Team 12 EnElPi: Report

Domain	Description
Machine Translation	ACL 2019 shared task - Machine Translation of News Articles

Mentor	Team Members
Saumitra Yadav	Gaurang Tandon and Yoogottam Khandelwal

Link to model weights **GitHub repository link**

Introduction

Our task is to implement a **neural translation model** that takes one German sentence as input and outputs one English sentence as its output (i.e., the corresponding translation) We are using the **EuroParl dataset**, which is a set of hand-translated sentences spoken in the discussions of the European Parliament. EuroParl dataset is provided as part of the **ACL 2019 shared task**. We are optimizing for the SacreBLEU scores on the target sentences, as it is used by all the participants of the shared task.

In this report, we will describe what we have understood from the SOTA approaches, the recurrent and the transformer-based ML approaches we have implemented, and the results from each of these approaches.

Dataset used

We looked through all the datasets that are listed on the WMT 2019 task page. We plan to use **Europarl v9 en-de** as our primary and only dataset. This dataset has 1,838,567 DE-EN sentence pairs (roughly **two million pairs**), which we believe would be sufficient to train our model. The dataset is available at this link.

The dataset consists of the proceedings of the European Parliament from 1996 to 2012. As such, it is expected to be discussing **many topics of relevance** to a modern society. It is not overly technical or specific to a particular topic. Moreover, since it covers a wide variety of discussions, it is difficult for a single model to easily *overfit* this dataset: which is to say that trained models should exhibit decent generalizability to real world sentences.

Here are some relevant sentences from the above dataset (DE-EN):

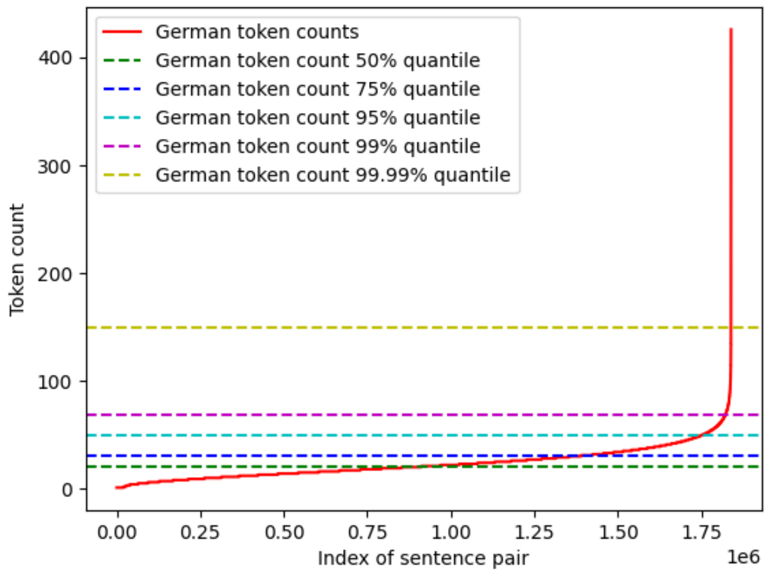
1. "Ich erteile dem Vorsitzenden dieses Ausschusses, Herrn Watson, gleich das Wort, damit er uns im Namen seines Ausschusses den Antrag vorstellt"
   "I shall now give the floor to Mr Watson, Chairman of this committee, so that he can present the request on its behalf."
   (entry number 27)
2. "Man denke auch an die Technik, an die Informationstechnologie und an die einzelnen Kulturen."
   "Consider, in addition, technology, information technology and the different cultures."
   (entry number 463)

Dataset Analysis

Token count in German

We have calculated the number of tokens per sentence in German language sentences of the dataset. Here are the statistics and graphs on this data:

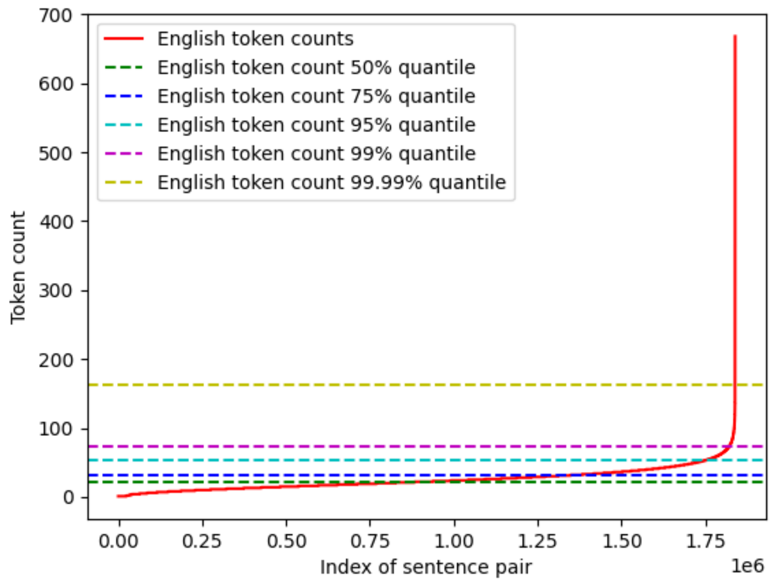
```
mean          23
std           14
min            1
25%           13
50%           21
75%           30
max          426
```



Token count in English

We have calculated the number of tokens per sentence in English language sentences of the dataset. Here are the statistics and graphs on this data:

```
mean            24
std             15
min              1
25%             14
50%             22
75%             32
max            668
```



Token difference

Key takeaway: Most sentences have almost equal number of tokens. 50% of the sentence pairs have lesser than five extra tokens. 95% of the sentence pairs have lesser than thirteen extra tokens.
Key inference: The token difference value has a higher negative magnitude than positive magnitude (the mean is also negative). This shows that, as German has more compound words, German sentences tend to have lesser tokens compared to those in English.

Full statistics:

```
mean             -1
std               4
min            -267
25%              -4
```

```
50%             -1
75%              1
max            211
Token difference 2% quantile -13.0
Token difference 10% quantile -7.0
Token difference 90% quantile 3.0
Token difference 97% quantile 6.0
```

Number frequency

We checked the frequency of top hundred most common numbers present in the dataset. This is what we found:

1. Numbers 1,2,3,4,5... make it to the top of the list without doubt. They are mostly used for referring to items in a sequence or just a general count of items in a situation. They are often used with qualifiers (million/thousand/etc.)
2. Year numbers, especially from 2000 to 2010, are the most common.
3. We did not find any floating-point numbers in the dataset. Most decimals were used to indicate timestamps (17.40hrs) instead of the usual colon-based notation (17:40hrs).

Code for reports

File `./analyzer.py` generates the graphs. The graphs are already provided in directory `./graphs`.

Approach 1: Literature review

We read several SOTA papers, by Microsoft Research, MSRA, and FAIR (links to papers). However, we quickly realized looking at the depth and breadth of these papers that it would be impossible to implement them as part of this course project.

Regardless of whether we would be able to implement these techniques or not, we read the papers because **we found the approaches interesting**. Here we will give a quick glimpse of these papers, which would make the reader aware that this is clearly way beyond a single course project:

1. **Backtranslation:** this is a **dataset augmentation** technique, and it was used by the FAIR team. They used an ensemble of three ML models, to translate target-language sentences back into the source-language. They used a 50-50 split of the human translated bitext and the model-generated back-translated text to train their actual model.
   In their analysis, they found that backtranslation helped gain additional 3 BLEU points.

2. **Document-Level systems:** Microsoft Translator team built deeper models to translate long paragraphs (upto 1000 characters long). They found that these *document-level* translator systems performed better than the sentence-level systems, by around one BLEU point. They attributed this improvement to the **large context** available to the model while encoding or decoding the paragraph.

3. **Sequence-Level Knowledge Distillation:** this is a **model compression** technique. The simplified idea of this technique – which was used by MSRA - is to first train a large model to obtain an accurate *teacher probability distribution,* and then train a smaller student model to mimic the teacher. This distribution is obtained on sentence level.

4. **Noisy Channel Model Re-ranking:** this is a **decoder technique.** In brief, the translator system outputs N hypotheses, on which we apply a Bayesian rule assumption to split the probabilities into three parts, which we then combine. This *re-ranked* score is then used to determine the optimal translation
   This technique was used by the FAIR team to gain 1.4 BLEU points on the validation dataset.

After reading these approaches, we were convinced that these models are beyond our one-month course project.

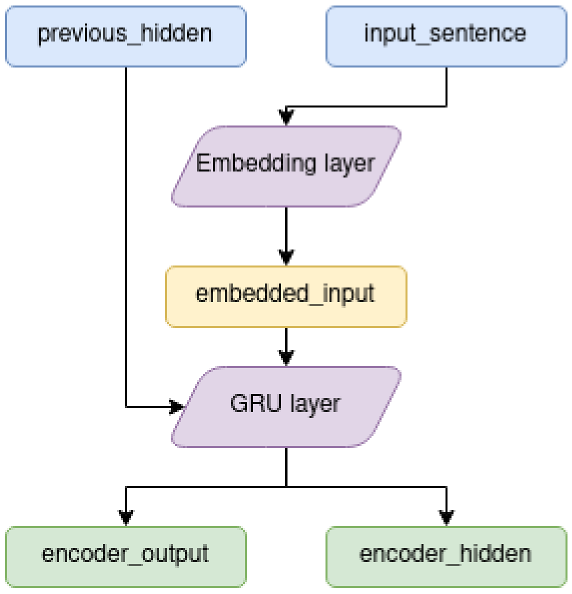
Approach 2: RNN+GRU baseline

For our baseline model, we trained a simple seq2seq RNN with attention using Gated Recurring Units (GRU).

Encoder

We use a two-layer encoder: an Embedding layer followed by a GRU layer. We use a hidden size of 256, as that is most feasible to train in reasonable time.

Importance of GRU: A GRU (gated recurrent unit) layer, as its name suggests, has few gates in each GRU cell to regulate the flow of information. These are reset and update gates, using tanh and sigmoid activations. These help the model maintain **long term** information which is useful for our dataset, as many of our sentences are quite long (22 words).



Encoder forward step flowchart

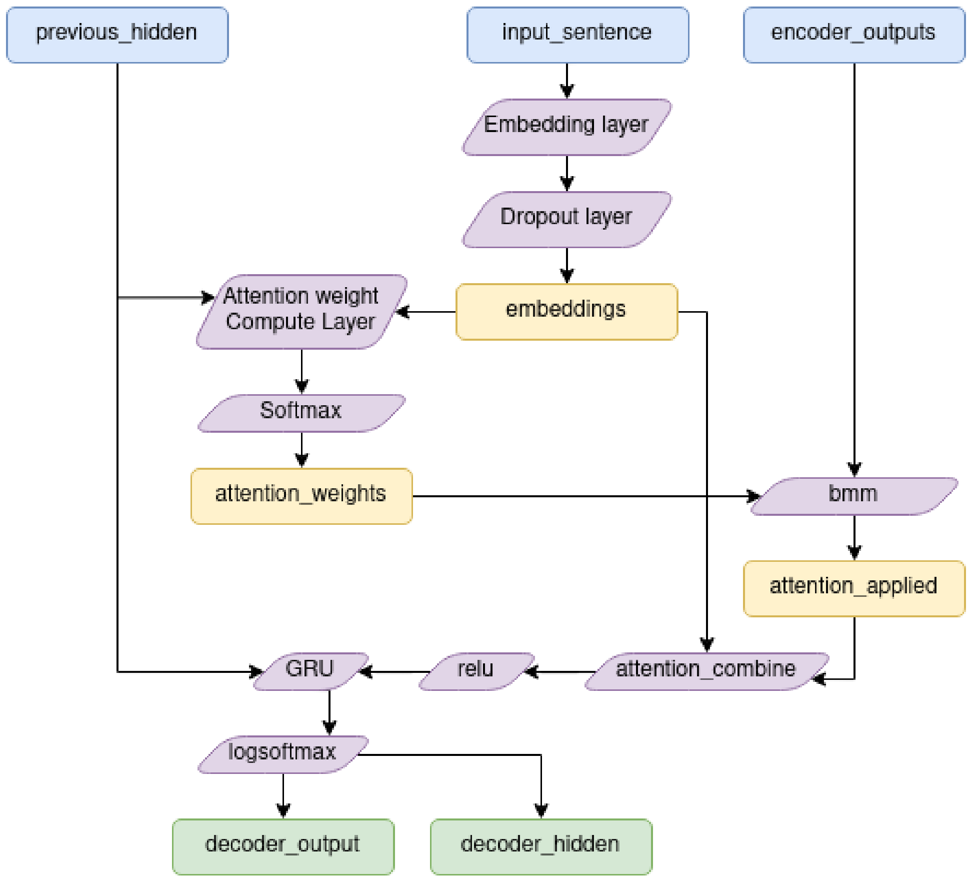
Attention Decoder

We wrote an attention-based decoder that uses an Embedding layer and a GRU layer, sandwiching attention heads between them. We also use Dropout to reduce model overfitting.

Importance of attention-based methods: In a regular decoder, we would only have a context vector passed from the encoder to the decoder. Usually, this is insufficient to capture all the information that we need to produce an accurate translation. Therefore, we need to be able to **dynamically *focus* on different parts of the sentence,** in order to produce the next word in the translation**.** The *focus* is performed by increasing the weights of the corresponding words in the sequence.

Therefore, in our case, we use **two attention layers.** The first layer is of size 2H (H=hidden size) and it generates the attention weights. This larger layer allows us to dynamically shift our focus based on the decoder's hidden state and its inputs. In the next smaller layer of size H, we simply multiply the obtained attention weights with the decoder embeddings to get the Tensor we require.

Note that all throughout we have been using the **standard ReLU** (rectified linear unit) and **log softmax** activation functions wherever applicable.



Decoder forward step flowchart

Seq2Seq pipeline

We wrap both our encoder and decoder in a seq2seq RNN model. Since we are using PyTorch Lightning, we must implement the training_step and the validation_step methods. Each of these steps consists of taking the input batch, moving it through the encoder and the decoder, then using the **negative log likelihood loss** function along with the output batch to compute the loss.

Importance of Adam optimizer: We are using Adam optimizer with a learning rate of 10^{-4}, and default values of the alpha and beta constants. **Adam** optimizer is based on **adaptive moment estimation.** It accounts for the *momentum* from the earlier steps and uses them to scale up the subsequent steps in gradient descent. This theoretically results in faster convergence to an optimal point.

Model training techniques

- We trained in batches of size 32. Larger batch sizes were giving us OOM.
- Since we are training in batches, throughout the code we have ensured to use proper torch Tensor **vectorization and broadcasting** techniques to write performant trainer/evaluator code.

Code

The code for seq2seq training is present in `./anlp_project/models/seq2seq.py`

Approach 3: Transformer model

For our baseline++ model, we fine-tuned a T5 model.

What is the T5 model?

T5 is the Text-to-Text Transfer Transformer that was trained on the C4 (Colossal Cleaned Crawled Corpus) - both of which are developed and trained by Google. T5 is a massive transformer-based ML model that takes input as text and outputs as text as well. It receives instructions as text, for example, in our case, we have to give input like so:

“translate German to English: <input sentence>”

It is also based on the structure of an encoder step followed by a decoder setup.

The detailed implementation of the T5 paper is out of scope for our project. However, we understand that it contains several MultiHeaded attention layers and Add&Norm layers.

Why not use BERT?

BERT is trained for masked language modeling whereas T5 is trained for autoregressive/generative language modeling.

Therefore, BERT would be good for classification or fill in the blanks tasks, whereas T5 is good for full NLG tasks like ours.

Model training techniques

- We trained the t5-small model, which is the only one we’re capable of training fast enough
- We trained in batches of size 4 as the T5 model is quite big it gets OOM easily on larger batch sizes

Code

The code for seq2seq training is present in
`./anlp_project/models/transformer.py`

Training model setup

In the following section we describe our model training setup.

Model implementation

We implemented the model in **PyTorch Lightning** – which is a ML research framework that makes it easy to write performant and well-structured ML code in PyTorch. We used the Wandb logger to get great loss visualizations in Weights and Biases, which is a website built to visualize various model parameters during training.

All our model parameters are defined in the file `anlp_project/hparams.yaml.` This serves as a centralized and easy to lookup reference table for all the project parameters. This file is read by our custom config py.

Model training system

We trained our model on the Ada machine that is provided to us by the IIIT Hyderabad institute. The machine has the following hardware specifications:

1. Four GTX 1080 graphics cards
2. 38 cpus
3. Processor clock speed is roughly 3GHz
4. RAM is 75G

While these machines are surely capable, they are nowhere near the capacity that we would require to train SOTA models from scratch. Therefore, in this project, we have stuck to training simpler seq2seq models or fine-tuning existing transformer models.

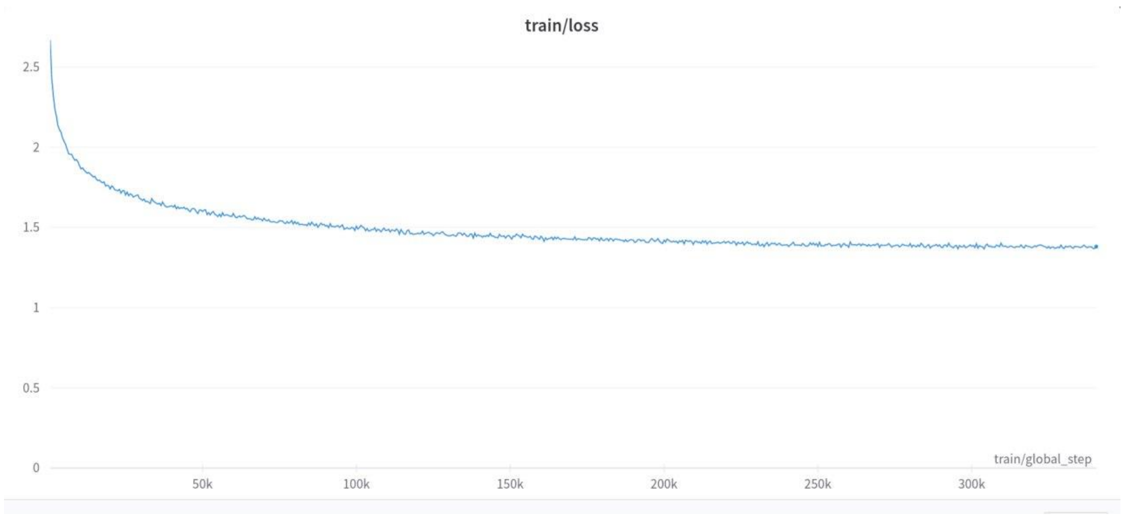
Evaluation of inferences

We intend to use SacreBLEU as the evaluation metric as this has been used by many teams and was developed for this specific type of shared tasks. Specifically, we will use `` `BLEU+case.mixed+lang.en-de+numrefs.1+smooth.exp+test.wmt18+tok.13a+version.1.3` `` as has been used by Microsoft's submission and is given in this GitHub repo.

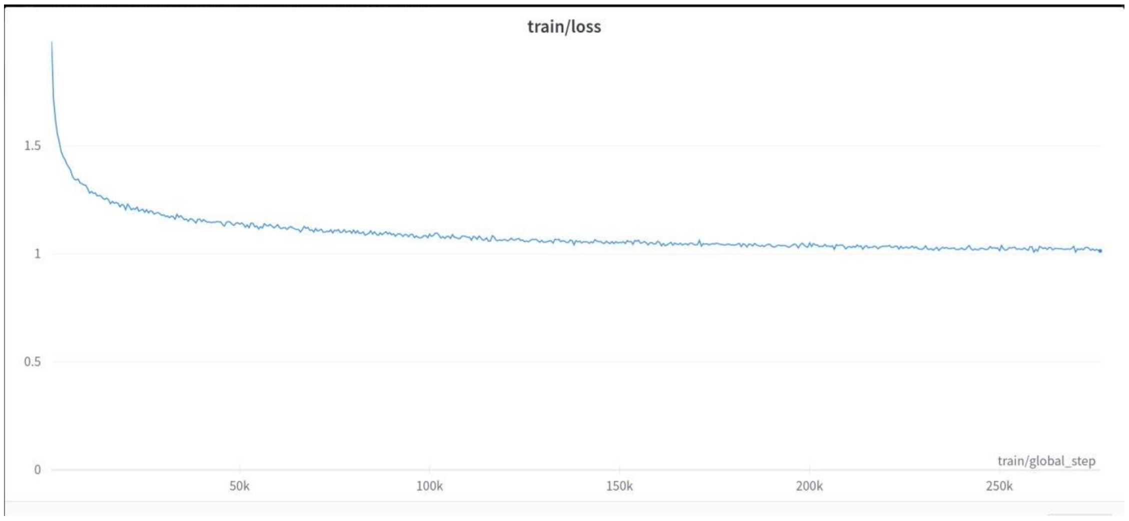
Wandb graphs

The following are our training loss graphs for the transformer model:

German to English



English to German



Model inferences

Code to run evaluation and get scores

Code is provided at `anlp_project/evaluate_outputs.py.` You may run it as: `python evaluate_outupts.py checkpoint_path` where checkpoint can be obtained from our google drive link.

We have already run the evaluations and all inference files are present at `documents/inference_en-de.txt` and documents/inference_en-de.txt Note the first sentence is our output and the second sentence is the reference target.

SacreBLEU scores

Our SacreBLEU score is:

- DE to EN pipeline: **BLEU** = **28.07** 62.5/36.9/24.4/16.7 (BP = 0.902 ratio = 0.906 hyp_len = 15203 ref_len = 16776
- EN to DE pipeline: **BLEU** = **25.30** 59.0/35.1/24.2/17.4 (BP = 0.828 ratio = 0.841 hyp_len = 10360 ref_len = 12312)

Here:

- The numbers in four slashes indicate 1-4 ngram precision
- BP is brevity penalty
- ratio indicates the ratio between hypothesis and reference lengths.
- A ratio close to 1 indicates healthy translations.
- We note that in both cases are outputs are slightly smaller in length than the reference texts.

Manual analysis

We manually picked 100 English output sentences and 50 German sentences and evaluated them for their grammaticality and fluency. To evaluate the German sentences, we had to first plug them through Google Translate because we do not speak German.

We noticed that most of the sentences are of good quality. We attribute this to two main reasons:

1. Quality of dataset: we noticed that the dataset is translated by professional translators which handle facts.
2. Pre-trained language models: T5 handles grammaticality of the language tokens very well.
3. High resource languages: both German and English are high resource languages.

DE to EN - Error Analysis

Number transfer errors

Example 1

Our output: i have received a motion for a resolution on the basis of article 42.
Reference output: i have received a motion for a resolution tabled pursuant to article 425.
As we can see, **our model** will occasionally mess up the numbers. However, in many other cases, **our model captures numbers nicely.** Here are some examples:

Example 2

Our output: for example, about 11 million bangladesh people are illegally staying in india.
Reference output: there are some eleven million bangladeshi living illegally in india.
Here our model correctly handled the numeric translation of the word *eleven.*

Example 3

Our output: the vote will take place today at 12 noon.
Reference output: the vote will take place at 12 noon tomorrow.
Here our model handled the hour value correctly, but omitted the date value of *tomorrow.*

Factuality errors

Large-scale fine-tuned models are known to be susceptible to **hallucinations** and creating extra facts that were not present in the input sentence (**false fact generation**). In other cases, certain facts are **completely omitted.** In this section we check that our model suffered from that.

Example 1

Our output: use of uranium-containing ammunition in bosnia and koso
Reference output: use of depleted uranium in bosnia and kosovo balkan syndrome

Here we notice that the words “balkan syndrome” are **omitted** in our output.

Example 2

Our output: my colleague, mr van den bos, has already mentioned this in his speech.
Reference output: mr van den bos has already mentioned them in his speech.

Here we notice our model is **hallucinating** about the fact that Mr. Bos is *his colleague.*

Example 3

Our output: these are shortcomings that should be re-examined.
Reference output: these are shortcomings that must be reviewed as a matter of priority.

Here we notice our model is **omitted** that the review must be done at *priority*.

EN to DE - Error Analysis

Factuality errors

In this case, we see that the pronoun guessed by our model was incorrect (herr => male, frau => female). We believe this is because the dataset had more data points for male parliament speakers compared to female ones.

Our output: vielen dank, herr kommissar.
Reference output: vielen dank, frau kommissarin!