Distilling the Knowledge of BERT For Text Generation

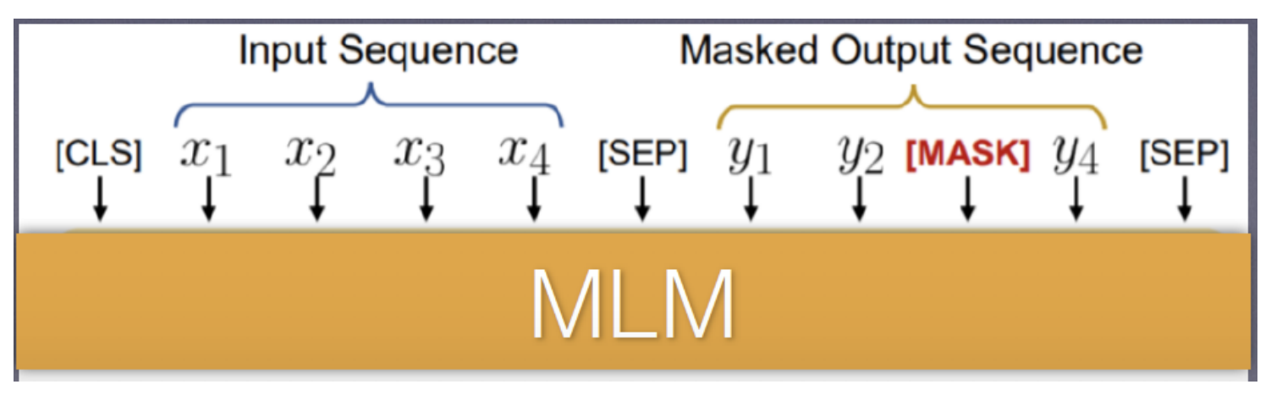
Ouarda FENEK - Sara Yasmine OUERK
Sorbonne University - DAC - 2019/2020

Introduction

Large-scale pre-trained language models, such as BERT, has recently achieved great success in a wide range of language understanding tasks. However, it remains an open question how to use BERT for text generation tasks. One may propose to initialize the parameters of a Seq2Seq model with pre-trained BERT, then, fine-tuning on the target dataset. However, this requires the model to have the same size as BERT, inevitably making it too large.
In the proposed approach we do not directly use the parameters of BERT in the Seq2Seq model. Instead, Bert acts as an effective regularization to the seq2seq training loss, by proactively injecting future information for predecting the present.
We distill the knowledge learned in BERT (the teacher) for text generation tasks (the student model).

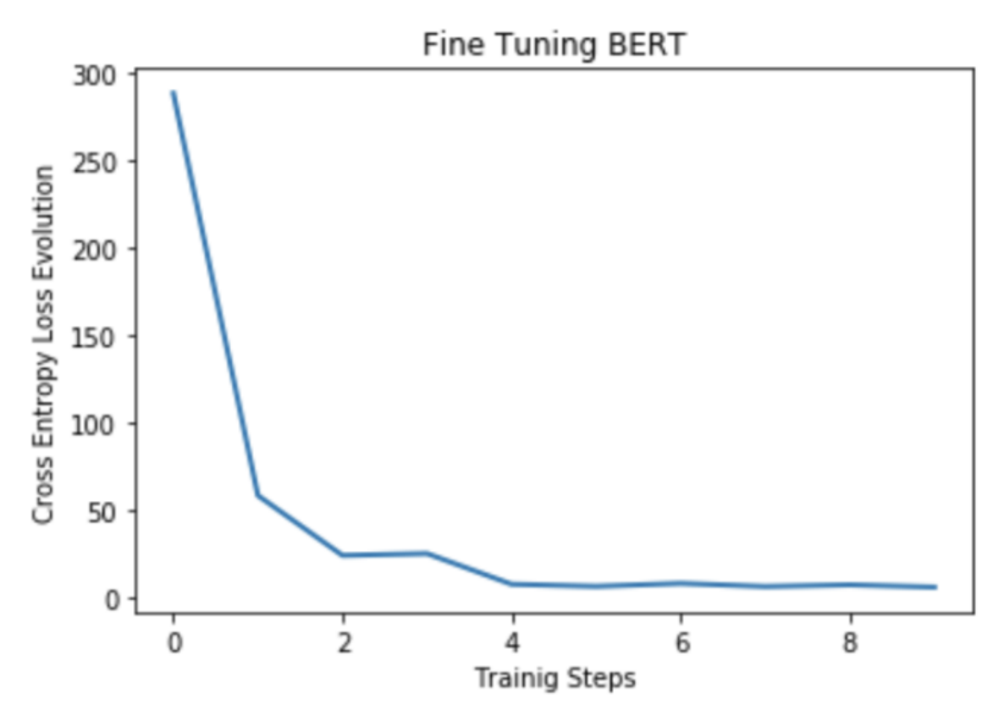
1 - Fine-Tuning BERT for the Task

- Having the training set $(x_i, y_i) \in (X, Y)$ where X are the original texts, and Y their summaries/translations.
- Use the pre-trained Masked LM strategy of BERT in which we mask 15% of tokens from y_i.
- Form inputs that corresponds to BERT requirements: $['CLS'] + x_i + ['SEP'] + y_{imasked} + ['SEP']$. It is interesting to keep the X because it is a crucial part of the context.



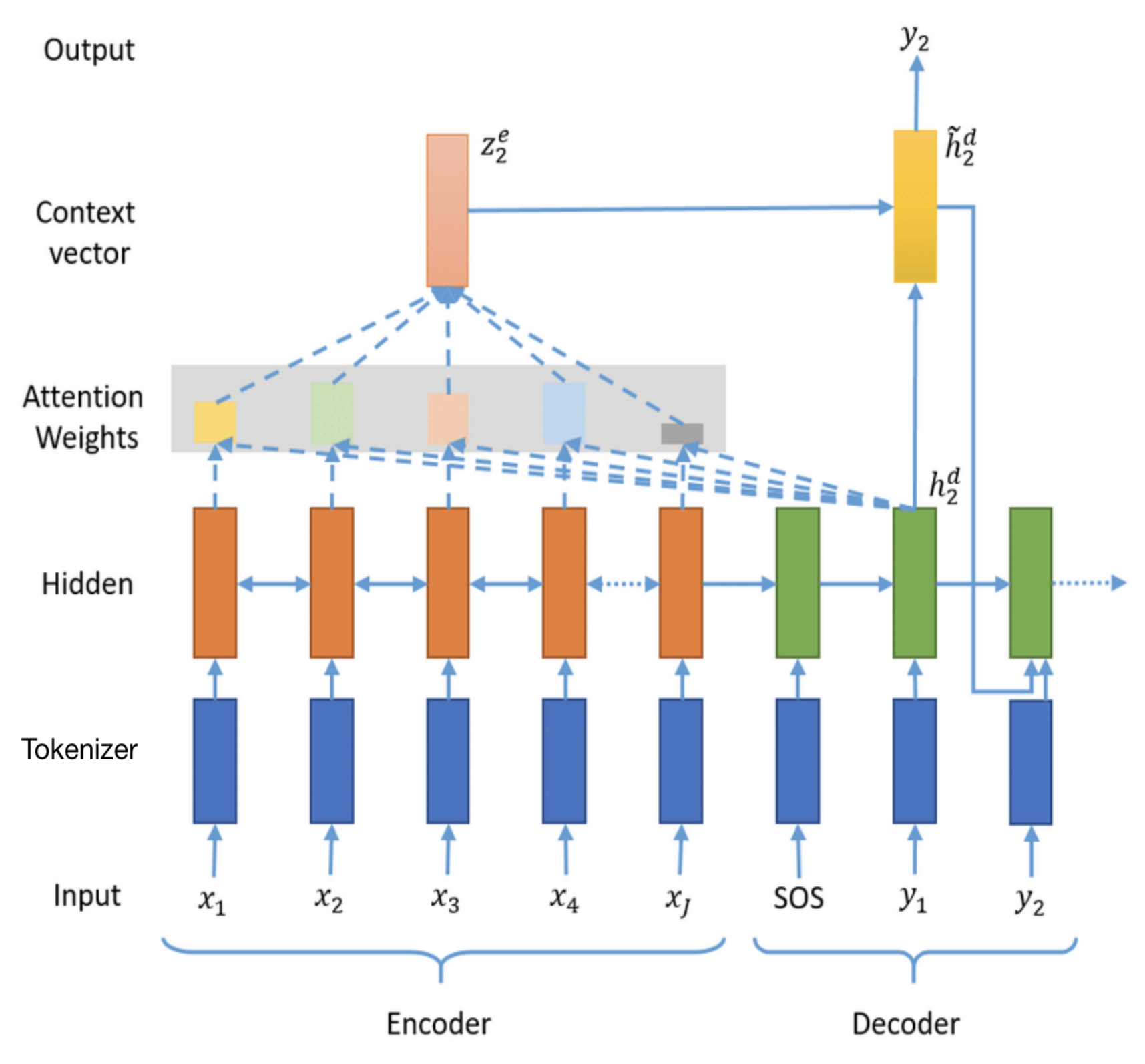
- Use the BERT tokenizer to get a numerical presentation of each token.
- Lunch a training loop on the model to learn parameters ϕ. The loss used is the Cross Entropy Loss.

Results:



2 - Building the Seq2Seq Model

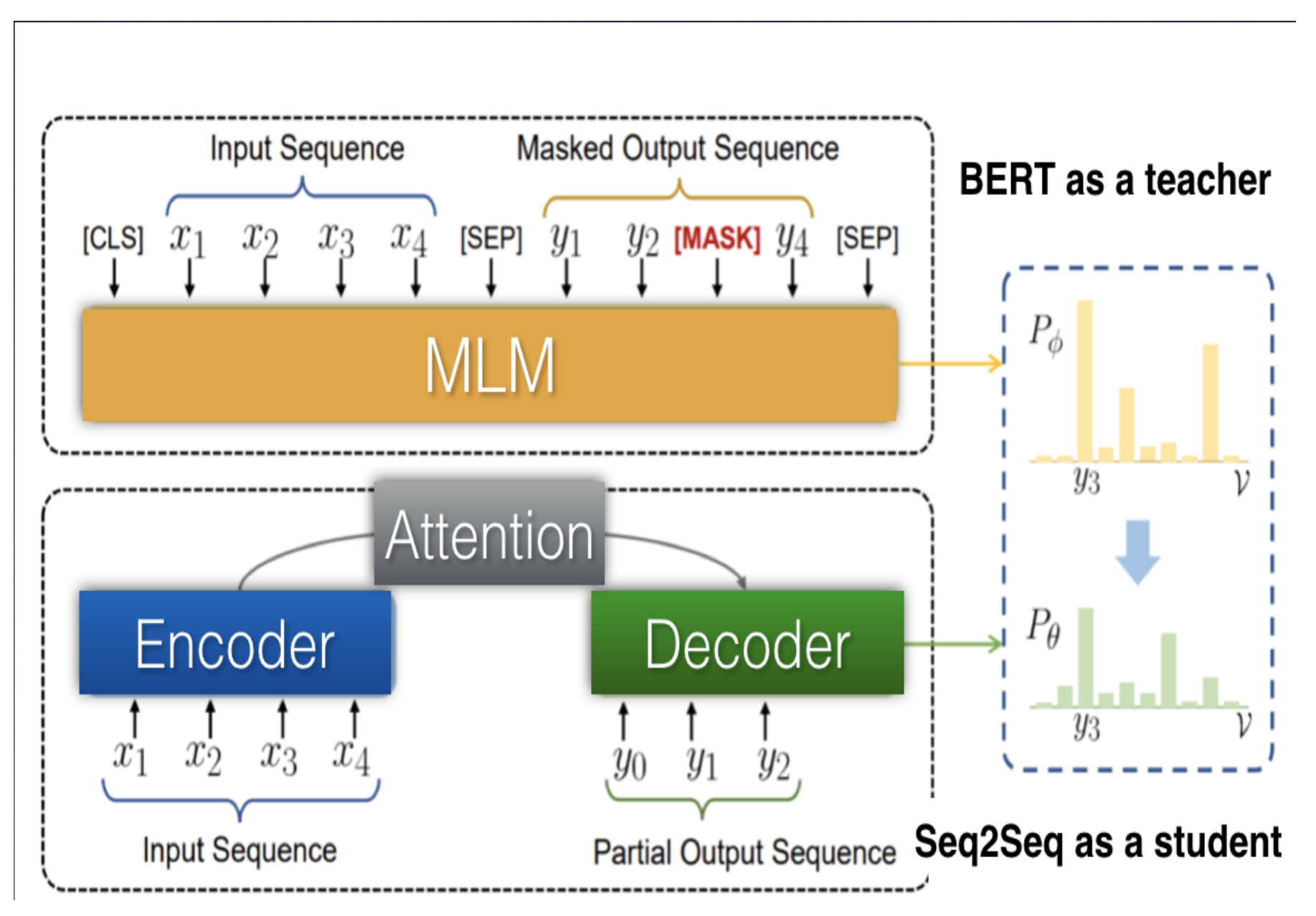
- Having the training set $(x_i, y_i) \in (X, Y)$ where X are the original texts, and Y their summaries/translations.
- Build the Encoder and Attention Decoder that will form together the Seq2Seq scheme. Prepare it to learn parameters θ.



- Don't forget to use the BERT tokenizer to have the same representation of the data.
- Define the loss to optimize as:

$$L_{xe}(\theta) = -\sum_{t=1}^{N} logP_\theta(y_t = w|y_{1:t-1}, x)$$

3 - Knowledge Distilliation



- Make a training loop for the Seq2Seq Model.
- At each step calculate the CE loss defined above $L_{xe}(y, y_{pred_{seq2seq}})$.
- Mask 15% of the current y, concatenate with x to have the adequate BERT format and call BERT predict to get $P_\phi(y_t = w|y_{unmasked}, x)$.
- Calculate the Cross Entropy Loss between the Seq2Seq predictions and BERT predictions, taking BERT's as the ground truth:

$$L_{bidi} = -\sum_w [P_\phi(y_t = w|y_{unmasked}, x).logP_\theta(y_t = w|y_{1:t-1}, x)]$$

- Calculate a general loss:

$$L_{total} = \alpha L_{xe} + (1-\alpha)L_{bidi}$$

- Compute the gradient on L_{total} with respect to θ.

Datasets

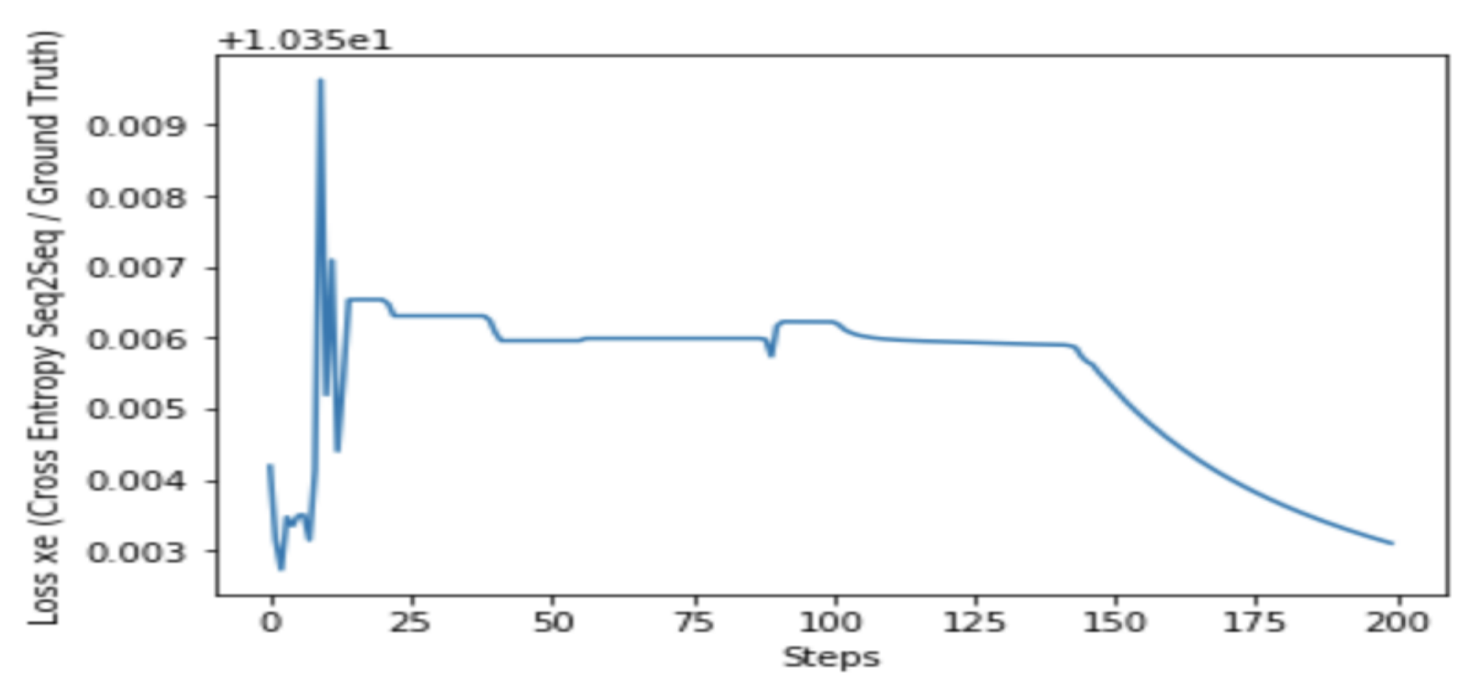
- **Giga Word for Summarization:**
  This is a comprehensive archive of newswire text data in English that has been acquired over several years by Linguistic Data Consortium. The texts of it that have summaries are splited to train, validation and test pairs, respectively 3.8M/190k/2k.
- **English-Vietnamese and German-English for Translation:**
  Many versions were used, En-Vi, 113k training samples, IWSLT14 De-En 60k training samples, WMT14 En-GDe 4.5M training samples. These samples were devided to train, validation and test pairs.

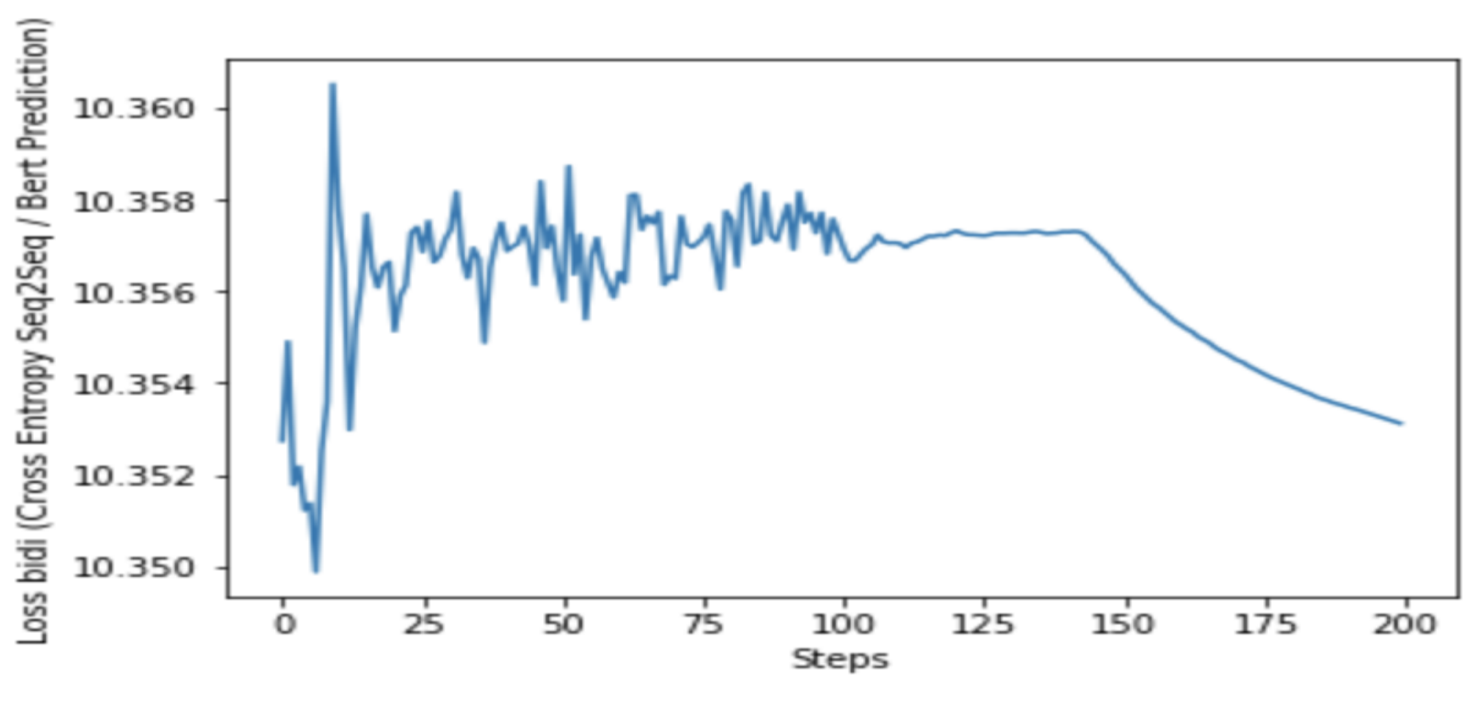
Metrics

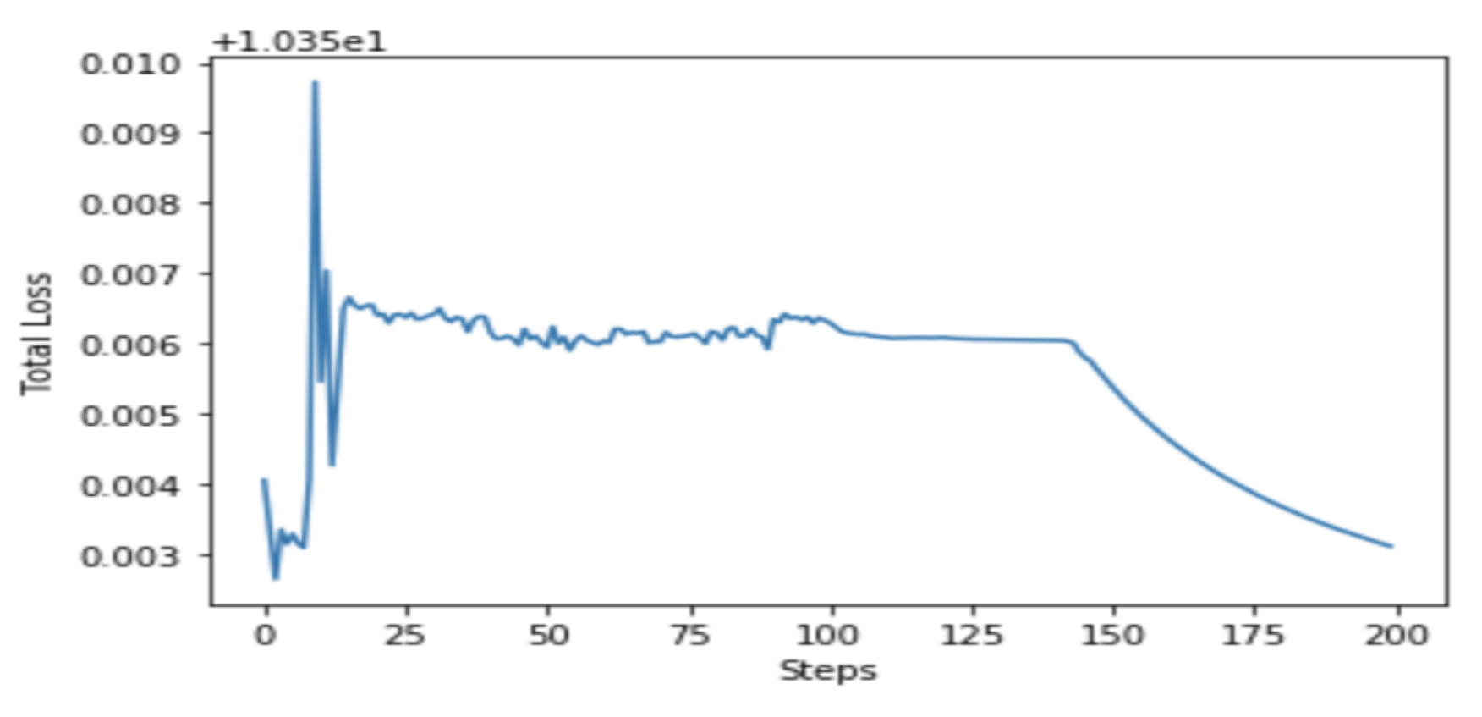
- **ROUGE:** Measures recall, how much the words (and/or n-grams) in the human reference summaries appeared in the machine generated summaries.
  - **ROUGE-N:** Overlap of N-grams between the system and reference summaries.
  - **ROUGE-L:** Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.

Training

We have trained our model for 200 *epochs* with a *learning_rate* = 2 and α = 0.1 on the GigaWord dataset.







The mean test loss was 10.346635971069336 witch corresponds to the final trainig loss. This means that the model is fitting well (there is no over-fitting).

Results

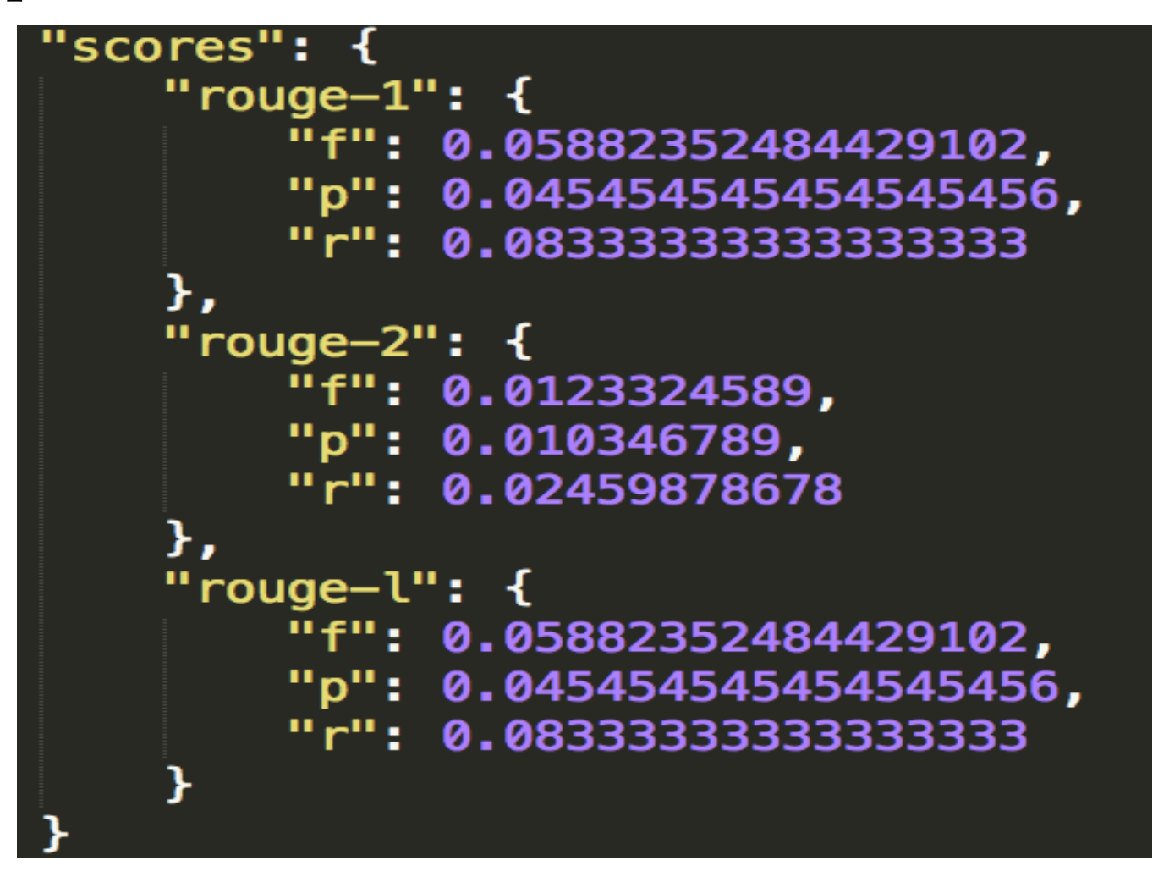
The results of reference models were:

GW Models	R-1	R-2	R-L
Seq2Seq[†]	36.40	17.77	33.71
CGU[‡]	36.3	18.0	33.8
FTSum$_g$[*]	37.27	17.65	34.24
E2T$_{cnn}$[◇]	37.04	16.66	**34.93**
Re^{3}Sum[•]	37.04	**19.03**	34.46
Trm + BERT teacher	**37.57**	18.59	34.82

It has significantly improved state of the art results.

GW Models	R-1	R-2	R-L
Dev			
Transformer (base)	46.64	24.37	43.17
+ BERT teacher	**47.35**	**25.11**	**44.04**
Test-Dev			
Transformer (base)	46.84	24.80	43.58
+ BERT teacher	**47.90**	**25.75**	**44.53**

Our implemented model didn't train enough due to lack of ressources, thus we couldn't report the same results. An example of one sentence:


```
"scores": {
    "rouge-1": {
        "f": 0.05882352484429102,
        "p": 0.045454545454545456,
        "r": 0.08333333333333333
    },
    "rouge-2": {
        "f": 0.0123324589,
        "p": 0.010346789,
        "r": 0.02459878678
    },
    "rouge-l": {
        "f": 0.05882352484429102,
        "p": 0.045454545454545456,
        "r": 0.08333333333333333
    }
}
```