Contributing to the Neural Machine Translation Project at CDLI

Rachit Bansal
Delhi Technological University
`rachitbansal2500@gmail.com`
`https://github.com/RachitBansal`

Abstract

With the advent of self-attention based Transformer architecture, sequence-to-sequence Translation has reached new heights. Moreover, subsequent techniques for Unsupervised and Semi-Supervised Neural Machine Translation have reduced the required amounts of parallel data considerably and in some cases, close to none. For Low Resource Languages such as Sumerian, the performance of NMT systems can be improved a whole lot by introducing monolingual corpus based model training and data augmentation.

In this proposal, I describe the models and algorithms that can be used to make this happen, along with a thorough description of how I will go about implementing it during Google Summer of Code, 2020, under the Cuneiform Digital Library Initiative (CDLI).

1 About Me

I am currently pursuing my Bachelors in Technology (B.Tech) from Delhi Technological University, India. I have been following Deep Learning research for a long time now, primarily focusing on Natural Language Processing. I am an avid practitioner of Mathematics and hold an intense zeal of studying Deep Learning models, Optimization Techniques, and Algorithms from the core.

I am especially interested in contributing to this project being undertaken at CDLI because of how well it aligns with my prior knowledge in the domain of Unsupervised NMT, as well as my willingness to learn more. I am well acquainted with implementing Deep Learning models from scratch and have elaborately worked with PyTorch, Tensorflow, Keras, Numpy, and other Python libraries. I was a mentor for Tensorflow during Google Code-In 2019 and thus am well versed with Open-Source.

My most recent research work titled 'Back2Back Translation', which introduced a unified encoder approach along with progressive transitions in transformers and differential sampling, complemented with a new loss for training an end-to-end Unsupervised and Semi-Supervised Neural Machine Translation system, is currently under review at ACL 2020 SRW[1]. A part of the abstract of the research paper is as below:

'...we aim to stretch the limits of semi-supervised NMT by introducing an end-to-end continuous training mechanism for monolingual data by employing a differentiable sampling to increase the correlation between the weights of primal and dual models. Effectively, we constraint the training of the two models to be inverse mappings and introduce the losses in such a way that the two models share more and more information over the training phase.'

[1] `drive/Back2Back_ACL.pdf (draft)`

2 Introduction

Previous models that have been used to carry out English-Sumerian Translation have only made use of the available parallel corpora. Presently[2], we have only about 50K extracted sentences for both languages in the parallel corpora, whereas around 1.47M sentences in the Sumerian monolingual corpus. This huge amount of monolingual data can be used to improve the NMT system by combining it with techniques like Back Translation (Sennrich et al. 2015a) and Dual Learning (He et al. 2016) which have proved specially useful for Low-Resource languages like Sumerian which have a limited amount of parallel data (Karakanta et al. 2018). Studies on Back Translation (Sennrich et al. 2015a, Poncelas et al. 2018) have shown that using the synthetic sentences on the source side proves more effective and thus to accomplish English $\rightarrow$ Sumerian translation, the Sumerian monolingual data can be used extensively. The limitless English monolingual data too can be used to assist Sumerian $\rightarrow$ English translation, further the two models can be made to complement each other, thus moving onto an Iterative form of Back Translation or Dual Learning (Xia et al. 2018).

Recently methods have been proposed to accomplish completely Unsupervised Neural Machine Translation (Lample et al. 2018), they rely on Language Modelling and then combining the hidden states of the two LMs. Though they are still very limited to similar language pairs, it would be interesting to experiment with it for our task.

Before moving onto using monolingual data, it would be wise to create a solid baseline by using the self-attention based Transformer architecture (Vaswani et al. 2017) purely on the parallel data, first. Moreover, additional techniques like Transfer Learning (Zoph et al. 2016) wherein weights from a High Resource Language (HRL) Pair are fine-tuned for the Low Resource Language (LRL) Translation have shown considerable improvement in performance and can be employed along with a model like XLM (Lample and Conneau 2019), which hasn't been tested on such a framework before.

3 Model Level Details

3.1 Transformer

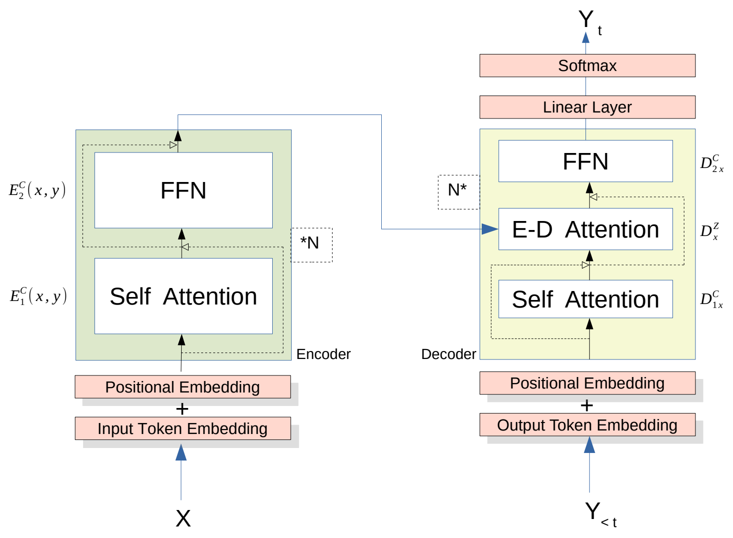


Figure 1: The Transformer Encoder-Decoder Architecture

As we would be using the Transformer (Vaswani et al. 2017) Encoder-Decoder architecture for most of our experiments, it is fair to briefly discuss how it works and its results in NMT.

[2] cdli-gh/Unsupervised-NMT-for-Sumerian-English

The Transformer architecture is wholly based upon attention, specifically self-attention. Talking in context with NMT, an embedded (concatenated positional and token embedding) source sentence would go into the Encoder, consisting of N (=6, by default) vertically stacked Encoder Blocks. In each Encoder Block, it first passes the multi-head self-attention layer, which modifies the n-dimension embedding of each word (n mostly being equal to 2048) with respect to the sentence context. This vector then moves onto a feed-forward layer which transforms it to d-model dimensions (usually being 512), this output then passes onto the next Encoder Block. Each sub-layer is followed by an Add and Norm function, as shown in Fig 1. The final vector from the last encoder block represents a high-level latent representation of the source sentence, which acts as an additional Encoder-Decoder self-attention layer in the Decoder.

The Decoder, which also consists of N vertically stacked Decoder Blocks, takes in input the embedding of $t-1$ tokens preceding the current t^{th} token. This embedding passes through all the layers as in the Encoder, in addition to the Encoder-Decoder (E-D) attention layer. The output of the Decoder, which is a vector of size d-model, passes through an external feed-forward layer of size equal to the vocabulary size, softmax over which gives the probability distribution over all the words in the target vocabulary. Sampling over this distribution gives the word at the t^{th} position of the output sentence.

From Table 1, it is evident that NMT techniques including Dual Learning and Back Translation perform exceptionally better when implemented using Transformers.

	Without Transformer	With Transformer
Convolution with Attention	25.2	NA
Vanilla Transformer	NA	27.74
XLM	NA	28.0
Dual Learning	25.4	28.9
Backtranslation	26.6	35.0

Table 1: Comparison of BLEU Scores of NMT Models on cross English-German translation.

XLM (Lample and Conneau 2019) makes some amendments to the original transformer encoder to better condition its working with Cross-Lingual tasks. It makes use of a unified vocabulary for the two languages, uses an extra embedding layer i.e. Language Embedding, and trains the Encoder using an additional Translational Language Modelling (TLM) task besides the conventional Masked Language Modelling (MLM) and Causal Language Modelling (CLM) tasks, which enforces a shared latent space for both languages.

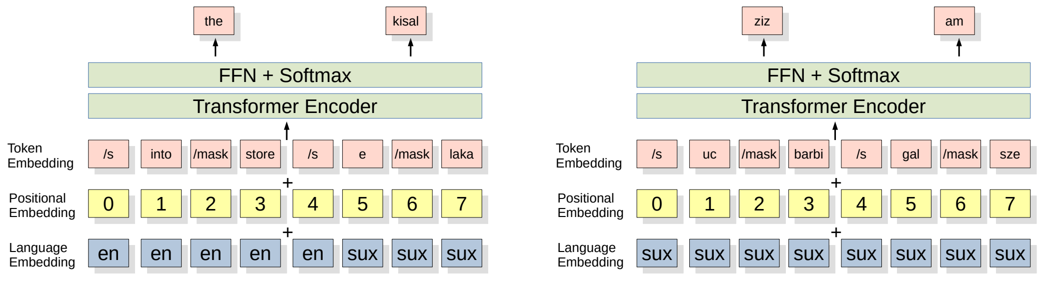


Figure 2: Comparison of Translational Language Modelling (TLM, left) and Masked Language Modelling (MLM, right) tasks for training transformer encoders and PLT embeddings. The XLM introduced TLM to make better cross-lingual latent representations of the textual data.

3.2 Dual Learning

Dual learning is a branch of machine learning that focuses on co-learning models corresponding to primal and dual tasks. Originally inspired by the dual nature of many conversion and transformation tasks for both symmetrical and asymmetrical cases, it has proved to be successful for Neural Machine

Translation tasks by learning both- a forward and a reverse transformation for the translation task at hand.

Model-level Dual Learning allows the usage of monolingual data by passing it through two cascaded models and back propagating the loss computed between the output and the source itself. Xia et al. 2018 uses a pair of weight shared Encoder-Decoder models which were used to execute the $A \leftrightarrow B$ Translations, wherein the Encoder from $A \rightarrow B$ acts as the Decoder for $B \rightarrow A$, i.e., during the reverse traversal, this is done by bridging the gap between an encoder and a decoder by using a zero context vector for the Encoder in place of the E-D Attention vector that needs to be present in the Decoder.

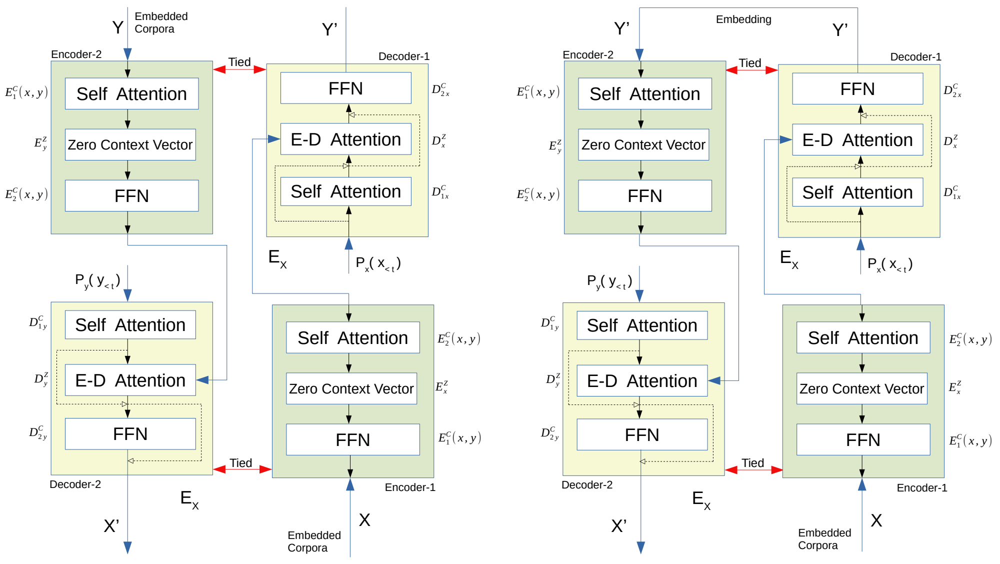


Figure 3: A close look at the working of the Encoder-Decoder model in Dual Learning. The figure on the left represents training on Parallel Data while the one on the right on Monolingual Data. X' and Y' represent the output from the Decoder after sampling. Some notations and symbols inspired by Xia et al. 2018.

The following equations with reference to Figure 2 encapsulate the training procedure on Parallel Data:

$Y = D_{2y}^c(\tilde{y}_{<t} + D_{1y}^Z(\tilde{y}_{<t}, C_y))$
where,
$\tilde{y}_{<t} = D_{1y}^c(P_y(y_{<t}))$
$C_y = E_x$

$X = D_{2x}^c(\tilde{x}_{<t} + D_{1x}^Z(\tilde{x}_{<t}, C_x))$
where,
$\tilde{x}_{<t} = D_{1x}^c(P_x(x_{<t}))$
$C_x = E_x$

Similarly, the following equations encapsulate the training procedure on Monolingual Data:

$Y = D_{2y}^c(\tilde{y}_{<t} + D_{1y}^Z(\tilde{y}_{<t}, C_y))$
where,
$\tilde{y}_{<t} = D_{1y}^c(P_y(y_{<t}))$
$C_y = E_x$

$X = D^c_{2x}(\tilde{x}_{<t} + D^Z_{1x}(\tilde{x}_{<t}, C_x))$
where,
$\tilde{x}_{<t} = D^c_{1x}(P_x(x_{<t}))$
$C_x = P_x(Y)$

Existing implementation of Dual Learning for Machine Translation that could be used for coding references: `yistLin/pytorch-dual-learning`

3.3 Back Translation

Back translation (Sennrich et al. 2015c Edunov et al. 2018), which is the current SOTA in NMT involves data augmentation by making use of a reverse model (Sumerian $\leftarrow$ English, when the task is to translate from Sumerian $\rightarrow$ English) trained on the existing parallel corpora and applying it on the monolingual corpus. The synthetic samples thus generated are added to the parallel corpora and a new reverse model is trained on the augmented dataset.

Saved weights of the transformer based models trained previously on the parallel corpora would be used to back-translate the monolingual data and then the models would be re-trained on this augmented parallel data, this process would be carried iteratively until the entire monolingual corpus is exhausted. Thus, implementation of Back Translation is **not** a complicated procedure once we have trained the transformer based models successfully.

This simple technique has been used extensively in the past for LRLs as well has HRLs and has shown considerable improvement in the performance of NMT systems. Though, large scale studies (Edunov et al. 2018) have shown the heavy dependency of their performance on aspects like Sampling Methods used as well as the amount of parallel data.

In general case scenarios, when the Back Translated samples are sampled using Non-MAP (Non-maximum a-posteriori) techniques like Nucleus Sampling (Holtzman et al. 2019) and beam search with noise (Lample et al. 2017) the performance of the final Data Augmented model is better due to the diversity of synthetic samples than when MAP techniques like Beam Search and Greedy are used.

Sampling Methodology	Description
Greedy	Picking the output with word-wise largest probability by picking the output corresponding to the expectation value of the Gaussian distribution.
Beam Search (OW and MORTON 1988)	Picking the output that maximizes the net probability product until the EOS token is encountered.
top-k sampling (Holtzman et al. 2018)	Restricting the sample size to the top-k probabilities and then re-normalizing to sample from this restricted set.
Nucleus Sampling (Holtzman et al. 2019)	Restricting the sample size until the cumulative probability reaches a threshold and then re-normalizing to sample from the restricted set.
Beam Search with Noise (Lample et al. 2017)	Applying noise to beam search outputs by transforming the source sentences by deleting, masking, and swapping some words with restrictions.

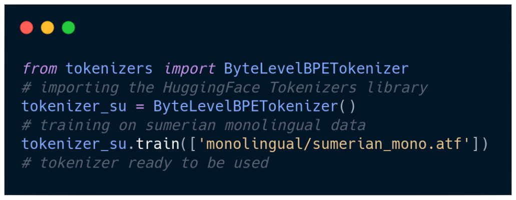
Table 2: Comparison of different sampling methods to generate synthetic source sentences. The first set consists of the MAP methods while the second set consists of non-MAP methods.

Non-MAP methods are not equally effective for Low-Resource settings (~80K sentence pairs), for such settings MAP methods have been reported (Edunov et al. 2018) to give better results. The effectiveness of non-MAP techniques improves almost-linearly with the amount of bitext. Thus, for the English-Sumerian Translation task MAP techniques might prove better. Experiments would be carried out both ways so as to compare the results and have a firm understanding.

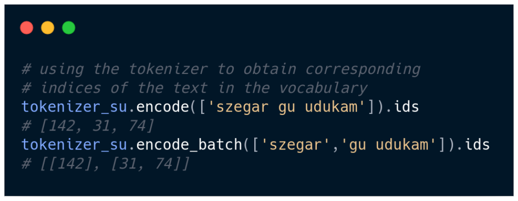
4 Implementation Details and Dependencies

Open-Source Python Libraries that will be used:

- PyTorch- For Deep Learning modules, Optimization methods, Loss Functions and working with tensors.
- HuggingFace Tokenizers[3]- For tokenizing our data using appropriate methods like Byte Pair Encoding (BPE) (Sennrich et al. 2015b).



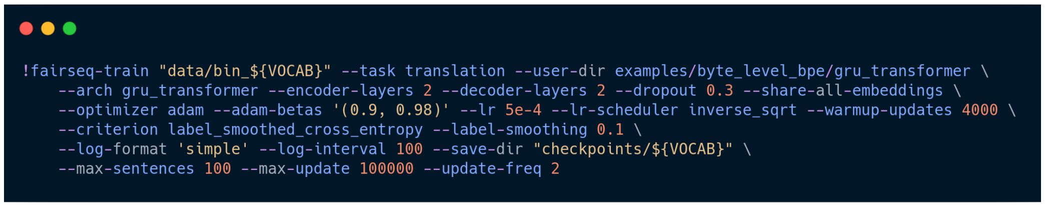
```python
from tokenizers import ByteLevelBPETokenizer
# importing the HuggingFace Tokenizers library
tokenizer_su = ByteLevelBPETokenizer()
# training on sumerian monolingual data
tokenizer_su.train(['monolingual/sumerian_mono.atf'])
# tokenizer ready to be used
```



```python
# using the tokenizer to obtain corresponding
# indices of the text in the vocabulary
tokenizer_su.encode(['szegar gu udukam']).ids
# [142, 31, 74]
tokenizer_su.encode_batch(['szegar','gu udukam']).ids
# [[142], [31, 74]]
```

Figure 4: The recently released HuggingFace Tokenizers library allows us to train tokenizers on our own data, thus we can easily use it for Sumerian.

- fairseq[4]- To perform additional experiments like Transfer Learning with Parallel Data, additional transformer based models like XLM-R and using the Bye-level Byte Pair Encoding (BBPE) (Wang et al. 2019).
- Pandas- For handling, modifying and visualising data.



```
!fairseq-train "data/bin_${VOCAB}" --task translation --user-dir examples/byte_level_bpe/gru_transformer \
    --arch gru_transformer --encoder-layers 2 --decoder-layers 2 --dropout 0.3 --share-all-embeddings \
    --optimizer adam --adam-betas '(0.9, 0.98)' --lr 5e-4 --lr-scheduler inverse_sqrt --warmup-updates 4000 \
    --criterion label_smoothed_cross_entropy --label-smoothing 0.1 \
    --log-format 'simple' --log-interval 100 --save-dir "checkpoints/${VOCAB}" \
    --max-sentences 100 --max-update 100000 --update-freq 2
```

Figure 5: Using fairseq to experiment on a Transformer model with Bi-GRU embedding contextualization along with BBPE.

My local working environment would be Linux (Ubuntu 18.06).

NMT tasks are often computationally expensive, though some of the experiments would suffice within the limits of free cloud based kernels like Google Colab and Kaggle IDE, we would require access to a GPU server with >16 GB of memory, cloud platforms like GCP shall also work.

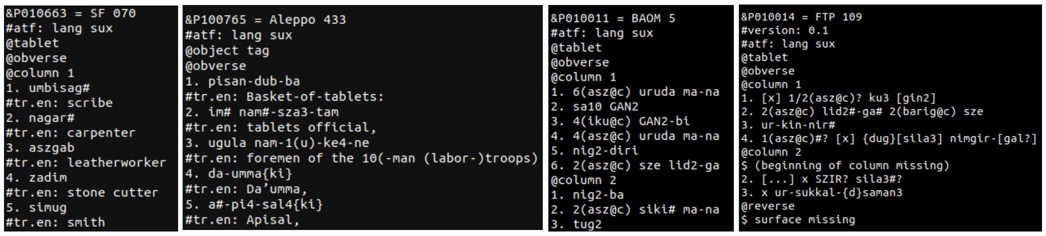
5 Data

As can be seen in Figure 6, there are many symbols and supplementary information given alongside the main text. Some of which are as follows:

- The hash-sign # is used to represent damaged text on the tablet.
- Ellipsis inclosed within square brackets ([...]) depict indeterminable number of missing characters.
- 'x' represents an unidentified sign.
- When a number is missing, the convention is to use n as $n(disz)$.

Presently, while cleaning the data, all such features are completely overlooked. Due to the large presence of these symbols, repeated ignorance of such features might lead to the model learning from an incorrect/incomplete context of the sentences leading to a wrong latent representation of

[3]huggingface/tokenizers
[4]pytorch/fairseq



```
&P010663 = SF 070
#atf: lang sux
@tablet
@obverse
@column 1
1. umbisag#
#tr.en: scribe
2. nagar#
#tr.en: carpenter
3. aszgab
#tr.en: leatherworker
4. zadim
#tr.en: stone cutter
5. simug
#tr.en: smith
```

```
&P100765 = Aleppo 433
#atf: lang sux
@object tag
@obverse
1. pisan-dub-ba
#tr.en: Basket-of-tablets:
2. im# nam#-sza3-tam
#tr.en: tablets official,
3. ugula nam-1(u)-ke4-ne
#tr.en: foremen of the 10(-man (labor-)troops)
4. da-umma{ki}
#tr.en: Da'umma,
5. a#-pi4-sal4{ki}
#tr.en: Apisal,
```

```
&P010011 = BAOM 5
#atf: lang sux
@tablet
@obverse
@column 1
1. 6(asz@c) uruda ma-na
2. sa10 GAN2
3. 4(iku@c) GAN2-bi
4. 4(asz@c) uruda ma-na
5. nig2-diri
6. 2(asz@c) sze lid2-ga
@column 2
1. nig2-ba
2. 2(asz@c) siki# ma-na
3. tug2
```

```
&P010014 = FTP 109
#version: 0.1
#atf: lang sux
@tablet
@obverse
@column 1
1. [x] 1/2(asz@c)? ku3 [gin2]
2. 2(asz@c) lid2#-ga# 2(barig@c) sze
3. ur-kin-nir#
4. 1(asz@c)#? [x] {dug}[sila3] nimgir-[gal?]
@column 2
$ (beginning of column missing)
2. [...] x SZIR? sila3#?
3. x ur-sukkal-{d}saman3
@reverse
$ surface missing
```

Figure 6: Some samples from the original data. The first two images are from the same file containing the Sumerian text along with the English translation, while the last two are from the file containing untranslated Sumerian text. The large variance between different texts of the same file can be noted

the language entirely. Thus, a major challenge of the project would be to extract relevant features from such identifiers given in the uncleaned text and to handle them effectively in the final data. We can assign certain special tokens (alongside tokens like <pad>, <sep>, <eos> etc) to such features so that the model learns how to adapt to their presence over training. All the symbols representing breakage/missing information could be represented by a unified token.

6 Timeline

I plan to undertake and divide the various sub-components of the project throughout the 13 weeks in the following manner.

- **Community Bonding Period**: Getting acquainted with the organisation, interacting with the mentors and getting to know their desired way of working in the coming weeks. Also finalising the computation availability, data to be used as well as the models and algorithms that we would be working on.
- **Week 1**: Preparing the parallel and monolingual texts for final usage by using Regular Expressions along with methods like BPE and BBPE to tokenize the text. Also implementing the devised methods to make efficient use of special characters and tokens in the original data[5] as discussed in Section 5.
- **Week 2**: Implementing the Vanilla Transformer (Vaswani et al. 2017) for English $\leftrightarrow$ Sumerian and training on the parallel corpora while saving weights for future use (for Back Translation). Performing experiments with various sets of hyper parameters and analysing the results in comparison to previously used models.
- **Week 3**: Compiling the results of Vanilla Transformer and doing further experiments solely on the parallel corpora using techniques like Transfer Learning (Zoph et al. 2016) by using pre-trained weights on other language sets.
- **Week 4**: Implementing the XLM Encoder (Lample and Conneau 2019), training it on Masked Language Modelling (MLM), Translational Language Modelling (TLM) and Causal Language Modelling (CLM) tasks on some part of the parallel and monolingual data to obtain the PLT (Positional, Language and Token) embeddings for both English and Sumerian languages.
- **Week 5**: Training the XLM Encoder along with a Transformer Decoder on English $\leftrightarrow$ Sumerian Translation and saving the weights.
- **Week 6**: Back translating sentences from Sumerian monolingual corpus using the saved models thus creating synthetic parallel data, re-training Encoder-Decoder models using this synthetic data for English $\rightarrow$ Sumerian translation task.
  Doing the same for Sumerian $\rightarrow$ English by back translating sentences from English monolingual corpus.
- **Week 7**: Implementing Model-Level Dual Learning (Xia et al. 2018) using tied Transformer Encoder-Decoder pairs. Alternating the training on parallel and monolingual corpora.

[5]Uncleaned Text File

- **Week 8-9**: Investigating the performance variation in Dual Learning with different Encoder-Decoder models such as LSTMs, Vanilla Transformer and XLM, and the effect with various combinations of hyper-parameters. Comparing the results.
- **Week 10-11**: Experimenting with Back Translation by studying the effect of sampling methods, amount of parallel data and model hyper-parameters on the translation quality.
- **Week 12**: Working on completely Unsupervised NMT Systems (Lample et al. 2018) based on Language Models.
- **Week 13**: Comparing and Analysing the results obtained from different models and techniques, finalising the documentation of the work done. Also drawing comparisons with last year's results and noting improvements made. Working on logical and mathematical reasoning behind the observed performances.

7 Contributions

I have been actively contributing to the project by solving issues, which have primarily revolved around data preparation and cleaning so far. (I have the most number of contributions among other contributors.)

Three of my pull requests have been merged in the project repository-

#5: Pushed clean parallel Sumerian-English data, monolingual Sumerian data and the python script used to carry out the cleaning.

#9: Made amendments to the cleaning methodology and thus updated the parallel as well as monolingual '.atf' files. Also arranged the data, files and scripts in appropriate sub-directories in the repository.

#11: Added more features to the preprocessing pipeline, introduced the placeholders like 'NUMB' in the new monolingual data.

8 Additional Information

1. I do not have any other commitments during the 3-months of GSoC (1st June- 31st August), I wish to spare all my time contributing to this project.
2. I shall be able to spend at least 45 hours each week, more if and when required.
3. I'll be mostly working full-time on the code on weekdays (Monday to Friday). On weekends, I'll be focusing on documentation, testing and bug fixing.
4. My awake hours would usually be in between 10 AM IST (4:30 AM UTC; 5:30 AM CET) to 3:30 AM IST the next day (10:00 PM UTC; 11:00 PM CET) and I'm comfortable working anytime during this period. Thus, the difference in time zones would **not** be a problem. I'll be available for communication at all times decided by my mentor and am open to discuss new ideas and methods for the project. I'll responsibly notify my mentor in case of any emergency, with enough detailing.
5. One of the reasons I have been active in CDLI is due to the immensely helpful community and I'll try to keep myself continuously updated with other developments going on.

9 Post Google Summer of Code

If any of the tasks which I have mentioned throughout this document including the additional experiments aren't completed during the GSoC period, I will undertake them post-GSoC. I would also be most interested to take this project ahead as a research study and to work along with the mentors to prepare it for top NLP venues.

I will continue to be an active part of the vibrant CDLI community by interacting with fellow members about future improvements as well as new projects.

References

S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. 2018.

D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma. Dual learning for machine translation. 2016.

A. Holtzman, J. Buys, M. Forbes, A. Bosselut, D. Golub, and Y. Choi. Learning to write with cooperative discriminators. 2018.

A. Holtzman, J. Buys, M. Forbes, and Y. Choi. The curious case of neural text degeneration. 2019.

A. Karakanta, J. Dehdari, and J. van Genabith. Neural machine translation for low-resource languages without parallel corpora. 2018.

G. Lample and A. Conneau. Cross-lingual language model pretraining. 2019.

G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. 2017.

G. Lample, M. Ott, A. Conneau, L. Denoyer, and M. Ranzato. Phrase-based & neural unsupervised machine translation. 2018.

P. S. OW and T. E. MORTON. Filtered beam search in scheduling†. 1988.

A. Poncelas, D. Shterionov, A. Way, G. M. d. B. Wenniger, and P. Passban. Investigating backtranslation in neural machine translation. 2018.

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. 2015a.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. 2015b.

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data, 2015c.

A. Vaswani, N. Shazeer, N. Parmar, and J. Uszkoreit. Attention is all you need. 2017.

C. Wang, K. Cho, and J. Gu. Neural machine translation with byte-level subwords, 2019.

Y. Xia, X. Tan, F. Tian, T. Qin, N. Yu, and T.-Y. Liu. Model-level dual learning. 2018.

B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. 2016.