIVIDUALLY and NOT in GROUPS.

The model developed by you will be tested using a separate test set. TAs will be carrying out that evaluation. You need to provide your code and scripts to the TAs and they will run those on the test set. During testing, you are required to submit only one final model. Of course, during the development and training you can experiment with as many models as you like.

Competition will be evaluated on the following parameters:

1. Data analysis and pre-processing: We have taken data from an existing corpora but still it may not be perfect. It might have noise and needs to be cleaned. Also for the NMT system, the data needs to be pre-processed to a suitable format. You can also use visualizations to do data analysis.
2. Implementations and Code Documentation: You would be experimenting with dozens of models and finally select one final model for testing. Your code should be very well documented. Since so many models are involved, it would be better to follow standard software project develop- ment practices. For example, data pre-processing code maybe in a different file, each model might have a separate file on its own, common functions might be part of a utility library, there might be one main file that could be used to call different models in different files, etc.

NOTE: This is a dynamic document, it will keep getting updated as and when required

3. Analysis of the experiments and results. Since so many models are involved it would make sense to do a thorough analysis to understand what works and what doesn't and why. For example, you could use attention maps or other visualizations to explain model performance, etc. This will also help you to select the best model.
4. Novelty and Creativity: Since it is an open problem we are looking for out-of-the-box solutions. Let your ingenuity and creativity take the lead. You can be creative about each of the above points mentioned above. We value unconventional and novel model architectures, something that has never been done before. You could also take hints of existing research papers but don't forget to cite your inspirations.
5. Performance of the model on the final test set. Performance will be relative i.e. your model performance will be ranked against others. Models are going to be evaluated using BLEU and METEOR metrics.
6. VIVA: We will conduct a in-depth VIVA about how you developed the models and about the code that you have submitted and the models you have experimented with. Basically the viva is going to be about everything you did in the project.
7. Project Report: In a brief report, outline all your attempts and experiences. We will tell more details about this later.

This competition carries around 40% weightage. So attempt it seriously.

It is needless to say that the course has a VERY STRICT policy about cheating. If we found that you have cheated from another student or you have copied from an external source, you and all those involved would be honored with an F grade. If you think pragmatically, it is better to get less marks in this competition than failing the course. The idea is you learn by doing, it's OK even if you do not have the best performing model. I honor authentic efforts.

8. Timeline

I am giving more than required time for the competition, so that you can explore and learn.

Competition Start Date: 22 March 2021
Weekly Development Set Evaluations: 29 March, 5, 12, 19 April 2021
Final Evaluation on the Test Set: 23 and 24 April 2021
VIVA: 27, 28, 29 April 2021

All deadlines are strict and if you do not submit on time you will get a zero in this competition. TAs will be coordinating with you regarding the evaluation.

For any queries/questions/doubts please contact TAs. We understand some of you are new to the topic, but the idea is to learn while doing this competition. Please

---

NOTE: This is a dynamic document, it will keep getting updated as and when required

contact TAs if you are having trouble understanding some concepts or need help with some tutorials on relevant topics, or you are stuck somewhere. Instead of cheating, it is better to seek help from us.

References:

[1] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” arXiv preprint arXiv:1409.3215, 2014.

[2] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[3] N. Reimers and I. Gurevych, “Making monolingual sentence embeddings multilingual using knowl- edge distillation,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 11 2020.

[4] P. Lison and J. Tiedemann, “Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles,” 2016.

[5] J. Tiedemann, “Finding alternative translations in a large corpus of movie subtitle,” in Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16), pp. 3518– 3522, 2016.

[6] J. Tiedemann, “Parallel data, tools and interfaces in opus,” in Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12) (N. C. C. Chair), K. Choukri, T. De- clerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, eds.), (Istanbul, Turkey), European Language Resources Association (ELRA), may 2012.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” arXiv preprint arXiv:1706.03762, 2017.

NOTE: This is a dynamic document, it will keep getting updated as and when required