Capstone Project: Machine Translation for India

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

Table of Contents:

This is a dynamic document and can get updated, please keep a close watch on it.

1. Your First Job in the Industry

You recently graduated and have been employed in the tech team of a mass-media conglomerate. The company has been primarily publishing articles in English. The company now wants to expand its operations and wants to focus on Indian languages. They want to publish previously published English articles in Indian languages. Moreover, they have also hired a team of content writers who would be writing articles in an Indian language.

The CTO of the company came to know about the wonders of deep learning-based Machine Translation (MT), 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 CS799 course**, the company wants you to develop a Neural Machine Translation (NMT) system that could automatically translate articles/editorials written in English to an Indian Language (Bengali and Hindi) . However, later someone else also approached the CTO and promised to develop a state-of-the-art neural machine translation system. However, the CTO has faith in you and your education; hence, to be fair with all, he has organized a competition to develop an English-Indian MT system.

There are different types of neural models, e.g., RNN-based models, sequence-to-sequence models (with attention), and transformer models. You plan to use these to implement MT systems for translation.

This competition requires you to implement Neural Machine Translation (NMT) models to translate from English to Indian Language. 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., English) 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., Hindi) 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, top-k sampling, speculative decoding, etc (see e.g., https://arxiv.org/pdf/2203.15721 ; https://arxiv.org/pdf/2508.13580 ; https://www.assemblyai.com/blog/decoding-strategies-how-llms-choose-the-next-word). **The only caveat is that due to some copyright issues you CANNOT use a pre-trained model; you need to train the model from scratch. You are allowed to use ONLY pre-trained static word embeddings (word2vec or GloVe).**

IMPORTANT NOTE: This is an individual competition and not a group competition. There will be a leaderboard on which your model will be ranked against others, so it is in your best interest not to share your code and model with friends unless you want to share your grades with them! Also, we would

be manually checking your code during a viva, if we find any two models/code are similar both would be honored with an F grade.

It is compulsory for everyone to take part in the competition (even during the training/dev phase); there should be at least 25 submissions during the competition in the training/dev phase (and minimum 4 submissions per week) for every individual. In case you fail to do so, it will be penalized. There will be a separate test phase.

2. Tasks Definition

Given a sentence in English, automatically translate it to an Indian Language (Bengali, Hindi)

The competition is organized in two phases: Development Phase and Test phase

3. 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. Open in Google CoLab, clone it, to experiment with it: Text Pre-Processing
2. Tutorial on implementing Seq2Seq models in PyTorch: https://github.com/lkulowski/LSTM_encoder_decoder
3. Tutorial on implementing Transformers: https://pytorch.org/tutorials/beginner/torchtext_translation_tutorial.html

4. Dataset

We are providing the train set, dev set here , evaluation script here , a Toy model for translation here. 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 any of the phases.

5. 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 cover seq2seq models in 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 the 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

In recent times, Transformers based architectures [3] have shown SOTA performances on the majority of NLP tasks. These are the magic behind the success of LLMs.

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 the original transformer [3]. But do remember transformers are resource hungry architectures!

I will cover Transformer models in the course. However, to get started, check out 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/torchtext_translation_tutorial.html), https://pytorch.org/hub/pytorch_fairseq_translation/

USE OF PRE-TRAINED LANGUAGE MODELS (e.g., from Hugging Face) IS STRICTLY PROHIBITED. USE OF SCIKIT-LEARN OR ANY OTHER LIBRARY TO OBTAIN PRE-BUILT MODELS IS PROHIBITED. YOU NEED TO TRAIN MODEL FROM SCRATCH. IF PRE-TRAINED MODELS ARE USED, YOU WILL GET ZERO! YOU ARE ALLOWED TO USE NON-CONTEXTUALIZED EMBEDDINGS: word2vec and GloVe. Use of LLM-based (or based on any other model) coding assistants is strictly forbidden! Use of LLMs is strictly forbidden! USE OF SCIKIT Library is prohibited. You are only allowed to use PyTorch functions for implementing and training various model architectures. You can train your own contextualized embedding model from scratch and use those embeddings.

c. Linguistic Information:

There are similarities between Indian languages, this could be useful for, or you can also include some Indian language-specific linguistic information in your models, check this paper for more details: https://arxiv.org/ftp/arxiv/papers/2003/2003.08925.pdf

d. 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.

6. Evaluation

We will be evaluating your MT system using chrF++, BLEU and ROUGE metrics. You can check out details in the references. Use the evaluation script (in Codabench) provided by us.

7. 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/) or Kaggle Notebook for running models on GPUs. You are NOT allowed to use the HuggingFace library for transformers. You will be zipping your google colab notebooks and sharing with us later for evaluation. Your colab notebook should have all the model codes including train and testing code. Also include a link to saved model parameters, which could be downloaded and used to initialize the model, so that the model can be tested during viva.

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 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.**

8. Codabench Leaderboard

You would be evaluating your model on a separate validation set on the Codabench (https://www.codabench.org/). **You will have to do a minimum of 25 submissions (and minimum 4 submissions per week) during the development phase of the competition and a minimum of 5 submissions during the test phase of the competition. A minimum of four submissions per week is required for everyone after 6th October 2025 during the development phase.** During the **development phase**, you can iteratively train and refine your model. After this, there will be a **test phase**, where you will evaluate your model on a separate test set. All evaluations will take place on **Codabench**, and your last final submission there will be considered as your official model. 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 Codabench (https://www.codabench.org/accounts/signup), you are required to keep your username as "FirstLetterofName_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.

Codabench Competition Link:

https://www.codabench.org/competitions/10523/?secret_key=f8fee6c8-e3c9-4961-b5de-fb557c69fb70

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

NOTE: The time mentioned on Codabench is in UTC, you will have to convert the timezone to IST (UTC+05:30 hours) at your end.

9. Grading Criteria

This capstone project 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. Submissions during Training stages: **We will monitor if you are submitting overall a minimum of 25 required submissions and minimum of 4 submissions per week after 6th October 2025.** If you submit more, then it will be an advantage for you. But it should not happen that you submit the same thing again and again, this will become evident from the results. This is for us to know that you are actually experimenting with various models.
3. 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 development practices. For example, data preprocessing code may be 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.
4. 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.
5. 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 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.

6. Performance of the model on the final test set. You will have to do a minimum of 5 submissions during the test phase of the competition. Performance will be relative i.e. your model performance will be ranked against others. Models are going to be evaluated using charF++, BLEU and ROUGE metrics. We will be evaluating the performance of the test set. NOTE that performance on the leaderboard is not the only criterion for grading!
7. VIVA: We will conduct an 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.
8. Project Report: In a brief report, outline all your attempts and experiences. We will tell more details about this later.
9. Minimum Submissions: **You are required to do a minimum of 25 submissions (and minimum 4 submissions per week) during the development/Training phase of the competition and a minimum of 5 submissions during the test phase of the competition.** A penalty will be imposed if you do not satisfy the minimum submission requirements.

This competition carries considerable 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.

10. Report and Code Submission

You are required to submit a report about your solution(s). The report has to be written in latex and compiled into pdf. The details of the report are given in this template: ReportTemplate

In addition to the report you also need to submit code for your solution. Since you have been training on Google CoLab or Kaggle Notebook, you can download it as a Jupyter notebook, zip it and submit it.

11. Timeline

Development Phase Start Date: 6th October 2025 to 3rd November 2025 11:59 PM IST

Evaluation on the Test Set: 4th November 2025 to 6th November 2025 (Codabench closes on 6th November 2025, 11:59 PM IST)

Report and Code Submission: 9th November 2025, 11:59 AM.

Viva: 10th November 2025 to 15th November 2025 (Tentative)

Report and Code will be submitted via google form (Note you can submit only once): https://forms.gle/6qjWyQVDNsptmi4y5

Naming convention to be used for the submitted files: "CS799-CP-[yourfirstname]-[yourlastname]-[rollno]-report.pdf" and "CS799-CP-[yourfirstname]-[yourlastname]-[rollno]-code.zip"

NOTE: The time mentioned on Codabench is in UTC, you will have to convert the timezone to IST (UTC+05:30 hours) at your end.

All deadlines are strict and forms/competitions will automatically close immediately after the deadlines. Do not email me any of your solutions, this will go into spam. TAs will be coordinating with you regarding the evaluation.

12. Lost? Whom to contact?

We understand some of you are new to the topic, but the idea is to learn while doing this competition. We also understand that the MT is a difficult task but don't lose hope, seek help. We are here to help you out and conquer MT! If you are able to develop an MT system you can boast about it in your CV and this will give a boost to your career!

For any queries/questions/doubts please contact TAs. Please 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. TAs are available on Discord and these are the contact emails:

Name	Email
Divyaksh Shukla	divyaksh@cse.iitk.ac.in
Sanjeet Singh	sanjeet@cse.iitk.ac.in

13. **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] 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.