POLITECNICO DI TORINO

Master degree course in Mathematics Engineering

Master Degree Thesis

VAE for NLP: Novel Architectures and Possible Applications

Supervisor
prof. Maurizio Morisio

Candidates
Felipe Gonçalves Marques
Student ID: 249996

Internship Tutor
LINKS Foundation
dott. ing. phd. Giuseppe Rizzo

Academic Year 2018 - 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Text Generation and VAE

Natural Language Processing consists in the study of how to automatically process information conveyed in Natural Language, that is, human language. This thesis is interested in a subset of this domain called Text Generation which consists in the elaboration of computational system capable of generating text comprehensible by humans.

More specifically, it is interest in exploring a specific model named Variational Autoencoders, which is one of the most popular models in the academia in the time of elaboration of this work.

1.2 Goals

In the context of the VAE, this thesis aims to accomplish two sets of goals. The first is: to improve the overall performance of the VAE in the text generation task, the understanding of the most new techniques and reflect upon how to evaluate Text Generation Models. This shall be done by combining recent advances in the VAE architectures and by proposing a new technique.

The second set of goals is to explore the applications of the text generation system to other tasks, analyzing its performance and giving guidance to future work in the application of VAEs and Text Generation Systems.

1.3 Thesis structure

This thesis is divided into 7 chapter. This one is the first chapter and summarizes the thesis domain and goals.

The second chapter consist in a listing of the related work done in the field. It specifies the scope of research and the state-of-the-art methodologies applied to that scope.

The third chapter contains the methodology used to develop the thesis. It list the used datasets, known baselines, the proposed models and the possible applications of the VAE.

The fourth chapter makes explicit the experimental setup, by specifying the model training methodology, how the metrics are calculated and giving explicit dimension of the tested models.

The fifth chapter list the results of all the experiments together with important baselines, highlighting qualitative examples and presents some model-specific analysis.

The sixth chapter presents the result for each proposed application together with some analysis.

Finally, the seventh chapter concludes the work, giving an overview of the overall result and suggesting directions for future work to be done.

Chapter 2

Related work

2.1 Scope

2.1.1 Artificial Intelligence

In general and abstract terms, Artificial Intelligence focus on creating machines capable of realizing complex tasks – normally, tasks that humans performs well. It is not a new idea and it has been present in the human imaginary for a long time.

There are Greek myths that talk about intelligent robots create by the god Hephastus. During the beginning of modern age, alchemists proposed ideas to put mind into matter called Homunculus. And in several societies, automatons were created to mimic human reasoning and intelligence. Finally, in recent years, there has being a paramount increase of Artificial Intelligence elements in pop culture.

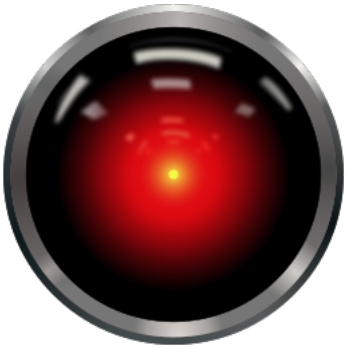
Humans always wondered if it is possible to add intelligence and ability of reasoning to artificial elements. But, the formalization and exploration of the field using scientific methodology flourished only with the rise of computer science.

The two main events that gave birth to Artificial Intelligence as a research field were Alan Turing's article "Computing Machinery and Intelligence" [1] and the Darmouth Summer Research Project [2], both in the early 50s. Although, those were water shedding events, they were result of a series of development in the 200 years before such as Boole algebra, Probability theory and much more.

Turing's article was innovative, because it devises a very simple procedure to evaluate an artificial intelligent actor. It is known as Turing test – or Imitation Game.

It consist of a questioner chatting through a textual interface with what

(a) Golden Automaton Talus



(b) HAL 9000 from Warner Bros's 2001: A Spacy Odyssey

Figure 2.1: AI representation through time

she thinks are two other people, a man and a woman. The goal of the questioner is to decide which one is the woman and which one is the man, while both of try to pass as a man. Turing's proposal is to evaluate how many times the questioner gets the game right, when one of the respondents is a machine. In other terms, the Artificial Intelligence would have to outperform the human competitor.

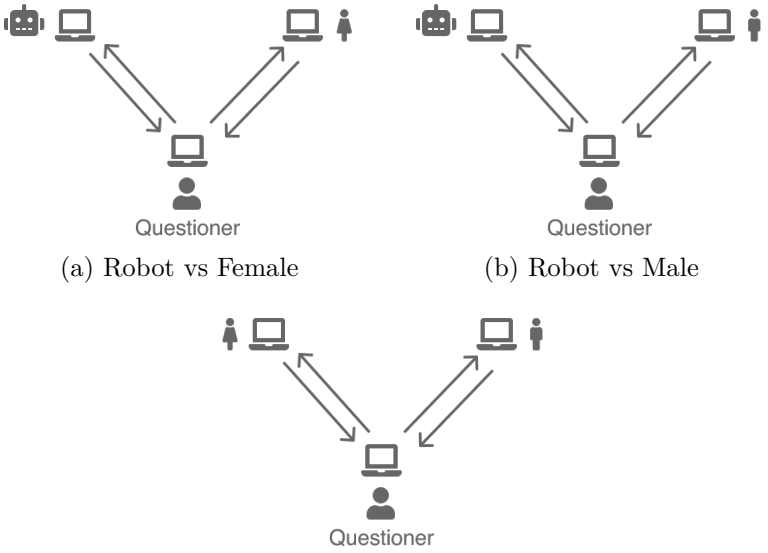
To engage in this full conversation setting, a machine would not only have to be able to understand and formulate natural language, but also demonstrate human traits such as authenticity, creativity and even emotion. It would also have to be able to solve any problem that normally a human would be able to do, such as math, logic problems, etc.

The other event was The Dartmouth Summer Research Project. It was a series of discussion sessions that spanned for two months with the attempt to – in their own words:

> "how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves" from [2]

One important aspect of this proposal is the ability of artificial intelligence to "improve themselves", in the most basic terms, this can be translated as the ability of learning and today represents one of the most important field of Artificial Intelligence research: Machine Learning.

Besides being 50 years old, both those events still influences modern Artificial Intelligence. As the field moves itself towards domains such as Natural



(a) Robot vs Female

(b) Robot vs Male

(c) Female vs Male

Figure 2.2: Turing test possible combinations

Language Processing or Image Processing – for example, text summarization or object recognition, the Turing's Test becomes the final validation of the system quality. And now days, the most prominent methodology for solving AI problems is Machine Learning, that focus on devising algorithms that, as the Dartmouth Project suggests, allows machines to improve themselves.

Another important aspect to notice is the duality of both proposals. While the Dartmouth Summer Research Project focus on abstractions, concepts and solving problems – that is, a rationalization of human thought, the Turing's Test requires a simulation of human behavior: not always pure rational, but also presenting characteristics such as emotion, creativity, etc.

Consequently, as explored in [3], it is normal to divide the overall ability of intelligent agents into two areas: humanly behavior and rational behavior. While some tasks under the Artificial Intelligence domain requires pure rational behaviour such as optimization problems, other tasks such as text generation, question answering requires some level of human-like behavior.

Bringing it to concrete world problems, take an example the use of Artificial Intelligence to make ChatBots for client support – something already being explored by companies such as Microsoft, Apple and Google. The expectation regarding this system has an implicit Turing's test. It is expected that the system provides a service with quality just as good as a human, with characteristics such as kindness and empathy.

In summary, the Artificial Intelligence as a research field is worried about creating intelligent actors which are able of both performing rational tasks in a human-level, but also mimics human behavior outside of the pure rational domain.

2.1.2 Natural Language Processing

Natural Language Processing is the domain that focus in working with human language, that is, languages used primarily to human-to-human communication. It covers both written language, spoken language and sign language in several formats, from simple image captions to long structured text such as medical reports.

It may be divided in understanding and generating natural language artifacts, from the most fined-grain to more holistic tasks. Examples range from making summaries of long documents or identify the morphology of a word. It may also combine different languages, such as text translation, or even interact with other domains such as image processing, for example, by automatically generating caption for images.

Text Generation

One of the main tasks in Natural Language Processing is text generation. It consist in generating text, with or without constraints. One possible mathematical formulation of this task is the following:

Be L the set with all valid text on a specific language, $\overline{L}$ the set of all possible text on the same language and Z an origin space, a text generation system is a function:

Definition 2.1.1. Text Generation Task Be L the set with all valid text on a specific language, $\overline{L}$ the set of all possible text on the same language and Z an origin space, a text generation system is a function:

$$\mathcal{F} : Z \to \overline{L} \text{ where } L \subset \overline{L} \tag{2.1}$$

That maps values from the origin space Z to text in the "language space".

This function $\mathcal{F}$ has a few desirable characteristics:

- The function should map only to valid texts
- The function should be able to map to all possible valid texts

They may be expressed as $Im\,\mathcal{F} = L$. In other terms, only valid term is desirable, but also diversity in generation. This goal allows to compare different text generation systems. Take for example, the following toy-language with:

- Word-set $W = \{$"a", "b", "c"$\}$
- Language space $\overline{L} = \{$"a b c", "a c c", "a b b"$\}$
- Valid text $L = \{$"a c c", "a b b"$\}$

And two text generating functions from $[0, 1] \to \overline{L}$ defined as:

$$\mathcal{F}_1(z) = \begin{cases} \text{"a c c", for } z \leq 1/3 \\ \text{"a b b", for } 1/3 < z < 2/3 \\ \text{"a b c", for } 2/3 \leq z \leq 1 \end{cases}$$
$$\mathcal{F}_2(z) = \text{"a c c", } \forall z \in [0, 1]$$

On one hand, F_2 only generates valid text, but it generates always the same text – also known as mode collapse. On the other hand, F_1 generates two valid text, but it also generates an invalid text.

Therefore, different systems may have different performance regarding those aspects. Besides all that, the text generation task as proposed is ill-posed: it is not possible to define L. The "language space" is not just infinitely large, but it is also subjective, the definition of a valid text may vary according to the person evaluating the text.

This problem is normally solved by using reference corpus of text generated or evaluated by humans. It is used to evaluate the system both in diversity, ie capability of generating different texts, and quality, ie capability of generating valid texts. The reference corpus consist on a large amount of valid examples in the language. Two metrics derived from this approach are explained below:

Average BLEU@n

The first metric is called BLEU (Bilingual Evaluation Understudy). It was first presented at [4] and it was created to evaluate Automatic Translation System, but it has also been used for evaluating text generation system. It is heavily used in both settings due to its correlation with human judgment.

It needs a set C of candidates generated by the system and a set $\mathcal{R} = \{R_1, ..., R_n\}$ of references.

The core idea of its calculations is the modified precision. But before explaining it, lets explore the traditional precision and its drawbacks.

The traditional precision would count each word in the candidate that is present in the references and then divide this by the number of words in the candidate. But, in this case a phrase with repeated words only, would have perfect precision. This computation can be reformulated as the following procedure:

1. Get the set of distinct words in the candidate
2. Filter the set of distinct words in the candidate by the ones that appear on the reference
3. Count how many times each word appears in the candidate
4. Sum those values and divide by the length of the sequence

The modified precision changes the procedure by adding a new step to avoid that the repetition of words increases the precision:

1. Get the set of distinct words in the candidate
2. Filter the set of distinct words in the candidate by the ones that appear on the reference
3. Count how many times each word appears in the candidate
4. **Clip those values by the maximum of number of times each distinct word appears on each reference**
5. Sum those clipped values and divide by the length of the sequence

The following example compute the precision using both the first definition and the procedure of the modified precision:

Table 2.1 Example of BLEU calculation

Candidate: a a man man man ski **Reference 1:** a man is skiing on the mountains **Reference 2:** a man is on skis going down a mountain	
Precision $P = \frac{1+1+1+1+1+0}{6} = \frac{5}{6}$	**Precision Procedure** 1. { "a", "man", "ski" } 2. { "a", "man" } 3. { 2, 3 } 4. $\frac{5}{6}$
Modified Precision Procedure 1. { "a", "man", "ski"} 2. { "a", "man" } 3. { 2, 3 } 4. { $min(2,2)$, $min(1,3)$} = { 2, 1 } 5. $\frac{}{6}$	

The modified precision penalizes sentences that repeats lot of words in other to improve precision. But, using just one word does not guarantee much fluency. Instead BLEU@n computes several BLEU metrics for different grams. A gram is a group of words. Below is the representation of the candidate and reference from the example before in Bigram ($n = 2$) and Trigram ($n = 3$):

Table 2.2 Example of n-grams

	Bigram	Trigram
Candidate	{ "a man", "man ski"}	{ "a man ski"}
Reference	{"a man", "man is", "is skiing", "skiing on", "on the", "the mountains"}	{"a man is", "man is skiing", "is skiing on", "skiing on the", "on the mountains"}

In this case, the phrase with repeated words would generate grams like "man man" that would not match with any gram on the references. A

BLEU@n computes the BLEU for grams representation from 1 to n and averages then geometrically or according to some set of weights:

$$\text{BLEU@n} = BP * \exp(\frac{1}{n}\sum_{i=1}^{n}\log p_i) \tag{2.2}$$

Where BP is a Brevity Penalty in case the candidate is too short regarding the sentences:

$$\text{BP} = \begin{cases} 1 \text{ if } c > r \\ \exp(1 - r/c) \text{ if } c \leq r \end{cases} \tag{2.3}$$

Where c is the candidate length and r is the length of the reference with the closest length to the candidate.

The BLEU@n normally used is BLEU@4 or BLEU@5, which has good correlation with human judgment. Finally, the Average BLEU@n consist on averaging the BLEU@n for several generated text from the system against the reference corpus.

In summary, the Average BLEU@n is a measure of the quality of the text generation system regarding the reference corpus, where higher is better. But it is not measure on diversity, since a system that generates only one sentence will have the BLEU@n of that sentence.

SelfBLEU

In order to measure system diversity, the SelfBLEU is used, which was proposed at [5]. As the name says, SelfBLEU consist in calculating the BLEU@n using the generated data as reference. It works as following:

Given a collection of generated text $G = [T_1, T_2, ..., T_N]$ we compute the SelfBLEU as:

$$\text{SelfBLEU@n} = \frac{1}{N}\sum_{i=1}^{N} BLEU@n(T_i, [T_1, ..., T_{i-1}, T_{i+1}, ..., T_N]) \tag{2.4}$$

For most text generation systems, it would be impossible to list all generate text, therefore a sampling approach is used, ie a sample of generated text is used as the collection. Bellow there are two examples demonstrating the SelfBLEU computation using BLEU@1 without brevity penalty for simplicity:

Table 2.3 Example of SelfBLEU@1 for different datasets

Text 1: A man is skiing	**Text 1:** A man is skiing
Text 2: A man is on skis	**Text 2:** A boy is skating
Text 3: A woman is skiing	**Text 3:** A woman is swimming
Text 1 1-BLEU: $\frac{4}{4} = 1$	**Text 1 1-BLEU:** $\frac{2}{4} = 0.5$
Text 2 1-BLEU: $\frac{3}{5} = 0.6$	**Text 2 1-BLEU:** $\frac{2}{4} = 0.5$
Text 3 1-BLEU: $\frac{3}{4} = 0.75$	**Text 3 1-BLEU:** $\frac{2}{4} = 0.5$
Self-BLEU: $\frac{2.35}{3} \approx 0.783$	**Self-BLEU:** 0.5

The SelfBLEU@n provides a way of measuring the diversity of a Text Generation system. In opposition with the Averaged BLEU@n, smaller is better.

2.1.3 Information Retrieval Task

Information Retrieval is an orthogonal task regarding Natural Language Processing. It consist in retrieving information based on a query. It can be posed as:

Definition 2.1.2. IR Task Given a set of documents $D = [d_1, ..., d_n]$ and a query space Q, a information retrieval system is a function IR such that:

$$\text{IR}(q) : Q \rightarrow G(D) = \pi(D) \tag{2.5}$$

Where $G(D)$ is the set of all permutations of D and $\pi(D)$ is one permutation. In this context, each permutation $\pi(D)$ defines a rank of the documents.

For example, given the following query and documents, one possible rank would be:

Table 2.4 Example of Information Retrieval Task

Query: Washington DC	**Rank:**
Document 1: Washington DC Wiki article	**1st:** Washington DC Wiki article
Document 2: Brazil Wiki article	**2nd:** USA Wiki article
Document 3: USA Wiki article	**3rd:** Brazil Wiki article

It is important to notice that in this context, documents mean any kind of data and it does not need to be the same type as the query. For example, some image search websites have text as query and images as document.

In this kind of task, the concept of **relevance** is important. If a document is related to the query presented, it is considered relevant to it. But again, as in the Text Generation task, to formalize this concept and measure it is a difficult task, mainly due to it subjectivity.

Therefore, in other to provide a evaluation system, references produced by multiple human judges excluding the ones without consensus.

A set of references consist in a query q and a set $D_q \subset D$ of relevant documents. The metrics to measure the quality are applied over this set. The two most common metrics are:

Accuracy

The Accuracy is one of the simplest measures for Information Retrieval systems. For each set of query and relevant documents, we compute if the most ranked document is one of the relevant documents. The accuracy is an average of those indicator variables, ranging from 0 to 1.

Mean Reciprocal Rank

The Accuracy has a problem as measure, because it does not quantify by how much the system got it wrong. The Mean Reciprocal Rank (MRR) tries to overcome this by using a different measure: the rank of the first relevant document. Formalizing the MRR is calculated as:

Definition 2.1.3. Mean Reciprocal Rank

$$\text{MRR} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{Rank_n} \tag{2.6}$$

Where $Rank_n$ is the rank of the first relevant document for the nth reference. It is important to know what is the naive baseline for the MRR. By naive, we mean a system that randomly order the documents.

Therefore, suppose there are N documents in the search space, the naive baseline randomly choose between the permutations and there is only one relevant document per query. The probability of this document being ranked in the position i is:

$$P(Rank = i) = \frac{1}{N}$$

Therefore, the expected MRR is:

$$\mathbb{E}(\text{MRR}) = \sum_{i=1}^{N} P(Rank = i)\frac{1}{i} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{i} = \frac{H_N}{N}$$

Where H_N is the harmonic number. This value is bounded by:

$$\frac{H_N}{N} \leq \frac{\log(N+1)}{N}$$

Which is decreasing and, for example, for N = 100, we have the expected MRR around 0.005.

In conclusion, the MRR provides a better measure for the IR task.

IR in Natural Language Processing

In the Natural Language Processing domain, the Information Retrieval task must contain natural language elements, that is, the query or the documents must consist or contain natural languages elements. More specifically, when the main information is stored in textual form, it is called Text Retrieval.

2.2 State of the art

2.2.1 Neural Networks and Deep Learning

Since the founding of Artificial Intelligence as a research fields, the methodologies used to confront the tasks changed over the decades. In the current moment, the most common and popular methodology is Neural Networks and Deep Learning.

The core of Neural Networks is in the creation of powerful system using basic computing units. By composing those units in different architectures, the network becomes able to learn complex patterns.

The rise and the popularity of this approach is not due only to an innovative way to tackle those problems, but also on the developments of more powerful hardware and in the rise of the quantity of data available to perform the training of those networks.

Neuron

The basic computing mentioned previously is called a Neuron. It has this name, because the Neural Networks are inspired in brain structure and its ability to learn different tasks. A neuron consist of input signals and an

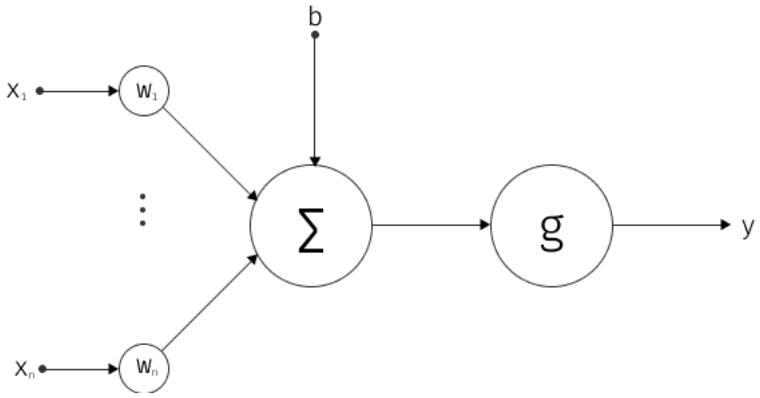


Figure 2.3: Artificial Neuron

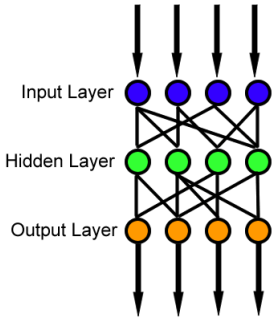


Figure 2.4: Feed forward Neural Network

output signal. The most common neuron form is composed by a weighted-sum and an activation function as seen in figure 2.3. Its equation may be written as:

$$y = f(x) = g(\sum_{i=1}^{n} w_i x_i + b) \tag{2.7}$$

The knowledge of the network is coded through the weights $w = [w_1, ..., w_n]$ of the neurons in the network. Those weights are updated during training. The neurons can be combined to create more complex networks such as the seen in figure 2.4.

They can also be used to construct layers that process an input vector into an output vector. One very common layer is the SoftMax, displayed in figure 2.5, which perform the following computation in order to obtain an stochastic vector:

$$p_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{2.8}$$

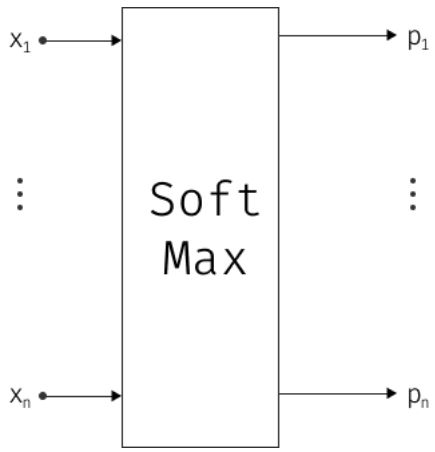


Figure 2.5: Softmax layer

In this case, the neurons are implementing a specific function and will have fixed weights.

In summary, the neurons are the building block of the neural networks, they are used to create complex trainnable layers or to implement useful functions.

Discrete tokens, Embedding Layer and Output layer

As pointed out before, a Neuron is a simple computational unit that performs weighted-sum, functions, etc. It works with continuous values and mathematical operations. This presents an obstacle to work with text.

When thinking on the elements of natural language, more specifically, written natural language, it is composed of tokens: characters, words and punctuation. Those elements are not only discrete, but they also do not have any of the most basic operations such as adding, subtracting, multiplying, etc.

Therefore, special layers are needed to process and output such elements. Below is described the most common way to confront those problems:

Word input and Embedding Layer

The most common method to deal with class as input of neural networks is through embeddings. Embeddings are basically a table that maps those values into real values or vector of real values.

Definition 2.2.1. Embeddings Given a set of tokens (word, sequence of words or even letters) W, a Embedding function map this to continuous

values:

$$\text{Emb} : W \to \mathbb{R}^n \tag{2.9}$$

An well-known example of this function is the one-hot encoding. The OneHot function can be defined as:

$$\text{OneHot} : W \to \mathbb{R}^{|W|} \tag{2.10}$$

For a toy language with $W = \{\text{"}hello\text{"}, \text{"}world\text{"}, \text{"}bye\text{"}\}$, a one-hot encoding would be:

$$\begin{aligned} \text{OneHot("hello")} &= [1, 0, 0] \\ \text{OneHot("world")} &= [0, 1, 0] \\ \text{OneHot("bye")} &= [0, 0, 1] \end{aligned}$$

This embedding is a fixed embedding, because the vectors does not change during training. A more sophisticated approach is to use learn-able vectors where the embedding would be:

$$\begin{aligned} \text{Emb} &= \text{OneHotToVector} \circ \text{OneHot} \\ \text{where OneHotToVector} &: \mathbb{R}^{|W|} \to \mathbb{R}^n \end{aligned}$$

For our toy example, we could have

Table 2.5 Example of OneHot Encoding and Embedding

Word	OneHot	Embedding
"hello"	$[1, 0, 0]$	$[0.5, 0]$
"world"	$[0, 1, 0]$	$[0.25, 0.25]$
"bye"	$[0, 0, 1]$	$[-0.5, 0]$

Which could be implemented by the network presented in figure 2.6.

The advantage of using this approach are two. The first consist in the relation that the embeddings have when done in large scale. For example:

$$\text{Emb("king")} - \text{Emb("man")} = \text{Emb("queen")} - \text{Emb("woman")}$$

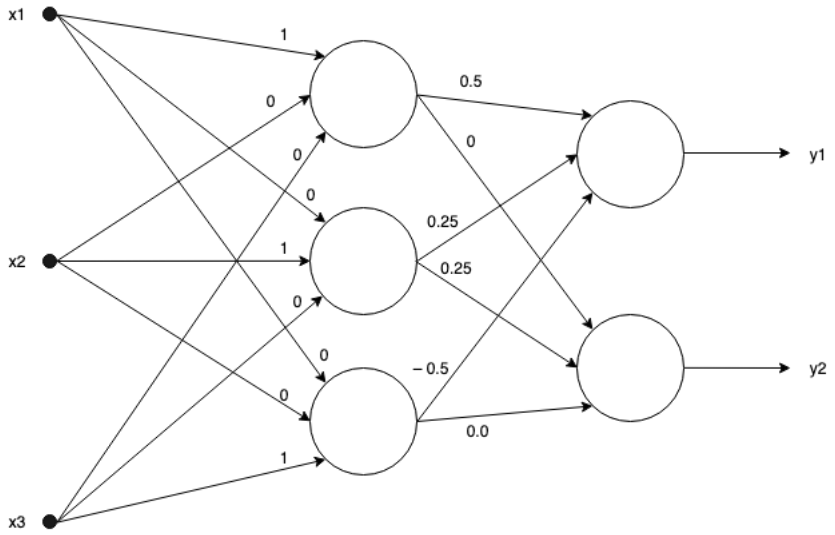


Figure 2.6: Embedding network toy example

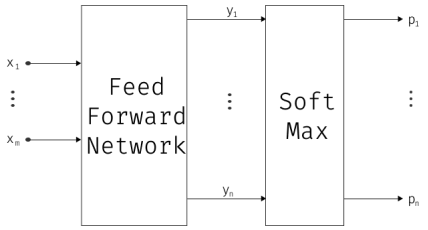


Figure 2.7: Output of probabilities over a set

The other advantages is this layer can be trained in generic tasks over large scale datasets – like datasets from web scrapping – and them be used for specific tasks systems. Two famous examples of pre-trained embeddings is GloVe [6] and FastText [7].

This layer presents a fundamental role in the development of Natural Language Systems.

Output Layer

On the other-hand, to output elements from a set, the Neural Network outputs a probability of each value. It does so by combining a Linear Feedfoward Layer and a Softmax function – as shown in figure 2.7, where the output has the same size of the set. In this case, the FeedForward network maps the input vector of size m to an output vector of size $n = |W|$ which is them normalized to be a stochastic vector.

This probability output vector is them used to choose the more likely

element. The most common approach is to select the element with the highest probability. For the toy example presented before, if $p = [0.8, 0.1, 0.1]$, the selected word would be **"hello"**.

Sequence processing and Recursive Neural Networks

As emphasized before, Natural Language Processing deals with several different types of data from short length to long structured text. Take example the Text Retrieval context, we have both small-size text, such as queries, and long-size text, such as the references that are used to search for the answer. But even if we focus on only one sub-domain, we would have different-sized questions or text.

This variation of length creates an obstacle to apply simple Feedfoward Networks, due to its constraint on input size. The two most common methods for dealing with varying size sequences are Convolutional Neural Networks and Recursive Neural Networks, being the latter the one explored by this thesis.

Recursive Neural Networks lifts the ban on different size input by processing a fixed-size chunk of the input at a time, and using the output as the input for the next step. Its input is normally divided into state and input. In the figure 2.8, we have both the basic representation of an RNN and its operation over an input.

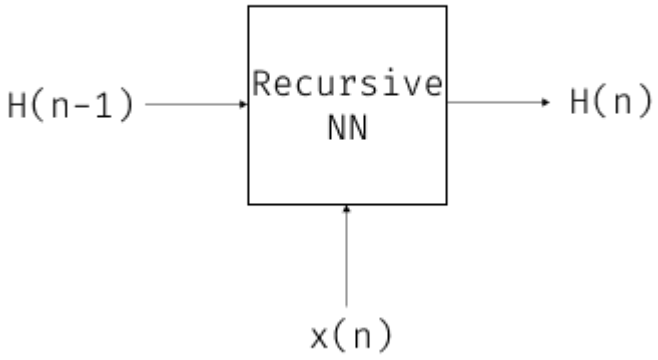
Using the elements presented above – embedding layers, probabilistic output layer and RNN, we can construct a full-functional model that works with Natural Language. To give an example, take the problem of translating sentences. A common approach is to use an Autoencoder which encodes the sentence using an RNN and then uses the hidden representation to generate the sentence in the targeting language. This system is represented in figure 2.9. In those kind of systems, the tokens "<sos>" and "<eos>" are used to denote the start of the sentence and the end of the sentence, respectively.

This system accepts as input and outputs Natural Language elements.

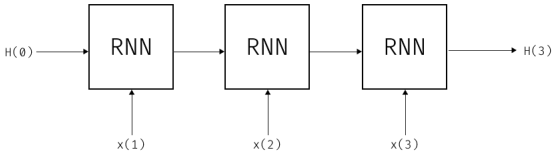
2.2.2 Neural Networks for Text Generation

To create a Text Generation function, a probabilistic model can be used in other to learn from available examples. Our goal is to fit the following probabilistic function:

$$P(t|z) \text{ where } z \text{ is our control signal and } t \text{ is the text} \tag{2.11}$$



(a) Recursive Neural Network (RNN)



(b) Unfolded RNN

Figure 2.8: Example of Recursive Neural Networks

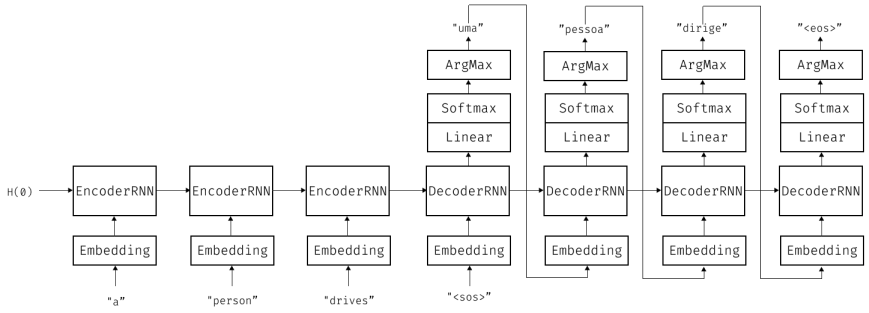


Figure 2.9: Neural Translation Model

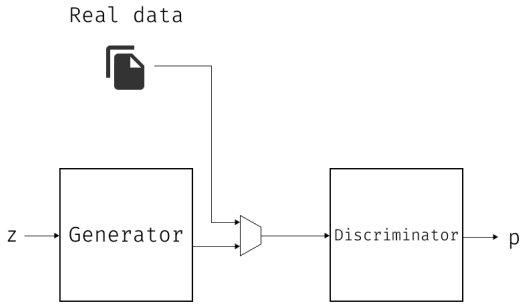


Figure 2.10: Generative Adversarial Network

And we may define our $\mathcal{F}$ as:

$$\mathcal{F} : Z \rightarrow \mathcal{L} = argmax_t P(t|z) \tag{2.12}$$

In the probabilistic formulation done, z is now called a latent variable, ie a hidden representation, that allows us to predict the most likely sentences.

In the deep learning context, there are two famous models to take this approach to learn $P(w|z)$ using real data. They are the Generative Adversarial Networks (GAN) and Variational-Autoencoders (VAE).

Generative Adversarial Networks

The Generative Adversarial Network was first presented in [8]. This model consist in a multiple turns two-players game, connected as describe in figure 2.10 . The players are a generator and a discriminator. The discriminator must learn do distinguish real examples from artificial ones, created by the generator. The generator, on the other hand, must try to fool the discriminator.

The discriminator produces as output a p which is the probability of the presented sentence to be real. When training the generator, we update its weights in the direction to make the p higher for the artificial examples – trying to fool the discriminator. On the other hand, when training the discriminator, we present it with examples from both the real data and from the generator, where for the former, it must output a high probability and, for the latter, it must output a low probability – trying to catch the generator forgery. Therefore, the networks are trained on opposed goals.

In the work presented in [8], they demonstrate that theoretically, this game has an equilibrium where the generator fits well the target function, but training the GAN to reach this equilibrium is far from trivial.

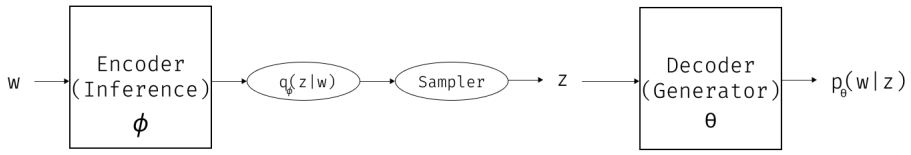


Figure 2.11: VAE model concept

Nonetheless, it was a really important breakthrough, because the learning happens in an indirect manner. The generator is never presented with a real example, but it learns to produce similar examples through feedback from the discriminator.

Another important aspect is the similarity of this setting with the Turing's test. It has an judge which is trying to distinguish between real data from data generated artificially by an intelligent actor. The research on this Adversarial Trainning sets seems really promising.

Variational Autoencoders

The Variational Autoencoders were first presented in [9]. It was inspired by both Autoencoders and Variational Inference. It is composed of two elements: an encoder and a decoder, displayed at figure 2.11. The VAE goal is to maximize the likelihood of the dataset:

Definition 2.2.2. Likelihood under VAE Given a dataset $X = [x_1, ..., x_n]$ of text sentences, the goal of VAE is to learn a parameters θ, such that the likelihood of generating the dataset W is maximal:

$$p_\theta(W) = \prod_{i=1}^{x_n} p_\theta(w_i) \tag{2.13}$$

Taking the probability for one example, it can be expressed as:

$$p_\theta(x_i) = \int p_\theta(x_i, z)dz = \int p_\theta(x_i|z)p_\theta(z)dz = \int p_\theta(z|x_i)p_\theta(z)dz \tag{2.14}$$

Such probability is intractable due to the integral, because the z is normally a vector in a multidimensional space/ In order to overcome it, the VAE also learns a distribution $q_\phi(z|w_i)$ and uses value sampled from those distribution to approximate the likelihood.

Therefore, the VAE tries to maximizes a lower bound of the log-likelihood:

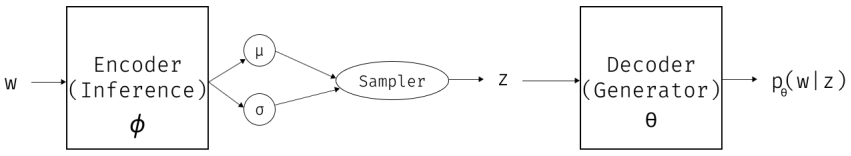


Figure 2.12: VAE with normal as prior and posteriori

Definition 2.2.3. VAE Goal

$$-D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{2.15}$$

Where $p_\theta(z)$ is normally taken as $\mathcal{N}(0, I)$. But still the term D_{KL} – which stands for Kullback-Leibler divergence – is still intractable because calculating involves computing an integral of a distribution without a close form, falling again on integrating over a multi-dimensional domain.

To solve this, the Encoder is constrained to produce parameters of a family of distributions, normally a Gaussian: $\mathcal{N}(\mu, \sigma)$. Therefore, the updated system is in the figure 2.12. With this close form, the lower bound becomes tractable. The final form becomes, where L is the number of samples from $q(z|x)$):

$$\mathcal{L}(X, \phi, \theta) = \frac{1}{n}\sum_{i=1}^{n}(-D_{KL}(q_\phi(z|x_i)||p_\theta(z)) + \frac{1}{L}\sum_{j=1}^{L}\log p_\theta(x_i|z_i^{(j)})$$

Where the D_{KL} term may be seem as a regularization term which forces the values to be around the 0 vector and the log term is the reconstruction error, since we try to maximize the probability of generating the seem text.

After training, the Decoder may be used as a Text Generating System.

VAE vs GAN

It is important to notice some difference between the GAN and VAE. The VAE uses the examples directly to learn how to generate text, while the GAN learns in a more indirect manner. Also, the VAE allows a mapping $\{x_i, z_i\}$ between sentences in the dataset and the latent variables. This allows for a more guided text generation, where , for example, one can interpolate between a z_i and z_j to generate a combination of the two sentences.

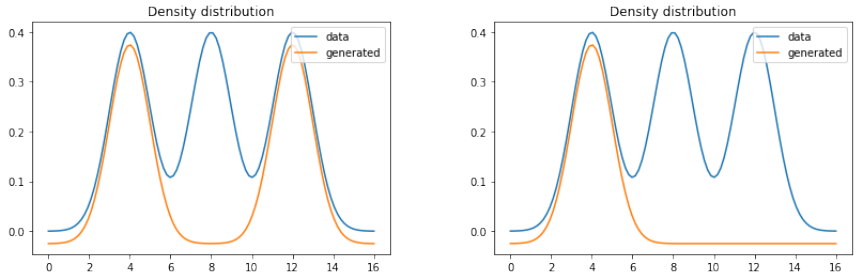


Figure 2.13: Examples of GAN Mode Collapse

Mode collapse and Fidelity

Although being state of the art models, both GAN and VAE suffers from problems to fit real data. The GAN has a problem called Mode Collapse while the VAE has a Fidelity problem. Each problem is well described at [10] and an overview is given below.

GAN and Mode Collapse

The Mode Collapse consist in the collapse of the diversity of the generative model. In other words, it concentrates itself in generating a few realistic examples. This is explained by the optimization objective of the GANs when the discriminator is fixed and optimal:

$$KL(P_G||P_{\text{data}}) - 2JS(P_{\text{data}}||P_G)$$

Where JS is the Jensen-Shannon distance which is constant for optimal discriminator. In this context, if we have $P_G(x) \to 0$ and $P_{data}(x) \to 1$, the KL term tends to 0. While with the opposite: $P_G(x) \to 1$ and $P_{data}(x) \to 0$ means $KL \to \infty$.

In the first case, the generator is **not** covering the realistic data in the dataset. While, in the second case it is generating data not similar to the real dataset. Due to the values of the KL term for each case, the generator is penalized for generating data not similar to the dataset, but it is not penalized for not generating real data. Therefore, it tries to generate few, realistic examples, since not covering the full dataset diversity is not penalized. It could, for example, learn the following distributions shown in figure 2.13.

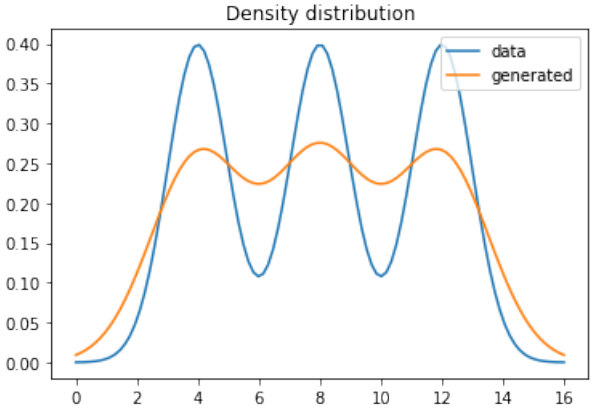


Figure 2.14: VAE Fidelity problem

VAE and Fidelity Problem

The VAE, on the otherhand, explicit forces the distribution to be as diffusive as possible, but this influences its ability to produce realistic examples. In the image context, it normally means obtaining blurred images. One possible learned distribution is shown in figure 2.14.

Combining VAE and GAN

There is a proposition of combining VAE and GANs in something called the VAE-GAN model. Since the VAE decoder may act as the Generator for GAN setting, it gives the possibility of training it as a GAN. This model was proposed at [11], but they applied it only to images.

Chapter 3

Methodology

3.1 Datasets and baselines

3.1.1 Datasets

As any deep learning model, it is necessary to use dataset for training it. The choice of the dataset has a big impact both in the quality of the model and in the time that it takes to be trained. With that in mind, three textual dataset were used for training the models: ImageCOCO [12] and Cranfield Collection.

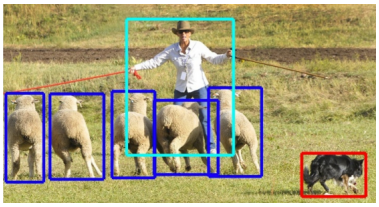
The factors took into account when choosing the datasets was their complexity, number of examples and the availability of baselines to compare our model to. Also, the possibility of using it for the proposed applications.

ImageCOCO

The Common Objects in Context, COCO, is a dataset presented by Microsoft Research at [12]. It consist of several images with related tasks such as Image Classification, Object Localization and Semantic Segmentation. In 2015, a new task was presented as a challenge: Image Captioning available at [13]. Since this task was evaluated by human judges, this generated a great number of high quality short-length text. Figure 3.1 displays some of those tasks.

Through out this work, it will be used the subset of ImageCOCO presented by the TexyGen: A Benchmarking Platform for Text Generations Models at [5]. It consist of $20,000$ samples drawn from the original dataset. Table 3.1 presents some examples.

(a) Image Classification



(b) Object Localization

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

(c) Image Caption

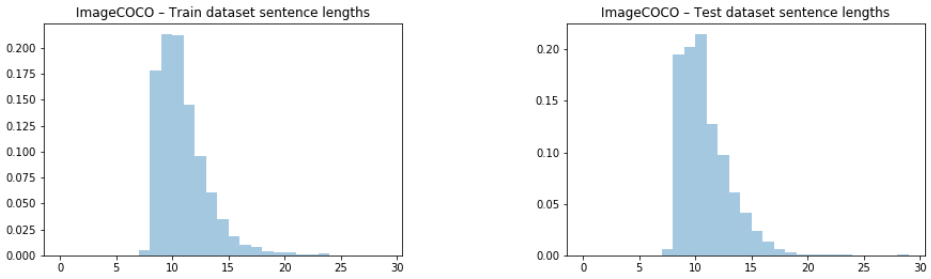
Figure 3.1: COCO tasks

Table 3.1 Examples of captions in ImageCOCO

Examples:
a bicycle replica with a clock as the front wheel.
a bathroom with a sink and shower curtain with a map print.
a cat eating a bird it has caught. a small box filled with four green vegetables.
little birds sitting on the shoulder of a giraffe.

This dataset was chosen due to the availability of baselines for several models and due to it small size, allowing for a rapid iteration and testing of different architectures.

One important feature is its homogeneity, it does not only have very similar phrase structures as in the examples shown, but also homogeneous length. Figure 3.2 shows the distribution of the length of the sequences for train and test dataset. It can be notice that around 70% of the length concentrates between 9 and 12.



(a) Train data

(b) Test data

Figure 3.2: ImageCOCO Lengths histogram

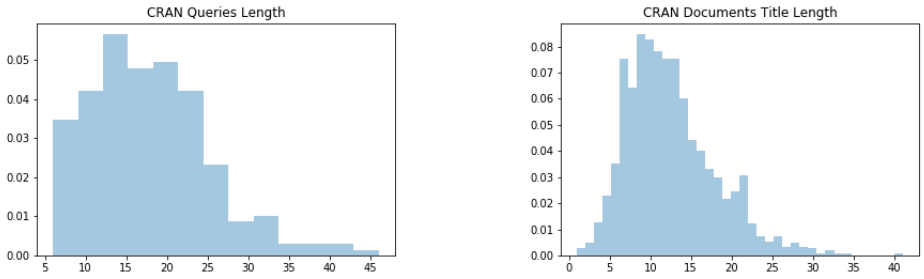
Cranfield Collection

The Cranfield Collection [14] is a small collection for Information Retrieval tasks. It contains 255 textual queries and 1400 documents. Table 3.2 shows an example of a query and a document. The ".I 001" line indicates that the relevant document is the document with ".I 1" id.

Table 3.2 Cranfield Colleaction Examples

Query example:	**Document example:**
.I 001 .W what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft .	.I 1 .T experimental investigation of the aerodynamics of a wing in a slipstream . .A brenckman,m. .B j. ae. scs. 25, 1958, 324. .W <body hidden due to size>

For this work, the Text Retrieval will be used over the title, because the query and title have similar lengths between them. Also, they have similar length to ImageCOCO phrases. In further works, one might explore the effect of the longer phrases in the VAE or the what happens when learning more heterogeneous datasets, that is, with both short and long phrases. Figure 3.3 displays the distribution of lengths. It is more diverse than ImageCOCO, but the most examples are in the same region as ImageCOCO.



(a) Queries (b) Documents

Figure 3.3: Cranfield Lengths histogram

To perform the VAE training, it is necessary to have both train and test data. To do so, 10% is separate from both queries and documents title to compose the test dataset.

3.1.2 Baselines

As mentioned in the **Related Work**, the main metrics for Text Generation tasks are: SelfBLEU and BLEU, with grams going from 4 to 5. Besides that, in the VAE context, the metrics of Perplexity, KL loss and NLLoss are important too, because they measure the quality of the reconstruction performed by the VAE.

The following table lists the available baseline for some GAN text generation models applied to the ImageCOCO captions dataset published at [5].

Table 3.3 GANs BLEU metrics from TexyGen

	SelfBLEU@4	SelfBLEU@5	Train BLEU@4	Train BLEU@5
SeqGAN	0.670	0.489	0.530	0.348
MaliGAN	0.606	0.437	0.482	0.312
RankGAN	0.762	0.618	0.601	0.414
LeakGAN	0.848	0.804	0.660	0.470
TextGAN	0.804	0.746	0.596	0.523

It is important to know that the smaller the SelfBLEU is better, since it indicates a better diversity. While a higger BLEU regarding the test and training dataset indicates a better similarity to the dataset, therefore a better fluency.

A BLEU and SelfBLEU Target

As pointed before, the higher the BLEU and the lower the Self-BLEU the better. Following this logic, and knowing that those quantities vary between 0 and 1, the perfect model would be one with (SelfBLEU, BLEU) = $(0, 1)$.

This target seems way to impossible to be hit. Given that, how could the SelfBLEU and BLEU metrics can be quantified? What are good values for that? The big advantage of BLEU is it correlation with human judgment, but beside that, it is not clear how this metric may be interpreted. And it may be influenced by several factors: the reference dataset, the number of references and the number of candidates.

Given this context, this thesis would like to propose a novel target for BLEU and SelfBLEU metrics: the train dataset metrics. The main reason for this choice is the quality of the train data. Since most of those datasets are normally produced by humans, they are real representation of world data. Therefore, their metrics represents a baseline based on real example.

And since that any machine learning model is dependent of the train data, this presents a normalization regarding the dataset. Take for example two models:

- A VAE trained with the ImageCOCO with BLEU@5 of 0.30.
- A VAE trained with the Cranfield dataset with a BLEU@5 of 0.23.

Using the current evaluation metric, one could argue that the first model is more fluent than the second. But the ImageCOCO dataset is a way more restricted dataset – consisting of image captions. While the Cranfield dataset is more diverse. By giving, also the metrics of the dataset, the information is better contextualized and more informative.

For example, the ImageCOCO dataset presents the following metrics:

Table 3.4 ImageCOCO BLEU metrics

	SelfBLEU@4	SelfBLEU@5	BLEU@4	BLEU@5
ImageCOCO	0.218	0.134	0.531	0.421

To calculate those metrics, 256 samples were randomly taken from the dataset and other 500 samples were randomly taken to work as references.

Comparing this result with the baselines presented before, no model is as good as the train dataset.

Finally, it is interesting to discuss what a model with BLEU@5 bigger than 0.531 would mean. It would be erroneous to say that this model is more fluent than the dataset given that the dataset is human generated and curated. Not because, it is expected for human to be better, but because the definition of fluency in language is extremely correlated with human judgement. Therefore, the alternative is to consider that the model is repetitive regarding the dataset, that is, it overuses patterns presents in the dataset.

In conclusion, the proposition is to calculate and use the train dataset BLEU and SelfBLEU scores as baseline/target for the text generation models. The two reasons is to provide more context to the Text Generation metrics and quantify those metrics, avoiding the over-optimization – such as growing the BLEU way above the train dataset metric.

3.2 Proposed models

3.2.1 Preliminary: State-of-the-art techniques

Two techniques are promising in the improvement of the performance of the VAE's. One is a general technique called Aggressive Training, while the other is specific for sequence processing which is Length-Aware Decoding and Bag-of-Word (BoW) loss. Below, each technique is described in detail.

Encoder Aggressive Training

One common problem of training VAE's in the NLP context is what is called Posterior Collapse. It consist in the following condition $q_\phi(z|x) = p_\theta(z|x) = p(z)$, in other words, in both the inference model (encoder) and the real model (decoder), the x and z become independent.

The presence of this problem in NLP context can be intuitively explained by the discreteness of this domain. Since the data consist in sequence of words, the decoders are recurrent neural networks – auto-regressive models. Therefore, this models can learn to code relationships between words. For example, it can learn that an "a" is normally followed by a noum, and therefore give more probability to nouns in the vocabulary. The possibility of deriving those relationships may give the network possibility of improving itself without deriving any information from the encoder and collapsing the distributions to bring to 0.0 the KL term.

Another factor is the loss function used. To explain it, it is necessary to use the following alternative form of the loss function:

$$\mathcal{L}(X, \phi, \theta) = \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \tag{3.1}$$

The term $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ forces the true posterior and the inference posterior to agree as distributions. While the term $\log p_\theta(x)$ forces x and z to be correlated. If the D_{KL} terms overwhelms the other term, q_ϕ and p_θ will agree before the network learns the relation between z and x, this mean that they will crash into the prior, that is $q_\phi(z|x) = p_\theta(z|x) = p(z)$.

One of the proposed solution for this problem is to enforce the dependence between the x and z by training only the encoder for some epochs. This was proposed at [15]. By doing that, it forces the model to improve only by deriving information from the encoder, while the decoder is fixed. Below is the training algorithm proposed by them, remembering that ϕ is the encoder parameters and θ is the decoder parameters.

Algorithm 1 VAE training with controlled aggressive training

$\phi, \theta \leftarrow$ Initialize parameters
aggressive $\leftarrow$ TRUE
repeat
  if *aggressive* **then**
    repeat
      $X \leftarrow$ Random data minibatch
      Compute gradients $g_\phi \leftarrow \Delta_\phi \mathcal{L}(X; \phi, \theta)$
      Update ϕ using gradients g_ϕ
    until *convergence*
    $X \leftarrow$ Random data minibatch
    Compute gradients $g_\theta \leftarrow \Delta_\theta \mathcal{L}(X; \phi, \theta)$
    Update θ using gradients g_θ
  else
    $X \leftarrow$ Random data minibatch
    Compute gradients $g_{\phi,\theta} \leftarrow \Delta_{\phi,\theta} \mathcal{L}(X; \phi, \theta)$
    Update ϕ, θ using gradients $g_{\phi,\theta}$
  end if
until *convergence*

This unbalance in the training between the Encoder and the Decoder improves the performance of the VAE for Natural Language Processing, but makes the training more costly.

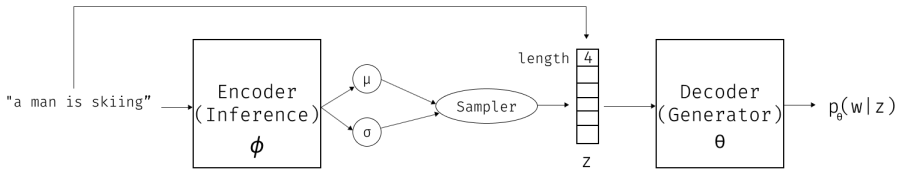


Figure 3.4: VAE Length-Aware

Length-Aware Decoding and Bag-of-Word Loss

Other two techniques to improve quality of VAE is having Length-Aware decoder and adding a Bag-of-Words(BoW) Loss over the latent variable. The first impacts the overall structure of the model and also impacts the sampling of the latent variable, the latter, on the other-hand, is a technique to improve encoding quality. Those techniques were explored in conjunction at [16] where they used VAE with controlled length to generate summaries of texts. Below, both are explained in details:

Length-Aware Decoding

In the pure VAE setting, information is passed only through the latent variable z. The Length-Aware Decoding consist in adding an input to the decoder that consist in the expected length of the sentence. Suppose it must decode the following sentence: "a man is skiing", in the Length-Aware decoding, it would be done like displayed in figure 3.4.

Obtaining the length as feature is something that can be done both on demand, through a computational unit that would count the number of inputs that the encoder received or simply by pre-processing the input.

In either way, this technique affects the architecture of the VAE. For generating new sentences, it is necessary not only to generate a latent variable from the prior, but also generating a length from some distribution, as described in figure 3.5.

This is something that may affect the quality measures, because there are several ways of generating a length. On the other-hand, this allows for a more direct text generating and may enable the usage of the VAE is other application as explored in future sections.

It is important to make explicit how the Decoder would use the length in the decoding phase. The most common way of using it is through a technique called positioning decoding where at each step the expected length is an input of the RNN. Figure 3.6 shows how this work for a sentence with length 4.

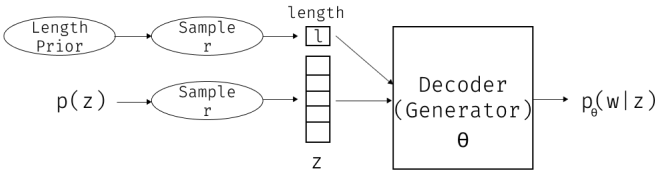


Figure 3.5: VAE Length-Aware Decoding

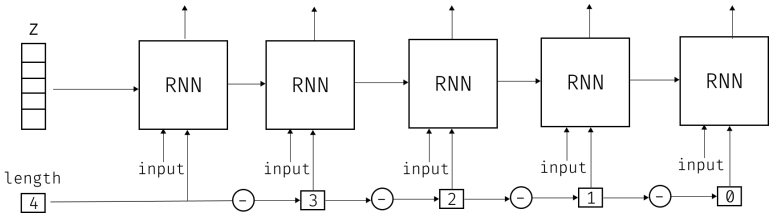


Figure 3.6: Decoding with length as a feature

Bag-of-Words Loss

This technique has the goal of improving the content representation of the latent variable. As explained before, due to the recurrent nature of the Decoder, it may end up learning just the most common succession of words. To avoid this, a new layer that tries to predict the bag-of-word vector is added together with a new loss regarding this prediction. This can be very effective in improving the reconstruction of the phrases. But, by adding a new loss term, the KL loss may be eclipsed and the continuity of the latent space compromised.

In more details, a bag-of-word vector is a vector with the dimension of the vocabulary with 1 in the position of the words that appear on the phrase and 0 in the position of the word that do not. The prediction vector must give high probability to the words that appear and low probability for the words that do not. For example, given the following toy vocabulary, the bag-of-word for the two phrases is presented at table 3.5:

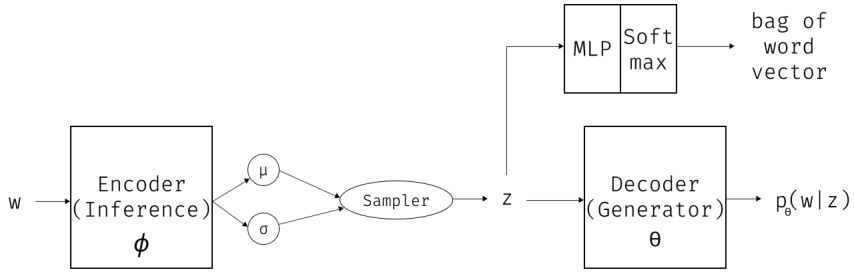


Figure 3.7: Bag of Word

Table 3.5 Example of Bag of Words encoding

Vocabulary	**Phrases:**
Hello: 0	"Hello world" $\rightarrow [1, 1, 0]$
world: 1	"Bye world" $\rightarrow [0, 1, 1]$
Bye: 2	

Figure 3.7 shows the new architecture. It is possible to see that this new architecture forces the latent to encode information about the word present on the phrase.

3.2.2 Preliminary: Disentangled representation

This last technique is based on two principles. The first consist on training a network on a easier task and then leveraging that network to help on a more complex task. The second consist in enforcing content representation on the latent variable by breaking it into two vectors: a grammatical vector and a content vector.

The easier task is predicting the grammatical form of the phrase. For example, given the phrase "a man is skiing", it can be grammatically represented as "DET NOUM VERB VERB", where **DET** means determinant ("a", "the", etc). In this case, the size of the vocabulary is the size of the Part-of-Speech (POS) vocabulary. For the well-known SpaCy library available at [17], the POS-vocabulary consist of 17 classes listed below, plus the start of sentence, end of sentence and padding tokens, respectively "<s>", "</s>" and "<pad>":

Table 3.6 Part-of-Speech classes on SpaCy

Acronym	Part-of-Speech tag	Examples
ADJ	Adjective	big, old, green
ADP	Adposition	in, to, during
ADV	Adverb	in, to, during
AUX	Auxiliary	in, to, during
CONJ	Conjunction	in, to, during
CCONJ	Coordinating Conjunction	in, to, during
DET	Determiner	in, to, during
INTJ	Interjection	in, to, during
NOUM	Noum	in, to, during
NUM	Numeral	in, to, during
PART	Particle	in, to, during
PRON	Pronoun	in, to, during
PROPN	Proper Noum	in, to, during
PUNCT	Punctuation	in, to, during
SCONJ	Subordinating Conjunction	in, to, during
SYM	Symbol	in, to, during
VERB	Verb	in, to, during
X	Other	to swim, singing, etc

The idea is to train a VAE in the Part-Of-Speech reconstruction task and then use this VAE as input for the more general VAE. Hopefully, this will not only improve performance, but will allow the other VAE to encode more content instead of grammatical features.

The overall-structure is represented in figure 3.8. It is important to understand the impact of this architecture on the latent space. First, as the latent space is doubled, it is expected for this to have a higher KL loss term. Second, this hopefully allows us to have more control over the text generation process, for example, the content vector or the grammar vector may be kept fixed while the other is disturbed. This, therefore, provides also a good manner of qualitative validating the effectiveness of this technique.

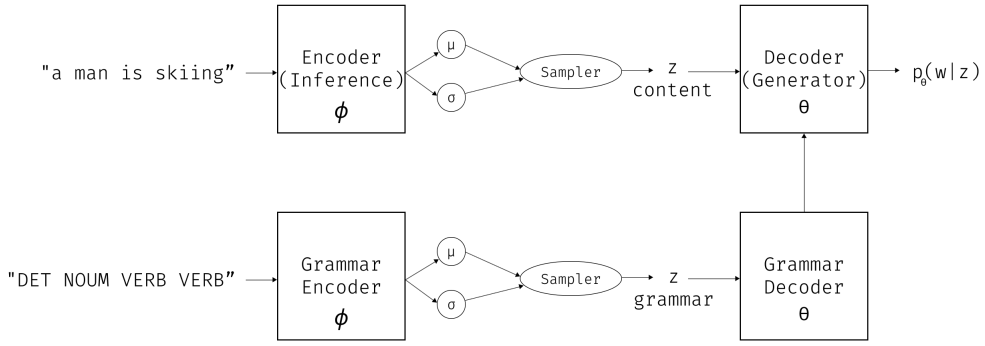


Figure 3.8: VAE with Disentangled representation

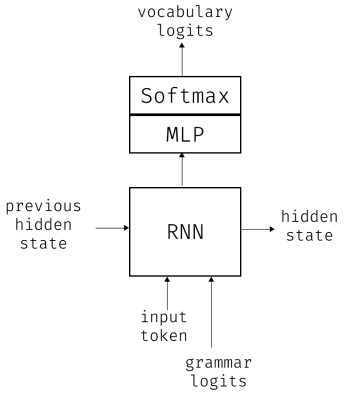
Finally, the figure 3.8 does not make explicit how the Content VAE Decoder uses the output of the Grammar Decoder. For each step. the decoder outputs a stochastic vector with the size of the POS vocabulary – 19 classes. There are several way that the Content VAE Decoder may use this vector. Three are explored. The first consist in adding it as input to the RNN, the second consist in adding it as input of the predictive layer and the last consist in scaling it to the size of the vocabulary and multiplying it with the output of the predictive layer. Figure 3.9 shows the three possibilities that will be explored.

3.2.3 Explored models

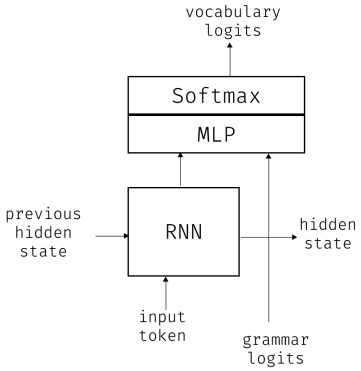
In the following subsections, the trained models are presented with their respective diagram. This models will be compared in the results section and according to each architecture, they will be chosen to be used on the suggested applications.

VAE

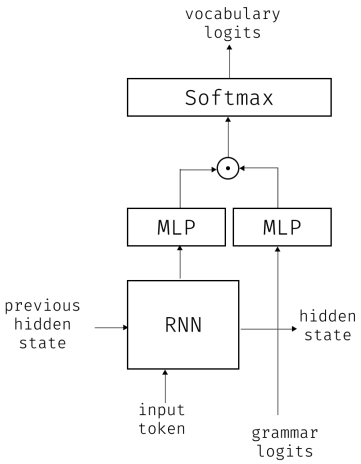
This model consist in the VAE on its most basic form. No aggressive training or architecture modification is made. It is used both as a baseline for the other models and as a implementation validation of our model. The proximity of its metric to the presented baselines on the previous sections gives an idea if it was implemented correctly. Figure 3.10 shows its architecture.



(a) Logits@Input: Grammar logits as RNN input



(b) Logits@Output: Grammar logits as Predictive layer input



(c) Logits@Dot: Grammar logits interacting directly with Content logits

Figure 3.9: Three explored ways of using Grammar logits as input

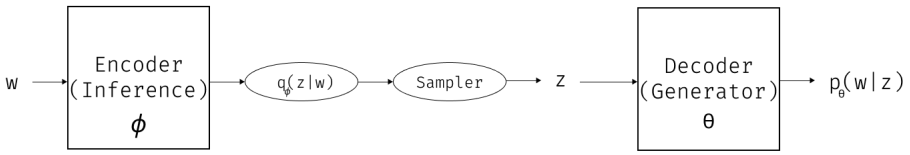


Figure 3.10: VAE architecture

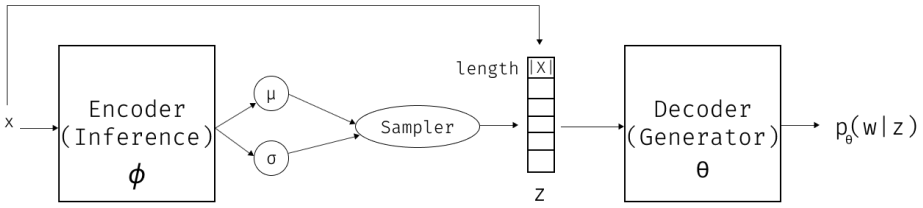


Figure 3.11: VAE Length-Aware Architecture

Aggressive VAE

This model consist in the one proposed at [15], it is therefore a validation and a baseline model to which the others will be compared to. There is no change in the architecture regarding the previous model, it just changes the algorithm used for training.

Aggressive Length-Aware VAE

This is the first novel model proposed in this work. It is a novelty, because combines two state-of-the-art techniques which have the potential of improving some aspects of the VAE performance. Its architecture is presented at figure 3.11

Aggressive Length-Aware + BoW Loss VAE

This model on the other-hand, combines all the first three techniques and therefore it is one of the most complex one. Its architecture is described at figure 3.12.

Double-level Aggressive VAE

This is a baseline model for the VAE Disentangled. The reason to add this double-leval, it is the dimension of the latent space in the VAE Disentangled. Since it has a latent dimension two times bigger then the one for the other models, it has more computing power and may therefore work better. By training a Double-level VAE, that is a VAE with RNN with 2-layers, a baseline with same computing power is used. Its architecture is presented at figure 3.13.

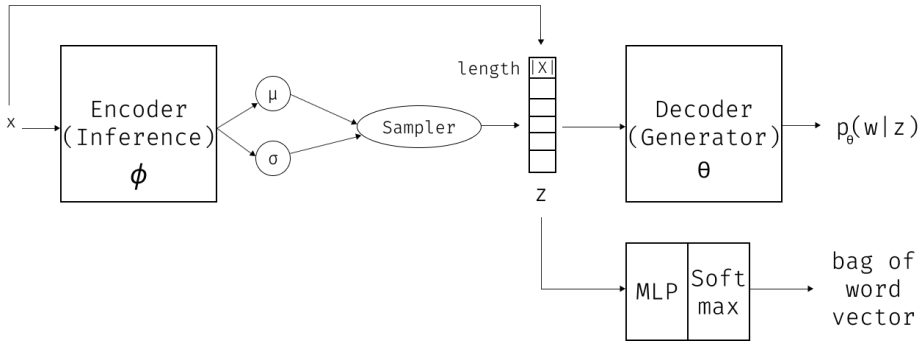


Figure 3.12: VAE Length-Aware + BoW Architecture

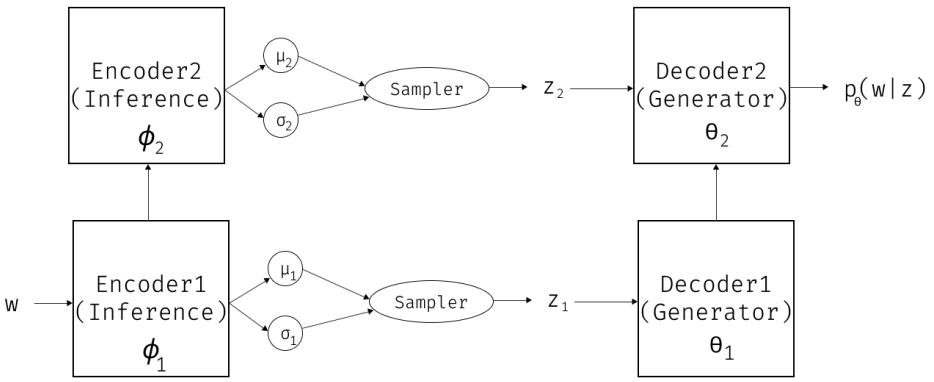


Figure 3.13: Double-level VAE Architecture

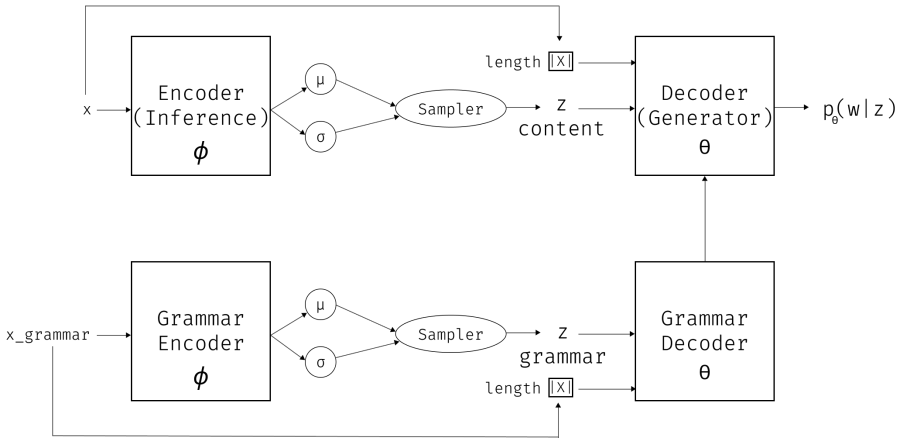


Figure 3.14: VAE Disentangled + Length-Aware Architecture

VAE Disentangled + Length-Aware

This is the most complex model propose, because it combines almost all the proposed techniques. It does not use technique is the Bag-of-Word Loss, because the disentanglement of representation has a similar goal as this technique: it tries to force the (content) representation to encode more information about the words.

It sets high expectations regarding its property – best performance and two unrelated representations, which must be validate. To validate the performance, the standard metrics will be used. The disentanglement of the representations will be validated qualitative in the following manner: the grammar latent vector will be fixed and several content latent vector will be sampled and vice-versa. According to the latent vector being sampled, the feature that the vector represents is expected to vary, while the other is supposed to be fixed. Its architecture is presented at figure 3.14. It also uses Aggressive Training.

3.3 Proposed applications

One of the big questions regarding Text Generation systems is their application to more concrete tasks such as Text Completation, Question Answering, Information Retrival, etc. This section explores the application of the VAE to some of those tasks. It uses the VAE as it is or with really small modification to the task. The goal is to identify possible applications of the VAE which can be further improved in order to work.

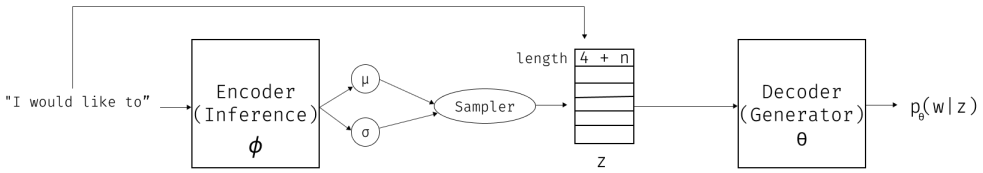


Figure 3.15: Predicting next words with VAE (n is the number of words)

3.3.1 Text Completion task

Writing assistants is one of the domain that has attracted lot of attention. From Word Grammar Correcting system to recently presented auto-complete feature on Google's Gmail service, this kind of system is becoming more common every day.

Since it is under the umbrella of the Natural Language Processing, this can be a interesting field to apply the VAE. If the general problem is posed as a language model where the next word or the next few words must be predicted, the VAE may be adapted to do so. More precisely, the length-aware VAE can be a good fit: one can encode the sentence until the known words, and them decode it by adding one to the length used as input for the decoder. This allows the network to control the number of predicted words. Specifically in this context, the model can be used online and retrained as the user generates more content.

To visually evaluate the quality of the system, phrases from the test dataset will be capped from 1 to 3 words and our system will be run to predict the same quantity of capped words. Figure 3.15 presents the operation in this task, where n is the number of next words, hopefully, this system will generate relevant suggesting such as: "I would like to set up a meeting" or "I would like a cafe". And to give a quantitative idea of the precision of the model, the Cross Entropy and the Accuracy of the system will be computed for predicting 1, 2 or 3 words.

Since this is similar to a reconstruction task, it is reasonable to expect that the model with less CrossEntropy loss will be the best. The tested models are: Aggressive VAE and Aggressive Length-Aware + BoW Loss VAE.

The ImageCOCO were used due to its low-level complexity and fast training, but also because it has a very homogeneous length – centered around 10, and phrases really similar.

3.3.2 Text Retrieval task

This task consist in retrieving related documents according to the query presented. The possibility of using VAEs to perform such task depends on the quality of the latent space representation. If the VAE performs well, its latent space representation is good and similar phrases will be grouped together. The measure of similarity to be used will be the distance between the latent representation of the document ant the query, computed using cosine similarity.

All VAE may be tested in this task, but the ones chosen were: Length-Aware + BoW Loss VAE and VAE Disentangled + Length-Aware. For the second, only the content representation will be used to query the documents. This selection were made due to the fact the architecture of those models favors a good content representation in the latent space, while in a "pure" VAE, the latent representation is responsible for more than just the topics in the sentence.

The main element of the VAE to be used is the encoder. Any element that should be in the search space is processed by the encoder generating a representational mean μ. After that, the selected query is used to order this documents based on the distance. The distance metric is the cosine similarity which consist in:

$$similarity = cos\theta = \frac{\mu_1 \mu_2}{||\mu_1||||\mu_2||} \tag{3.2}$$

The evaluation will consist in quantitative evaluation over the Cranfield Collection dataset. In the second, the question will be used to query the documents and the accuracy and Mean Reciprocal Rank will be computed to the dataset.

Chapter 4

Experimental setup

4.1 Training methodology

In other to select a model that fits well the test dataset and avoids over fit, a validation dataset is used. At the end of each epoch, the model is evaluated over the validation dataset. If the loss is not improved after two consecutive epochs, the learning rate is reduced and the system is brought back to the smallest loss, that is, the best model insofar. This dynamic is represented in figure 4.1.

If the learning rate decays 5 times, the training is stopped. This dynamics starts after a specified epoch, that is, the system has an warm-up period where it is updated freely without checking the performance in the validation dataset.

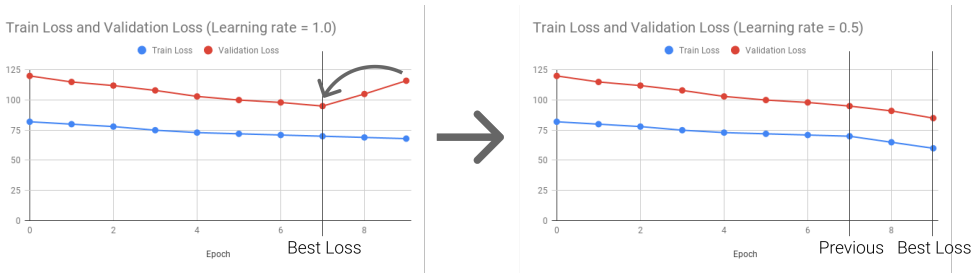


Figure 4.1: Learning rate annealing

4.2 KL Loss Annealing

The KL term is a regularization term that makes the posteriori distribution to be close to the prior distribution. This terms affects the continuity of the Latent Space which affects the quality of the text generation.

Optimizing this term too early, takes the ability of the network of learning high-level features of the text and may cause posteriori collapse. Therefore this term is annealed for a few epochs.

4.3 Training Algorithm

The listing 2 details the training algorithm in a high-level, giving a full understanding of the training dynamic.

4.4 BLEU and SelfBLEU Calculation

In other to compute the BLEU and the SelfBLEU, it is necessary to generate a few sample from the VAE. The number of samples generated for the computation were 256 and a total of 500 samples were taken randomly from the training dataset and test dataset. The same process was used to calculate the metrics of the train dataset, which will be used as baseline. The obtained numbers are:

Table 4.1 ImageCOCO BLEU metrics

	Self 4-BLEU	Self 5-BLEU	Train 4-BLEU	Train 5-BLEU	Test 4-BLEU	Test 5-BLEU
COCO	0.218	0.134	0.531	0.421	0.218	0.134

Those values will be the ones used to compare and evaluate the proposed models.

4.5 Models Dimension

Every model has basically the same dimension regarding the common layers. The latent space of every model is 32. The hidden space, that is the dimension of the state in the RNNs, are 1024. And every decoder has a dropout layer

Algorithm 2 Full VAE training algorithm

model ← Initialize model
learning_rate ← Initialize learning rate
KL_weight ← 0.0
KL_anneal_rate ← Initialize KL Anneal rate
decay_count, *epoch* ← 0, 0
number_of_decay ← 0
train_dataset, *validation_dataset* ← Load data
best_loss ← ∞
repeat
    Train *model* with *train_dataset*
    loss ← *validate*(*model*, *validation_dataset*)
    KL_weight ← *max*(1.0, *KL_weight*)

    if *best_loss* > *loss* **then**
        best_loss ← *loss*
        number_of_decay ← 0
        Save best model so far
    else
        if *epoch* > 15 **then**
            number_of_decay ← *number_of_decay* +1
        end if
    end if

    if *number_of_decay* = 2 **then**
        number_of_decay ← 0
        model ← Load best model
        learning_rate ← $\frac{learning\_rate}{2}$
    end if
until *decay_count* = 5

on input and output of 0.5. Finally, the embedding layer is of size 512. Those values were taken from the models at [15].

Chapter 5

Results for Explored Models

In this section, the results for each explored model is presented together with some specific analysis. The metrics mentioned before and some qualitative examples are displayed both for the reconstruction and generation task.

Finally, the last subsection presents a summarizing table together with the baselines and an analysis.

5.1 VAE

This model consist in the most simple VAE and as it will be shown, presents a model collapse problem. The training metrics resulting are presented in table 5.1.

Table 5.1 Training results for VAE

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.2794

It is important to notice the low value of the KL-loss indicating a posteriori collapse, since very little information is provided in the encoding phase. This suspicion is confirmed by the examples for the Reconstruction Task presented in table 5.2 and table 5.3.

Table 5.2 Reconstruction Task examples in train dataset for VAE

Train dataset
Expected: a doorway looking into a demolished bathroom **Actual:** a man and woman sit on a bench in the middle of the street
Expected: two people riding horses along a trail **Actual:** a man is sitting on a bench next to a bike with a bike next to it
Expected: two cosplaying people pose for a photo **Actual:** a man in a suit and tie is standing in front of a motorcycle
Expected: an airplane just landed on the runway **Actual:** a man and woman riding a motorcycle down a street with a woman on the back
Expected: a kitchen with a window some cupboards **Actual:** a man and woman riding a motorcycle on a race track

Table 5.3 Reconstruction Task examples in test dataset for VAE

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a man in a helmet is standing in front of a motorcycle with a woman on the back
Expected: a <unk> bear partly submerged in water **Actual:** a man and woman sit on a bench in front of a building
Expected: some pasta with broccoli <unk> and sauce **Actual:** a man and woman sit on a bench in front of a building with a steeple
Expected: two bears touching noses standing on rocks **Actual:** a man and woman sit on a bench in front of a building
Expected: a statue of a three horse carriage **Actual:** a man and woman riding a motorcycle down a street with a woman on the back

The examples for the Text Generation task also provide this insight. They are presented in table 5.4.

Table 5.4 Generation Task examples for VAE

Generated:
a man and woman sit on a bench next to a bike and a river
a man riding a motorcycle with a woman on the back of it
a man in a blue shirt and a black motorcycle with a red blanket in the background
a man in a suit and tie is standing in front of a motorcycle

This gives a sense of what to expect for the BLEU and Self BLEU. Since the examples are extremely repetitive, but very fluent, one might expect a high BLEU and SelfBLEU, which is the case. The results are presented at table 5.5.

Table 5.5 BLEU metrics for VAE

	SelfBLEU@4	SelfBLEU@5	Train BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579

As discussed before, here is an example of a model of a BLEU higher than the computed BLEU for the dataset. In this case, although very fluent, the examples are extremely reptitive, and the high value of BLEU is not an indicative of high quality, but of repetitiveness.

5.2 Aggressive VAE

This model should present way better results than the previous, since the Aggressive training would avoid the posteriori collapse. The training results are presented at table 5.6.

Table 5.6 Training results for Aggressive VAE

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.27
AggressiveVAE	43.32	5.89	46.11

Therefore, it is reasonable to expect more diversity and a better result in the reconstruction task. The results are presented in table 5.7 and 5.8.

Table 5.7 Reconstruction Task examples in train dataset for Aggressive VAE

Train dataset
Expected: a door way looking into a demolished bathroom **Actual:** a white toilet sitting in a bathroom next to a white sink
Expected: two people riding horses along a trail **Actual:** a giraffe standing next to a bike with a fence in the background
Expected: two cosplaying people pose for a photo **Actual:** two men in a kitchen with a dog in a kitchen with a dog in her hand
Expected: an airplane just landed on the runway **Actual:** a group of people sitting on a snowy slope with a large commercial airplane on the tarmac
Expected: a kitchen with a window some cupboards **Actual:** a large passenger jet taking off from an airport runway with passengers on either side

Table 5.8 Reconstruction Task examples in test dataset for Aggressive VAE

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a young man standing next to a woman in a kitchen
Expected: a <unk> bear partly submerged in water **Actual:** a person that is holding a camera in a bowl with a camera in her hand
Expected: some pasta with broccoli <unk> and sauce **Actual:** a table with multiple monitors and a cup of coffee and papers
Expected: two bears touching noses standing on rocks **Actual:** there are two people riding bikes on a side walk
Expected: a statue of a three horse carriage **Actual:** an old photo of a plane flying in a clear sky

One can also notice the diversity of the generated phrases – displayed at table 5.9 – when compared with the previous example.

Table 5.9 Generation Task examples for Aggressive VAE

Generated:
a man is sitting on a bench next to a black dog on a leash a bathroom with a white toilet , sink , and shower curtain
a large group of sheep in front of a group of motorcycles
a woman in a kitchen with a carton of food and a pot of food
a living room with a television , chair , and a tv on the side

Most of the examples are fluent and somehow related with the input sentence, but there are some problems. Some sentences are repetitive – such as "a living room with a television , chair , and a tv on the side", or incoherent – such as "a person that is holding a camera in a bowl with a

camera in her hand" and finally, the generated phrases show a tendency of been very verbose. Therefore, it is expected a better SelfBLEU, while keeping or reducing the fluency. Those results are presented in table 5.10.

Table 5.10 BLEU metrics for Aggressive VAE

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579
Aggressive VAE	0.743	0.650	0.617	0.484

The closeness of the BLEU@4 and BLEU@5 to the baseline, indicates a better fitness of this model to the dataset. It is also interesting to point the correlation between the SelfBLEU and the Train BLEU: a model with lower BLEU also has a lower SelfBLEU.

5.3 Aggressive Length-Aware VAE

This model makes the Length an explicit feature, and therefore, it is expected to improve the reconstruction task and be less verbose. Also, it creates a discussion about how to sample the length for the Text Generation task. Due to that, in the Text Generation results, three length prior distribution will be explored.

The training results are presented in the table below:

Table 5.11 Training results for Aggressive Length-Aware VAE

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.2794
AggressiveVAE	43.32	5.89	46.11
Length-Aware Aggressive-VAE	34.71	4.60	43.40

Given the value of the Cross-Entropy Loss, the results for the reconstruction task are expected to be better:

Table 5.12 Reconstruction Task examples in train dataset for Aggressive Length-aware VAE

Train dataset

Expected: a door way looking into a demolished bathroom
Actual: a bathroom with a sink and toilet

Expected: two people riding horses along a trail
Actual: two people riding motorcycles and a bus

Expected: two cosplaying people pose for a photo
Actual: two men are standing by a picture

Expected: an airplane just landed on the runway
Actual: an airplane is taking off the runway

Expected: a kitchen with a window some cupboards
Actual: a kitchen with a washing machine ...

Table 5.13 Reconstruction Task examples in test dataset for Aggressive Length-aware VAE

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a small bathroom with a toilet
Expected: a <unk> bear partly submerged in water **Actual:** a giraffe in the snow covered
Expected: some pasta with broccoli <unk> and sauce **Actual:** the desk with many snacks and sprinkes
Expected: two bears touching noses standing on rocks **Actual:** two men are standing on a table
Expected: a statue of a three horse carriage **Actual:** a building with a clock tower

5.3.1 Length Prior Distribution

For a few phrases, the reconstruct phrase is pretty similar to the expected one, showing the improvement. The results for the generated phrases are:

Table 5.14 Generation Task examples for Aggressive Length-aware VAE

Generated:
a white toilet seat
a kitchen with a toilet , sink and a window
a man riding a motorcycle down the street in the city
a bathroom that is very clean

As pointed before, this model raises the questions of which prior to use for sampling length, when generating text. Three options were explored:

normal-fitted prior, discrete-fitted prior and uniform distribution. The last two have the advantage of generating only discrete length values that were present on the dataset. The results are:

Table 5.15 BLEU metrics for different Length Prior Distributions on the context Aggressive Length-aware VAE

	SelfBLEU@4	SelfBLEU-5	Train BLEU@4	Train BLEU@5
Normal	0.720	0.595	0.552	0.433
Discrete fitted	0.772	0.670	0.575	0.459
Uniform	0.795	0.715	0.608	0.482

There are not big variations regarding the prior used. In this thesis, the diversity will be favored, therefore, in the following models were is needed to sample the length, the normal distribution will be used.

Then, the metrics for this model are:

Table 5.16 BLEU metrics for Aggressive Length-aware VAE

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579
Aggressive VAE	0.743	0.650	0.617	0.484
Length-Aware Aggressive VAE	0.720	0.595	0.552	0.433

This models improves by little the previous model, but it approach more the baseline, therefore, it is a good improvement.

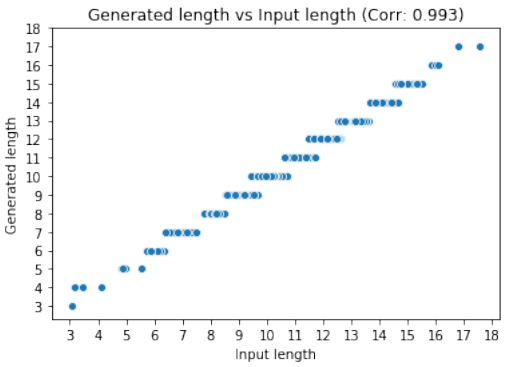
[H]


Figure 5.1: Input length vs Generated length

Length effectiveness

Finally, it is important to evaluate the effectiveness of the length as a feature, to do so, the length of the generated sentence is plot against the input length to check the correlation between the two which can be seen in 5.1.

As one can see in figure 5.1. The input length is extremely correlated with the output length – with a correlation coefficient of 0.993. Therefore, the network learned to use the input length to control the generation.

5.4 Aggressive Length-Aware + BoW Loss VAE

This model adds a new loss with the goal to improve the latent representation and at the risk of obfuscating the KL loss. Therefore, it is expected a bigger KL-loss and a smaller Cross-Entropy loss:

Table 5.17 Training results for Aggressive Length-Aware VAE + BoW Loss

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.2794
AggressiveVAE	43.32	5.89	46.11
Length-Aware AggressiveVAE	34.71	4.60	43.40
Length-Aware Aggressive-VAE + BoW Loss	19.67	20.01	36.45

The reconstruction tasks has the following results:

Table 5.18 Reconstruction Task examples in train dataset for Aggressive Length-aware + BoW Loss VAE

Train dataset

Expected: a door way looking into a demolished bathroom
Actual: a doorway looking into a bathroom

Expected: two people riding horses along a trail
Actual: two people riding horses along a trail

Expected: two cosplaying people pose for a photo
Actual: two old photo of a women pose

Expected: an airplane just landed on the runway
Actual: an airplane on the runway unoccupied

Expected: a kitchen with a window some cupboards
Actual: a kitchen with a window and cupboards

Table 5.19 Reconstruction Task examples in test dataset for Aggressive Length-aware + BoW Loss VAE

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a person standing next to a field
Expected: a <unk> bear partly submerged in water **Actual:** a man in a big kitchen
Expected: some pasta with broccoli <unk> and sauce **Actual:** a modern kitchen with some kind
Expected: two bears touching noses standing on rocks **Actual:** two domestic - animals on a table
Expected: a statue of a three horse carriage **Actual:** a statue of a three horse carriage

For the Generation Task, a worse result is expected. The reason is the high KL term. It indicates that during training, the latent vectors were distant from the prior. While, during the generation they will be close, because they are sampled from it. Since the model will be working on an "unexplored" latent area, the results expected will work poorly.

Table 5.20 Generation Task examples for Aggressive Length-aware + BoW Loss VAE

Generated:
a crowd of people standing at his crowd of people
a woman crosses a motorbike as a woman and a bullhorn
a jet on the wall next to the ocean
a knife on a shiny side car

As expected, the phrases here, although grammatically correct, are very incoherent, correlating words weirdly. The metrics for this model are:

And the metrics:

Table 5.21 BLEU metrics for Aggressive Length-aware + BoW loss VAE

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579
Aggressive VAE	0.743	0.650	0.617	0.484
Length-Aware Aggressive VAE	0.720	0.595	0.552	0.433
Length-Aware Aggressive VAE + BoW Loss	0.446	0.306	0.330	0.225

This is the first model that presents a lower BLEU regarding the COCO baseline. And as we see, it indicates a low level of fluency. But on other-hand, it presents a very low SelfBLEU indicating a better diversity if phrases. Finally, until now, this model is the better on the reconstruction task.

5.5 Double-level Aggressive VAE

As pointed before, this model was chosen due to the computational power of it which is closed to the next proposed models. As the following table can show, just adding a more units – without changing dataset – did not improve the performance.

Table 5.22 Training results for Double-level Aggressive VAE

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.2794
AggressiveVAE	43.32	5.89	46.11
Length-Aware AggressiveVAE	34.71	4.60	43.40
Length-Aware AggressiveVAE + BoW Loss	19.67	20.01	36.45
Double-level Aggressive VAE	84.44	3.64	54.28

The reconstruction tasks has the following results:

Table 5.23 Reconstruction Task examples in train dataset for Double-level Aggressive VAE

Train dataset
Expected: a door way looking into a demolished bathroom **Actual:** a white toilet sitting in a bathroom next to a white sink and a window
Expected: two people riding horses along a trail **Actual:** two people are walking a dog on the side of a road with a crowd of people walking by
Expected: two cosplaying people pose for a photo **Actual:** a group of people sitting on a bench in front of a large church
Expected: an airplane just landed on the runway **Actual:** an airplane is parked on the runway with a service truck parked in the background in the background
Expected: a kitchen with a window some cupboards **Actual:** a kitchen with a stove , microwave , and wooden cabinets with a large window in the background

Table 5.24 Reconstruction Task examples in test dataset for Double-level Aggressive VAE

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a large passenger jet flying through a cloudy sky in a cloudy sky
Expected: a <unk> bear partly submerged in water **Actual:** a man standing in a kitchen with a dog watching her hands and him is in the background
Expected: some pasta with broccoli <unk> and sauce **Actual:** an airplane with people on the ground and waiting at an intersection for a city street with two cars
Expected: two bears touching noses standing on rocks **Actual:** a group of people sitting on a bench in front of a small church
Expected: a statue of a three horse carriage **Actual:** a small bathroom with a toilet , sink , and a shower stall , with a shower , and a shower stall

For the Text Generation task:

Table 5.25 Generation Task examples for Double-level Aggressive VAE

Generated:
a person riding a motorcycle down the middle of a street with a red light in the background
people are walking across the street while a dog licks a dog that is walking the bike
an orange and white cat sitting on top of a wooden table near a microwave
the large aircraft is on the runway at an airport

It is interesting to note how verbose this model is. Both in the reconstruction task and generation task, it generates phrases with very high length.

The results for the metrics are:

Table 5.26 BLEU metrics for Double-level Aggressive VAE

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579
Aggressive VAE	0.743	0.650	0.617	0.484
Length-Aware Aggressive VAE	0.720	0.595	0.552	0.433
Length-Aware Aggressive VAE + BoW Loss	0.446	0.306	0.330	0.225
Double-level Aggressive VAE	0.764	0.690	0.605	0.489

5.6 VAE Disentangled + Length-Aware

This model uses a network that tries to predict the POS and other that tries to predict the phrase. To construct this model, we first train the model on the POS task and after that we use this model to construct the more general one.

The results for the Grammar model are:

Table 5.27 Training results for Grammar VAE

Perplexity	KL-loss	Cross-Entropy Loss
1.87	4.90	7.72

The low perplexity value indicates a very good success in predicting the right token.

To construct the general model, three model were suggested to integrate the grammar model: Logits@Input, Logits@Output and Logits@Dot. Before presenting the qualitative examples of this model, below is the result for each option:

Table 5.28 Training results for all options of VAE Disentangled + Length-aware

	Perplexity	KL-loss	Cross-Entropy Loss
Logits@Input	33.74	4.74	43.50
Logits@Output	34.86	4.33	43.90
Logits@Dot	47.55	2.16	47.74

In order to compare the KL-loss of this model to the KL-loss of other models, one must add the Grammar Model KL-loss, in this case: 4.90. This is done in the general model tables.

The metrics for the text generation:

Table 5.29 BLEu metrics for all options of VAE Disentangled + Length-aware

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
Logits@Input	0.628	0.468	0.434	0.300
Logits@Output	0.536	0.383	0.353	0.232
Logits@Dot	0.630	0.478	0.339	0.221

The Loigits@Dot is the worst option, because it has a very low BLEU and a very high Self BLEU. While the others presents a trade-off between high BLEU and high Self BLEU, due to the correlation of this two models. In this case, we will opt for the Logits@Input due to its BLEU values closer to the dataset baseline.

Therefore, the results for the model in question compared to others are:

Table 5.30 Training results for VAE Disentangled + Length-Aware

	Perplexity	KL-loss	Cross-Entropy Loss
VAE	71.68	0.032	52.2794
AggressiveVAE	43.32	5.89	46.11
Length-Aware AggressiveVAE	34.71	4.60	43.40
Length-Aware AggressiveVAE + BoW Loss	19.67	20.01	36.45
Double-level Aggressive VAE	84.44	3.64	54.28
VAE Disentangled + Length-Aware	33.74	9.64	43.50

The examples for the Reconstruction Task:

Table 5.31 Reconstruction Task examples in train dataset for VAE Disentangled + Length-Aware

Train dataset
Expected: a door way looking into a demolished bathroom **Actual:** a bathroom with a white toilet
Expected: two people riding horses along a trail **Actual:** two people are on the ground
Expected: two cosplaying people pose for a photo **Actual:** a kitchen with white and a ceiling
Expected: an airplane just landed on the runway **Actual:** an airplane is at the airport terminal
Expected: a kitchen with a window some cupboards **Actual:** a kitchen with a window above it

Table 5.32 Reconstruction Task examples in test dataset for VAE Disen-tangled + Length-Aware

Test dataset
Expected: a <unk> herself to catch a frisbee **Actual:** a man is being flown by
Expected: a <unk> bear partly submerged in water **Actual:** a large bathroom with two stuffed animals
Expected: some pasta with broccoli <unk> and sauce **Actual:** this bathroom with toilet paper and toilets
Expected: two bears touching noses standing on rocks **Actual:** two people sitting on the side
Expected: a statue of a three horse carriage **Actual:** a close up of a car

For the Generation Task:

Table 5.33 Generation Task examples for VAE Disen-tangled + Length-Aware

Generated:
a kitchen with a kitchen is white
a black and bathroom is on to
a car in a a city at a city street
a motorcycle a motorcycle in front next to a bicycle

Finally, the metrics compared to other models:

Table 5.34 BLEU metrics for VAE Disentangled + Length-Aware

	SelfBLEU@4	SelfBLEU@5	Tain BLEU@4	Train BLEU@5
COCO	0.218	0.134	0.531	0.421
VAE	0.984	0.979	0.677	0.579
Aggressive VAE	0.743	0.650	0.617	0.484
Length-Aware Aggressive VAE	0.720	0.595	0.552	0.433
Length-Aware Aggressive VAE + BoW Loss	0.4464	0.3061	0.330	0.225
Double-level Aggressive VAE	0.764	0.690	0.605	0.489
VAE Disentangled + Length-Aware	0.628	0.468	0.434	0.300

Therefore, this model presents reasonable results regarding BLEU and SelfBLEU. One question that arises is the effectiviness of the Grammar representation.

Grammar and Content Representation Effectiveness

To analysis it, the sensibility of the output regarding noise added to the latent variable will be tested in a qualitative manner.

The procedure consist in adding noise to the latent representation of a phrase. The Noise values vary from $[0.01, 0.1, 0.5, 1.0]$ and we disturb one

representation at a time – content and grammar. The result for the first 3 examples in the train dataset are below:

Table 5.35 Analysis of effectiveness of latent disentangled representation

Example 1: "a motorcycle parked on a road with a surfboard on the side"		
Noise	Content	Grammar
0.01	"a motorcycle parked on a road with a surfboard on the side"	"a motorcycle parked on a road with a surfboard on the side"
0.10	"a motorcycle parked on a road with a surfboard on the side"	"a motorcycle parked on a road with a surfboard on the side"
0.50	"a motorcycle parked on a road with a surfboard on the side"	"a motorcycle parked on a road with a surfboard on the side"
1.00	"a motorcycle parked in a parking lot with trees in the background"	"a motorcycle parked in front of a building with graffiti on it"

Example 2: "a bathroom with a white sink and a sink and shower"		
Noise	Content	Grammar
0.01	"a bathroom with a white sink and a sink and shower"	"a bathroom with a white sink and a sink and shower"
0.10	"a bathroom with a white sink and a sink and toilet"	"a bathroom with a white sink and a sink and shower"
0.50	"a bathroom with a toilet , sink , and a shower"	"a bathroom with a toilet , sink , and a shower"
1.00	"a bathroom with a large mirror and a large mirror above"	"a bathroom with a white sink and white and white toilet"

Example 3: "three giraffes standing in front of a large building"		
Noise	Content	Grammar
0.01	"three giraffes standing in front of a large building"	"three giraffes standing in front of a large building"
0.10	"three giraffes standing in front of a large building"	"three giraffes standing in front of a large building"
0.50	"three giraffes standing in front of a large church"	"several giraffes standing in front of a large building"
1.00	"three people are standing in a kitchen holding food"	"several different motorcycles parked in front of a fenced area"

Although not perfect, we can notice that the representation is a little effective. If one takes the example 2 into consideration, it may notice the change from "," to "and", or the phrase moving from simple nouns to the combination adjective + noun: "sink" to "white sink". While only in the content new elements are added such as "mirror".

In a less degree, this can also be notice in the example 3, the change from "giraffes" to "people" or "building" to "church". Finally, it is importance to notice that the sensitivity for both representation is around 0.5.

5.7 Conclusions

As it was shown, both the length-aware technique and the disentangled are somehow effective in their propose. They allow for the generation task to be more controlled, giving the generator the ability to control length, content and grammar. But they are not as effective as expected.

It is also to important to compare all the proposed models. To do that, the plots in figure 5.3 was produced. It contains the Self BLEUs versus the BLEU for all model.

As we can see, the disentangled produces results similar to the addition of the Bag-of-Word loss, while keeping the KL weight more low (9.64 vs 20.01). Which can be seen as an improvement.

Finally, all the models distribute themselves along a line that correlates SelfBLEU and BLEU, while the dataset presents a reasonable high BLEU while keeping the SelfBLEU. This gives an idea of a possible direction that generative models research should work on: making the Self BLEU and BLEU uncorrelated.

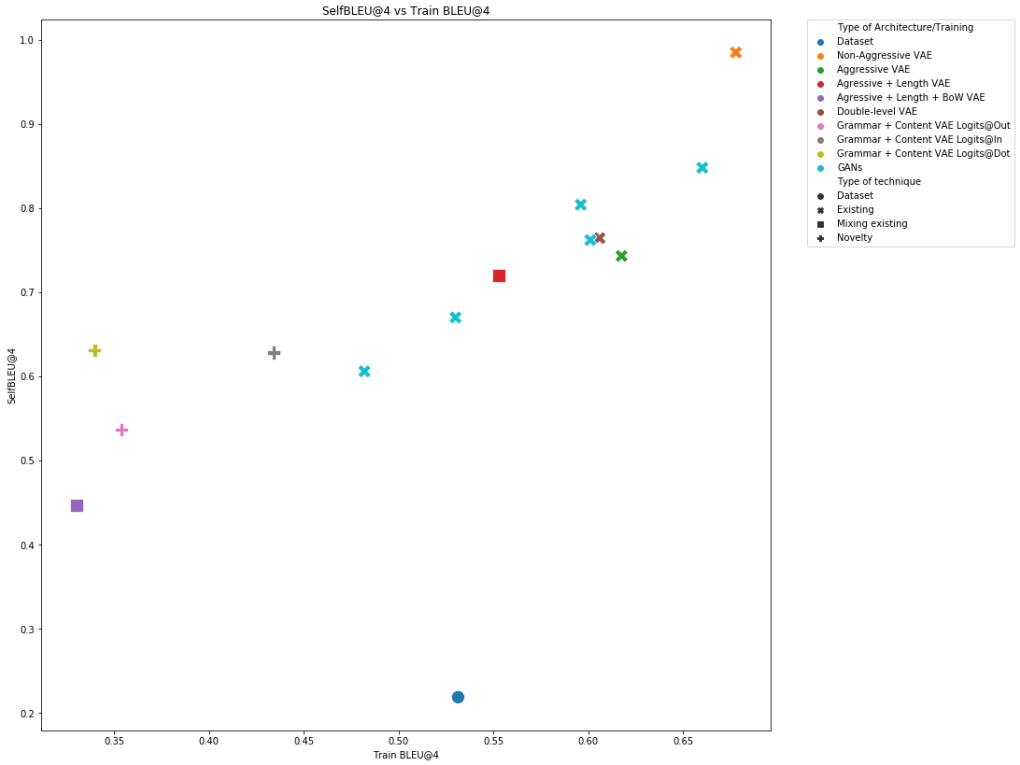


Figure 5.2: **All models metrics for BLEU@4**: The models show a correlation between BLEU and SelfBLEU. Aggressive + Length VAE and Aggressive + Length + BoW VAE are inspired by other papers. Grammar + Content VAE is the novel architecture suggested in this work.

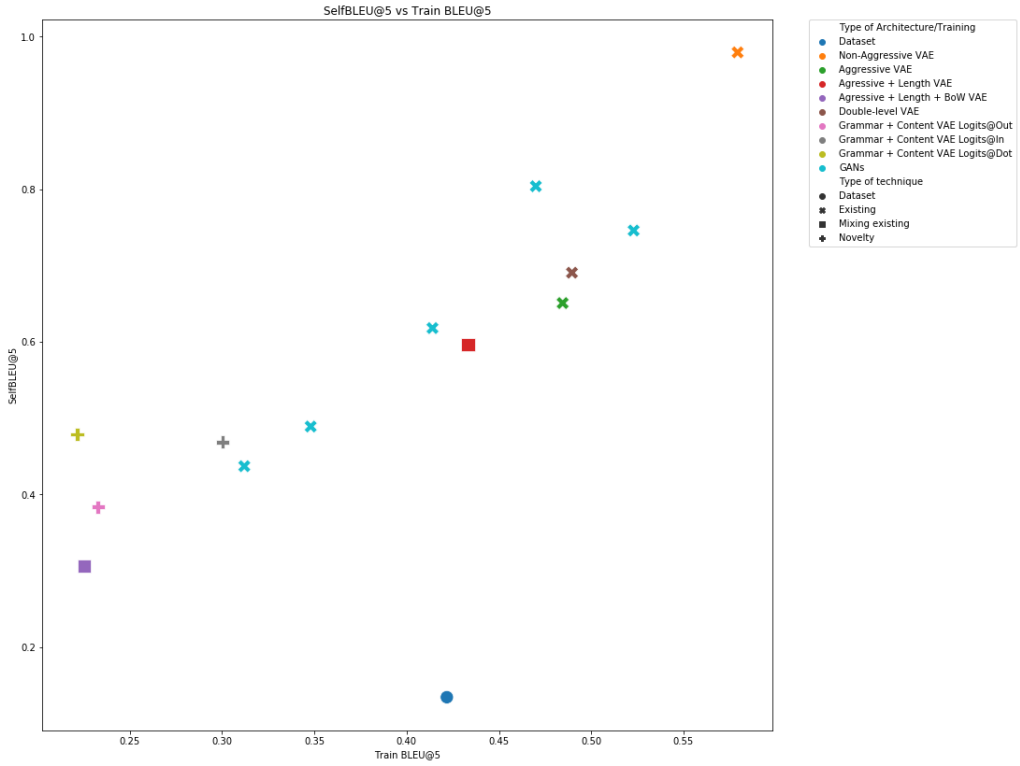


Figure 5.3: **All models metrics for BLEU@5**: The models show a correlation between BLEU and SelfBLEU. Aggressive + Length VAE and Aggressive + Length + BoW VAE are inspired by other papers. Grammar + Content VAE is the novel architecture suggested in this work.

Chapter 6

Results for Proposed Applications

6.1 Auto-complete

As pointed before, the VAE may be used to predict next words, by encoding the words known so far and them predicting the next one. It is intuitively to think that length-aware VAE will perform better in this use case and this will be one of the ideas that will be validated. Two models will be tested: Aggressive VAE and Agressive VAE Length-Aware with BoW loss, which will be referred as AVae and AVaeBow in the tables. This has the goal of both validating the impact of using length as a feature for this task.

Besides that, there are two possible decoding strategy: forcing or loop. The first consist in using the known words as input of the decoder at each step. The second consist in looping the predicted word into the decoder at each step. The former is expected to work better in terms of accuracy.

Below are a few examples for each model and decoding strategy, varying from 1 to 3 predicted words. First the examples, with forcing decode strategy:

Table 6.1 Examples of auto-completing with forcing decode

Predicting 1 word:

Example 0: some people that are hanging out in the "***snow***"
AVae: "kitchen"
AVaeBow: "distance"

Example 1: a close-up of an orange on the side of the "***road***"
AVae: "road"
AVaeBow: "road"

Example 2: this little boy wearing a jacket is holding a red "***apple***"
AVae: "umbrella"
AVaeBow: "shirt"

Predicting 2 words:

Example 0: some people that are hanging out in "***the snow***"
AVae: "the <pad>"
AVaeBow: "the room"

Example 1: a close-up of an orange on the side of "***the road***"
AVae: "a <pad>"
AVaeBow: "the road"

Example 2: this little boy wearing a jacket is holding a "***red apple***"
AVae: "banana phone"
AVaeBow: "little shirt"

Predicting 3 words:

Example 0: some people that are hanging out "***in the snow***"
AVae: "of <pad> to"
AVaeBow: "on side view"

Example 1: a close-up of an orange on the side "***of the road***"
AVae: "of <pad> <pad>"
AVaeBow: "of side walk"

Example 2: this little boy wearing a jacket is holding "***a red apple***"
AVae: "a <pad> <pad>"
AVaeBow: "a little cat"

And without forcing:

Table 6.2 Examples of auto-completing without forcing decode

Predicting 1 word: **Example 0:** some people that are hanging out in the "***snow***" AVae: "with" AVaeBow: "red" **Example 1:** a close-up of an orange on the side of the "***road***" AVae: "of" AVaeBow: "walk" **Example 2:** this little boy wearing a jacket is holding a red "***apple***" AVae: "standing" AVaeBow: "shirt"
Predicting 2 words: **Example 0:** some people that are hanging out in "***to them***" AVae: "with a" AVaeBow: "to some" **Example 1:** a close-up of an orange on the side of "***road walk***" AVae: "street </s>" AVaeBow: "empty snowy" **Example 2:** this little boy wearing a jacket is holding a "***red apple***" AVae: "banana in" AVaeBow: "is smiling"
Predicting 3 words: **Example 0:** some people that are hanging out "***air to them***" AVae: "bench next to" AVaeBow: "on the side" **Example 1:** a close-up of an orange on the side "***of the road***" AVae: "the side of" AVaeBow: "an empty snowy" **Example 2:** this little boy wearing a jacket is holding "***a red apple***" AVae: "a banana standing" AVaeBow: "a jacket and"

As expected, the using forcing during decode, provides better results.

Also, the presence of "<pad>" or "</s>" in the simple Aggressive VAE indicates it has problems to get the correct length right. While this does not happen with the AVaeBow. As a general result, the AVaeBow performs well, getting the right word in a few examples, or at least generating consistent phrases.

To compare this two models in a quantitative manner, we compute the Accuracy and NNLoss for each model and decoding strategy.

Table 6.3 Results for auto-complete predicting 1 word

	Forcing		Non-Forcing	
	Perplexity	NLLLoss	Perplexity	NLLLoss
AVae	90.01	4.50	80,017	11.29
AVaeBow	7.38	2.00	91.83	4.52

Table 6.4 Results for auto-complete predicting 2 words

	Forcing		Non-Forcing	
	Perplexity	NLLLoss	Perplexity	NLLLoss
AVae	849.79	13.49	350.72	11.72
AVaeBow	37.52	7.25	41.47	7.45

Table 6.5 Results for auto-complete predicting 3 words

	Forcing		Non-Forcing	
	Perplexity	NLLLoss	Perplexity	NLLLoss
AVae	38.60	10.96	44.40	11.38
AVaeBow	7.26	5.95	15.69	8.26

It is important to notice that the system is not predicting bigrams or trigrams. It is predicting each word and computing the loss for each word. If for example, it predicts "the room" and the expected result is "the snow", the correct prediction of the word "the" affects the NLLLoss and the perplexity, which explain in part why the perplexity varies from predicting one word or three words. When predicting two or three words, normally there are words such as "the", "to" and "of" which are easier to predict. Nonetheless, it would be more correct to predict the bigrams and trigrams, obtaining a more precise evaluation.

As expected the AVawBow has better values of loss being more adequate for this application. But unfortunately, the accuracy is still very low, indicating the need to improve the model to be useful for such setting. Some of the alternatives would be to change the decoding strategy for more words, by using techniques such as beam-search.

6.2 Text Retrieval

To analyze the quality of the VAE for the Text Retrieval, the Cranfield Dataset will be used. In this section, models that favors good content representation in the latent space will be tests, more specifically, the Agreesive VAE with Length-Aware and BoW loss and the Agressive VAE Disentangled with Logits@In will be tested and compared.

The first step consist in training the models with the Cranfield Dataset. The table 5.4 shows a few examples of queries and documents titles. It is easy to notice the difference regarding the ImageCOCO. In this case, since the dataset consist of title of documents, there is a bigger diversity on the format of the phrases. On the otherhand, the queries repeat an certain structure.

Table 6.6 Cranfield Colleaction Queries and Titles

Queries:	**Documents titles:**
what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft .	experimental investigation of the aerodynamics of a wing in a slipstream .
what problems of heat conduction in composite slabs have been solved so far .	simple shear flow past a flat plate in an incompressible fluid of small viscosity .
what chemical kinetic system is applicable to hypersonic aerodynamic problems.	one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal .

This difference between this dataset and ImageCOCO may bring different performance. Table 5.5 presents the result of training for each model:

Table 6.7 Training with Cran Collection

	Perplexity	KL-loss	Cross-Entropy Loss
Length-Aware AggressiveVAE + BoW Loss	42.96	10.45	57.64
VAE Disentangled + Length-Aware	60.64	2.18	62.92

The results are worse than with ImageCOCO, given that the vocabulary of the Cranfield Collection is smaller. Besides that, it was necessary to do two modification to obtain VAE where the posteriori did not collapse. The

start KL Weight had to be reduced to 0.001 and the KL Weight was clipped to 0 every time the Kullback-Leibler Loss was smaller than 2. This indicates that the VAE is extreme sensible to an increase in the length of the phrases. This will definitly be a research topic for extending the usage of VAEs.

Regarding the performance Text Retrieval over the Cran Collection, the Accuracy and Mean Reciprocal Rank (MRR) obtained are displayed on table 5.6.

Table 6.8 Text Retrieval in Cran Collection using VAE

	MRR
Length-Aware AggressiveVAE + BoW Loss	0.0072
VAE Disentangled + Length-Aware	0.0038

As we can see the results are really poor, our Text Retrieval system would be most of the time wrong. And due to the poorly results, it is impossible to compare the two models. If eventually one would like to explore the VAE in such context, it is necessary to explore several twists both in training and architecture to obtain better results.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis was started with two goals: improve the performance and understanding of the most recent advances in the Variational AutoEncoders for Text Generation and explore possible application of such system.

Regarding the first goal, three main contributions were made. The first consist in proposing a twist in how we evaluate text generation performance. Instead of using the maximum BLEU and minium Self-BLEU as targets, it was suggested using the values computed for the training dataset using samples drawn from it. As presented in figure 5.3, some models presents a BLEU higher than the dataset baseline which was human generated, therefore, improve those models regarding BLEU would be a misleading target. The dataset baseline provides a more reasonable target that may help direct efforts in a more promising direction.

The second consist in demonstrating the correlation between the Self-BLEU and BLEU in VAEs models. It is clear that when the Self-BLEU drops the BLEU also does it, while the dataset baseline has a low Self-BLEU and high BLEU. This correlation is a problem for the improvement of VAE and efforts to develop techniques that improve only one metric without affecting much of the other is a interesting research topic.

Finally, the third contribution consist in the Disentangled VAE. This technique has a similar goal as the Bag of Word loss – improve content representation on the latent space, but accomplishes it in a different manner. It provides results similar regarding metrics to the BoW technique while mantaining the Kullback-Leibler loss smaller them the BoW, which is a good indicative. Unfortunately only one architecture/training configuration were tested, leaving plenty space for more experimentation.

Regarding the application of Text Generation model, two possibilities were explored: text completion and text retrieval. In the first, we can see an improvement of one model regarding the other, but the accuracy and quality of the results seems way too low to be used as a product.

In the Text Retrieval, the VAE performed extremelly poor, not being able to get the correct document more them 2 times in 340.

Regarding application, although this work presented ways of using VAE in such context, much more study and exploration is necessary to maybe consider the VAEs in such setting.

In conclusion, since the first presentation of the VAE, lot of improvement were made and for smaller datasets some really coherent phrases were generated. But there are still open question of how to improve this model to be more useful and without doubt there is one big question of what is the usage of such models. Although some application were explored, the results are way low to compete against tailored system for such tasks. Lot of improvement must be made for such general-purpose model – Text Generation, to be used in more specific tasks.

7.2 Future Work

Given the results obtained in this works, a few lines of works would be interesting to explore. This thesis would like to highlight a few:

7.2.1 Dataset's BLEU and Self-BLEU as target

One of the minor proposals of this work was to use the BLEU and Self-BLEU of the train dataset as target for the text generation models. Although intuitively coherent, this must still be explored to see if it provides better classification of Text Generation models.

Such exploration must be aimed in comparing BLEU and Self-BLEU of several datasets and seeing how they differ. Comparing the effect of changing the number of samples generated or references used, for several datasets and models. Finally, checking if there is correlation between the distance of the metrics to the dataset target with human judgment. If proven coherent, such target might provide more information for the develop of Text Generation models.

7.2.2 Disentanglement Architectures

The most novel contribution of this work is the Disentanglement VAE architecture. It is seed idea is to have two networks representing two different, and hopefully, orthogonal, parts of the sentences. Such architecture was relatively successful to provide results similar to the application of the Bag of Word loss. Therefore, it might be interesting to explore some variations. Some suggestion is use other type of grammatical features, below is a list of some we find interesting:

- Rule-based morphology such as Verb Form and Verb Tense
- Dependency parsing results
- Entity recognition

Most of those are fully available in SpaCy library, allowing for rapid iteration.

Another interesting exploration topic is how the two networks are trained. Focusing on simplicity, we opted for training the Grammar network separately and keep it fixed during the training of the content network. A few variations of this setting could be:

- Keeping the Grammar network trainable
- Pre-training the Content network in a Bag of Word prediction task
- Training both together in a multi-task setting

The multi-task setting seems as a very challenging one. As we saw with the Bag of Word loss, the add of other losses tends to shadow the Kullback-Leibler loss – a fundamental aspect of the VAE. Therefore, exploring this might need some study and analysis on how to assign different weights to each loss.

7.2.3 Training strategy for VAE

During the exploration of the VAE for the Text Retrieval task, we notice the difficult to parameterize it to avoid mode collapse – even already using Aggressive learning, when the dataset changed. Therefore, more robust and less dataset dependent strategies are needed. One interesting aspect is the value of the Kullback-Leibler loss for best models. It tends to be around 3 to 6 and intuitively we believe this value is correlated with the size of the latent space, an aspect interesting to explore. Below are some suggestions of research topics:

- Check for estimates of Kullback-Leibler ideal loss values.
- Change the Kullback-Leibler weight according to the distance of the Kullback-Leibler to a giver range target
- Perform Aggressive training on-demand, according to the value of the Kullback-Leibler loss.

But as saw before, this problem is created in part due to the architecture used in the decoder (RNNs). Such architecture drives the network to learn the most common succession of words while ignoring the latent space. Several techniques mitigate this problem such as Drop-out layer of the decoder and adding the latent vector as input in every step.

7.2.4 Combinging VAE and GANs

As proposed in [11], it is possible to VAE and GAN in training. Given this and the results obtained in the BLEU metrics, we think that combining the two VAE models with better SelfBLEU – Aggressive Length-aware + Bag-of-Word Loss and VAE Disentangled + Length-aware Logits@Output, with some of the GAN models might drive the model to have good results regarding the SelfBLEU and BLEU metrics. But this creates the complexity of training GAN in Text that normally involves doing reinforcement learning as in [18]

7.2.5 General VAE Architecture

As pointed before, the architecture of the decoder influences a lot the effectiveness of training the VAE. Therefore, it is interesting to suggest a few alternatives that allow for a more high-level training inspired in the architecture of the Question Answering system proposed at [19]:

- Multi-level decoder with high-level network, that is, that receives the output of all previous levels
- Attentive encoder and decoder, that is, instead of receiving previous state use attention layer in all previous states

Other aspects to be explored the usage of pre-trained embedding layer in both the encoder and decoder. It is also interesting to explore the effect of using the same layer in both encoder and decoder.

7.2.6 Usage of VAE and Text Generation Systems

Finally, one part explored was the application of the VAE. For the proposed application, in deep studies are necessary to understand how to use the VAE in such context. It is also interesting to explore the usage trained VAE as part of another network.

7.2.7 Conclusion

Its been 6 years that the VAE were presented. Since them, lot of research has be done to improve and apply those models, but nonetheless, most of the models applied to Natural Language Processing are not text generative – that is, focused on predicting classes or learning vectors – or are attention-based and huge models such as GPT-2 [20], which was trained with 40GB of data, the Transformer [21] and BERT [22].

That setting is a little bit overwhelming, because as generative models become bigger, the need for resources to train it also grows, forcing most of the research to concentrate in big hubs with the disposition of resources to train such models.

It is in our opinion that models such as VAE and GAN may give the possibility of developing useful generative models that are less demanding regarding resources. But it has not been an easy task to do so.

Bibliography

[1] Turing, A. M., Computing Machinery and Intelligence. Mind 49: pages 433 – 460 (1950)

[2] McCarthy, J., Minsky, M. L., Rochester, N., Shannon, C. E., A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence (1955)

[3] Russel, S., Norvig, P., Artificial Intelligence: A Modern Approach. Prentice Hall (2009)

[4] Papineni, K., Roukos, S., Ward, T., Zhu, W., BLEU: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics, pages 311 – 318 (2002)

[5] Zhy, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., Yong, Y., Texygen: A Benchmarking Plataform for Text Generation Models. CoRR (2018)

[6] Pennington, J., Socher, R., and Manning, C. D. GloVe: Global Vectors for Word Representation

[7] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., Enriching Word Vectors with Subword Information. CoRR (2016)

[8] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., Generative Adversarial Nets. NIPS Proceeding (2014)

[9] Kingma, D. P. and Welling, M., Auto-encoding Variational Bayes. (2013)

[10] Mi, L., Shen, M. and Zhang, J. A Probe Towards Understanding GAN and VAE Models. CoRR (2018)

[11] Larsen, A. B. L., Sønderby, S. K. and Winther, O.. Autoencoding beyond pixels using a learned similarity metric. CoRR (2015)

[12] Tsung, Y. L., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., et al, Microsoft COCO: Common Objectives Context. CoRR (2014)

[13] Cui, Y., Ronchi, M. R., Tsung, Y. L., Dollár, P., and Zitnick, L. COCO 2015 Image Caption Task, at cocodataset.org, (2015). Accessed the September 5 2019.

[14] Richmond A. P. Review of the cranfield project. Wiley Periodicals (1963)

[15] He, J., Neubig, G., and Berg-Kirkpatrick, T. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. CoRR (2019)

[16] Schumann, R. Unsupervised Abstractive Sentence Summarization using Length Controlled

[17] SpaCy library at https://spacy.io/, (2015). Accessed the September 15 2019

[18] Yu, L., Zhang, W., Wang, J. and Yu, Y. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. CoRR (2016).

[19] Zhu, C., Zeng, M. and Huang X. SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering. CoRR (2018)

[20] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. OpenAI (2019)

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. and Polosukhin, I. Attention is All you need. CoRR (2017)

[22] Devlin, J, and Chang, M., and Lee, K., and Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Model. CoRR (2018)