Deep Latent Variable Models for Text Generation and Summarization

Semester Project by

Tushar Goel

Supervised by: **Diego Antognini and Prof. Boi Faltings**
Laboratoire d'Intelligence Artificielle

Abstract

The ever increasing growth in the usage of Social Media and the E-Commerce has led to a burgeoning volume of textual data. This growth has resulted in the need for efficient processing of these text corpuses. The aim of this semester project is to find smooth and continous latent representations for language processing which can be used to generate and summarize text. We begin by exploring the performance of the state of the art Sequence to Sequence Deep Learning models and then switch to adversarial and generative methods to perform this task.

...

Contents

List of Figures

List of Tables

1 Introduction

Opinion Summarization can be defined as the process of automatically summarizing opinions on the same topic through different sources. Text summarization in general is a very well studied problem in Computer Science and Linguistics. The first attempt at text summarization can be traced back to the 1950's ([1]). Since then, text summarization has seen many use cases in areas such as document search, report generation and Question Answering ([2], [3]). In recent years, two major approaches for generating text summaries have become popular:

- **Extractive Summarization** where important content is extracted from the text corpora using the exact same words.
- **Abstractive Summarization** involves generating new phrases, possibly rephrasing or using words that were not in the original text

In our work, we will focus exclusively on Abstractive text summarization methods, owing to their ability of synthesising/paraphrasing information thereby avoiding redundancies. The proliferation of large-scale datasets and computing power have enabled the development of neural architectures s for summarizing single and multiple documents. . Several approaches ([4], [5], [6]) have shown promising results with sequence-to-sequence models that encode one or several source documents and then decode the learned representations into an abstractive summary.

Most of these works have relied on highly annotated data. Unfortunately, opinion summarization datasets, unlike news or document datasets, are largely unannotated. Thus, we try to find intelligent unsupervised methods to summarize opinions/reviews.

The Mean-Sum paper([7]) was the first attempt at using unsupervised Deep Learning methods to find abstractive summaries, where the authors

use an Auto-Encoder Decoder neural architecture to generate reviews. The problem with this approach is that the latent space of the Auto-encoder is not smooth and the reviews do not take into consideration any labels or sentiment that might add more information to the sentences. Hence, in our work we aim to use a Sequence to Sequence model with a Variational Auto Encoder-Decoder mechanism to find continous latent representations using Pre-trained Language Models. Further, we will try to use Generative and Adversarial training techniques to inject labels into the dataset to find more meaningful summaries.

1.1 Pre-Trained Language Models

Pre-trained language models (PLMs) have substantially advanced the state-of-the-art across a variety of natural language processing (NLP) tasks ([8]). PLMs can generally play two different roles: (i) a generic encoder such as BERT ([9]) to provide contextualized representations for language understanding tasks, and (ii) a powerful decoder such as GPT-2 ([10]) to generate text sequences in an auto-regressive manner.

As described in the previous section, we aim to find smooth latent spaces which are capable of generating text. Variational Auto-Encoders ([11]) provide a tractable method to train latent-variable generative models. In NLP, latent variables may assume the role of higher-level sentence representations, which govern a lower-level word-by-word generation process, thus facilitating controlled text generation.

Thus, we try to use Optimus([12]), a large-scale pre-trained deep latent variable models for natural language. OPTIMUS enjoys several favorable properties as it combines the strengths of VAE, BERT and GPT, and supports both natural language understanding and generation tasks.

The model architecture of OPTIMUS is composed of multi-layer Transformer-based encoder and decoder mechanisms. Further, while training the encoder and decoder weight parameters are initialized by using the weights of Bert

and GPT-2 . The following figure illustrates the Optimus architecture:

Figure 1.1: Optimus Architecture

The loss function minimised by Optimus is given by:

$$L_E + \beta L_R \tag{1.1}$$

$$L_E = -E_{q_\phi(z|x)} log[p_\theta(x|z)] \tag{1.2}$$

$$L_R = KL(q_\phi(z|x)||p(z)) \tag{1.3}$$

It is pretty easy to see that the latent variable z acts as an information bottleneck for the encoding and decoding mechanism. Hence, tightening the information bottleneck (using lesser bits to encode z) leads to smoother and more controllable text generation but at the cost of higher reconstruction error.

While OPTIMUS, provides smooth representations for sentences, we also need to find ways to inject labels into the training dataset. Adversarially Regularized Auto-Encoders (ARAE) ([14]) have traditionally been employed for conditional text generation. ARAEs have been quite successful at capturing latent smooth representations and are good at learning label independent latent representations but they are prone to suffering from conflicting objectives which can lead to bad reconstruction. We hope to alleviate this issue through CARA.

Complementary Auxiliary classifier Regularized Auto-encoder (CARA) ([13]) is used to learn latent sentence representations which are disentangled

with the labels. CARA achieves this by adding two classifiers to predict labels: a classifier in the latent space is used for disentangling feature learning as in ARAE, and a complementary classifier in the observation space is proposed to encourage the generator to contain the conditional label. The following figure illustrates the CARA architecture

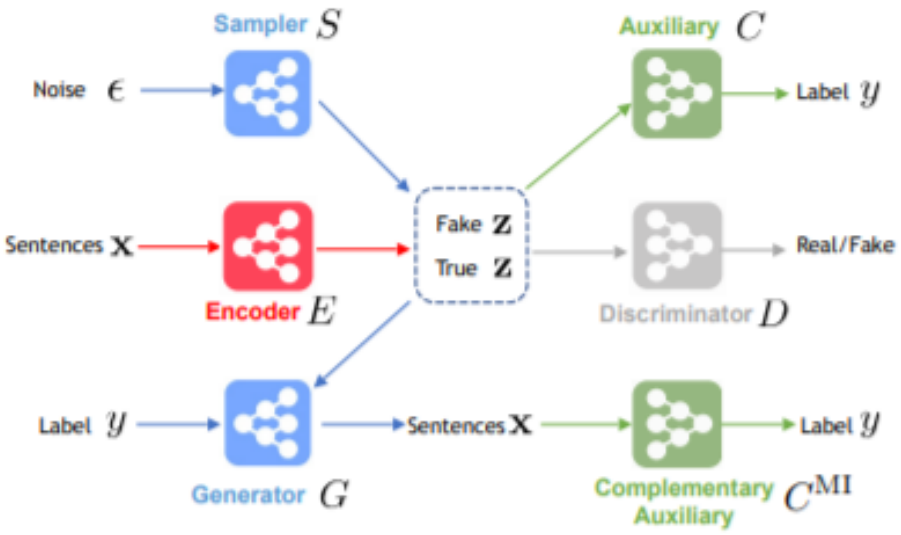

Figure 1.2: Training mechanism of CARA

The loss function minimized by CARA is given by:

$$
\begin{aligned}
& L_{rec} + L_{disentangle} + L_{adversarial} \\
& min_{E,G}\ L_{rec} = -E_{q_E(z|x),p(y)}\ log\ p_G(x|z,y) \\
& min_E\ max_C\ L_{disentangle} = E_{q_E(z|x)}\ log\ [p_C(y|z)] \\
& min_{E,S}\ max_D L_{adversarial} = E_{x\sim p(x)}\ log\ p_D(d=1|E(x)) \\
& +E_\epsilon\ log\ p_D(d=0|S(\epsilon))
\end{aligned}
$$

1.2 Decoding Algorithms for NLP

In the models we have described above, the task of text generation can be given by:

$$p(x) = \prod_i p(x_i|x_{i-1}, x_{i-2}....x_0, z)$$

In the above equation, x_i represent the words of a sentence and z represents the hidden state obtained from the seq2seq models. One of the most obvious ways to go about the text prediction problem is to pick the word with the highest probabilities. This is called as Greedy Search. Unfortunately, Greedy search leads to repititive sentences and words. This is true because the maximum likelihood text is not the same as a text where every token is the maximum probability

$$\begin{aligned} argmax\ p(x) &= argmax \prod_i p(x_i|x_{i-1}, x_{i-2}....x_0, z) \\ &\leq \prod_i argmax\ p(x_i|x_{i-1}, x_{i-2}....x_0, z) \end{aligned}$$

Beam Search ([15]) is one way of solving the above mentioned optimization problem. Beam Search is a graph search algorithm that maximizes the probability of a sequence of tokens by doing a best-first search on the graph formed by selecting m best tokens at each selected token recursively (where m is the beam size).

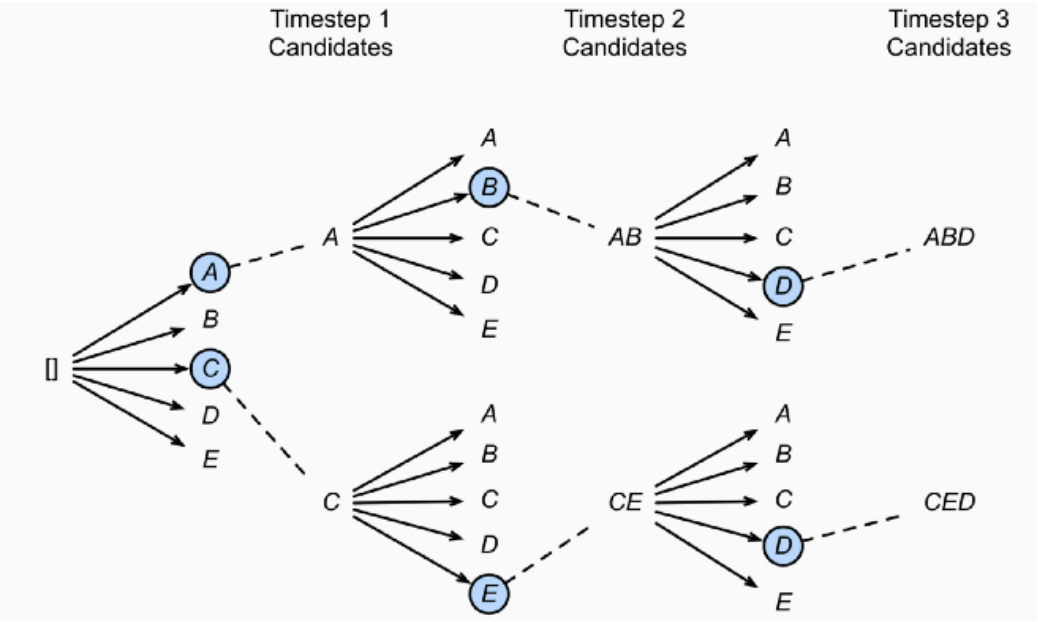

Figure 1.3: Beam search for beam width=2

To alleviate the problem of repitition, we can introduce randomness in our selection by picking words with a probability proportional to their likelihood. Random sampling, by itself, could potentially generate a very random word by chance. Temperature is used to increase the probability of probable tokens while reducing the one that is not. Usually, the range is $0 < temp \leq 1$ To strike a balance between greedy search and random-sampling, Top-K sampling is proposed where only k words, which have the highest probability, are considered for random sampling.

$$p(x_i|x_{i-1}, x_{i-2}....x_0) = \frac{exp(u_j/Temp)}{\Sigma_i exp(u_i/Temp)}$$

2 Dataset and Methodology

In our work, we will use the HotelRec dataset. This dataset contains 500k reviews of different hotel rooms. The following features are provided in the dataset:

- The review of a hotel room
- Aspect: 5 bit encoding which indicates the aspect of the hotel room about which the opinion was written. The following are the aspects used: 'Service', 'Cleanliness', 'Value', 'Location', 'Room'
- Polarities: 3 bit encoding which indicates the polarity of the opinion. The following polarities are used: 'Neutral', 'Negative', 'Positive'

The following is an outline of the strategy we have employed in our methodology:

- Tokenize and embed the reviews into both the BERT and GPT-2 Embeddings
- Train the Optimus Model to reconstruct the hotel reviews without any labels
- Train the CARA model using the weights obtained in Optimus as an initialization. Inject the aspects and polarities tag in the model to create disentangled features.
- Use the injected labels to generate controlled text as well as make linear interpolations in the latent space to summarize opinions

3 Experiments and Results

We use the following two metrics to assess the quality of our reconstructions:

- Bilingual Evaluation Understudy Score (BLEU): It is a performance metric to measure the performance of machine translation models. It evaluates how good a model translates from one language to another. It assigns a score for machine translation based on the unigrams, bigrams or trigrams present in the generated output and comparing it with the ground truth, In our work, Bleu is used to assess how similar the reconstructed sentences are compared to the original sentences
- Recall-Oriented Understudy for Gisting Evaluation (ROUGE): ROUGE is very similar to the BLEU definition, the difference being that Rouge is recall focused whereas BLEU is precision focused. ROUGE-N measures the overlap of N-grams between the generated and real text while ROUGE-L checks for Longest Common Subsequences

In the following experiments we generate hotel reviews by encoding the original hotel reviews in our dataset and then passing these encodings through a decoder. We observe the difference between the generated text and the real text through the above defined metrics.

3.1 Fine-Tuning on Optimus

The loss function minimised by Optimus is given by:

$$L_E + \beta L_R \tag{3.1}$$

$$L_E = -E_{q_\phi(z|x)} log[p_\theta(x|z)] \tag{3.2}$$

$$L_R = KL(q_\phi(z|x)||p(z)) \tag{3.3}$$

It should be noted that L_E represents the reconstruction error while L_R represents the regularization term in Variational Auto Encoders. Hence, if we turn β to 0, we will essentially make the objective function into a standard Auto-Encoder.

Hence, in our first experiment, we observe what happens when the model is run $\beta = 0$ and $\beta = 0.5$. As we'd expect, the KL Loss, which represents L_R, decreases to almost 0 when β is 0.5. We also observe from the testing metrics that $\beta = 0$ performs better as compared to $\beta = 0.5$ since it only tries to minimize the reconstruction loss. It is interesting to note here that while we might get better reconstruction for $\beta = 0.5$, the latent space for these models is much less interpolable.

Figure 3.1: Reconstruction Loss

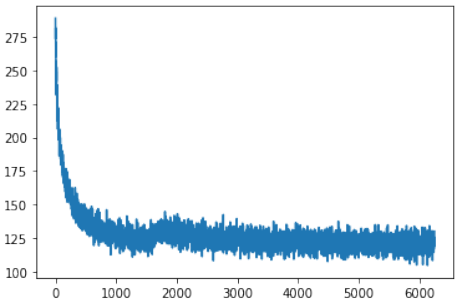

Figure 3.2: KL Loss

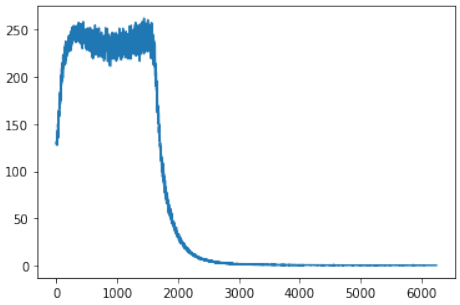

Figure 3.3: Training loss function for $\beta = 0.5$

Figure 3.4: Reconstruction Loss

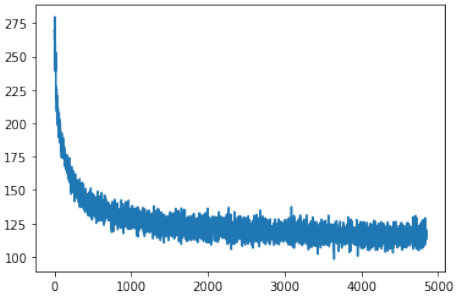

Figure 3.5: KL Loss

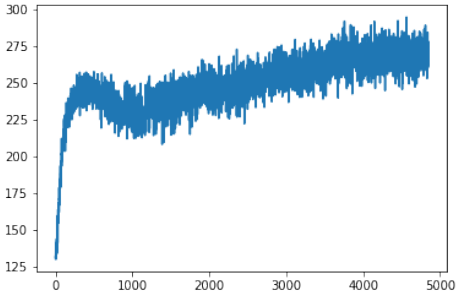

Figure 3.6: Training loss function for $\beta = 0$

Table 3.1: Performance of the Optimus Model for text generation

Beta	Rouge-1	Rouge-2	Rouge-L	Bleu
0	17.6	5.2	16.7	18.2
0.5	16.3	2.1	16.1	10.3

3.2 Training on CARA

The loss function minimized by CARA is given by:

$$L_{rec} + L_{disentangle} + L_{adversarial}$$

$$min_{E,G}\ L_{rec} = -E_{q_E(z|x),p(y)}\ log\ p_G(x|z,y)$$

$$min_E\ max_C\ L_{disentangle} = E_{q_E(z|x)}\ log\ [p_C(y|z)]$$

$$min_{E,S}\ max_D L_{adversarial} = E_{x\sim p(x)}\ log\ p_D(d=1|E(x)) + E_\epsilon\ log\ p_D(d=0|S(\epsilon))$$

We try running the CARA model on our dataset with the aspects being used as the labels. The pretrained weights from fine-tuning are used for the initialization of CARA. We try to observe the difference in the training between initialized and uninitialized models.

Figure 3.7: Pretrained weights

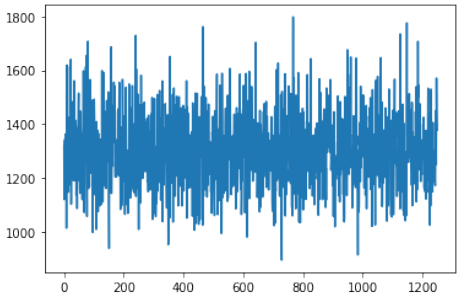

Figure 3.8: Random Initializations

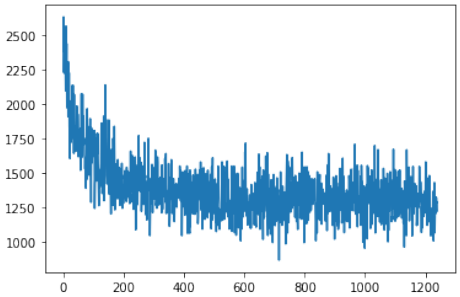

Figure 3.9: Training loss function for CARA

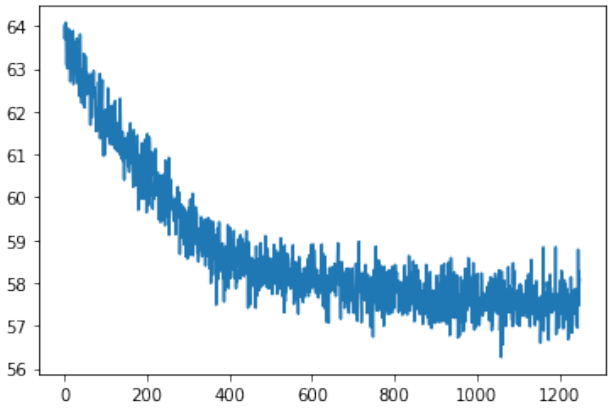

Figure 3.10: Training loss function for CARA minus the Reconstructed Loss

We can see from the above graphs that the pretrained initialization model has almost no loss reduction. This is due to the fact that the term L_{rec} dominates the total loss function and the pre-training of the model parameters with Optimus, which optimizes the same loss function, have made L_{rec} to converge. Visualizing the loss landscape without reconstruction loss, we notice the convergence of the other loss variables.

In the above training mechanism, we update the weights simultaneously for all the model parameters. This does not always lead to convergence for min-max problems ([16]). Hence, we instead try to first optimize the classifier and the discriminator weights and then take a gradient step for the Encoder and the Generator. This is similar to the training methodology employed by the authors of Wasserstein GAN paper. ([17]). The following algorithm

describes the training process through this methodology:

Algorithm 1: Training the CARA model Asynchronously

Initialize the Encoder E, Latent space classifier C, Discriminator D, Generator G and Sampler S appropriately;
Count = 0;
while *count* $\leq$ *number of minibatches* **do**
    // Compute $Loss_{CARA}$;
    if *count*%5! = 0 **then**
        //Freeze the weights of E, G, S ;
        // Backprop for $Loss_{CARA}$
    else
        //Freeze the weights of C, D ;
        // Backprop for $Loss_{CARA}$;
    end
end

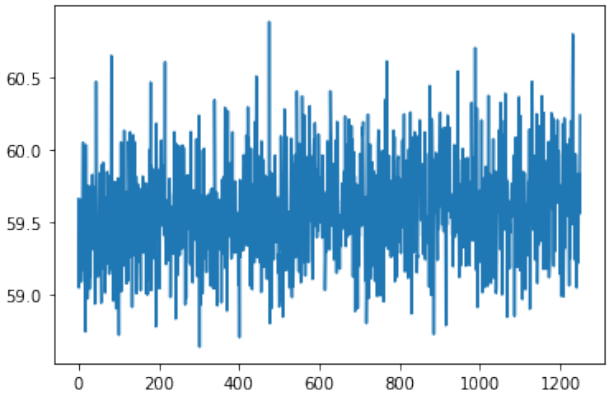

Figure 3.11: Training loss function for CARA minus the Reconstructed Loss

Table 3.2: Performance of CARA Model for text generation

Training	Discriminator Acc.	Classifier Acc.	Rouge-1	Rouge-2	Rouge-L	Bleu
Synchronous	59.1	27.3	19.3	4.7	17.8	18.7
Asynchronous	74.3	32	14.9	3.6	14.3	17.9

We observe in the above graphs that the training loss for Asynchronous training is cyclical without any convergence. Further, we observe in the case of Synchronous training that the classifier and discriminator in the latent space have almost converged to the ideal accuracies (The ideal discriminator accuracy is 50% and ideal classifier accuracy is 20%). Consequently, we observe slight improvements in the testing metrics for Synchronous training as compared to the text generated by OPTIMUS.

3.3 Visualizing the effects of Decoding Strategies

In the following experiment we try to understand how the decoder assigns probabilities to words at different time steps. To do this, we first take a sample of real text and try to observe what probability the decoder assigns to the immediate next word in the text. We then generate text and observe the probability of observing the next word. The following graphs illustrate the difference in the generated and real text:

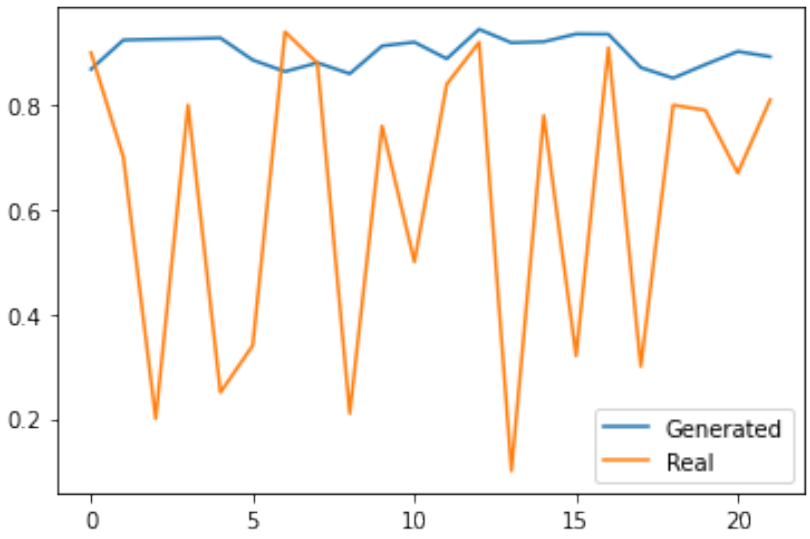

Figure 3.12: Difference in porbability estimates of real and generated text

We observe two things from this graph:

- Real/Natural text has a mix of high and low probability words. Moreover, it is the low probability words which convey the meaning of the sentence
- The generated text keeps producing the same words again and again to maximize the one step prediction problem which is harmful for overall text generation.

We next try to understand the sentences beam search considers at different positions. As it can be seen in the following figure, the decoder initially considers varied sentences but eventually the algorithm considers a homogenous set of sentences at the end. This is one of the main reasons for repitition and lack of variability in the generated text

Figure 3.13: T=1

```
tensor([[50258,  1169],
        [50258,   464],
        [50258,  2990],
        [50258,  1026],
        [50258,  1135],
        [50258,   392],
        [50258,    40],
        [50258,  1212],
        [50258,  1858]],
```

Figure 3.14: T=4

```
tensor([[50258,   464,  3823,  9776, 27773],
        [50258,   464,  3823,  9776,   548],
        [50258,   464, 37648,  3823,  9776],
        [50258,   464, 28120,   533,   548],
        [50258,   464, 28120, 22474,   548],
        [50258,   464,  3823,   271, 27773],
        [50258,  1135, 18108,    64, 18223],
        [50258,  1135, 22474,   548, 34191],
        [50258,    40,  9776,   548, 34191]],
```

Figure 3.15: T=8

```
tensor([[50258,   464,  3823,  9776,   548, 27773,   392, 27773,    13],
        [50258,   464,  3823,  9776, 27773,   392, 27773,    13, 50259],
        [50258,   464,  3823,  9776,   548, 27773,   392, 27773,   392],
        [50258,   464,  3823,  9776,   548, 27773,   392,   548, 27773],
        [50258,   464,  3823,  9776,   548, 27773,    13, 50259,   392],
        [50258,   464,  3823,  9776, 27773,    11,   392,  1169,  3823],
        [50258,   464,  3823,  9776,   548, 27773,   392, 27773,    11],
        [50258,   464,  3823,  9776,   548, 27773,   392, 13120,    13],
        [50258,   464,  3823,  9776, 27773,   392,  4053, 32924,    13]],
       device='cuda:0')
```

Figure 3.16: Text generation from CARA at different positions of the sentence

To further understand the optimization problem faced by the decoder, we demonstrate a simple experiment. We try to see the one step time prediction of the decoder given the correct labels and context at each position. Then we compare it with the text generated by beam search. The following table illustrates the results:

```
['Late night room service and strong options at the bar / rest aur ant are w el coming pieces ',
 'Location is not near any sh opping are as though . ',
 'The hot el rooms are quite sp acious with large bed s and a so fa . ',
 "I 'd give them a B - or a C + when you consider the price . ",
 'The room was super clean and every body was super nice to us . ',
 'Only thing was the ne igh bour ing t ents are too close to each other and talking l oud ly is very dist ur bing
```

Figure 3.17: Original Text

```
['The hot els are very com fortable and very com fortable . ',
 'The hot el was very clean and the room was very clean . ',
 'The hot el is very clean and clean . ',
 'The service is very good . ',
 'The room was very clean and the room was very clean . ',
 'The room was very clean and clean . ']
```

Figure 3.18: Generated Text

```
['The the night hot , is hot staff for all hot bec rest aur ant , very el coming ',
 'The the is very very the where ores room . well it ',
 'The The room el and have very clean acious with room rooms s and a large fa and ',
 'They The can love this a great & and a C - and they have it price . ',
 'We I service was clean clean and the place was very happy . stay and ',
 'I I a we needed staff ither bour hood ']
```

Figure 3.19: Text Generated with original labels

While we observe that the text generated by the correct labels seems to be meaningless, the text produced in the second half of these sentences is much closer to the real text. This illustrates that the decoding process in itself is a very complicated optimization problem. Also, if we get the first few words of the sentence right, we can reconstruct the original sentences with their preserved meanings.

4 Conclusion

4.1 Summary

It is easy to see from our results that we still have a long way to go to truly generate human-like text. In the first half of our work, we tried to find smooth continous representations for sentences using OPTIMUS. We later experiment with Generative and Adversarial methods to learn latent representations which are disentangled from labels and can subsequently be used not just for summarization but also for style transfer.

It is easy to see that the paucity of data is not the reason for the poor performance of our models since all the training plots have converged. We observe that our decoders are able to make coherent sentences and can capture some of the original meaning of the inputs but tend to fall prey to the problem of repitition. Finally, we show some insights into the decoding mechanisms and how they might be causing the problems we observe.

4.2 Recommendations for Future Work

We can experiment with the information bottleneck of the Variational Auto Encoder in OPTIMUS. We might find an improvement in the reconstruction loss by loosening this bottleneck. Further, as illustrated above, we might see sharp improvements in the generated text by working on more efficient decoding mechanisms. Posing the decoding problem as a Reinforcement Learning problem might help us in exploring more sentences from a time-horizon perspective.

Bibliography

[1] H.P. Luhn, *"The Automatic Creation of Literature Abstracts.*

[2] Kristjan Arumae, Fei Liu, *"Guiding Extractive Summarization with Question-Answering Rewards"*. In: NAACL 2018

[3] Byron C. Wallace, Sayantan Saha, Frank Soboczenski, Iain J. Marshall, *Generating (Factual?) Narrative Summaries of RCTs: Experiments with Neural Multi-Document Summarization*. In: Arxiv 2019

[4] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, Yejin Choi *Deep Communicating Agents for Abstractive Summarization*. In: NAACL 2018

[5] Yang Liu, Mirella Lapata, *Text Summarization with Pretrained Encoders*. In: Arxiv 2019

[6] Laura Perez-Beltrachini, Yang Liu, Mirella Lapata *Generating Summaries with Topic Templates and Structured Convolutional Decoders*. In: Association for Computational Linguistics 2019

[7] Eric Chu, Peter J. Liu, *MeanSum : A Neural Model for Unsupervised Multi-Document Abstractive Summarization*. In: ICML 2019

[8] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, Xuanjing Huang, *Pre-trained Models for Natural Language Processing: A Survey*. In: Arxiv 2020

[9] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* . In: Arxiv 2019

[10] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever *Language Models are Unsupervised Multitask Learners*. In: Arxiv 2018

[11] Diederik P Kingma, Max Welling, *Auto-Encoding Variational Bayes*. In: Arxiv 2014

[12] Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, Jianfeng Gao, *Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space*. In: EMNLP 2020

[13] Yuan Li1, Chunyuan Li2, Yizhe Zhang, Xiujun Li, Guoqing Zheng, Lawrence Carin , Jianfeng Gao *Complementary Auxiliary Classifiers for Label-Conditional Text Generation*. In: Association for the Advancement of Artificial Intelligence 2020

[14] Jake (Junbo) Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, Yann LeCun *Adversarially Regularized Autoencoders*. In: ICML 2018

[15] Sam Wiseman, Alexander M. Rush *Sequence-to-Sequence Learning as Beam-Search Optimization*. In: EMNLP 2016

[16] Naveen Kodali, Jacob Abernethy, James Hays Zsolt Kira *ON CONVERGENCE AND STABILITY OF GANS*. In: Arxiv 2017

[17] Martin Arjovsky, Soumith Chintala, Léon Bottou *Wasserstein GAN*. In: Arxiv 2017