DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Learning to predict text quality using Generative Adversarial Networks

WASEEM SHERIFF

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Learning to predict text quality using Generative Adversarial Networks

WASEEM SHERIFF

Degree Projects in Scientific Computing (30 ECTS credits)
Degree Programme in Computer Simulations for Science and Engineering (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at Peltarion: Anders Arpteg, Carl Thomé, Pawel Herman
Supervisor at KTH: Michael Hanke
Examiner at KTH: Michael Hanke

TRITA-SCI-GRU 2019:397
MAT-E 2019:91

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

Generating summaries of long text articles is a common application in natural language processing. Automatic text summarization models tend to find themselves generating summaries that don't resemble the quality of human written text, even though they preserve factual accuracy. In this thesis, we propose a quality metric using convolutional neural networks to evaluate the quality of text produced. Using this metric, we finetune a baseline generative model (Pointer Generator Networks) in a GAN-based approach by optimizing a combined loss function - one from the baseline generative model and another from the discriminative model. The discriminator's loss serves as a qualitative training signal for finetuning the generator. The model is tested on the CNN / Daily Mail dataset of news articles. The results show that the summaries generated by the model are more well-written as well as accurate in content.

Sammanfattning

En vanling application inom naturlig språkbehandling är automatisk sammanfatning av långa textartiklar. Ett problem med existerande teknier är att genererade sammanfatningar kan överensstämma faktamässigt med referenssammanfatningen, men det finns brister med kvaliteen i den genererade texten. I denna avhandlingen föreslår vi en ny metod för att mäta textkvalite med hjälp av ett faltningsätverk (convolutional neural network). Denna metod kan användas inte bara för att förbättra kvaliteen på genererade texten. Vi utvärderar denna metod via ett generativt pekarnätverk (Pointer Generator Networks), som vi förbättra på ett GAN-baserat (Generative Adversarial Network) tillvägagångssätt. En kombinerad felfunktion används, där fel från den generarede modellen kombineras med fel från en diskriminerande. Den diskriminerande modellens felvärden fungerar som en kvalitativ signal för generatorn, som då kan tränas att förbättra textkvaliteen ytelligare. Metoden, utvärderas med CNN/Daily Mail-datasetet som innehåller nyhetsartiklar samt manuellt skrivna sammafattningar. Resultat från experimentan visar att den genererade texten blir mer välskriven och även korrekt faktainnehåll.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability for computers to generate text that is close to the quality of human written text has a lot of real world applications such as translation from one language to another, chat-bots, question answering, summarization etc. However, generating human quality text is a challenging problem because of inherent complexity of human language structures, ambiguity of meaning and difficulty in modeling long term semantic connections. In this thesis, the challenge of producing text summaries of news articles is that resemble human-written text considered.

1.1 Text Summarization

Today's information age has brought massive amounts of information to our fingertips. While this access to limitless information is transformative, we often find ourselves stifled by information overload. People need to read increasingly more text in seemingly less time than ever before. The growth of the web has only increased the many sources of text. The need to have tools that can condense these volumes of content into shorter yet accurate versions has become increasingly vital.

This has become an active field of research in Natural Language Processing (NLP). NLP is a branch of machine learning that is dedicated to modeling interactions between human (natural) language and computers. In NLP, an interesting problem is that of automatic text summarization - the task of automatically reducing a piece of text into a shorter and accurate version.

Manually generating summaries can be time consuming and expensive. Meanwhile computers excel at data processing. Enormous datasets can be computationally analyzed to reveal patterns in sequences, threads and associations. Documents, articles or other sources of text can be considered as long sequences of word with innate meanings, patterns, dependencies and inter-connected meanings. If algorithms can be trained to effectively understand these relationships, automatic generation of summaries can be closer to realization.

1.2 A difficult and necessary task

Even for a human, producing a summary of a long piece of text can be a difficult task. A good summary proves that the content was understood. According to mathematician Gregory Chaitin [1], "compression is comprehension. "The higher the level of compression that is achieved, the better a system's predictions will be, and the greater the extent to which it can be said to 'understand' ". While this was stated in the context of data compression, summarization can be viewed as a compression problem too. The source is reduced in size by trying to keep the most relevant information.

The need for summarizing appears in multiple ares of modern science. Reducing the laws of a theory to a few ones makes inconsistencies easier to find and the evaluation of how complete the theory is simpler. Therefore, it improves the understanding of the entire theory. Mathematicians write axioms and physicists try to find the equation that explains the system in entirety. Just like summarizing can be a challenge for humans, it can also be a challenge for machines. Algorithms that are able to understand the same way humans do, is at the core of artificial intelligence (AI) and NLP. In this thesis work, the focus will be set on automatic text summarization.

1.3 Approaches

The two main approaches for automatic text summarization are extractive and abstractive [2]. Extractive approaches generate summaries by the concatenation of extracts from the original text. Although they are limited by passages or sentences of the source, this property allows extractive approaches to be widely applicable. Abstractive approaches uses linguistic models to decompose and interpret the text so it is able to generate an original summary based on meaning of the content. Abstractive approaches typically are paraphrased versions of the original source. A common drawback of abstractive approaches is that they need to be trained with a text corpus which tends to make a particular method specific to the domain it is trained on. Today, the best algorithms that for text summarization are based on deep neural networks [3] .

1.4 Research Question

Despite the recent advancements in the domain of abstractive text summarization, the current state of the art models cannot be reliably used to consistently generate high quality summaries. The major problems that can be identified with the available models are: lack of coherence in longer summaries, incorrect reproduction of facts, and lack of novelty in generated sequences. In other words, producing summaries that resemble human-written text is a challenge especially when trying to preserve the nuances of grammar and subtle qualitative aspects of language.

This is particularly challenging because current de facto standards for automatic summarization evaluation, such as *ROUGE, BLEU, METEOR*, (see Section 2.7) are quantitative by design. Generating good quality summary is a subjective task and can be written in multiple different ways. Evaluating quality by applying a formula is inadequate. This is often the method used by evaluation metrics - they measure the extent of overlap of word sequences between a human written text and machine generated text and assign a score based on the overlap. While this could be an indicator of quality, it poses many limitations. For example, none of these quantitative metrics are capable of handling synonyms. It is therefore possible to have a generated sentence that captures the same semantic meaning as the reference human written text but with different words, and as a result be assigned poor BLEU/ROUGE/METEOR scores. Furthermore, these standards often fail to pick up nuances of language which lead poor quality of text. It is possible to obtain a good ROUGE score with a low quality of text. Thus, often *ROUGE* scores do not reflect the true quality of summaries and prevents multi-faceted evaluation of summaries (i.e. by topics, by grammar, by overall content coverage etc.). This is the challenge that this thesis work hopes to investigate.

The research question motivating this thesis is if applying alternative, task-specific, training signals to neural networks can improve the performance of text summarization models in context of the previously mentioned problems. In other words, if a neural network be trained to distinguish real human text from model-generated text, can this learning be used as a loss function as an additional loss function for the original summarization model to produce summaries that resemble human-like text? In this thesis we investigate if adversarial training of an existing competitive baseline abstractive summarization model (Pointer Generator Networks [4]) using a loss from GAN-like discriminator (based on SeqGAN [5]) produce better quality summaries which are closer in quality to human written summaries? These concepts will be further discussed in Chapters 2, 3, 4 and 5.

1.5 Organization of the thesis

In Chapter 2, the necessary mathematical concepts that form the building blocks of text summarization, NLP, neural networks are presented. Chapter 3 reviews the body of work done in the field of abstractive text summarization and methods to train neural networks for structured sequence generation. Chapter 4 introduces the models, datasets and tools used in this thesis work. The baseline architecture and the proposed extensions are presented in detail. Chapter 5 describes the conducted experiments and results obtained. Chapter 6 summarizes the project work, draws conclusions, discusses encountered problems, and suggests future directions in the domain.

1.6 Contribution

Work completed as part of this thesis is a continuation of research made by the deep learning community, especially [4] and [5] in search of better training approaches for text summarization. The contributions brought by this thesis can be summarized as

- Demonstrating the limits of current de facto standard metrics for automatic summarization evaluation
- Proposing a classifier model to distinguish between human written text and machine generated text
- Proposing a novel training approach for improving abstraction of generated summaries
- Qualitatively evaluating the effect of the proposed extensions and comparing the new models with previous work in the field

Chapter 2

Background

This section explains the theory needed to understand the model that is proposed by this thesis and to evaluate the results. Since the reader is assumed to have basic prior knowledge of artificial neural networks, only some aspects of neural networks are presented here. As this is a vast subject, if the reader wishes to learn more about the deep neural networks parts of the text described, he/she is referred to the books *Fundamentals of Deep Learning* by Buduma [6] and *Deep Learning* by Goodfellow [7].

2.1 Text Summarization

We call automatic text summarization the task of generating a shorter version of a given document. The main goals are to first extract the most meaningful information and then to write it using a human language. There are broadly two approaches to automatic text summarization: extractive and abstractive.

- **Extractive** approaches select passages from the source text, then arrange them to form a summary.
- **Abstractive** approaches use natural language generation techniques to write novel sentences.

As sentences are taken from the source text, it is easier to have meaningful human language in the summary with an extractive summarization model. However in this approach, sometimes sentences may seem unrelated, not organized in a natural way or lack a logical flow between them. In abstractive summarization, novel sentences are generated from scratch. This way, the summary will look more natural, reads more fluent and only relevant information is kept. At the same time, this task is harder.

Most of existing summarization techniques are of an extractive kind – mostly because it is much easier to select text than it is to generate text from scratch. For example, if the extractive approach involves selecting and rearranging whole sentences from the source text, it is almost certain to produce a summary that is grammatically correct, fairly readable, and related to the source text. These systems can be reasonably successful and robust when applied to mid-length factual text such as news articles. Extractive methods are strictly limited to the contents of the source and tend to lack human-written flair especially when applied to longer and more complex text.

A more sophisticated technique is abstractive text summarization, in which text is generated based on the content of the document. As text is generated using an internal language model, there are infinite possible summaries to each document, which means that summaries have the potential to be very good, but also to be very poor in quality. One of the limiting factors of abstractive summarization is that the results of natural text generation is far from perfect. Previous work within abstractive summarization has shown the ability to generate accurate summaries, but suffers from inconsistency in the generation [8]. Abstractive summarization is a more complex and challenging task, but more essential at the same time.

To illustrate the above approaches as an analogy, we can compare extractive summarization and abstractive summarization to a highlighter and a pen. Extractive is done with a highlighter by marking pieces of the text and choosing the best pieces as the summary of the document. Abstractive is done with a pen, with which a summary is composed from the ground up. In the former, the quality is dependent of the quality of the text in the document, in the latter, the quality depends entirely on the writer.

2.1.1 Abstractive Summarization Main Tasks

There are three main phases that need to be tackled to write a good abstract summary:

- the content needs to be **selected**. A text contains a lot of information. Some are relevant, some are less important than others and some are of less interest to the reader. So, a method to select relevant content is required.

- once the components of the summary are chosen, the information has to be **ordered**. In fact, if the information is not ordered correctly, it may change the meaning of the original text. So one needs to be careful with it. One may assume that the order of sentences in the original text implies the order of the information but this is a trivial solution and may not apply in every scenario (especially when summarizing across multiple different text sources)

- sentences need to be **generated**. This step is important to make the sentences easy to read for the human. The final result has to look fluent. It is also important to correctly express the original content. Another aspect of this stage is to compress: the information needs to be expressed the most efficiently possible.

2.2 Natural Language Processing (NLP)

Natural language processing is a term used to describe the field of machine learning focusing on the interaction between computers and human languages. While computers are superior to humans in many tasks, such as remembering facts or performing numeric calculations, a human being's ability to process natural language is unrivaled. There are several reasons why teaching language to computers is challenging. In addition to being ambiguous, languages are highly variable, meaning that we can express the same meaning by constructing completely different sentences. Further, humans are great at using languages, but worse at understanding and expressing why certain sentences are more valid than others.

Apart from natural languages being ambiguous and irregular, NLP is also difficult for computational reasons. Languages are discrete - words can't be related to each other simply by the characters they are made up of. Instead, their meaning has to be inferred from surrounding context. Languages are also compositional. As we form sentences by combining words, a sequence of words can carry more meaning than the words themselves. The fact that languages are discrete and compositional leads to data sparsity. Every day, we form sentences we have never heard before without hesitation. For these reasons, we must teach the computers how to interpret and express languages in ways which suit their strengths. The structure of human languages have to be represented in some numerical form, in order for computers to understand and process them.

NLP is this research field that treats human language from a computational perspective. There are many sub-fields within NLP, such as translation, entity extraction, text summarization, question-answering, sentiment analysis etc. NLP is a vast discipline, and a complete review of it is beyond the scope of this thesis. Only parts which are considered relevant to this thesis are described in this section.

2.2.1 Language Modeling

A human regards language as natural unconsciously, if the text they read or the words they hear sound plausible, in the context they are read or heard. Language models are models which try to calculate the probability of a sentence, i.e. the probability of a sequence of words to appear in the order they do in the evaluated sentence. Formally, given a sequence of words $w_{1:N}$, a language model tries to estimate $P(w_{1:N})$ using the chain rule of probability:

$$P\left(w_{1:N}\right) = \prod_{i=1}^{N} P\left(w_i | w_{1:i-1}\right)$$

The conditional probabilities are then estimated by evaluating a set of training data and counting the number of appearances each combination has. As a sentence becomes longer, the number of words grows and the conditional probability approaches zero. To avoid zero probabilities and to make the estimation more tractable, a common assumption made is that the probability of each word only depends on the k words that precede it. This is known as a Markov assumption.

The task of calculating the probability of a given sentence is equivalent to the task of calculating the most probable upcoming word, given a history of words. This follows since if we are able to calculate the probability of any sequence, we can also calculate the probability of a given word conditional to the sequence that precedes it as

$$P\left(w_i|w_{1:i-1}\right) = \frac{P\left(w_{1:i}\right)}{P\left(w_{1:i-1}\right)}$$

This means that languages models can also be used for generative purposes. The probabilities of words and sequences are typically estimated by counting how many times a word or a sequence appear in a corpus, i.e. a collection of texts easily readable to a computer. The counts are normalized by the total number of words, to lie between 0 and 1. This estimation method is called maximum likelihood estimation (MLE).

To evaluate language models, it is common to evaluate a model's perplexity [9]. The perplexity measures a variant of the probability of generating a sequence. The perplexity of a sequence $w_{1:N}$ is defined as $w_{1:N}$

$$\mathrm{PP}\left(w_{1:N}\right) = P\left(w_{1:N}\right)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P\left(w_{1:N}\right)}}$$

Using the chain rule, we can expand this to,

$$\mathrm{PP}\left(w_{1:N}\right) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P\left(w_i|w_{1:i-1}\right)}}$$

2.2.2 N-gram models

A set of models utilizing the Markov assumption is the *n-gram* models, where an n-gram represents a set of n continuous words. This reduces the sequence probability to the following form:

$$P\left(w_{1:N}\right) = \prod_{i=1}^{N} P\left(w_i|w_{i-n:i-1}\right)$$

The related perplexity would thus be using the chain rule, we can expand this to,

$$\mathrm{PP}\left(w_{1:N}\right) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P\left(w_i|w_{i-n:i-1}\right)}}$$

The conditional probability of an n-gram $w_{i-n:i}$ is estimated by counting all appearances of the n-gram in the training set, and dividing by all n-grams starting with the *n-1*-gram $w_{i-n:i-1}$.

Ultimately, the Markov assumption only gives an estimation of the true state. In practice, a test dataset might very well contain n-gram occurrences that never appear in the training data, which is a symptom of the curse of dimensionality. Assigning a zero probability to this sentence would be detrimental to the model. For that reason, there are techniques such as smoothing by assigning a small base probability to every possible n-gram.

2.2.3 Tokenization

For text processing purposes, it is of great importance to be able to identify individual units, so called *tokens*, of a given text. The act of dividing a text into tokens is called *tokenization*. These basic units, which can be defined on several levels, from morphemes (the smallest meaningful unit in a language) to words and even phrases [10], form sequences when grouped together. The total set of tokens form a vocabulary. In this thesis, sequences correspond to sentences and the tokens almost exclusively correspond to words, with a few exceptions such as punctuation.

Mathematically, a vocabulary *V* consists of $|V|$ tokens $y_1, \ldots, y_{|V|}$ which can be combined into sequences. We choose to denote a human produced sequence as $s_{1:T}$ or s, and let $\tilde{s}_{1:T}$ denote machine generated sequences, where T is the number of tokens in sequence. A sequence that is partly generated is denoted as $s_{1:t}, t < T$. s and $\tilde{s}$ are often referred to as *real* and *generated*, respectively.

2.2.4 Word Embeddings

Once a text is tokenized, it can be processed further by using the concept of *word embeddings*. This is the act of mapping tokens (which are words in our case) into real valued vectors in a continuous vector space which computer can handle. The purpose of this mapping is to be able to capture syntactic and semantic meanings and similarities between words. For instance, similar words can be identified by computing the Euclidean dot product, i.e. the cosine similarity d between two word embeddings *a* and *b* as

$$d = \frac{a \cdot b}{\|a\|_2 \|b\|_2} \tag{2.1}$$

A common approach is to use already pre-trained word embeddings, such as Word2Vec [11] and GloVe [12] as the embedded representation of the vocabulary of choice. An alternative approach would be to initialize the word embeddings randomly in a vector space, and then optimize the distribution of word embeddings as the training of the model progresses. This method is adopted by SeqGAN [5].

2.3 Feedforward Neural Networks

The feedforward neural network (FNN) [7], or fully-connected network, is one of the most popular and powerful machine learning models. Early research on the subject was inspired by and related to work done in the field of neuroscience [6].

The basic building block of the intelligence in the human brain is the *neuron*, which is a *cell* that "listens" to its surrounding neurons by sensing electrical impulses using several antennae called *dendrites*. The inputs from the dendrites are combined in the cell body to form a resulting output signal that is transmitted to other neurons via an axon [6]. The capability of the human brain does not come from a single neuron, but from an enormous network thereof. In 1943, McCulloch and Pitts introduced a model [13] set to mimic the neurons in the human brain. The so called artificial *neuron*

$$y = f(Wx + b), \tag{2.2}$$

which is shown in Figure 2.1

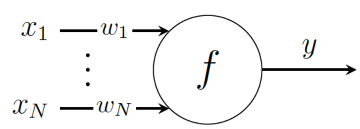

Figure 2.1: An artificial neuron with inputs $x_1, ..., N$, weights $w_1, ..., N$, activation function f and output y.

While a single unit is simple, a large collection (a network) of artificial neurons placed side-by-side in a layer can approximate advanced mathematical functions. At their very essence, FNNs are parametric mathematical functions, mapping data from an input space to an output space Y, $f\colon X \to Y$ Layers can also be stacked, forming deep networks that can approximate even more advanced mathematical functions. A group of artificial neurons that collectively form a network is called an artificial neural network (ANN). A simple ANN is shown in Figure 2.2

The weights and biases of the artificial neurons in an ANN can be updated in different ways. A common approach is to use the backpropagation algorithm, which was introduced in 1986 by Rumelhart et al. [14]. During training, a sample x with a corresponding correct output y is transmitted through the ANN, which outputs,

$$\tilde{y} = \phi\left(\sum_i w_i \cdot x_i + b\right)$$

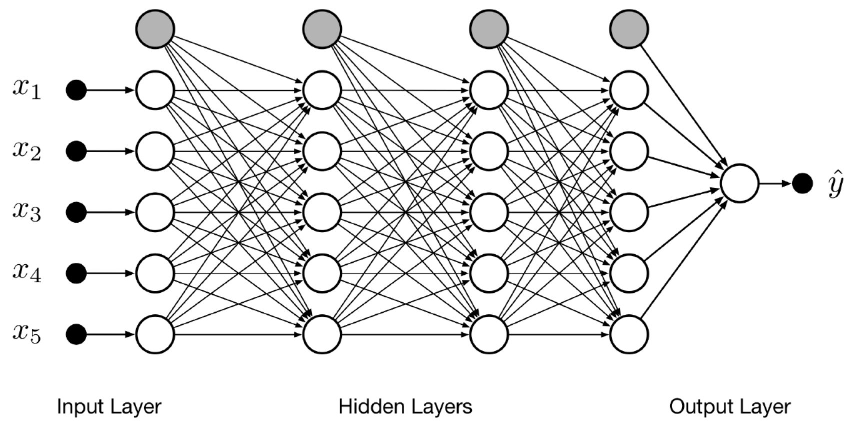

Figure 2.2: Illustration of a feedforward network. Black nodes represent the input and outputs of the network, gray nodes represent the bias terms b, and white nodes indicate neurons. Connections between the nodes are the parameters (weights w) of the model which are learned in the training procedure.

Unless the ANN is perfect, there will be a

$$\text{loss } \mathbb{L} = \tilde{y} - y$$

which the network has to minimise in order to improve its performance. Backpropagation does this by differentiating $\mathbb{L}$ with respect to the weights w, which can then in turn be updated via various methods such as stochastic gradient descent (SGD) [15] or Adaptive Moment Estimation (Adam) [16].

2.3.1 Training a Neural Network

While a neural network can approximate any continuous function well, the actual performance of a neural network depends on how well it is trained i.e how well its parameters (values of the weights and biases) are calculated. The loss function $\mathbb{L}$ models the distance between the networks output and the target. Training the neural network is an exercise in minimizing the loss function through gradient-based optimization. The loss function is not necessarily convex, and a solution found is not guaranteed to be a global solution [17]. However, gradient based optimization methods still yield good results in practice.

The gradient informs in which direction each parameter should be changed to approach the local optimum. Denoting the set of all weights and bias parameters by $\boldsymbol{\Theta}$ the parameter update can then be described as,

$$\begin{aligned}\boldsymbol{\Theta}^* &= \arg\min_{\Theta} \mathbb{L}(\Theta) + \lambda * \Phi(\Theta) \\ &= \arg\min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, \tilde{y}^{(i)}\right) + \lambda * \Phi(\Theta) \\ &= \arg\min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}, \Theta\right)\right) + \lambda * \Phi(\Theta)\end{aligned}$$

where f denotes a parametrized function which produces an output $\tilde{y}^{(i)}$ given the input vector. The parameter update also includes a regularization parameter λ to avoid overfitting and Φ is a regularization function penalizing large terms.

2.3.2 Softmax

A common task for ANNs is to classify a sample as belonging to one of N categories, $N > 1$. An intuitive way to do this is to assign a probability for each category. However, the outputs α_i, $i = 1, \ldots, N$ (often called *logits*) from a layer of neurons, do not necessarily add up to one, which is necessary if the logits are to represent a probability distribution. A common way to ensure that the logits meet this requirement is to use the softmax function.

$$\sigma\left(\alpha_i\right) = \frac{\exp\left(\alpha_i\right)}{\sum_{j=1}^{N} \exp\left(\alpha_j\right)} \tag{2.3}$$

The softmax function can be viewed as a distribution over logits, where $[\sigma\left(\alpha_i\right), \ldots, \sigma\left(\alpha_N\right)]$ denotes the probability for selecting an outcome related to one of the logits $\alpha_1, \ldots, \alpha$. If one seeks to change the relative differences within the distribution, a common practice is to introduce a temperature τ. The softmax function is then defined as

$$\sigma\left(\alpha_i\right) = \frac{\exp\left(\alpha_i/\tau\right)}{\sum_{j=1}^{N} \exp\left(\alpha_j/\tau\right)} \tag{2.4}$$

If $\tau > 1$, it will result in a flatter distribution and if $0 < \tau$, the probability differences will instead be amplified.

The shape of the softmax function strengthens the differences between predictions such that the ratio between two logits $\alpha_1$andα_2 with large and small values, respectively, is smaller than the ratio $\sigma\left(\alpha_1\right) : \sigma\left(\alpha_2\right)$. A special case of the softmax is when $N = 2$ when the softmax function is referred to as a *logistic sigmoid*.

2.4 Discriminative Models

Within the field of artificial neural networks, a common category of models are known as *discriminative models*. Their primary objective is to learn how to classify and perform regression on data. In a mathematical sense, a discriminative model learns how to compute the conditional probability $p(y|x)$ for the outcome y, given input data x.

A kind of neural network that has grown very popular when building discriminative models is the convolutional neural network, as presented below.

2.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), were originally invented [18] for computer vision to recognize and classify images by learning features and patterns unique to categories of images. CNNs have been shown to achieve strong performance on text classification tasks as well, even when considering relatively simple one-layer models.

A *Convolutional Neural Network* (CNN), is a type of feed forward network in which the hidden layers consist of convolutional layers (see [6] for a thorough explanation). An example of a CNN used for text classification is presented in Figure 2.3

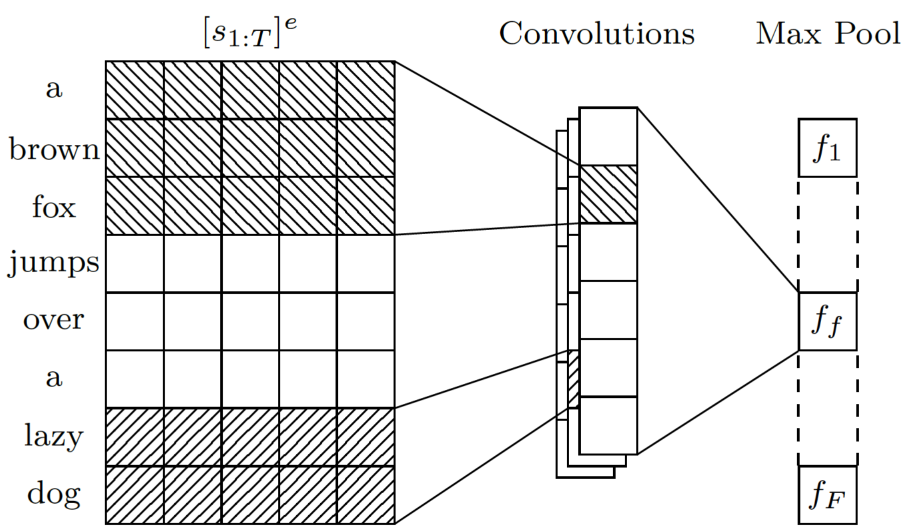

Figure 2.3: A CNN used to extract features $f_{1,\ldots,F}$ from a text

We begin with a tokenized sentence which we then convert to a *sentence matrix*, the rows of which are word *embeddings* of each token. These might be, e.g., outputs from trained word2vec [19] or GloVe [20] models. We denote the dimensionality of the word vectors by d. If the length of a given sentence is s, then the dimensionality of the sentence matrix is $s * d$.

We can then effectively treat the sentence matrix as an 'image' [21] , and perform convolution on it via linear *filters*. In text applications there is inherent sequential structure to the data. Because rows represent discrete symbols (namely, words), it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e., d). Thus we can simply vary the 'height' of the filter, i.e., the number of adjacent rows considered jointly. We will refer to the height of the filter as the *region size* of the filter.

A method for reducing the size of the date in between layers is a technique called *down-sampling* or *pooling*, in which the general idea is to divide the input tensor into several subsets, and perform an operation on each set to obtain a single output element per subset. One common pooling option is the *max pooling* where the maximum value of each subset is selected in order to produce a new output which is smaller in size than the input. In Figure 2.3, the max pool operation is performed on the entire output from each filter, but could also have been performed on subsets of the outputs, in which case the dimensionality reduction would not have been as drastic.

Suppose that there is a filter parameterized by the weight matrix w with region size h; w will contain $h \cdot d$ parameters to be estimated. We denote the sentence matrix by $A \in \mathcal{R}^{s*d}$, and use $A[i : j]$ to represent the submatrix of A from row i to row j. The output sequence $o \in \mathcal{R}^{s-h+1}$ of the convolution operator is obtained by repeatedly applying the filter on submatrices of A:

$$o_i = w \cdot A[i : i + h - 1]$$

where $i = 1 \dots s - h + 1$, and $\cdot$ is the dot product between the sub-matrix and the filter (a sum over element-wise multiplications). We add a bias term $b \in \mathcal{R}$ and an activation function f to each o_i, inducing the *feature map* $c \in \mathcal{R}^{s-h+1}$ for this filter:

$$c_i = f(o_i + b)$$

One may use multiple filters for the same region size to learn complementary features from the same regions. One may also specify multiple kinds of filters with different region sizes (i.e., 'heights')

The dimensionality of the feature map generated by each filter will vary as a function of the sentence length and the filter region size. A pooling function is thus applied to each feature map to induce a fixed-length vector.

A common strategy is 1-max pooling [22], which extracts a scalar from each feature map. Together, the outputs generated from each filter map can be concatenated into a fixed-length, 'top-level' feature vector, which is then fed through a softmax function to generate the final classification. At this softmax layer, 'dropout' [23] can be applied as a means of regularization. This entails randomly setting values in the weight vector to 0. It is also possible to impose an *v* norm constraint, i.e., linearly scale the *l2* norm of the vector to a pre-specified threshold when it exceeds this. Figure 2.4 provides a schematic illustrating the model architecture just described.

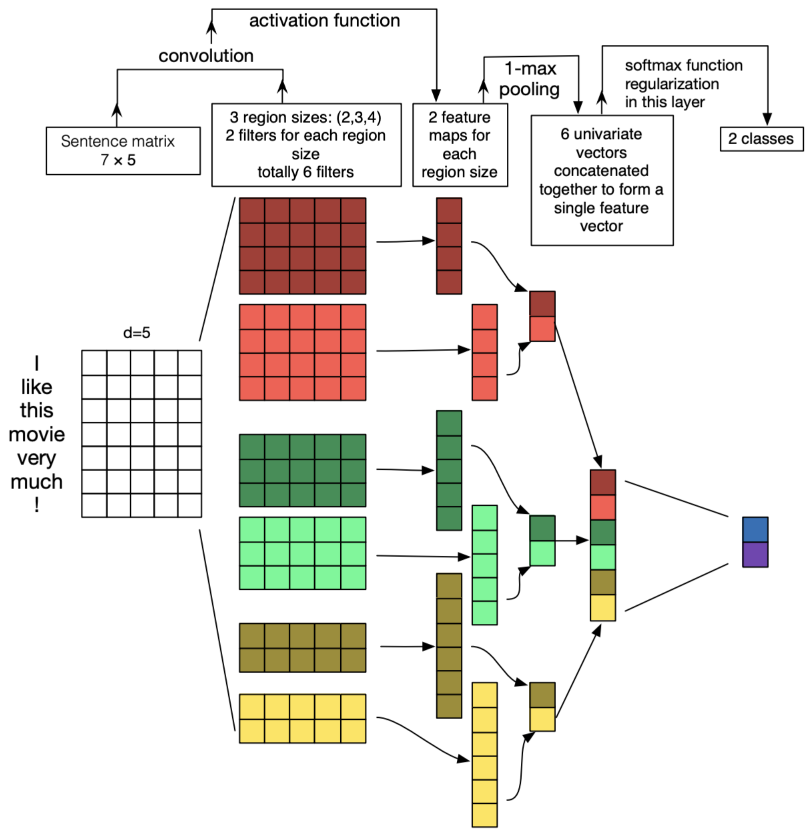

Figure 2.4: Illustration of a CNN architecture for sentence classification. We depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Filters perform convolutions on the sentence matrix and generate (variable-length) feature maps; 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states.

When training deep neural networks with many layers, a common challenge is *vanishing gradients* which means that the gradients used in the propagation algorithm become very small (vanish) when calculated through several layers [6]. A remedy for this, proposed by Srivastava et al. is the *Highway Layer* [24] architecture, in which a *transform gate* and a *carry gate* are introduced for passing some of the input data through layers without any nonlinear function applied to it. See Section A.1 for details. The result is that the network can vary how much of the input x that should be transferred through the network $H(x, W_h)$, and consequently where the backpropagation should be in effect, which tackles the problem of vanishing gradient. Highway networks have been used with success in text generation [5] [25].

2.5 Generative Models

In contrast to discriminative models, *generative* models generate examples of a given type of data by trying to model its distribution. In mathematical terms, the model tries to approximate the distribution $p_r(x; \theta^r)$, of the true data set in question, with parameters θ^r, by finding set of parameters θ^{g*} which maximizes the resemblance between the approximated distribution $p_g(x; \theta^g)$ and $p_r(x; \theta^r)$. With the optimal θ^{g*} known, sample generation is performed by drawing samples from $p_g(x; \theta^{g*})$

There are several different neural network architectures that are designed for generative applications. Some common techniques used for text generation, which is the underlying concept for producing summaries, are described in the following sections.

2.5.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNN) is a type of neural network that specializes in modelling sequential data with temporal dependencies. This could range from anything between natural language to time-series. In contrast to feed forward networks, RNNs have cyclic connections within layers, i.e *recurrent* connections. This gives the network the ability to handle a data sequence of variable length by maintaining a recurrent *hidden* state. The activation of a state thus depends on the previous state, which introduces the notion of memory in the neural network. This is illustrated in Figure 2.5 which shows a simple RNN in a compact format (left) and rolled out (right). In the figure, $x_t, t = 1, \ldots, T$ is the input data sequence, y_t are the output values and h_t are internal *hidden* states that are the means with which information is passed on to other time steps. The network's hidden state can be interpreted as a summary of all inputs seen by the network in previous steps, and is maintained by the network in an unsupervised fashion.

Given an input $x_t = [x_t^1, \ldots, x_t^n]$, the hidden state $h_t = [x_t^1, \ldots, x_t^k]$ is updated as

$$h_t^j = \begin{cases} 0 & \text{if } t = 0 \\ g\left(W\left(x_t \oplus h_{t-1}\right)\right)^j & \text{otherwise} \end{cases} \qquad j = 1, \ldots, k \tag{2.5}$$

where g is an activation function, e.g the *logistic sigmoid function*, and W is a weight matrix associated with input and the previous hidden state which are concatenated, denoted

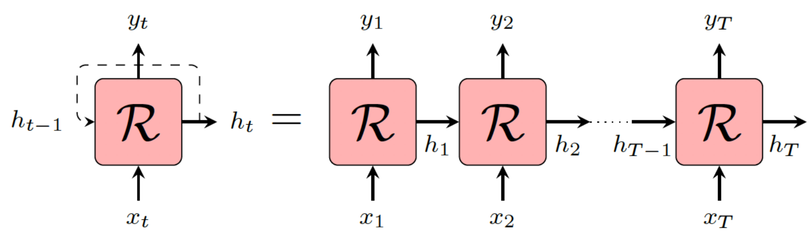

Figure 2.5: Illustration of a recurrent network. The states h is passed between consecutive time steps and allows the network to effectively model temporal data. The left illustration shows the network in rolled form, while the right illustrates in unrolled form.

by $\oplus$

RNNs have been shown to be useful in different types of sequence modelling problems, e.g. problems regarding text generation. In recent years, RNN in general, (and long short-term memory (LSTM) units in particular, see section 2.5.2 below) have been used to generate close to realistic text that uses correct words and semantics, but lacks real meaning, as demonstrated by, among others, Karpathy [26].

When generating text with RNNs, the input x_t, e.g. a word embedding vector, can be viewed as a function of the previously generated token $\tilde{y}_{t-1}$, i.e $x_t = f(\tilde{y}_{t-1})$. The RNN then computes the conditional probability distribution for each token $f(\tilde{y}_{t-1})$ given the previous tokens generated in the sequence as

$$p(y_t|y_{t-1},\ldots,y_1) = g(h_t) \tag{2.6}$$

with g being an activation function and $h_t = [h_t^1, \ldots, h_t^k]$ as defined in (2.5)

For example, if the purpose of a RNN is to generate a sequence of tokens $y_1, \ldots, y_t$ from an existing vocabulary V, equation (2.6) would give the probability of selecting token y_t, given all the previously selected tokens $y_{t-1}, \ldots, y_1$. Selecting the next token y_t, could be done by either selecting the token with the highest probability in (2.6), or a less greedy approach, sampling from the vocabulary with the computed probabilities assigned to the available tokens.

Given all the conditional probabilities of generating all the individual tokens y_j, the joint distribution of generating an entire sequence of t tokens is then calculated as

$$p\left(y_t, y_{t-1}, \ldots, y_1\right) = p\left(y_1\right) \prod_{i=2}^{t} p\left(y_i|y_{t-1}, \ldots, y_1\right) \tag{2.7}$$

2.5.2 Long Short Term Memory (LSTM) Units

A *Long Short Term Memory* (LSTM) network is a network using *LSTM units*. One such unit is illustrated in Figure 2.6, which is inspired by an illustration in an essay by Olah [27]. There are several different LSTM implementations [27]. Below, the version used by our baseline model [4] is described.

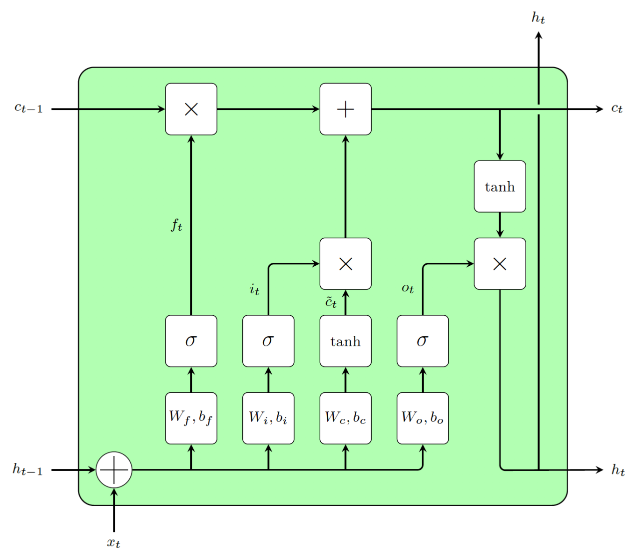

Figure 2.6: The inner workings of the LSTM unit.

In contrast to recurrent units defined earlier, the concept behind LSTM units is to let each update of the j^{th} hidden state h_t^j at time t be computed as

$$h_j^t = o_t^j \tanh\left(c_t^j\right)$$

where o_t^j is the so called *output gate* defined as

$$o_t^j = \sigma\left(W_o\left[x_t, h_{t-1}\right] + b_o\right)^j$$

with weight matrices W_o and bias b_o.

The variable c_j^t is a *cell state* in the j^{th} neuron at time t defined as

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j$$

where c_{t-1}^j represents the cell state from the previous time step, and $\tilde{c}_t^j$ is a candidate contribution to the memory, consisting of a combination of the input x_t and the current hidden state h_{t-1}^j as

$$\tilde{c}_t^j = \tanh\left(W_c\left[x_t, h_{t-1}\right] + b_c\right)^j.$$

In order to control the dynamics of the change of the memory c_t^j, a *forget gate* f_t^j and an *input gate* i_t^j are introduced as

$$f_t^j = \sigma\left(W_f\left[x_t, h_{t-1}\right] + b_f\right)^j$$
$$.$$
$$i_t^j = \sigma\left(W_i\left[x_t, h_{t-1}\right] + b_i\right)^j$$

With these, the network can control the duration of its memory and thereby learn how to remember information for several time steps.

2.5.3 Encoder-Decoder Attention Model

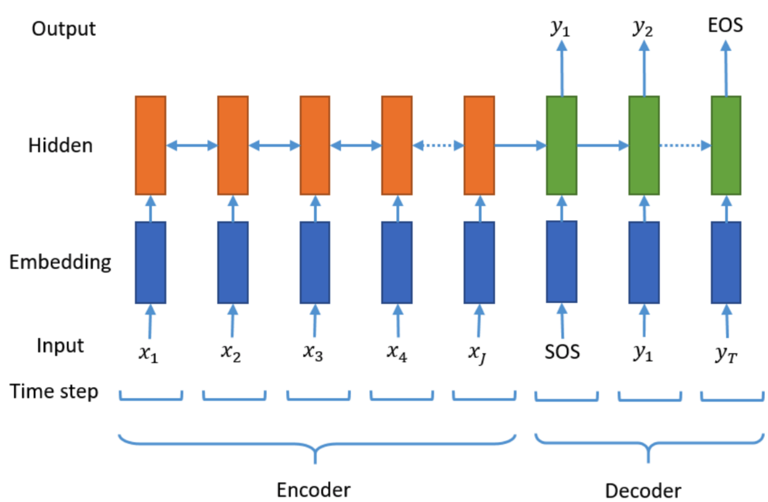

Figure 2.7: The basic Encoder-Decoder model. SOS and EOS represent the start and end of a sequence, respectively.

One common approach to generating sequences of text based on some input translation, summarization is to use an *encoder-decoder* model. The idea is that an *encoder* reads a source article, denoted by $x = (x_1, x_2, \ldots, x_J)$ and transforms into hidden states $h^e = (h_1^e, h_2^e, \ldots, h_J^e)$ while the decoder takes these hidden states as the context input and outputs a summary $y = (y_1, y_2, \ldots, y_T)$ where J and T are the lengths of the source and summary respectively.

A simple illustration of this is shown in Figure 2.8 as applied to a simple translation sequence-to-sequence task. The sequence "his name is karl" is encoded. The encoded embedding, denoted h_5 in the illustration, is then fed to the decoder$\mathcal{D}_e$ which processes the embedding vector and outputs a decoded version of the input, in this case the Swedish translation of the input sequence - "han heter karl". The words *SOS* and *EOS* denotes the *start of sentence* and *end of sentence* (see Section A.4).

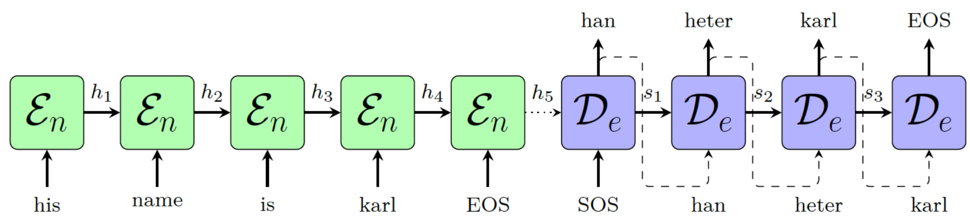

Figure 2.8: A simple encoder-decoder model consisting of an encoder $\mathcal{E}_n$ and a decoder $\mathcal{D}_e$.

Encoder-decoder models are often applied to *sequence to sequence* problems that are present within various areas of NLP, such as machine translation (the case in illustration) or text summarization, which motivates the use of encoder-decoders in this thesis. For more information on encoder-decoder models, see [6]

One problem with encoder-decoder models is that all information in the input must be encoded into the final state of the encoder in order for the decoder to be able to produce something meaningful. For short text such as "his name is karl", this is not a problem, but for longer sequences, or in more date intense applications such as sound to sequence, it quickly becomes an issue.

To deal with this issue, an *attention mechanism* can be utilized. The attention mechanism can be explained by viewing the encoder-decoder as a human translator with a memory - all hidden states $[h_1, \ldots, h_T]$ of the encoder $\mathcal{E}_n$. As illustrated in Figure 2.9, the encoder-decoder first encodes the entire sequence and stores it in its memory. During the decoding, the decoder network does, in addition to its internal state s_{t-1}, not only receive the previously generated token as in the simple encoder-decoder in Figure 2.8, but rather a concatenation of the previously generated token and the output from the attention mechanism.

The inner workings of the attention mechanism varies between implementations, but a common practice presented by Bahdanau et al. [28] is to weigh the influence of each of the encoder outputs by its relevance in regard to the inner state s_{t-1} of the decoder from the previous time step, as shown in Figure 2.9. This is done by scoring the importance of each encoder output h_j as the dot product between h_j and s_{t-1}, normalizing the scores with a softmax, and then weighting h_j with the resulting, normalized score. As defined in Figure 2.9, the output of the previous recurrence y_{t1}, which is used as input to $\mathcal{D}_e$ in the simple encoder-decoder model presented in Figure 2.8 is concatenated with the outcome of the attention mechanism, forming y_{t1}^{+}, which is instead used as an input to $\mathcal{D}_e$.

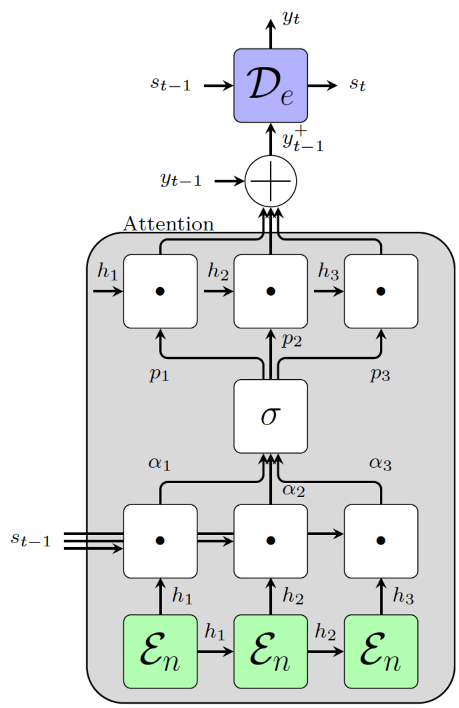

Figure 2.9: An encoder-decoder model with an attention mechanism.

$$
\begin{aligned}
y^+_{t-1} &= y_{t-1} \oplus h \cdot p \\
p &= \sigma(\alpha) \\
\alpha &= s_{t-1} \cdot h
\end{aligned}
\tag{2.8}
$$

In the below example, the source text is a news article that begins *Germany emerge victorious in 2-0 win against Argentina on Saturday*, and we're in the process of producing the abstractive summary *Germany beat Argentina 2-0*. First, the encoder RNN reads in the source text word-by-word, producing a sequence of encoder hidden states. (There are arrows in both directions because the encoder is bidirectional, but that's not important here).

Once the encoder has read the entire source text, the *decoder RNN* begins to output a sequence of words that should form a summary. On each step, the decoder receives as input the previous word of the summary (on the first step, this is a special <SOS> token which is the signal to begin writing) and uses it to update the *decoder hidden state*. This is used to calculate the *attention distribution*, which is a probability distribution over the words in the source text.

The attention distribution is calculated as:

$$\begin{aligned} e_i^t &= v^T \tanh\left(W_h h_i + W_s s_t + b_{\text{attn}}\right) \\ a^t &= \text{softmax}\left(e^t\right) \end{aligned} \tag{2.9}$$

where v, W_h, W_s and b_{attn} are learnable parameters. Intuitively, the attention distribution tells the network where to look to help it produce the next word. In the diagram above, the decoder has so far produced the first word "Germany", and is concentrating on the source words "win" and "victorious" in order to generate the next word "beat".

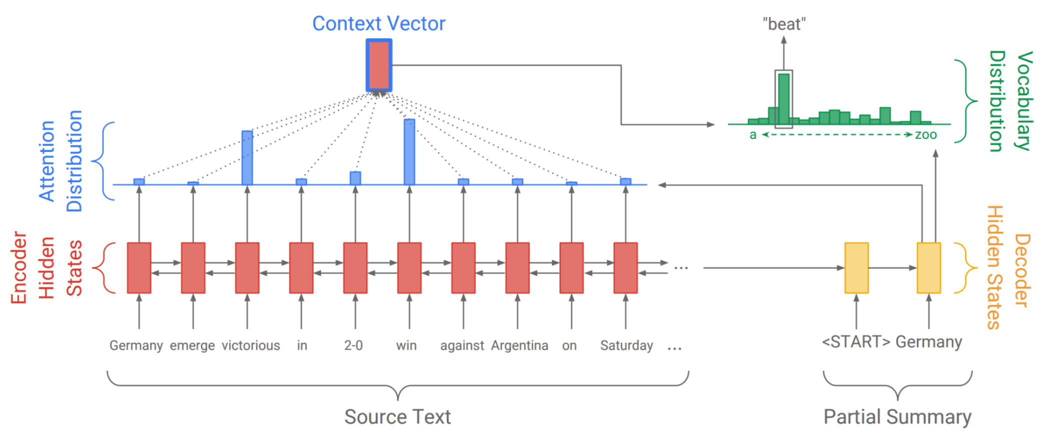

Figure 2.10: An encoder-decoder model with an attention mechanism.

The attention distribution is then used to produce a weighted sum of the encoder hidden states, known as the context vector h_t^*:

$$h_t^* = \sum_i a_i^t h_i \tag{2.10}$$

The *context vector* can represent fixed representation of "what has been read from the source text" on this step of the decoder. Finally, the context vector and the decoder hidden state are used to calculate the *vocabulary distribution* P_{vocab}, which is a probability distribution over all the words in a large fixed vocabulary (typically tens or hundreds of thousands of words).

$$P_{\text{vocab}} = \text{softmax}\left(V'\left(V\left[s_t, h_t^*\right] + b\right) + b'\right) \tag{2.11}$$

where V, V', b and b' are learnable parameters.

The final distribution from which words are predicted is given by w:

$$P(w) = P_{\text{vocab}}\left(w\right) \tag{2.12}$$

During training, the loss for timestep t is the negative log likelihood of the target word w_t^* for that timestep:

$$\text{loss}_t = -\log P\left(w_t^*\right) \tag{2.13}$$

and the total loss of the whole sequence is:

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T} \text{loss}_t \tag{2.14}$$

The decoder's ability to freely generate words in any order – including words such as "beat" that do not appear in the source text – makes the sequence-to-sequence model a potentially powerful solution to abstractive summarization.

2.5.4 Pointer Mechanism

Often in texts with many named entities such as names of people, places, and organizations, many of the words are uncommon resulting in the probability of them being generated is small. A solution to this is the use of a *pointer mechanism*. This can be the thought of as a binary switch that determines if at a given time step t the next token should be generated by the generative model, or by copying the token from the source text (the article) from which it draws the most *attention*. By this method, the generator is capable of generating rare tokens as well as common ones.

The decision can be formulated as a binary variable g indicating whether a token is copied or not. In the baseline model [4] this decision is made by a single dense layer with a sigmoid activation, taking the hidden state of the decoder h_t^{dec} and the the context vector c_t as inputs. The output value ranges from 0 and 1, and represents the confidence if the pointer mechanism should be used. The probability distribution over the input sequence decides which token i from the input should be 'copied'. In our model, the probability from attention mechanism α_t is reused for this. The pointer mechanism is fully differentiable and can be trained along with the rest of the model.

$$p_g(g=1) = \sigma\left(W_{copy}\left[h_t^{\text{dec}}; c_t\right]\right)$$

$$p_{\text{copy}}\left(y_t^i\right) = \alpha_{ti}$$

The final distribution over the output of the decoder module computed as in eq. 2.15

$$p\left(y_t\right) = p_g(g=1)p_{\text{copy}}\left(y_t\right) + p_g(g=0)p_{\text{vocab}}\left(y_t\right) \tag{2.15}$$

2.5.5 Beam Search

Once the neural network is trained, we need to perform a sequence generation process to produce the final result. At a given time step t, the decision on which token should be returned, or , can be made in different ways, and is not trivial. In order to find the best possible summary, we have to explore the space of possible summaries. What a model typically outputs is not one correct solution but rather a probability distribution of possible solutions. Thus, the solution with the best probability needs to be found.

An exhaustive *breadth first search* is not feasible. For example, if we consider a sequence of length 10 (which is short) with 100 possible tokens at each step, it results in a search of 10^{100} possible sentences. In the case of summarization, the sequence length and possible tokens is even larger so this approach is intractable.

One possible solution to search the output space of a model is to perform a *greedy search* where during decoding the returned token for any time step is determined solely by the output of the network at the given time step. This is usually sub-optimal and naive.

As shown in Fig. 2.11, we can generate a sequence with greedy search, i.e.,

$$y_t^* = \arg\max_{y_t \in \mathcal{V}} \log P_\theta\left(y_t | y_{<t}, x\right) \tag{2.16}$$

at each decoding step t. Although greedy search is computationally efficient, human-readability of generated summaries is low.

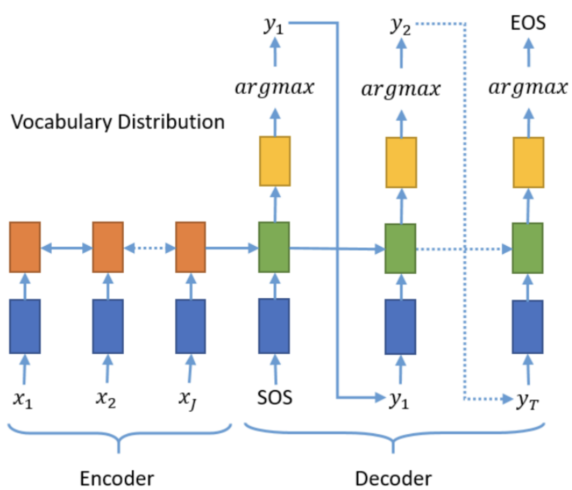

Figure 2.11: Generation process with greedy search

A common alternative produce the most probable sequence is *beam search*. Beam search algorithm is a compromise between greedy search and exact inference and has been commonly employed in different language generation tasks [29] [30]. Beam search is a graph-search algorithm that generates sequences from left to right by retaining only B top scoring (top-B) sequence-fragments at each decoding step. More formally, we denote decoded top-B sequence fragments, also known as hypotheses [30], at time-step $t-1$ as $y_{<t,1}, y_{<t,2}, \ldots, y_{<t,B}$ and their scores as $S^{bm}_{<t,1}, S^{bm}_{<t,2}, \ldots, S^{bm}_{<t,B}$.

For each fragment $y_{<t,b}$, we first calculate $P_\theta\left(y^{\text{cand}}_{t,b,b'}|y_{<t,b}, x\right)$ which determines B most probable words $y^{\text{cand}}_{t,b,1}, y^{\text{cand}}_{t,b,2}, y^{\text{cand}}_{t,b,B}$ to expand it

$$S^{\text{cand}}_{t,b,b'} = S^{\text{bm}}_{<t,b} \times P_\theta\left(y^{\text{cand}}_{t,b,b'}|y_{<t,b}, x\right)$$

$$\text{where } S^{\text{bm}}_{<t,b} \text{ is initialized with } 1, \text{ or} \tag{2.17}$$

$$S^{\text{cand}}_{t,b,b'} = S^{\text{bm}}_{<t,b} + \log P_\theta\left(y^{\text{cand}}_{t,b,b'}|y_{<t,b}, x\right)$$

where $S^{\text{bm}}_{<t,b}$ is initialized with 0. Here b and b' are labels of a current hypothesis and a word candidate, respectively. This yields $B * B$ expanded fragments, i.e., new hypotheses, in which only the top-B of them along with their scores are retained for the next decoding step. This procedure will be repeated until 'EOS' token is generated. In Figure 2.5.5, we show pseudocodes of a beam search algorithm for generating summaries with attention-based seq2seq models given the beam size of B and batch size of 1.

2.5.6 Pointer Generator Network

The pointer generator is a hybrid between encoder-decoder with attention and a pointer network [31], as it allows both copying words via pointing, and generating words from a fixed vocabulary. In the pointer-generator model (shown in Figure 2.13) the attention distribution a^t and context vector h^*_t are calculated as in Section 2.5.3 . In addition, the *generation probability* $p_{gen} \in [0, 1]$ for time step t is calculated from the context vector h^*_t, the decoder state s_t and the decoder input x_t

$$p_{\text{gen}} = \sigma\left(w^T_{h^*} h^*_t + w^T_s s_t + w^T_x x_t + b_{\text{ptr}}\right) \tag{2.18}$$

where vectors w_{h^*}, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab} or copying a word from the input sequence by sampling from the attention distribution a^t. For each document let the *extended vocabulary* denote the union of the vocabulary, and all words appearing in the source document. We obtain the following probability distribution over the extended vocabulary

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a^t_i \tag{2.19}$$

Note that if w is an out-of-vocabulary (OOV) word, then $P_{vocab}(w)$ is zero; similarly if w does not appear in the source document, then $\sum_{i:w_i=w} a^t_i$ is zero. The ability to produce

Algorithm 1: Beam search algorithm for decoding the basic attention-based seq2seq models.

Input: Source article x, beam size B, summary length T, model parameters θ;
Output: B-best summaries;

Initialize:
Output sequences $Q^{\text{seq}} = [\text{SOS}]_{B\times T}$;
Accumulated probabilities $Q^{\text{prob}} = [1.0]_{B\times 1}$;
The last decoded tokens $Q^{\text{word}} = [\text{SOS}]_{B\times 1}$;
States (hidden and cell states for LSTM) $Q^{\text{states}} = [0.0]_{B\times |h_t^d|}$;
Context vectors $Q^{\text{ctx}} = [0.0]_{B\times |z_t^e|}$;

Compute $(h_1^e, h_2^e, ..., h_J^e)$ with encoder;
Update Q^{states} with encoder states;
for t=1, T **do**
  Initialize candidates $Q^{\text{cand,seq}}$, $Q^{\text{cand,prob}}$, $Q^{\text{cand,word}}$, $Q^{\text{cand,states}}$, $Q^{\text{cand,ctx}}$ by repeating Q^{seq}, Q^{prob}, Q^{word}, Q^{states} and Q^{ctx} B times, respectively;
  for b=1, B **do**
    Compute $P_\theta(y_{t,b}^{\text{cand}}|y_{<t,b}, x)$ using decoder LSTM cell with input $(h_1^e, h_2^e, ..., h_J^e)$, Q_b^{word}, Q_b^{states} and Q_b^{ctx};
    Select the top-B candidate words $y_{t,b,b'}^{\text{cand}}$, where $b' = 1, 2, ..., B$;
    Select corresponding probability $P_\theta(y_{t,b,b'}^{\text{cand}}|y_{<t,b}, x)$, hidden states $h_{t,b,b'}^d$, cell states $c_{t,b,b'}^d$ and context vector $z_{t,b,b'}^e$;
    Update elements of $Q_{b',b,t}^{\text{cand,seq}}$, $Q_{b',b}^{\text{cand,word}}$ with $y_{t,b,b'}^{\text{cand}}$;
    Update elements of $Q_{b',b}^{\text{cand,states}}$ with $h_{t,b,b'}^d$ and $c_{t,b,b'}^d$;
    Update elements of $Q_{b',b}^{\text{cand,ctx}}$ with $z_{t,b,b'}^e$;
    Update $Q_{b',b}^{\text{cand,prob}}$ with Eq.(77);
  end
  Flatten $Q^{\text{cand,prob}}$ and choose B best hypotheses;
  Update Q_t^{seq}, Q^{prob}, Q^{word}, Q^{states}, Q^{ctx} with corresponding candidates.
end

Figure 2.12: Beam Search Algorithm

OOV words is one of the primary advantages of pointer-generator models; in contrast to being restricted to a pre-set vocabulary. The loss function is as described in equations 2.13 and 2.14, but with respect to our modified probability distribution $P(w)$ given in equation 2.19.

Repetition is a common problem for sequence-to-sequence models and is especially pronounced when generating multi-sentence text. A coverage model is used to solve the problem. In the coverage model, a coverage vector c^t is maintained which is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{2.20}$$

Intuitively, c^t is a (unnormalized) distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far. Note that c^0 is a zero vector, because on the first timestep, none of the source document

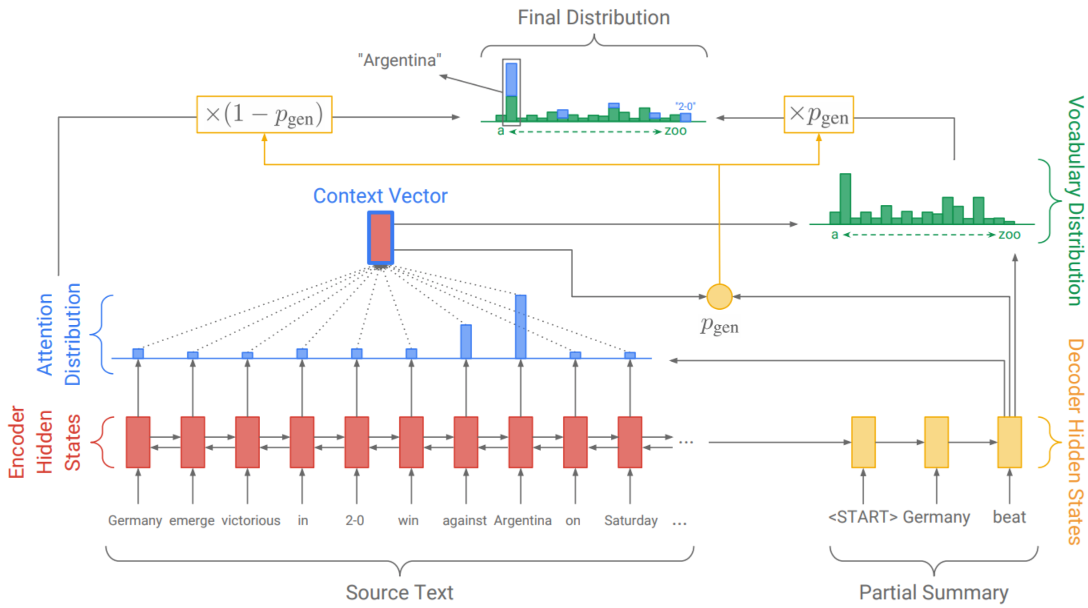

Figure 2.13: Pointer-generator model. For each decoder time step a generation probability $p_{gen} \in [0,1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article words such as *2-0* are included in the final distribution.

has been covered.

The coverage vector is used as extra input to the attention mechanism, changing attention distribution to:

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}\right) \tag{2.21}$$

where w_c is a learnable parameter vector of same length as v. This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarized in c^t). This should make it easier for the attention mechanism to avoid repeatedly attending to the same locations, and thus avoid generating repetitive text.

Its necessary to define an additional *coverage loss* to penalize repeatedly attending to the same locations:

$$\text{cov loss}_t = \sum_i \min\left(a_i^t, c_i^t\right) \tag{2.22}$$

The coverage loss is bounded:

$$\text{cov loss}_t <= \sum_i \min(a_i^t, c_i^t) \tag{2.23}$$

The coverage loss function is different and more flexible to the coverage loss in machine translation. This is because summarization should require uniform coverage, but only penalize the overlap between each attention distribution and the coverage so far - preventing repeated attention. Finally, the coverage loss, reweighted by some hyperparameter λ, is added to the primary loss function to yield a new composite loss function:

$$\text{loss}_t = -\log P\left(w_t^*\right) + \lambda \sum_i \min\left(a_i^t, c_i^t\right) \tag{2.24}$$

2.5.7 Reinforcement Learning

An interesting approach to training models for sequential data is to look at the problem from the perspective of reinforcement learning (RL). In this interpretation, the parametric model represents the agent that follows a policy, the previously generated tokens are treated as the state of the environment, and the problem of generating the next token is viewed as choosing the next action. The goal of the training procedure is to find a policy that will allow the agent to maximize a reward that is accumulated through time. The reward can be any user-defined metric without the requirement of being differentiable.

One RL method that has been used within text generation is the REINFORCE algorithm [32] which focuses on maximizing the immediate reward r, which is a one step Markov Decision Process (MDP). The algorithm's objective is to find an optimal set of parameters θ^* for a stochastic policy $\pi_\theta(\sigma, a)$.

Considering a one step MDP, starting from a state σ drawn from the state distribution $d(\sigma)$ and the one step reward $r_{a,\sigma}$, the loss function $\mathcal{L}_{\pi_\theta}$ to be minimized with respect to the parameters θ^* of the policy $\pi_\theta(\sigma, a)$ can be formulated as the negative expected reward r:

$$\mathcal{L}_{\pi_\theta}(\theta) = -\mathbb{E}_{\pi_\theta}[r] = -\sum_{\sigma\in\mathcal{S}} d(\sigma) \sum_{a\in\mathcal{A}} \pi_\theta(\sigma, a) r_{\sigma,a} \tag{2.25}$$

where $\mathcal{S}$ and $\mathcal{A}$ are all available states and actions, respectively.

In order to be able to update θ, an expression $\nabla_\theta \mathcal{L}_\pi(\theta)$ is needed. By using the fact that $\nabla_\theta \log \pi_\theta(\sigma, a) = \pi_\theta(\sigma, a)\frac{\nabla_\theta \pi_\theta(\sigma,a)}{\pi_\theta(\sigma,a)}$ in combination with the gradient of a logarithm, $\nabla_\theta \log \pi_\theta(\sigma, a) = \frac{\nabla_\theta \pi_\theta(\sigma,a)}{\pi_\theta(\sigma,a)}$, an analytic expression of the gradient of the loss function can be defined as

$$\begin{aligned}\nabla_\theta \mathcal{L}_\pi(\theta) &= \sum_{\sigma\in\mathcal{S}} d(\sigma) \sum_{a\in\mathcal{A}} \pi_\theta(\sigma,a)\nabla_\theta \log \pi_\theta(\sigma,a) r_{\sigma,a} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(\sigma,a) r\right]\end{aligned} \tag{2.26}$$

where r is a reward. $\nabla_\theta \mathcal{L}_\pi(\theta)$ is called the *policy gradient* for the REINFORCE algorithm since it enables the parameters θ of the policy $\pi_\theta(\sigma,a)$ to be updated using backpropagation.

When evaluating a policy $\pi_\theta(\sigma,a)$, it can be of interest to evaluate the policy after performing a subsequence of actions $a_{1:T} = [a_1,\ldots,a_t], t < T$ in the state space i.e before a complete action sequence $a_{1:T}$ has been performed. This can be done using *Monte Carlo Tress Search*, in which a subsequence of actions $a_{1:T}$, the rest of the action sequence $a_{t+1:T}$ is then sampled using the policy $\pi_\theta(\sigma,a|\sigma_t,a_t)$ with the current parameters θ. This is referred to as *rollout* and can be performed for multiple values of t in order to evaluate different sub parts of the action sequence.

Using repeated Monte Carlo Tree Search, the expected reward $\mathbb{E}[r_t]$, sometimes referred to as an *action-value function* $Q(\sigma,a)$, can be approximated for π_θ. $\mathbb{E}[r_t]$ measures the cumulative reward, and thereby the overall quality of the given policy.

2.6 Generative Adversarial Network (GAN)

A major advancement in the field of machine learning based generation that has been made in recent years is the invention and cultivation of the concept of generative adversarial networks (GAN). The concept, which was originally presented by Goodfellow et al. in [33], introduces the concept of using two adversarial networks for generative tasks. The original approach which was designed for generating images has subsequently been improved to enable text generation. Examples of such augmented approaches are SeqGAN [5], TextGAN [34] and LeakGAN [25].

GAN can be pictured as a game between two networks, a *generator* $\mathcal{G}$ and a *discriminator* $\mathcal{D}$ as illustrated in Figure 2.14. An analogy which is commonly used is that $\mathcal{G}$ is a counterfeiter who tries to create fake money and $\mathcal{D}$ is a police who tries to tell fake money from real. To win the game, $\mathcal{G}$ must learn how to manufacture money that looks exactly like real money, whereas $\mathcal{D}$ must find measures to identify money as real or fake

In a stricter sense, the aim of $\mathcal{G}$ is to produce samples $\tilde{x}$ such that it is impossible for $\mathcal{D}$ to tell if $\tilde{x}$ is drawn from pr or pg, which are the distributions of real and generated data, respectively. The aim of $\mathcal{D}$ is thus to learn enough about the features of pr in order to know the difference between p_r and p_g.

Mathematically, $\mathcal{G}$ is represented as a function that is differentiable w.r.t. its input z and its parameters $\theta^{\mathcal{G}}$. Typically, $\mathcal{G}(z)$ is represented by a deep neural network. Associated with

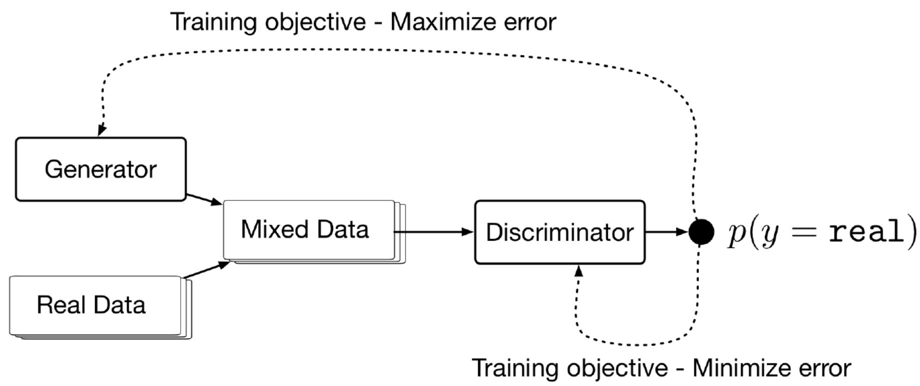

Figure 2.14: A schematic representation of the GAN training strategy

$\mathcal{G}(z)$ is a cost function $\mathcal{L}_{\mathcal{G}}(\theta^{\mathcal{G}})$ which the generator wishes to minimize.

$\mathcal{D}$ is also typically represented as a deep neural network, however not necessarily of the same kind as $\mathcal{G}$, and has the same requirements as $\mathcal{G}$: it is differentiable w.r.t. its input, the observed sample x, and its parameters $\theta^{\mathcal{D}}$. The discriminator has a corresponding cost function $\mathcal{L}_{\mathcal{D}}(\theta^{\mathcal{D}})$.

One of the simplest formulations of a GAN is when $\mathcal{G}$ and $\mathcal{D}$ can be viewed as a zero sum game, or a minimax game, where a value function $\mathcal{L}_{GAN} = \mathcal{L}_G = -\mathcal{L}_D$ can be defined to describe the total payoff of the game. Solving (i.e. finding the Nash equilibrium of) the minimax game, corresponds to finding the optimal generator parameters

$$\theta^{\mathcal{G}^*} = \arg\min_{\theta^{\mathcal{G}}} \max_{\theta^{\mathcal{D}}} \mathcal{L}_{GAN}\left(\theta^{\mathcal{D}}, \theta^{\mathcal{G}}\right) \tag{2.27}$$

There are various ways of defining the loss functions for $\mathcal{G}$ and $\mathcal{D}$ Goodfellow [7] introduces the *cross entropy* as the loss function

$$\mathcal{L}_{GAN} = \mathbb{E}_{x\sim p_r} \log \mathcal{D}(x) + \mathbb{E}_{z\sim p_g} \log[1 - \mathcal{D}(\mathcal{G}(z))] \tag{2.28}$$

The training process of a GAN consists of alternating backpropagation. Two batches of samples are drawn: x from the reference data set described by p_r and $\tilde{x} = \mathcal{G}(z)$ from $p_g(z; \theta^{\mathcal{G}})$. The discriminator parameters $\theta^{\mathcal{D}}$ are then updated using the gradient

$$\nabla_{\theta^D} \mathcal{L}_{\mathcal{D}}\left(\theta^{\mathcal{D}}, \theta^{\mathcal{G}}\right) = -\frac{1}{N} \sum_{n=1}^{N} \left[\nabla_{\theta^{\mathcal{D}}} \log \mathcal{D}\left(x^n\right) + \nabla_{\theta^{\mathcal{D}}} \log\left(1 - \mathcal{D}\left(\tilde{x}^n\right)\right)\right] \tag{2.29}$$

Once the discriminator is updated, new samples $\tilde{x}$ are sampled from $p_g(z;\theta^{\mathcal{G}})$, and the generator is updated by stochastic gradient descent with the gradient

$$\nabla_{\theta^G}\mathcal{L}_{\mathcal{G}}\left(\theta^{\mathcal{G}}\right)=\frac{1}{N}\sum_{n=1}^{N}\nabla_{\theta^{\mathcal{G}}}\log\left(1-\mathcal{D}\left(\tilde{x}^n\right)\right)\Bigg) \tag{2.30}$$

A common problem with GAN is that the set of hyperparameters for which training is successful is small. This is well known, but not thoroughly researched, as mentioned by Arjovsky and Bottou [35]. Another major problem with GAN is mode collapse, which results in $\mathcal{G}$ generating a few distinct set of samples. The cause of *mode collapse* is that the generator encompasses only a small subset of the modes in p_r [36].

2.6.1 Text Generation with GAN

One limitation of GAN is that they are designed to generate continuous, real-valued data, such as images [5]. Text, however, consists of discrete tokens (words, letters, punctuation etc), which makes the original GAN approach [33] inapplicable due to the non-differentiability of discrete samples. It is thus impossible to propagate the gradient from the discriminator back to the generator using the original GAN approach as described in Section 2.6.

Another issue related to text generation with GAN is vanishing gradient in the sense that as the discriminator $\mathcal{D}$ improves, the gradient of the generator $\mathcal{G}$ disappears, as proven theoretically and demonstrated in experiments by Arjovsky and Bottou [37]. The implication is that the contribution of $\mathcal{G}$ to the learning signal vanishes as $\mathcal{D}$ approaches a local optimum [36].

2.6.2 SeqGAN

SeqGAN, proposed by Yu et al. [5], is an attempt at generating text with GAN which addresses the issue of the discrete nature of text, as described in Section 2.6.1, by treating the generator as an agent in reinforcement learning, which was presented in Section 2.5.7. SeqGAN is also the foundation for more advanced emerging techniques in text generation such as *LeakGAN* and *TextGAN*. The work undertaken in this thesis is motivated by the SeqGAN approach for text generation.

The sequence of generated tokens is viewed as the state σ of the system and the next token to be generated is the action that the agent must decide upon. The agent is controlled via a stochastic parametrised policy which is trained using policy gradient and Monte Carlo search. Since the agent in this case is the generator, the generator policy is denoted as $\mathcal{G}_\theta := \pi_\theta$. The objective of the stochastic policy model (the generator) $\mathcal{G}_\theta(y_t|\tilde{s}_{1:t-1})$ is to generate a sequence $\tilde{s}_{1:T}=[y_1,\ldots,y_t,\ldots,y_T]\,,y_t\in V$, where V is the vocabulary of available tokens, from the start state σ_0 that maximises its action value function $Q(\sigma,a)$, which, in the case of SeqGAN, is decided by the discriminator $\mathcal{D}_\phi$. Thus, the cost function which the generator seeks to maximise is

$$\mathcal{L}_\mathcal{G}(\theta) = \sum_{y_1 \in V} \mathcal{G}_\theta\left(y_1|\sigma_0\right) \cdot Q\left(\sigma_0, y_1\right)$$

In the case of SeqGAN, for an entire sequence of length T,

$$Q\left(\sigma = \tilde{s}_{1:T-1}, a = y_T\right) = \mathcal{D}_\phi\left(\tilde{s}_{1:T}\right)$$

which means that the reward for an entire generated sequence is the verdict of the discriminator $\mathcal{D}$

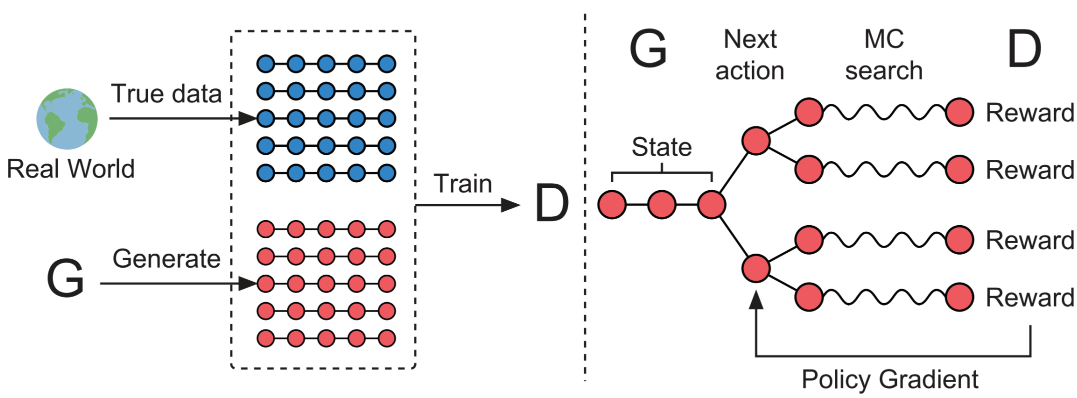

Figure 2.15: Left: Discriminator gets real and fake examples for training. Right: Shows the current state and next action representation. Also shows how Monte-Carlo search is used to reach the reward and policy gradient is used to update the policy.

The generator does, however need feedback for the intermediary time steps $(t < T)$ as well. To get a measure of the quality of the generator policy, Monte Carlo search is utilised by, starting at time step $t < T$, following $\mathcal{G}_\theta N$ times to generate N different sequences $MC\widetilde{S}^n_{1:T}$ as described in Section 2.5.7. Each sequence is then rewarded by $\mathcal{D}_\phi$. The total expected reward is the average value of the N scores, wherefore the action-value function

$$Q\left(\sigma = \tilde{s}_{1:T-1}, a = t_T\right) = \left\{ \frac{1}{N} \sum_{n=1}^{N} \mathcal{D}_\phi \left(\begin{array}{ll} MC^n_{1:T} & \text{for } t < T \\ \mathcal{D}_\phi\left(\tilde{s}_{1:t}\right) & \text{for } t = T \end{array} \right. \right. \tag{2.31}$$

$\mathcal{D}_\phi$ is updated by minimizing the cross entropy cost function.

To update the generator with gradient descent $\nabla_\theta \mathcal{L}_\mathcal{G}(\theta)$ has to be approximated. Yu et al. [5] do this in accordance with REINFORCE algorithm described in Section 2.5.7 as

$$\nabla_\theta \mathcal{L}_\mathcal{G}(\theta) \approx \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{y_t \sim \mathcal{G}_\theta(y_t|\tilde{s}_{1:t-1})} \left[\nabla_\theta \log \mathcal{G}_\theta \left(y_t | \tilde{\boldsymbol{s}}_{1:t-1} \right) \cdot Q \left(\tilde{s}_{1:t-1}, y_t \right) \right] \tag{2.32}$$

Since the expectation value $\mathbb{E}[.]$ can be approximated via sampling, the parameters θ of $\mathcal{G}_\theta$ can be updated as

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_\mathcal{G}(\theta) \tag{2.33}$$

with learning rate α.

The SeqGAN algorithm can be summarized as below in Fig 2.16:

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
    for g-steps **do**
        Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
        for t in $1:T$ **do**
            Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
        end for
        Update generator parameters via policy gradient Eq. (8)
    end for
    for d-steps **do**
        Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
        Train discriminator D_ϕ for k epochs by Eq. (5)
    end for
    $\beta \leftarrow \theta$
until SeqGAN converges

Figure 2.16: SeqGAN Algorithm

2.7 Evaluation Methods

An important aspect of text generation is evaluation of the generated samples. Using a well working metric not only aids in parameter tuning, but also enables comparison to previous achievements within the field. For text generation, two commonly used metrics are the *Bilingual Evaluation Understudy* (BLEU) score and the *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) score. They differ in that the *precision* based BLEU measures how much of the generated text appears in the reference text and that the *recall* based ROUGE measures the opposite; how much of the reference text appears in the generated text. ROUGE is used for evaluation in this thesis and is presented in greater detail below. When using both methods to evaluate generated sequences, each sequence is first stripped of the special tokens SOS, EOS and PAD, which are described in Section A.4.

2.7.1 Recall-Oriented Understudy for Gisting Evaluatiion (ROUGE)

The metric *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) introduced by Lin [38] is commonly used for evaluation of text summarisations and is used by, among others, [4]. A ROUGE score of 1.0 corresponds to a perfect match with the reference sequence, whereas a score of 0.0 means no overlap at all. ROUGE consists of several different metrics, which are presented below:

- ROUGE-N compares the token overlap of N-grams between the generated and reference sequences, where N = 1, 2, 3, 4 are the most common.

- ROUGE-L Uses the concept of *Longest Common Subsequence* (LCS) to compare sequences. This way, sequence level structure is captured. The measure is thus good at detecting sequences with similar tokens ordered in similar ways, but not good at finding syntactically different sequences with similar semantics.

- ROUGE-W is an augmented version of ROUGE-L that takes the spatial relations of the tokens within the sequences into account. This is best explained with an example used in the original paper on ROUGE [38]: picture the reference sequence $s = [ABCDEFG]$ and the two generated candidates $\tilde{s}_1 = [ABCDHIJ]$ and $\tilde{s}_1 = [AHBICJD]$ where $A, \ldots, J$ are different tokens. ROUGE-L assigns $\tilde{s}_1$ and $\tilde{s}_2$ the same score since they both contain the subsequence [ABCD], which is counterintuitive. From a semantic perspective, $\tilde{s}_1$ is clearly superior to $\tilde{s}_2$. ROUGE-W tackles this situation by using a weight which favours consecutive sub-sequences while not disregarding other sub-sequences.

- ROUGE-S counts the number co-appearances of skip-bigrams (pairs of tokens ordered as in the sequence) in the candidate and reference sequences.

- ROUGE-SU extends ROUGE-S to include unigrams in addition to bigrams.

Of the available metrics ROUGE-1 and ROUGE-2 (ROUGE-N with N = 1 and 2 respectively), ROUGE-L, and ROUGE-SU are the most commonly used. ROUGE-1 and ROUGE-2 are the ones that are used in this thesis. Formula for calculating ROUGE scores are described in Section 4.4.1

Since ROUGE measures only the degree of overlapping n-grams between two sets of summaries, it is often debated by NLP researchers what exactly is a good ROUGE score? And even if that is established, it still questionable if it naturally describes what a human's idea of a good quality summary is.

2.7.2 Limitations of ROUGE

While models for abstractive summarization have improved by leaps and bounds, their performance against the ROUGE score seem to be plateauing. The limitations of evaluating summaries against the ROUGE score have been illustrated in Schluter's work [39]. The paper proves that building a model that is optimising to maximize the ROUGE score is an NP-hard problem. Finally, ROUGE is dependent on a accompanying reference human-written summary. This is not feasible especially when summarization models are run on large non-standard datasets where human-written summaries do not exist. It is necessary to evaluate a model's summarization performance on its own. For these reasons, until better alternative evaluation scores to the ROUGE become widely accepted, one might have to evaluate a model's performance, especially qualitative, through human assessment of a sample from the data set.

2.7.3 Qualitative Evaluation

Since GANs are trained to produce more realistic human-like summaries, they do not optimize for traditional loss functions such as cross-entropy which is used in maximum likelihood methods. Therefore in order to evaluate the qualitative performance of samples generated, models are typically evaluated using human evaluation on various language criteria. This is undertaken in many different ways depending on the linguistic aspect under consideration.

Chapter 3

Related Work

The work completed as part of the thesis project touches on four topics:

- the task of abstractive text summarization
- strategies for training recurrent neural networks for structured sequence generation
- use of pointer networks
- generating text using methods motivated by reinforcement learning and generative adversarial networks

All the mentioned topics are considered unsolved problems and are the subject of intense research. This chapter briefly summarizes the work done in each of these sub-domains.

3.1 Neural Abstractive Text Summarization

The concept of automated text summarization dates back to the fifties when Luhn proposed that a convenient measure of word significance is in fact its frequency in a document [40]. Through the seventies and eighties linguistic and statistical methods emerged. These approaches represented word for word extraction of sentences. This group can be categorized as heuristic approaches, where words or sentences are rank-conformed features [41]. In the more recent years attention has been shifted towards abstractive approaches and multidocument summarization i.e. machine learning methods like Naive-Bayes, Hidden Markov models, Neural Networks and deep language analysis, where discourse structure is taken into account with e.g. lexical chains and rhetorical structure theory [42] . Before the popularization of neural network models in the domain of natural language processing, the main approaches to the task of abstractive text summarization relied on classical NLP methods, such as statistical models [43] [44] or parsing [45]. Publications of Sutskever et al. [46] and Cho et al. [47] describing the *encoder-decoder* architecture and its use in the context of *sequence-to-sequence* modeling enabled the application of neural models to tasks within the field of NLP. Their novel architecture allowed to replace multi-component solutions that heavily depended on human-crafted features and domain knowledge, with models that were trainable in an end-to-end approach and which was capable of scaling to large corpora.

Rush et al. [30] were the first to apply modern neural networks to abstractive text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets. Their approach, which is centered on the attention mechanism, have since been enhanced further with the use of recurrent decoders (Chopra et al., [48]), abstract meaning representations (Takase et al., [49]), hierarchical networks (Nallapati et al., [8]), variational autoencoders (Miao and Blunsom, [50]), and direct optimization of the performance metric (Ranzato et al., [51]), thus further improving performance on those datasets.

Nallapati et al. [8] adapted the DeepMind question-answering dataset (Hermann et al., [52] for summarization, resulting in the CNN/Daily Mail dataset (the current de facto standard dataset), and established the first baselines for abstractive text summarization. The same authors then published a neural extractive approach (Nallapati et al., [53]), which uses hierarchical RNNs to select sentences, and found that it significantly outperformed their previous result with respect to the ROUGE metric.

Prior to modern neural methods, abstractive summarization was less common than extractive summarization, but Jing [54] explored cutting unimportant parts of sentences to create abstract summaries, and Cheung and Penn [55] explored sentence fusion using dependency trees.

3.2 Training Recurrent Neural Networks

An often used approach for training recurrent neural networks for modeling sequential data is maximum likelihood estimation (MLE) combined with a procedure called teacher forcing [32]. The goal is to maximize the likelihood of each successive token present in the target sequence where the predictions are conditioned on an input token and a history of previously generated tokens. In teacher forcing, the input token, and effectively the history of previous tokens on which the prediction is conditioned, are taken directly from the ground-truth sequence. Despite its wide use, the mentioned method has been shown to be a suboptimal strategy in the context of training recurrent neural models for language generation and related tasks [56] [57] [58]

The main issue, which is caused by teacher forcing, is called *exposure bias* [59]. This refers to the situation where during training the prediction of the next word conditioned on the previous word becomes infeasible since the previous word may not have been seen in the training data. If the generator makes an error early on in the generation process, the generated sentence will keep diverging further away as more words are generated. Since the inputs to the trained model come directly from the ground-truth sequence, the model is not exposed to its errors during the training procedure. At inference time target, where labels are not available, tokens decoded at the previous time step are fed into the network as input in place of the ground-truth sequence. An example of exposure bias is shown in Figure 3.1

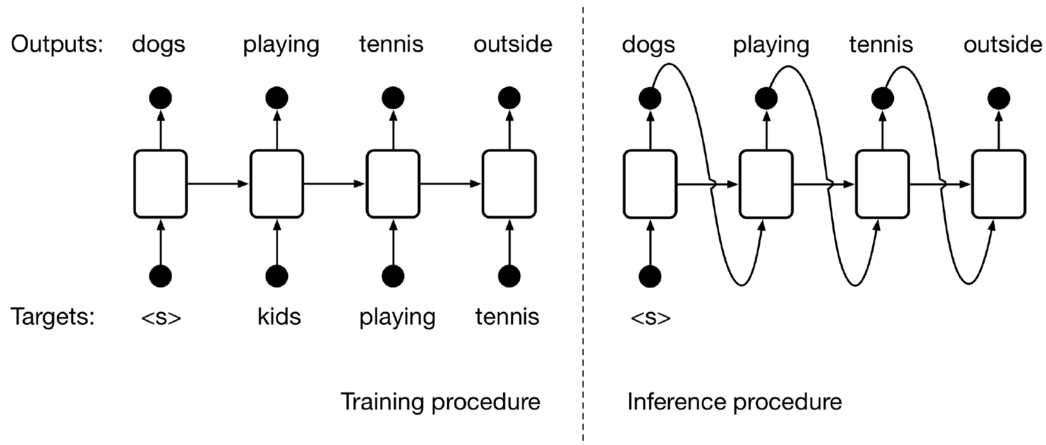

Figure 3.1: Illustration of the exposure bias which is the cause of suboptimal performance of generative models at inference time. During training the network is not exposed to its own outputs, at inference the generated tokens are passed back into the network. The autoregressive nature of the model at inference time might cause it work with sequences it has not seen during training

As the decoding progresses in time, errors made early in the decoding process might accumulate and the model might condition its prediction on sequences that it has never seen during training, thus yielding low quality output sequences. From a probabilistic perspective, the problem can be seen as using the models with data coming from different probability distributions during training and inference time. As observed in [57], the dynamics of the hidden states of the recurrent neural network are also affected and significantly differ between training and inference usage. The second drawback of the described training strategy is the *loss-evaluation mismatch* [59] where the metric used to optimize the model during training is in most cases distinct from the metric used to evaluate the performance after the training is finished. The task of abstractive text summarization can serve as one example of this issue. During training, the optimized metric is the sum of the negative loglikelihoods of the ground-truth sequence tokens, but the same models are later evaluated and compared based on sequence-level metrics such as BLEU [60], ROUGE [38], or METEOR [61]

Many extensions to the MLE-teacher forcing strategy as well as alternative methods of training sequential models have been proposed by the research community to address the issues. An early attempt at extending the teacher forcing strategy was made by Bengio et al. [56]. The authors proposed a technique called *scheduled sampling* that aims to prepare the network for working with its own outputs during the training procedure. The main intuition in the work was that during training, the source of the token that is fed as input to the network, ground-truth or sampled, should be decided by a coin flip. The method used a *curriculum learning* strategy, where the probability of the token being sampled from distribution output by the model in the previous step rather than copied from the target sequence increased with the length of the generated sequence. This caused the training procedure to go from fully guided by the ground-truth labels to a more free-running one. The hope was to expose the network to its own outputs and errors that it makes during the training procedure, which should bridge the gap between how the model is used during training and inference. Although empirical results on three independent sequence modeling tasks show that scheduled sampling improves the performance of trained models, a thorough analysis of the method done in Huszar [58] suggested that this strategy is a biased estimator and may not yield optimal models.

With regards to beam search, Wiseman and Rush [59] propose using beam search not only during test time, but also as part of the training procedure. The intuition behind the work was to unify the usage of the model across the mentioned modes. Similarly to test time, at each decoding step, the model keeps track of K most promising subsequences (paths) in the output space (state graph). A specially designed loss function penalized the model only in situations where the groundtruth sequence ranked below the K most promising sequences generated by the model. The loss function also incorporated a factor that evaluated the subsequences using sequence-level metrics, such as the mentioned ROUGE or BLEU, and provided additional guidance during the training procedure.

3.3 Pointer-generator networks

As the pointer-generator network of See [4] is chosen as the baseline abstractive summarization model for this thesis, it is worth highlighting prior work in the field, specifically related to pointer networks.

The pointer network (Vinyals et al., [31] was originally proposed to solve problems such as convex hull, delaunay triangulation, and traveling salesman problems. Since then it has found application in s NLP tasks such as Neural Machine Translation (NMT) [62], language modeling [63], and summarization [64] [36]. It is a sequence-to-sequence model that uses the soft attention distribution of Bahdanau et al. [65] to produce an output sequence consisting of elements from the input sequence. The pointer network has been used to create hybrid approaches for

The pointer approach used by the baseline model of See et al. [4] chosen for this thesis is close to the forced-attention sentence compression model of Miao and Blunsom [50] and the CopyNet model of Gu et al. [64], with some minor variations: (i) The model calculates an explicit switch probability p_{gen}, whereas Gu et al. induce competition through a shared softmax function. (ii) The model recycles the attention distribution to serve as the copy distribution, but Gu et al. use two separate distributions. (iii) When a word appears multiple times in the source text, the model sums probability mass from all corresponding parts of the attention distribution, whereas Miao and Blunsom do not. The reasoning of See et al. [4] is that (i) calculating an explicit p_{gen} usefully enables to raise or lower the probability of all generated words or all copy words at once, rather than individually, (ii) the two distributions serve such similar purposes that the simpler approach suffices, and (iii) the pointer mechanism often copies a word while attending to multiple occurrences of it in the source text.

The approach is considerably different from that of Gulcehre et al. [62] and Nallapati et al. [8] . Those works train their pointer components to activate only for out-of-vocabulary words or named entities (whereas See et al.'s model is allowed to freely learn when to use the pointer), and they do not mix the probabilities from the copy distribution and the vocabulary distribution.

3.4 Text Generation using Reinforcement Learning and GAN

Some recent advancements in the field of text generation [5] [25], have approached text generation using *Reinforcement Learning* (RL), which is based on the concept of having an agent acting in an environment, i.e. a state space. Each *action a* in the state space corresponds to a *reward r*, and the object of the agent is to, based on its *state* σ, take actions that maximizes its cumulative reward. In order to decide which actions to take, the agent uses a stochastic *policy* π_θ parametrised by θ, which recommends certain actions based on the state of the agent. This perspective on generating structured sequential outputs was explored in multiple works by the research community.

One of the early works utilizing such an approach was the mixed incremental cross-entropy reinforce MIXER algorithm proposed by Ranzato et al. [51]. In their work, the authors challenged both of the problems related to teacher forcing, they aimed to prepare the model for working with its own outputs, instead of the ground-truth labels, and directly optimized the test-time metric during training. The proposed algorithm was based on the policy gradient method REINFORCE [32] and applied extensions to make training feasible for tasks with large action spaces, e.g. language generation where the action space is equal to the size of the vocabulary. One of the key concepts of the proposed algorithm was a hybrid loss that used standard cross-entropy to evaluate prefixes of sequences, and the REINFORCE rewards to evaluate the remaining suffixes. An incremental (curriculum) training procedure was used to choose the dividing point between the prefix and the suffix of the sequences. This allowed to smoothly transition from using exclusively teacher forcing, to benefit from token-level supervision, into using exclusively REINFORCE, to utilize sequencelevel supervision, as the training progressed in time. The intuition of this procedure was to use the available ground-truth sequences to guide the model (policy) in the early phases of training and allow it to work with its own outputs in the later phases. Based on an empirical investigation, the authors reported that the MIXER algorithm with greedy decoding outperformed several other training strategies, including an optimized teacher forcing approach combined with beam search decoding, on common benchmark datasets.

Bahdanau et al. [66] proposed a different RL-based algorithm aimed to reach the same goals as Ranzato et al. [51]. The presented method was inspired by the actor-critic [67] approach, which is a training strategy based on policy gradient methods where two models, the actor and the critic, interact with each other. In their work, the authors used an encoder-decoder architecture [23], for both the actor (generative) model and the critic. During training, the actor model operates as a regular generative model, consuming the output it generated in the previous step, and the critic evaluates each generated token using a user-defined metric. Choosing a metric that will be used to evaluate the generative model at test time allows to the model to directly optimize it during training. The scores generated by the critic were treated as estimates of gradients with regards to the chosen metric and used as training signal by the actor model. Utilizing the fact that most NLP tasks are in the domain of supervised learning, the author proposes to condition the critic model on the ground-truth sequence, which can be seen as a sequence of perfect actions, which is a step away from a pure reinforcement learning approach. In the mentioned work, one of the tested tasks was neural machine translation, where the authors optimized the BLEU [60] score and achieved competitive results on benchmark datasets.

As mentioned in Section 2.6, Generative Adversarial Networks (GANs) is a different approach to training recurrent neural networks, which has recently gained the interest of the research community, makes use of the adversarial network training framework. Due to the specific architecture used in GANs, adversarial network training is a natural choice for generative models with continuous outputs, such as images. However, recent work in the field has shown that the constraint of having discrete outputs, such as text, is possible to overcome, which has enabled expanding adversarial network training to text.

Yu et al. [5] introduced a novel approach for text generation using GANs in the paper *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. Text generation is treated as a reinforcement learning problem where the generator uses a stochastic policy which is optimised using gradient policy update. SeqGAN was tested on real as well as synthetic data and achieved promising results on both.

Lin et al. [68] extended the SeqGAN model under the name *RankGAN*. In their work, the authors replaced the binary-classification discriminator of SeqGAN with a ranker. The task of the ranker was ordering (ranking) a set of sequences by their similarity to a single reference sequence. The reference sequence was always sampled from the ground-truth dataset and the ranked set consisted of a single sequence coming from the generator and multiple sequences sampled from the dataset. From an adversarial training perspective, the goal of the generator was to output sequences that would be ranked high by the ranker and the goal of the ranker was to distinguish the generated sequences and rank them lower than human-written ones. According to the authors using a ranking approach as opposed to of binary-classification served as a better signal for training the generator. The idea of using ranking mechanism was inspired by the learning to rank [67] framework that utilizes machine learning techniques for tasks in the domain of information retrieval (IR). Training of the RankGAN model was done using policy gradient methods.

Fedus et al. [69] proposed *MaskGAN*, a training task and neural architecture that together directly encourage generation of high quality textual sequences. The task used for training was text "in-filling", where the goal of the network was to reproduce the masked (removed) parts of a sequence, given the available context. The MaskGAN model consisted of three networks, an encoder-decoder-based generator and discriminator, and a critic network which shared part of its architecture with the discriminator. During the training procedure, the generator was given a masked sequence as input and expected to produce an output sequence with missing tokens filled in. The task of the discriminator was to assess whether each token was in the original sequence or was filled-in by the generator. From the reinforcement learning perspective, the values predicted by the discriminator were treated as partial rewards for each token. The critic network approximated the discounted reward per token which was used as the signal for training the generator network. Given the specifics of the architecture, the model was not fully differentiable and policy gradients [67] have been used for training the generator. Designing the discriminator to operate on a token-level, as opposed to sequence-level, allowed the authors to avoid the credit assignment problem present in other approaches. The authors claimed that the specific architecture and choice of task allowed the model to avoid the mode-collapse problem which is one of the main issues associated with the GAN framework.

In *Long Text Generation via Adversarial Training with Leaked Information* Guo et al. [25] extend the model by Yu et al. by letting the discriminative network leak features of partly generated text in order to steer the generator towards better performance. The proposed model, LeakGAN, outperforms SeqGAN on synthetic and real data.

Zhang et al. [34] approach text generation with GAN in a different way than Seq-GAN and LeakGAN. Their solution, TextGAN, is presented in Adversarial Feature Matching for Text Generation [34]. Instead of training a policy, the network tries to match high-dimensional latent feature distributions. According to Zhang et al. the approach avoids the common problem of mode collapse.

A common trait of all the presented papers related to adversarial training was that both the generator and discriminator models were pre-trained using MLE before the actual Adversarial training took place. The majority of work describing applications of the GAN framework to problems with discrete output spaces is focused on the task of language modeling, where the generated sequences were not conditioned on any external information. Wu et al. [35] applied the adversarial training paradigm to train models for the task of neural machine translation. In NMT models, as in many other tasks in NLP, the generative process is conditioned on the source sequence, given as input to the model. The authors utilized an encoder-decoder architecture [46] as the generator and a convolutional network as the discriminator. Taking inspiration from professor sampling [57], the discrimination process was designed in a way to force the generated sequences to be as close as possible to the human-written target sequences. The training was done via policy gradient methods, similarly to other work in the domain. The empirical results showed that NMT model trained using adversarial training can achieve competitive performance on standard machine translation benchmarks.

Chapter 4

Methodology

4.1 Data

In order to obtain successful results with deep neural networks, a large set of training data is needed. This section describes the datasets chosen and pre-processing steps taken in order to prepare data for training and running experiements.

4.1.1 Datasets

In the field of text generation, a few datasets are commonly used for model evaluation. Nallapati et al. [8] used a dataset consisting of news articles from the news publishers CNN/Daily Mail[1] and human written summaries. The summaries comprise of multiple sentences and are abstractive in nature. The would aid the model during training to produce novel summaries. Rush et al. [30] used the DUC-2004 and Gigaword datasets.

For the above mentioned reasons, we use the CNN/Daily Mail dataset, which contains online news articles (781 tokens on average) paired with multi-sentence human-written summaries (3.75 sentences or 56 tokens on average). For the purpose of training models, the human-written summaries are assumed to represent 'gold-standard' quality of writing - the quality to which the summarization models are trained to match.

The dataset is divided into three parts - 287,226 training examples, 13,368 validation examples and 11,490 test examples. Scripts supplied by Nallapati et al. [8] were used to obtain the same version of the the data. Both the dataset's published results use the anonymized version of the data, which has been pre-processed to replace each named entity, e.g., *The United Nations*, with its own unique identifier for the example pair, e.g., @*entity5*. By contrast, we operate directly on the original text (or non-anonymized version of the data), which is the favorable problem to solve because it requires no pre-processing. The relatively large size of

[1] https://cs.nyu.edu/ kcho/DMQA/

this dataset also enables experimenting with multiple different neural network architectures which typically require large amounts of training data. The basic statistics of the data set are summarized in the below table,

	Train	**Validation**	**Test**
# of article-summary pairs	287,227	13,368	11,489
Article Length	751	769	778
Summary Length	55	61	58

Table 4.1: Summary of the CNN/Daily Mail dataset

4.1.2 Pre-processing

Data contained in a dataset has to be pre-processed before it is accepted by the model. The pre-processing, which consists of several parts, is described below.

Tokenization

The first step in the pre-processing process is tokenization, which was introduced in Section 2.2.3 and can be performed in different ways, so choosing a good tokenization procedure is important to the outcome of a model.

Pointer-generator networks [4] used a simple tokenization approach to split a sequence on spaces and newlines. This being our baseline mode, for maintaining consistency and comparison of results, this tokenization approach was maintained throughout this work.

However, it is worth mentioning that while this approach is easy to grasp, it does have limitations such as inferior handling of punctuation, capital words and contractions. To illustrate this, consider the sequence *You're not worried, are you?* and the tokenized version in Figure 5.2. In the first token of the sequence, "You're", the capital "Y" means that the token is not equal to "you're", which is troublesome. One simple improvement is thus to lower-case all words in the sequence before splitting into tokens. The result of the naive approach with lower-casing is also presented in Figure 5.2.

Vocabulary Map

The text data that is fed to, processed by, and produced by the model does not consist of strings. Instead, each token is translated to a corresponding integer value. This conversion process is commonly called "one-hot encoding" and is an established practice in NLP [6].

When the generator generates text, it draws integer values from a distribution which is the result of a dense layer with a number of outputs equal to the number of tokens in the

Approach	Tokenised sequence
Naïve	[“You’re”, “not”, “worried,”, “are”, “you?”]
Naïve lower case	[“you’re”, “not”, “worried,”, “are”, “you?”]
NLTK	[“You”, “’re”, “not”, “worried”, “,”, “are”, “you”, “?”]
Augmented NLTK	[“you”, “’re”, “not”, “worried”, “,”, “are”, “you”, “?”]

Figure 4.1: Limitations of naive tokenization

vocabulary $|V|$. A large vocabulary results in a large output dimension of the final dense layer of the generator, which increases the complexity of the generator. Thus, for practical reasons, the vocabulary of the generator is usually limited to a fix amount of tokens, much smaller than the total number of unique tokens in the corpus. When encoding documents, though, this problem is not present, wherefore a larger vocabulary is allowed. In theory, the vocabulary of the encoder $\mathcal{E}_n$ could be equal to the total vocabulary of the corpus. In reality, some tokens are extremely rare, e.g. because they are misspelled. Therefore, the vocabulary of the encoder is also limited to a certain number of tokens, but substantially more than for the decoder.

The process of extracting a vocabulary from a corpus is straightforward: first all sequences in the corpus are tokenized. Then the number of occurrences of each token in the corpus are counted and the tokens are sorted by number of occurrences in descending order. The vocabulary is then extracted by selecting the $|V|_{encoder}$ most frequent tokens from the sorted list.

4.1.3 Padding

As indicated above, pre-processed summaries end with an EOS token (see Section A.4) followed by a number of PAD tokens. The reason is that in the model presented in Section 4.3, the generator must output a fix number of tokens. If the generator generates an EOS token as token $t < T$, it continues to generate the PAD token until $t = T$. This act of *padding* is a common practice in NLP.

There is a drawback to this approach, though. Buduma [6], describes that naively applying padding on a set of sequences is wasteful and degrades the performance of any model since every sequence would be as long as the longest sequence in the corpus. For the generator, doing this is inevitable because of the hard constraint on the length of the generated sequences. For the encoder, however, there is no such constraint. An alternative approach is therefore to let the encoder consume documents of varying size, which would mean no wasteful padding. The implementational consequence of this, however is that the encoder must re-adjust itself before each encoding, which is computationally expensive.

Fortunately, there is a golden mean: bucketing, in which documents are divided into buckets according to their respective lengths. Each bucket thus consists of sequences of lengths $L \in N, N+1, ..., N+B$ where N is the number of tokens in the shortest sequence in the bucket and B is a pre-defined bucket size. Documents are then padded up to length $N+B$, which means that, over the entire dataset, a lot fewer PAD tokens need to be added. Implementation wise, the encoder has to re-adjust itself fewer times than if no padding is used. All-in all, as presented by Buduma [6], bucketing reduces computation time significantly during the training phase, which is the reason why it is used in this thesis.

4.2 Discriminator

As mentioned in Section 2.6 , the role of the discriminator is to act like a human Turing tester. It is trained to distinguish between human and model-generated text. The loss from the discriminator is used to guide the training of the generative model to produce better text.

The discriminative model used in this thesis is a convolutional neural network (CNN). CNNs has recently shown great effectiveness in text classification [70]. Unfortunately, a downside to CNN-based models – even simple ones – is that they require practitioners to specify the exact model architecture to be used and to set the accompanying hyperparameters. To the uninitiated, making such decisions can seem like something of a black art because there are many free parameters in the model. This is especially true when compared to, e.g., Support Vector Machines and logistic regression. Furthermore, in practice exploring the space of possible configurations for this model is extremely expensive, for two reasons: (1) training these models is relatively slow, even using GPUs (2) The space of possible model architectures and hyperparameter settings is vast. Indeed, the simple CNN architecture we consider requires, at a minimum, specifying: input word vector representations; filter region size(s); the number of feature maps; the activation function(s); the pooling strategy; and regularization terms (*dropout/l2*).

In practice, tuning all of these parameters is simply not feasible, especially because parameter estimation is computationally intensive. Emerging research has begun to explore hyperparameter optimization methods, including random search [71], and Bayesian optimization [72]. However, these sophisticated search methods still require knowing which hyperparameters are worth exploring to begin with (and reasonable ranges for each). Furthermore, we believe it will be some time before Bayesian optimization methods are integrated into deployed, real-world systems.

Thus, for comparability, the architecture adopted for this thesis is from the discriminator architecture of SeqGAN, which had shown good results in distinguishing between human and model-generated text. The relative simplicity of SeqGAN's architecture coupled with strong observed empirical performance makes this a strong contender for our real/fake classification task. Since our model extends the SeqGAN model, we chose to train our word embeddings without any pre-trained vectors.

As far as we know, except for some specific tasks, most discriminative models can only perform classification well for a whole sequence rather than the unfinished one. In case of some specific tasks, one may design a classifier to provide intermediate reward signal to enhance the performance of our framework. But to make it more general, we focus on the situation where discriminator can only provide final reward, i.e., the probability that a finished sequence was real.

We first represent an input sequence $x_1, ..., x_T$ as:

$$\epsilon_{1:T} = x_1 + \oplus x_2 + \oplus + \ldots \oplus x_T$$

where $x_t \in \mathbb{R}^k$ is the k-dimensional token embedding and $\oplus$ is the vertical concatenation operator to build the matrix $\epsilon_{1:T} \in \mathbb{R}^{Txk}$. Then a kernel $\mathbf{w} \in \mathbb{R}^{lxk}$ applies a convolutional operation to a window of size of l words to produce a new feature map:

$$c_i = \rho(w \otimes \epsilon_{i:i+l-1} + b)$$

where $\otimes$ operator is the summation of elementwise production, b is a bias term and ρ is a non-linear function. We can use various numbers of kernels with different window sizes to extract different features. Specifically, a kernel w with window size l applied to the concatenated embeddings of input sequence will produce a feature map,

$$c = [c_1, \ldots, c_{T-l+1}]$$

Finally we apply a max-over-time pooling operation over the feature map $\tilde{c} = max(c)$ and pass all pooled features from different kernels to a fully connected softmax layer to get the probability that a given sequence is real.

To enhance the performance, we also add the highway architecture [24] before the final fully connected layer:

$$\begin{gathered}\boldsymbol{\tau} = \sigma\left(\boldsymbol{W}_T \cdot \tilde{\boldsymbol{c}} + \boldsymbol{b}_T\right) \\ \tilde{\boldsymbol{C}} = \boldsymbol{\tau} \cdot H\left(\tilde{\boldsymbol{c}}, \boldsymbol{W}_H\right) + (\mathbf{1} - \boldsymbol{\tau}) \cdot \tilde{\boldsymbol{c}}\end{gathered} \tag{4.1}$$

where W_T , b_T and W_H are highway layer weights, H denotes an affine transform followed by a non-linear activation function such as a rectified linear unit (ReLU) and τ is the “transform gate” with the same dimensionality as $H\left(\tilde{\boldsymbol{c}}, \boldsymbol{W}_H\right)$ and $\tilde{c}$. Finally, we apply a sigmoid transformation to get the probability that a given sequence is real:

$$\tilde{y} = \sigma(W_o \cdot \tilde{C} + b_o)$$

where W_o and b_o is the output layer weight and bias.

When optimizing discriminative models, supervised training is applied to minimize the cross entropy, which is widely used as the objective function for classification and prediction tasks:

$$\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

where y is the ground truth label of the input sequence and $\hat{y}$ is the predicted probability from discriminative models.

The discriminator which is parameterized by ϕ takes in full sequences and outputs $D_\phi(Y_{1:T})$ the probability between 0 and 1 that $Y_{1:T}$ comes from real data.

In the implemented convolutional discriminators, the rewards for each of the tokens in the sequence were based on a fixed-sized context window, the width of the window are solely dependent on the sizes of the CNNs filters. In the most basic case, the context for a token at time step t covered both tokens occurring before and after that token, however, more advanced padding schemes could also be applied to make the context consider only tokens occurring before or after the token at time t.

4.3 Generator

The architecture of the baseline generator was chosen in a way to make the model powerful enough to create realistic looking summaries. Since the task of abstractive text summarization can be viewed as conditional language generation, a natural design choice was to use the encoder-decoder architecture as the core of the baseline model. The goal of the encoder network is to represent the summarized article as an information-rich, fixed-sized vector that is later passed to the decoder to guide the generative process. The decoder network is a neural language model that allows the generated sequence to be conditioned on input data. For this purpose, the authors adapted the pointer-generator architecture proposed by See et al. [4] and trained it using the adversarial network framework. The generator has two different processes for generating text, depending on whether the model is in pre-training or adversarial training.

4.3.1 Pre-training

The purpose of the pre-training is to make the generator learn the weights of the network for generating human like text via maximum likelihood training. This is done with pointer-generator networks where the real summary sequence $s_{1:T}$ is used as training input.

We tried to pre-train the generator from scratch using the open-sourced code of pointer-generator networks [4]. Results showed that while this showed signs of convergence towards the final result, it was computationally very expensive, extremely time consuming and imposed high hardware requirements. The authors of the baseline model state in their publication that training their model took 4 days and 14 hours a Teskla K40m GPU. This was not feasible to reproduce. However, See et al. [4] released the final weights of the trained network along with their code. As this project builds on the work of See et al. [4], we initialized our pre-training model with the weights shared by the network used in [4]

Given that the baseline architecture closely followed the summarization model described in [4], a search for model hyperparameters wasn't necessary. The summarization model was trained using the same parameter values found in the mentioned publication.

4.3.2 Adversarial Training

After the pre-training phase, the summarization model was fine-tuned using the loss from the discriminator model. The goal of adversarial training is to finetune the neural language models using an adversarial network framework which can bring improvements to the quality of generated sequences. The discriminator network was used as one of the reward signals for training the baseline text summarization model. The proposed training procedure utilized a mixture of rewards coming from a discriminator model and the baseline generator model. The final total reward was a linear combination of the mentioned partial rewards, with the mixing coefficients allowing to put emphasis on either of the training signals.

4.4 Evaluation Metrics

4.4.1 ROUGE

Summarization models are commonly evaluated using the ROUGE metrics described in the previous Section 2.7. These metrics measure the overlap of sequences present in the candidate text to be evaluated and a set of human-written sequences. In ROUGE terminology, this measure of overlap is called *co-occurence*. ROUGE offers a number of different calculation approaches but the most commonly used one for text summarization is ROUGE-N, which defines *n-grams* as the basic unit for computing co-occurence statistics, with ROUGE-1 using unigrams, ROUGE-2 using bigrams, etc.

The formulae for calculation of ROUGE-N variations are presented below.

$$\mathrm{ROUGE}-\mathrm{N}_{rec} = \frac{\#_{\text{match}}(\text{Candidate, Reference, N})}{\#(\text{ Reference}, N)}$$

$$\mathrm{ROUGE}-\mathrm{N}_{prec} = \frac{\#_{\text{match}}(\text{Candidate, Reference, N})}{\#(\text{ Candidate}, N)}$$

$$\mathrm{ROUGE}-\mathrm{N}_{f} = \frac{(1+\beta^2)\cdot \mathrm{ROUGE}-\mathrm{N}_{rec}\cdot \mathrm{ROUGE}-\mathrm{N}_{prec}}{\mathrm{ROUGE}-\mathrm{N}_{rec}+\beta^2\cdot \mathrm{ROUGE}-\mathrm{N}_{prec}}$$

where *Candidate* referes to the summary generated by the model, *Reference* to the human written summary and $\#_{\text{match}}$ computes the number of co-occurrences between the candidate and reference.

4.4.2 Qualitative Evaluation

As one of the key goals of this work was to highlight the limitations of evaluation using ROUGE from a qualitative perspective, it was found necessary to have the generated summaries evaluated qualitatively by manually inspecting the semantic and syntactic correctness, and comprehensibility.

4.5 Implementation Details

In the field of machine learning, a very common language for research purposes is Python, which was the reason why it was chosen for the thesis work. Python is a general purpose language with a highly active community as indicated by the annual developer survey by the large online developer platform Stack Overflow. In the survey, Python ranked as the seventh most used programming language and was described as the "fastest-growing major programming language". The Python community has, over the years, created a multitude of frameworks for a wide variety of applications. Some examples of well documented and renowned frameworks for machine learning are Scikit-learn, Pytorch, Caffe, Theano and Tensorflow. Of these (and more) alternatives, Tensorflow was chosen for several reasons. First, most of the research work on which the thesis work is based, used Tensorflow. Of these, some even published their code online, which made for a good opportunity of code re-use. Second, Tensorflow is one of the most used frameworks overall in the developer community, which means that the knowledge gained during the work would be valuable in the future for the author of this report.

In order to evaluate the quality of the text as described in Section 2.7, the Python package pyrouge using the original Perl script ROUGE-1.5.5.pl was used for computing ROUGE scores.

If the reader who wishes to reproduce the results presented in this report, important hardware and software specifications are listed in table below. Due to the many matrix operations, training deep learning are computationally heavy tasks. Using basic computers is therefore not an option. For training to be feasible, a computer with a graphics processing unit, commonly known as GPU, (or even better, a tensor processing unit (TPU)) has to be used. The training of the model in this thesis was performed on a computer equipped with a good consumer grade NVIDIA GPU. Detailed specifications of the GPU, central processing unit (CPU) and memory configuration of the computer are listed in table along with the operating system of the computer. All experiments were conducted on virtual machines on the Google Cloud Platform.

Part	Specification
Operating System	Ubuntu 16.04.4
GPU	NVIDIA Tesla K80
CPU	Intel Core i7-7700 CPU 3.60GHz
Memory	15GB
Python	3.6.5
Tensorflow	Tensorflow 1.4.0
ROUGE	1.5.5

Table 4.2: Hardware and software specifications of the machine which was used for training of the model.

Chapter 5

Experiments and Results

This chapter describes the experiments that were conducted as part of the thesis project. Experiment results are presented and analyzed. The first subsections comments on data preprocessing and model training procedures followed by a quantitative and qualitative analysis of obtained results.

5.1 Data Preprocessing

Experiments were conducted on the full-text version of the CNN/DM dataset. The data were pre-processed as described by See et al. [4]. All characters were converted to lower-case and sequences were tokenized using the Stanford CoreNLP toolkit [73].

5.2 Training Procedure

In the majority of available work on training language models, alternative training strategies were applied to models pre-trained with the standard maximum likelihood and teacher forcing strategy. This approach was followed in this thesis project. The summarization model and the discriminator was first trained to achieve satisfying performance on the evaluation metrics and test accuracy. The summarization model is later fine-tuned via adversarial training with the hope to enhance the quality of summaries. The pre-trained model was also treated as the baseline for further experiments.The following subsection reports the training procedures of the summarization and discriminator models.

5.3 Baseline model Pre-training

This section describes the performance of the baseline model of [4] which is used in this thesis. It uses 256-dimensional hidden states and 128-dimensional word embeddings. We use a vocabulary of 50,000 words. As the model is capable of handling out of vocabulary (OOV) words, it is possible to use a smaller vocabulary size than previous summarization models such as [8] which used 150,000 source and 60,000 target words. The baseline model has 21,499,600 parameters, with only 1153 extra parameters as a result of the pointer-generator network and an additional 512 parameters due to coverage. The word embeddings are learned from scratch during training i.e we do not adopt any pre-trained word embeddings. The model is trained using *Adagrad* gradient descent algorithm with a learning rate 0.15. The model also uses gradient clipping with a maximum gradient norm of 2, but do not use any form of regularization. The loss on the validation set is used to implement early stopping. The length of the summary is truncated to 100 tokens for training. At test time our summaries, are produced using beam search with beam size 4.

As discussed in 2.7.2, what is considered a good ROUGE score is still an open question. Today, the best models [1] for summarization obtain mean ROUGE-1 scores around 0.40-0.42 across the entire dataset. Thus for our reference and assessing model performance, we consider a ROUGE-1 score of 0.4 or higher, a satisfactorily good summary - the higher the score, the better the summary.

5.3.1 Results

Below are some examples of summaries generated by the baseline model. The examples listed in all the subsequent sections are merely a small sample picked to illustrate findings and the results obtained [2]

Example 1

Original Article: in the united states , women earn an average of 78 cents for every dollar that men make — and a new pop-up shop in pennsylvania is using this blatant gender-biased wage gap in order to charge its female customers lower prices than its male visitors . pittsburgh-based pop-up store , less than 100 , has priced its entire stock to reflect the local gender pay gap , meaning that female customers are only charged 76 per cent of the retail price while men pay in full . more active protest than business venture , less than 100 is not-for-profit and aims to highlight the disparity in salaries between the sexes . scroll down for video . woman 's worth : elana schlenker has opened a pop-up shop in pittsburgh , pennsylvania where women are charged only 76 per cent as much as men in order to highlight the problem of wage inequality . 'it's incredible how deeply unconscious biases still permeate the ways in which we perceive (and value) women

[1] http://nlpprogress.com/english/summarization.html

[2] The complete results of our model containing original article, human summaries, baseline summaries and the finetuned summaries can be found on Github

versus men , ' owner elana schlenker told refinery29 . ' i hope the shop's pricing helps to underscore this inherent unfairness and to create space for people to consider why the wage gap still exists . ' ms. schlenker can afford the gender-conscious pricing because her shop is n't aiming to make money ; all sales from the ceramics , textiles , publications , art prints , and stationery sole will go back to the female artists who made them . ' this is a small way that i can do something about [the pay gap] , and i think many of the artists involved are coming from a similar place , ' ms. schlenker added . price adjustment : less than 100 sells ceramics , art , and textiles by female artists as a discount to female customers . in addition to adjusting its prices , less than 100 is also holding events promoting wage equality at the pop-up space , including one for april 14 's equal pay day , which represents how far into the year a woman must work to earn as much as her male counterpart did the previous year . ms. schlenker said she is concerned that women are undervalued in the workplace and underrepresented in executive positions and wants to draw attention to the problem through this venture , and added that she 's been excited about how much support her shop has received . ' everyone that 's walked in , from women to men , has said " i love this idea " , ' she told wpix . ' women are worth what men are worth , of course , and just that community that 's sprung up around this has been amazing . ' she has also reached out to local organizations for grants to help maintain the pop-up shop , which will be open in pittsburgh for the entire month of april . ladies pay less : at the original location , women pay 76 per cent ; in new orleans this fall , women will pay 66 per cent , since women in louisiana make even less compared to men than women in pennsylvania . fans of the idea who want to get in on the action but do n't live in pennsylvania can continue to check in on the organization 's website , as ms. schlenker intends to travel with the shop , changing prices everywhere she goes to reflect the local pay gap . she will be re-opening the the store in new orleans this fall , where prices for female shoppers will drop even more to 66 per cent of the original retail value , corresponding with louisiana 's pay cap . male customers will continue to pay 100 per cent of the price for items like a $224pitcher, whichcosts$ 195 for female customers in pittsburgh , and a $ 120 gamma folk necklace , for which women this month are charged $ 92 .

Human Summary: a store in pittsburgh , pennsylvania is charging women 76 per cent of what they charge men , reflecting the local pay gap . owner elana schlenker says she hopes the project , called less than 100 , draws attention to wage inequality . she plans to travel with the shop and will open up in new orleans , louisiana , this fall .

Baseline Summary: elana schlenker has opened a pop-up shop in pittsburgh , pennsylvania where women are charged only 76 per cent of the retail price while men pay in full . more active protest than business venture , less than 100 is not-for-profit and aims to highlight the disparity in salaries between the sexes .
ROUGE-1 Score: 0.5

Example 2

Original Article: when clare hines found out she was pregnant she was faced with an agonising choice – cancel an operation to remove the tumour growing on her brain or terminate her unborn baby . ms hines , then 27 , from darra in brisbane , had suffered epileptic seizures for 10 years but had recently noticed that her balance was gone and she was struggling to hear out of her right ear . after a number of tests doctors discovered she had a benign tumour that had wrapped itself around her acoustic , facial and balance nerve – and told her she had have it surgically removed or she risked the tumour turning malignant . scroll down for video . clare hines , who lives in brisbane , was diagnosed with a brain tumour after suffering epileptic seizures . one week before brain surgery she found out she was pregnant . she went against doctors ' warnings to keep the baby and give birth to her son noah . but in february 2012 , one week before she was due to go into surgery , she took a pregnancy test that turned out positive . ' i went to doctor booked and told her i 've been the pill for 10 years - this was n't planned , ' ms hines told daily mail australia . ' she asked me ' what do you want to do , do you want to carry on with pregnancy or have your surgery ? ' ' i felt sick - how can you ask somebody that ? ' doctors warned ms hines that the baby would not survive being under general anaesthetic for 12 hours if she had the operation . they also warned her that if she kept the baby , her pregnancy hormones would likely cause the tumour to grow significantly . her baby was also at a higher risk of being born with birth defects , but she decided to push ahead with the pregnancy . seven weeks after noah was born , ms hines and their newborn flew back to her home of manchester in the uk to finally undergo surgery to remove her brain tumour , which had almost doubled in size . ms hines is pictured seeing noah for the first time in three weeks , after her brain surgery . ' i 'm a strong believer in fate – i thought this is meant to happen and maybe this baby is going to be my strength to get me through , ' she said . ' all my friends were really shocked - friends said ' how could you be so stupid ? ' ' but i thought you know what , it 's nothing to do with you it 's my choice . ' ms hines ' pregnancy was difficult and she suffered from exhaustion , stress and anxiety – but the real test came when she went into labour . ' it was the worst labour ever , i had three seizures in the middle , ' she said . ' when you seizure all your muscles contract and they could basically squash the baby . ' doctors made the decision to perform an emergency caesarean and ms hines gave birth to her son noah on october 16 , 2012 . ' he was n't breathing when he was born he had to be resuscitated , that was the shock of the birth , ' she said . ' at first we did n't hear any cries . doctors had to perform an emergency caesarean after ms hines suffered three seizures during labour . ms hines said she and her husband marcus were very worried about noah , but tests showed he was healthy . ' but then he opened his mouth and his vocal cords started working . he only cried for like 30 seconds and he did n't cry for five days afterwards . ' ms hines said she and her husband marcus hines were very worried about noah , but tests showed he was healthy . ' he had 10 toes , 10 fingers , his hearing and eyesight were good - everything seemed to be fine , ' she said . seven weeks after noah was born , mr and ms hines and their newborn flew back to her home of manchester in the uk to finally undergo surgery to remove her brain tumour , which had almost doubled in size . ' i wanted to go back to the uk to have the surgery done there – i thought that if something happens to me i do n't want to die here i want to be around family , ' she said . ' my husband took me to hospital and i just remember feeling nauseous , dizzy and just really , really frightened . ' after returning to australia following her surgery , ms hines suffered another accident at the gym . in january 2103 , after a 10 and a half hour operation , she woke up in terrible pain and had completely lost hearing in her right ear . she also contracted bacterial meningitis forcing her to stay in hospital for four weeks without seeing noah . but ms hines said the moment they were reunited made it all worth it . ' i was n't supposed to hold him but i just wanted to see him , ' she said . ' as soon as i did the pain just went away . ' i thought " this is why he happened . he happened to make me better . " the young family returned to australia and ms hines returned to work . b . but in an effort to get healthy again she went to the gym , where she had another seizure while running on a treadmill causing massive injuries to her face . she had to undergo more surgery to have a metal plate put in her cheek and eye socket . ms hines is now 30 and has been seizure-free for two years . she has started a new job as an account manager at a marketing company but she struggles to communicate with her colleagues without hearing in her right ear . ms hines and husband marcus (pictured with noah) arein tens of thousands of dollars ' worth of debt due to the cost of flying to the uk for her operation , her medication and the multiple specialists she sees regularly . ms hines is now 30 and has been seizure-free for two years . she has started a new job as an account manager at a marketing company but she struggles to communicate with her colleagues without hearing in her right ear . ms hines - who is in tens of thousands of dollars ' worth of debt due to the cost of flying to the uk for her operation , her medication and the multiple specialists she needs to see regularly - is trying to raise money to buy a hearing aid . ' i need a hearing aid to help me – i ca n't hear noah in the night when i lay on my side , ' she said . ' i ca n't drive because ca n't hear traffic coming on the right side , i ca n't get a bike because ca n't hear cars . ' she has started a mycause page in the hope of raising funds – with any extra money she raises to be donated to the brain foundation , epilepsy action australia and the queensland acoustic neuroma association . the family is trying to raise money so ms hines can buy a hearing aid . her son noah is now two-and-a-half years old and ms hines says she does n't regret the decisions she made . her son noah is now two-and-a-half years old and although he is seeing a speech therapist he is a healthy toddler . ' he 's not talking , and we think he 's getting frustrated because he ca n't explain what he wants , ' ms hines said . ' we do worry about it because of the pregnancy and labour , you do n't know if it caused long-term effects . ' he has terrible mood swings but we just do n't know if that 's a two-year-old being a two-year-old . ' but ms hines says she does n't regret the decisions she made . ' i would not change having my son . i would n't change coming to australia , would n't change going back to uk for me operation , ' she said . ' i 'm in a lot of debt and it 's caused a lot of stress but your health is more important than anything . ' ms hines has started a mycause page to raise money for a hearing aid and for charity .

Human Summary:clare hines , who lives in brisbane , was diagnosed with a brain tumour . she found out she was pregnant a week before scheduled surgery . the 27-year-old had to decide to keep the baby or remove the tumour . ms hines went against warnings from doctors to give birth to son noah . she has since had many operations but has lost hearing in her right ear . ms hines , originally from the uk , is trying to raise money for a hearing aid .

Baseline Summary: clare hines , 27 , was diagnosed with a brain tumour after suffering epileptic seizures for 10 years . she had suffered epileptic seizures for 10 years but had recently noticed that her balance was gone and she was struggling to hear out of her right ear . but in february 2012 , one week before she was due to go into surgery , she took a pregnancy test that turned out positive . her baby was also at a higher risk of being born with birth defects .
ROUGE-1 Score: 0.42

Example 3

Original Article: southampton are the club punching heaviest above their weight in the premier league in terms of performance compared with wages paid to players while chelsea are also over-achieving , latest figures have revealed . saints are only ranked 16th in the top flight in terms of their wage bill - £55.2million - and yet are currently seventh in the premier league . chelsea are 10 points clear at the top of the table but are only the third-highest payers with a wage bill of £192.7million behind manchester united (£215.8m) and manchester city (£205m) . bottom-club burnley 's wage bill is just £21.5m including £6m in promotion bonuses , one-tenth the size of united 's . despite being seventh in the league , southampton have the fourth lowest wage bill based on last season . third-placed manchester united have have the highest wage bill at £215.8m , based on last season 's figures . chelsea are 10 points clear at the top of the table but are only the third-highest wage payers with £192.7m . chris ramsey 's qpr side salary bill was almost twice what the club earned in total last season . the biggest under-achievers are qpr , with a £75.3million wage bill even from a season when they were in the championship making them the eighth-highest payers - yet they are down at 19th in the premier league table . qpr 's salary bill was almost twice what the club earned in total last season . the wages costs and profits or losses of all top-flight clubs for 2013-14 have now been confirmed via annual accounts posted at companies house and overall there is a close correlation between total salary bill and league position , with the current top four in the premier league also the four biggest payers . how the premier league table currently stands . the combined accounts of the 20 clubs shows over overall turnover rose to £3.07billion from £2.3billion in 2012/13 with wages increasing too but at a slower rate and totalling £1.84billion compared with £1.59billion . the latest figure shows salaries account for 59.9 per cent of turnover compared with 71.7 per cent for the same 20 clubs a year before . the increase in income is mainly down to the premier league 's lucrative television deal that came into effect for the first time last season . the cash injection has led to six clubs who were in the red in 2012/13 now being in the black . apart from those clubs who were promoted from the championship last season , only manchester city , aston villa and sunderland ended the 2013/14 season having made a financial loss . premier league director of communications dan johnson said the clubs ' decision two years ago to introduce spending controls had also contributed to a positive financial outlook . johnson said : ' there are two reasons for this . the first is increasing revenues and the second is the financial criteria the clubs have voted in two seasons ago which put financial sustainability at the heart of how they want to go forward . ' the measures introduced by the clubs capped the amount they could use television money to pay for player costs . it also put a long-term limit on a club 's overall losses . the figures also show sunderland are the ninth-highest payers but are 16th in the table . burnley have the lowest wage bill with £21.5m based on the financial figures from last season .

Human Summary:seventh-placed southampton 's wage bill for last season was £55.2million . third-placed manchester united have the highest wage bill at £215.8m . qpr 's salary bill was almost twice what they earned in total last season . click here for the latest premier league news .

Baseline Summary: chelsea are only ranked 16th in the top flight in terms of their wage bill . chelsea are only ranked 16th in the top flight in terms of their wage bill . chelsea are 10 points clear at the top of the table but are only the third-highest wage payers .
ROUGE-1 Score: 0.21

5.4 Discriminator Training

The task of the discriminator is to accurately distinguish "real" human-written text from "fake" model-generated text. To train such a model we used the Convolutional Neural Network (CNN) architecture of SeqGAN [5] with the following model parameters:

- **Number of Filters:** 100, 100, 100, 100
- **Filter Sizes:** 2,3,4,5
- **Strides:** 1,1,1,1
- **L2 Regularization:** 0.2
- **Batch size:** 256
- **Non-liner Activation Function:** ReLU

From the CNN-Daily Mail dataset, a training set of 10,000 pairs of human-written and baseline model generated summaries were sampled. Similarly 1,000 pairs were sampled for the validation and test set. It is important to note that the random sampling to select the datasets, resulted in a well mixed set comprising of summaries ranging from a spectrum of high to poor quality. Thus for the model to perform well on the test set, it truly would have to learn the nuances in quality between good and bad quality texts during training.

5.4.1 Classification Quality

This section delves into understanding what and how CNNs learn during training. However, interpreting the inner workings of Convolutional Neural Networks (CNNs) for text classification is not straightforward unlike the case with images [3]. This is because intuitions about how CNNs work on an abstract level for images does not carry over to text. In the case of text, according to Goldberg [74], a CNN classifies text in the following way:

- 1-dimensional convolving filters are used as n-gram detectors, each filter specializing in a closely-related family of n-grams.
- Max-pooling over time extracts the relevant n-grams for making a decision.
- The rest of the network classifies the text based on this information.

Each CNN model is equipped with multiple filters of varying widths allowing the network to find both short and long-term dependencies and patterns present in the sequences. Each filter searches for a specific class of n-grams which it marks by assigning them high scores. There is a threshold for each filter where values above the threshold signal informative information regarding the classification, while values below the threshold are uninformative and can be ignored for the purpose of classification. For example, a filter may specializing in detecting n-grams of a semantic class such as "had no issues", “had zero issues”, and “had no problems” [75]. While this is a good way to interpret the performance of CNNs in text classification, it is quite impossible to uncover all the hidden inferences CNNs make for every filter and layer.

In the following section, a few examples of predictions on the test set made by the discriminator are described. As described in 2.6 each prediction is a probability score of the generated summary being real or fake.

5.4.2 Results

Example 1

[3] https://cs.nyu.edu/ fergus/papers/zeilerECCV2014.pdf

In this example, the generated summary is of poor quality and the discriminator rightly assigns it a low probability of being real (human-written).

Original Article: there was more drama than expected at a thespian awards ceremony this week , as an honoree slipped off the stage and nearly crushed musicians in the orchestra pit . harrison poe was announced as best supporting actor at the annual tommy tune awards on tuesday in houston , texas , but the high school student had a little difficulty navigating the stage . tv cameras caught him confidently getting up to accept the accolade before slipping and falling headfirst into the darkness . according to ktrk news , which was filming the event , poe fell into a net covering the orchestra pit and luckily no one was hurt below . scroll down for video . caught on camera : there was more drama than expected at a thespian awards ceremony this week , as an awardee slipped off the stage and nearly crushed musicians in the orchestra pit . going , going , gone : harrison poe was announced as best supporting actor at the annual tommy tune awards on tuesday in houston , texas , but the high school student had a little difficulty navigating the stage . the young bow tie-wearing actor went on to make a swift recovery and bounced back to the podium . however , he appeared to be a little shaken up from his fall as he proceeded to give his acceptance speech . ’ thank you ! at least i ’m alive , ’ poe chuckled , while the packed auditorium continued to cheer . red-faced : the young bow tie-wearing actor went on to make a swift recovery and bounced back to the podium - ’ thank you ! at least i ’m alive , ’ poe chuckled . ’ he ’s just practicing to be jennifer lawrence one day , you know - tripping off the oscar stage , ’ a news anchor later said of poe ’s faux pas . ironically , poe was recognized for playing paul in his school ’s production of a chorus line . in the musical , paul falls and suffers a knee injury during a tap dancing audition and is carried off to hospital . luckily , poe did not require medical treatment this time around . the 13th annual tommy tune awards will air on abc-13 on sunday , june 21 at 12pm .

Human (real) Summary: harrison poe was announced as best supporting actor at the annual tommy tune awards on tuesday in houston , texas . but the high school student had a little difficulty navigating the stage . tv cameras caught him confidently getting up to accept the accolade before slipping and falling headfirst into the darkness . however , the young bow tie-wearing actor went on to make a swift recovery and bounced back to the podium .

Baseline (fake) Summary: harrison poe announced as best supporting actor at the annual tommy tune awards on tuesday in houston , texas , but the high school student had a little difficulty navigating the stage . the young bow . actor went on to make a swift recovery and bounced back to the podium **Probability of real**: 0.02
ROUGE-1 Score: 0.71

Example 2

In the below example, the summary generated by the model ends abruptly where the sentence does not in a natural way.

Original Article: the green party this morning pledged an end to austerity - with a £90 billion a year spending splurge . the green party today demanded an end austerity with a radical £176 billion a year spending splurge - paying for everything from free prescriptions to organic school dinners for all . state pensions would be increased by more than £60 a week , child benefit doubled , tuition fees scrapped and train fares cut by 10 per cent . the party also pledged to pump an extra £12billion a year into the nhs from day one , introduce universal free elderly care and nationalise the railways . but its election manifesto also revealed a host of left-field pledges - including a ' complete ban ' on cages for hens and rabbits , an end to whips in horse racing and mandatory ' equality and diversity lessons ' at school . immigrants will also be given free english classes , while fruit and vegetables will be subsidised and children will be kept out of school until they are seven . green party leader natalie bennett , unveiling her party 's manifesto this morning , defended her ' vision of a fair economy ' . she said : ' that fair economy demands the end to austerity . it demands we restore and enhance the essential public services we all but particularly the most vulnerable . ' ms bennett 's list of policy pledges will see government spending rise by £89 billion next year - and by £176billion by the end of the parliament . the extra spending would see an extra £11 billion going on debt interest . under the party 's proposal , an extra £198 billion of taxes will be introduced to pay for the raft of giveaways – worth an extra £6,600 for every taxpayer in the uk . scroll down for video . green party leader natalie bennett and mp caroline lucas unveiled the green party general election campaigning at the arcola theatre , in london , this morning . green party leader natalie bennett , unveiling her party 's manifesto this morning , defended her ' vision of a fair economy ' . the new taxes include £25 billion in new ' wealth taxes ' , levied at two per cent of the entire estate of someone worth £3million or more . it means someone with £10 million in assets would pay an extra £200,000 a year . ms bennett also proposed hitting those earning £150,000 or more with a new 60 per cent tax rate . there would also be a ' robin hood tax ' on financial transactions and a promise to close tax loopholes . corporation tax would rise to 30 per cent with duties on cigarettes and alcohol increasing by £1.4bn every year . the party would save cash by scrapping £15 billion of road improvement works . instead the railways will be renationalised , a move the greens claim will save £1 billion a year . some £8 billion will be spent cutting public transport fares and hs2 will be scrapped . tuition fees will be scrapped at a cost of £4.5 billion , while school funding will be increased by £7 billion . grammar schools will be turned into comprehensives and the 'cruel' bedroom tax abolished . ms bennett told activists next month 's election was a landmark for the party . she said it was a ' new kind of politics ' where the economy and society could be run for the many and not the few . the leader said : ' it sees the end of the disastrous policy of austerity that is making the poor , the disadvantaged and the young pay for the greed and the fraud of the bankers . ' this is a vision that is exciting increasing numbers of britons . ' turning to her jobs policy , ms bennett said : ' that fair economy demands every worker is paid a living wage . it is really not a radical statement to say that if you work full time you should earn enough money to live on . the party 's manifesto listed a series of radical policies , including an extra £12billion for the nhs , free social care for the elderly , the nationalisation of the railways and an end to tuition fees . ' and yet we are the only uk party who is saying the minimum wage should immediately be lifted to a living wage and should reach £10 an hour by 2020 . ' the green leader said protection of the nhs was at the heart of the party 's election offer as she vowed to strip all private operators from the organisation . she said the ' expensive , destructive ' market would be removed from the health service . ms bennett said : ' behind that is an understanding of what privatisation has really meant for so many of our public services . ' it 's meant the cutting of the pay and conditions of workers , it 's meant the cutting of the quality of services and it has meant the shovelling of public money into private hands . ' other policies include re-nationalising the railways and policies to tackle climate change .

Human (real) Summary:greens pledge to increase government spending by £89bn next year . by 2020 , the party would hike spending by £176bn to almost £1 trillion . nhs spending would go up by £12bn , with the railways nationalised . free care for the elderly introduced and tuition fees scrapped . travellers would get new ' nomadic rights ' and foreign aid increased .

Baseline (fake) Summary: state pensions would be increased by more than 60 a week , child benefit

doubled , tuition fees scrapped and train fares cut by 10 per cent . the party also pledged to pump an extra £ 12billion a year into the nhs from day one , introduce universal free elderly care and nationalise the
Probability of real: 0.06
ROUGE-1 Score: 0.45

Example 3

In the below example, the summary generated by the model does not include the subject of the article. This is a minor difference from ROUGE score perspective but critical for factual accuracy. The example illustrates how a ROUGE score can be misleading. Our discriminator rightly classifies this with a low probability of being real

Original Article: as one of the world 's most successful female designers , victoria beckham was unlikely to turn up at the london marathon in trainers . despite it being an off-duty day , the 41 year old wore a pair of skyscraper heels to cheer on her son romeo , 12 , in the children 's marathon earlier today . she accessorised her skinny jeans , leather jacket and ' team romeo ' t-shirt with a pair of alaïa boots , which retail at approximately £1,500 . scroll down for video . victoria beckham at the london marathon today . r13 kate distressed skinny jeans . get a similar style at farfetch ! visit site . just when we thought we knew victoria beckham 's sense of style better than our own , she goes and surprises us . we 're so used to seeing her in midi skirts and dresses from her own collection that seeing her in a sleeveless wrap top and jeans takes a bit of getting used to . but when it comes to basics , like the rest of us , she has her favorites . she actually has worn these super skinny jeans countless times before , and it 's easy to see why . they 're by r13 , a label loved by a-listers like lindsay lohan , beyoncé and rihanna and they fit like a dream . when paired with platform ankle boots , vb 's legs look like they go on forever ! this season , the biggest denim trend has been jeans that are ripped to shreds . but we love this much more subtle approach - look closely and you 've notice that these dark wash jeans feature distressed detailing all over . click the link (right) to buy a similar pair now at farfetch . but if you want the beckham look on a budget , check out the edit below for our favorite high street styles . topshop distressed skinny jeans . visit site . asos distressed skinny jeans . visit site . citizens of humanity distressed jeans at selfridges . visit site . asos ridley skinny jeans in clean black with ripped knees at asos . visit site . despite the shoes having an eye-watering six inch heel , the star appeared unhindered and strolled easily around the finishing line with her husband david and her three sons . victoria is a devotee of huge heels - in her early years of fame she was never seen without a pair on and once said of flat shoes : ' i just can't get my head around the footwear . ' but recently she admitted in an interview with the independent that she was now a convert , thanks to her busy lifestyle . she told the newspaper : 'i'm busier now – so i couldn't totter around in a tight dress and a pair of heels ! i think i feel a little more relaxed . ' the star wore a pair of alaïa boots , which retail at approximately £1,500 . she continued : ' i suppose if i'm being honest , i would have been scared , right at the beginning , to wear lots of layers and to wear a flat shoe . ' i would have been scared to do that . now i'm not . ' but victoria must find these boots extremely comfortable because she has been wearing them on a number of occasions recently . the spice girl wore them for a flight to london from la last week and she also showcased them during an appearance on the ellen show last week . victoria wore the boots when she took a flight from la to london last week . victoria also wore the boots for an appearance on the ellen show .

Human (real) Summary: victoria beckham was at the london marathon to cheer on her son romeo . the star wore a pair of alaia boots , which retail at approximately £1,500 . she was joined by husband david and sons brooklyn and cruz .

Baseline (fake) Summary: the 41 year old wore a pair of skyscraper heels to cheer on her son romeo . she accessorised her skinny jeans , leather jacket and ‘ team romeo ’ t-shirt with a pair of alaia boots , which retail at approximately £ 1,500
Probability of real: 0.19
ROUGE-1 Score: 0.59

5.5 Adversarial Training

5.5.1 Results

Below the results from our finetuned model for summarization are shown.

Example 1

The following is an example where our finetuned summary has a lower ROUGE score compared to the baseline's summary but is of a better quality and more relevant.

Original Article: google wallet says it has changed its policy when storing users ’ funds as they will now be federally-insured (file photo) . for those who use google wallet , their money just became safer with federal-level insurance . google confirmed to yahoo finance in a statement that its current policy changed - meaning the company will store the balances for users of the mobile transfer service (similar to paypal and venmo) in multiple federally-insured banking institutions . this is good news for people who place large amounts of money in their wallet balance because the federal deposit insurance corporation insures funds for banking institutions up to $ 250,000 . currently , google ’s user agreement says funds are not protected by the fdic . however , a google spokesperson told yahoo finance that the current policy has changed . the policy states that ’ funds held by google payment corp. or its service providers (including any bank service providers) in connection with the processing of payment transactions are not deposit obligations of buyer , and are not insured for the benefit of buyer by the federal deposit insurance corporation or any other governmental agency ’ , according to 9to5 . as a non-banking institution , google wallet , along with paypal and venmo , is not legally required to be federally insured . but with the new change to its policy , funds in wallet balance are protected if anything were to happen to the company like bankruptcy . for example , for those who use mobile payment services where funds are not fdic insured , if a company files for bankruptcy , the user would become their creditor and would be required to go through bankruptcy court to get their money back . but for users with fdic insurance , they would receive their money from the government in a matter of days , according to yahoo finance . details on when google wallet ’s user agreement will be updated were not provided . how google wallet works : google wallet , launched in 2011 , stores credit and debit card information , and lets shoppers pay for goods by tapping their phones against special terminals at store checkout counters . it works with any debit or credit card , on every mobile carrier . the service allows users to send money to anyone in the us with a gmail address , or request money on-the-go , directly from their bank account , debit card or wallet balance . users can spend received money through the service in stores or withdraw cash from atms with the google wallet card , as well as transfer the funds directly to their bank account . it comes with 24/7 fraud monitoring , and users can disable their wallet app or card from their google wallet account if they ’re ever lost or stolen . source : google wallet . earlier this month , a federal judge rejected google inc ’s request to dismiss a lawsuit accusing the technology company of invading the privacy of users of its google wallet electronic payment service by sharing their personal information with outside app developers . the judge said google must face claims it breached users ’ contracts , violated the federal stored communications act which limits disclosure of electronic records , and violated a california consumer protection law . judge beth labson freeman in san jose , california said google wallet users may try to show that google ’ frustrated ’ the purpose of its own privacy policy by allowing ’ blanket , universal disclosure ’ of their personal information to app developers whenever they bought apps in google ’s play store . the lead plaintiff is alice svenson , an illinois resident who said google sent unnecessary personal information about her to ycdroid when she paid that developer $ 1.77 for an email app . she said google raised the risk of identity theft by routinely sending information about google wallet users such as addresses and zip codes , phone numbers and email addresses to app developers . svenson said the mountain view , california-based company ceased the practice soon after the lawsuit was filed in september 2013 . her lawsuit seeks class-action status , damages of $ 1,000 per violation , punitive damages and other remedies .

Human Summary: google spokesperson confirmed current policy changed meaning funds will be protected by the federal deposit insurance corporation . as a non-banking institution , google wallet , along with competitors paypal and venmo , is not legally required to be federally insured . with the new change to its policy , funds in wallet balance are protected if anything were to happen to the company like bankruptcy .

Baseline Summary: google confirmed to yahoo finance in a statement that its current policy changed - meaning the company will store the balances for users of the mobile transfer service . this is good news for people who place large amounts of money in their wallet balance because the federal deposit insurance corporation insures funds for banking institutions up to $ 250,000 .
ROUGE-1 Score: 0.47

Finetuned Summary: google wallet says it has changed its policy when storing users ’ funds as they will now be federally-insured (file photo) . for those who use google wallet , their money just became safer with federal-level insurance . google ’s user agreement says funds are not protected by the fdic .
ROUGE-1 Score: 0.40

Example 2

Revisiting the same example from the previous section, below our model not only rightly identified the error and corrected it but also obtained a higher ROUGE score.

Original Article: as one of the world ’s most successful female designers , victoria beckham was unlikely to turn up at the london marathon in trainers . despite it being an off-duty day , the 41 year old wore a pair of skyscraper heels to cheer on her son romeo , 12 , in the children ’s marathon earlier today . she accessorised her skinny jeans , leather jacket and ’ team romeo ’ t-shirt with a pair of alaïa boots , which retail at approximately £1,500 . scroll down for video . victoria beckham at the london marathon today . r13 kate distressed skinny jeans . get a similar style at farfetch ! visit site . just when we thought we knew victoria beckham ’s sense of style better than our own , she goes and surprises us . we ’re so used to seeing her in midi skirts and dresses from her own collection that seeing her in a sleeveless wrap top and jeans takes a bit of getting used to . but when it comes to basics , like the rest of us , she has her favorites . she actually has worn these super skinny jeans countless times before , and it 's easy to see why . they 're by r13 , a label loved by a-listers like lindsay lohan , beyoncé and rihanna and they fit like a dream . when paired with platform ankle boots , vb 's legs look like they go on forever ! this season , the biggest denim trend has been jeans that are ripped to shreds . but we love this much more subtle approach - look closely and you 've notice that these dark wash jeans feature distressed detailing all over . click the link (right) to buy a similar pair now at farfetch . but if you want the beckham look on a budget , check out the edit below for our favorite high street styles . topshop distressed skinny jeans . visit site . asos distressed skinny jeans . visit site . citizens of humanity distressed jeans at selfridges . visit site . asos ridley skinny jeans in clean black with ripped knees at asos . visit site . despite the shoes having an eye-watering six inch heel , the star appeared unhindered and strolled easily around the finishing line with her husband david and her three sons . victoria is a devotee of huge heels - in her early years of fame she was never seen without a pair on and once said of flat shoes : ' i just can't get my head around the footwear . ' but recently she admitted in an interview with the independent that she was now a convert , thanks to her busy lifestyle . she told the newspaper : 'i'm busier now – so i couldn't totter around in a tight dress and a pair of heels ! i think i feel a little more relaxed . ' the star wore a pair of alaïa boots , which retail at approximately £1,500 . she continued : ' i suppose if i'm being honest , i would have been scared , right at the beginning , to wear lots of layers and to wear a flat shoe . ' i would have been scared to do that . now i'm not . ' but victoria must find these boots extremely comfortable because she has been wearing them on a number of occasions recently . the spice girl wore them for a flight to london from la last week and she also showcased them during an appearance on the ellen show last week . victoria wore the boots when she took a flight from la to london last week . victoria also wore the boots for an appearance on the ellen show .

Human Summary: victoria beckham was at the london marathon to cheer on her son romeo . the star wore a pair of alaia boots , which retail at approximately £1,500 . she was joined by husband david and sons brooklyn and cruz .

Baseline Summary: the 41 year old wore a pair of skyscraper heels to cheer on her son romeo . she accessorised her skinny jeans , leather jacket and ' team romeo ' t-shirt with a pair of alaia boots , which retail at approximately £ 1,500
ROUGE-1 Score: 0.59

Fintuned Summary: victoria beckham wore a pair of skyscraper heels to cheer on her son romeo . she accessorised her skinny jeans , leather jacket and ' team romeo ' t-shirt with a pair of alaia boots , which retail at approximately £ 1,500 google 's user agreement says funds are not protected by the fdic .
ROUGE-1 Score: 0.64

Example 3

In the below example, the baseline summary was stuck in a repetitive loop producing identical sentences consequently. Our model corrected this and produced a more realistic summary.

Original Article: a device has been created that can scan entire planes for contraband , bombs and weapons . the mobile x-ray scanner , heralded as the world's first ' airplane scanner' , sweeps down a plane and can detect various objects inside . while the radiation emitted by the scanner is not safe yet for humans to be in the plane when it is operated , the company behind it thinks it could become a vital tool in aircraft safety . scroll down for video . romanian company mb telecom has revealed the roboscan 2m aeria . it uses a cone of radiation to sweep across planes (shown) and look inside . the device is accurate enough to find a filament in a light bulb . but the radiation it emits is not safe for passengers yet . the roboscan 2m aeria is being developed by romanian company mb telecom (mbt) . it costs 3 million (£2.1 million , $ 3.2 million) , reported the times , and is apparently already being used to check trucks at borders in eastern europe , where it has helped seize 76,000 packs of illegal cigarettes . on its website , mbt says the aeria is ‘ the very first solution for aircraft security inspection . ’ it can scan planes from small private jets to medium-sized commercial airplanes . the company notes that , while passengers and their luggage are thoroughly inspected at airports , there is no such inspection process for jets arriving at small private airports with low security . but with their device , the company says this ‘gap’ in aviation security can be plugged , and it can scan an entire plane in about 20 minutes . the device can be used to detect weapons (shown) and other contraband in small or medium-sized jets that land at low-security airports . an operator working inside the truck can look inside the entirety of a plane , although currently under regulations some areas of the plane - such as the fuselage and wings - are restricted . there are four main components that allow an aircraft to be scanned . armed with a sharp sense of smell , dogs have a long history of detecting explosives for their human handlers , and soon there could be sniffer elephants too . researchers have found that the large mammals excel at identifying explosives by smell , stirring speculation about whether their extraordinary ability could save lives . elephants could be used to detect land mines in areas that have seen conflict and could even do so at a distance with the aid of drones . the research was conducted in south africa and involved the u.s. military . the first is the operating truck , from which an operator can use the system and look inside the plane . attached to this is a robotic arm with an x-ray generator at the end . the plane is then pulled forwards by a battery-operated tug over a wide detection module . by firing x-rays out of the arm , a cone of radiation sweeps over the plane . the operator inside the truck is then given live images of the interior of the plane , and is able to scan its entirety for hidden items . it is apparently so accurate that it could pick out keys on a seat or even a filament in a light bulb . one issue , though , is that aircraft operators do not authorities to inspect the whole plane - areas of the fuselage and the wings are off limits , even though roboscan can examine these areas . ‘ you can not touch it , but you can scan it , ’ said radu tudor of mbt at the counter terror expo in london this week . there are four main components that allow an aircraft to be scanned . the first is the operating truck (shown) , from which an operator can use the system and look inside the plane . the plane is pulled forwards through the scanner by a battery-operated tug (shown) over a wide detection module . by firing x-rays out of the arm , a cone of radiation sweeps over the plane .

Human Summary: romanian company mb telecom revealed the roboscan 2m aeria . it uses a cone of radiation to sweep across planes and look inside . the device is accurate enough to find a filament in a light bulb . but the radiation it emits is not safe for passengers yet .

Baseline Summary: romanian company mb telecom has revealed the roboscan 2m aeria . it uses a cone of radiation to sweep across planes . it is accurate enough to find a filament in a light bulb . it is accurate enough to find a filament in a light bulb .
ROUGE-1 Score: 0.71

Fintuned Summary: romanian company mb telecom has revealed the roboscan 2m aeria . it uses a cone of radiation to sweep across planes (shown) and look inside . the device is accurate enough to find a filament in a light bulb .
ROUGE-1 Score: 0.84

Example 4

Below is an example where both the baseline and our model produce a relatively good quality summary but obtain below average rouge scores as they differ in words used from the human summary.

Original Article: natalie bennett said the so-called ' citizens income' – which critics say would cost £280billion – would take longer than five years to bring in , and could take even longer . every adult in britain would be paid £72 a week under a commitment in the green manifesto – but the party leader admitted her pledges could take decades to implement . natalie bennett said the so-called ' citizens income' – which critics say would cost £280billion – would take longer than five years to bring in , and could take even longer . she said another green pledge – to dismantle the armed forces and use weapons factories to build wind turbines – was also a 'long-term aspiration' . and the party's policy of leaving nato and abandoning nuclear weapons was ' many decades into the future' . the revelations came in another disappointing performance by miss bennett , when she was interviewed on the bbc radio 4 today programme . in february she suffered what she called a ' brain fade' on lbc radio when she forgot the details of her party's key housing policies . on today , she was challenged about policies listed in the green policy book , which is set by party members at the annual conference . one of these is the citizens' incomes , which would be given to everyone in britain as a ' right of citizenship' , regardless of wealth or whether they are seeking work . miss bennett , the green party leader in england and wales , said the policy would take time to implement . she said : ' let me say this very slowly – the commitment is in the manifesto . we don't think we can introduce it in the term of the next parliament ; it's a big change . ' the green party has a long-term vision about what britain should look like . i don't apologise for that . ' we are saying we need a britain that's built on principles of politics ; we need a different kind of politics and having that long-term vision , looking into the future and saying what should britain look like in five , 10 , 20 years' time is something that is very much central to green party politics . ' claiming the citizens' income was backed by an ' increasing number of economists' , she said that as ' the sixth richest economy , we have to provide for everyone 's basic needs and we have to do that within the limits of our one planet' . ' the citizens ' income , the commitment to that , is in this manifesto – but we 've seen the difficulties that iain duncan smith has got into in introducing universal credit , so we 're saying it would take more than the term of one parliament to actually introduce this , ' she said . claiming the citizens' income was backed by an ' increasing number of economists' , ms bennett said that as ' the sixth richest economy , we have to provide for everyone 's basic needs and we have to do that within the limits of our one planet' . after two-and-a-half years as green party leader , natalie bennett would have liked to think she has made a name for herself . but after another faltering media performance she suffered the indignity of being confused with her predecessor . radio 4 today presenter john humphreys ended the grilling saying : caroline lucas , thank you very much . ' he quickly apologised for his gaffe . ms lucas was green leader from 2008-12 , but stood down to concentrate on being the party 's first mp . miss bennett was also challenged over a pledge , contained in the same policy book , to dismantle the armed forces – saying it was a 'long-term' aspiration . she also said that abandoning nuclear weapons and leaving nato was a 'long-term plan many decades into the future' . the book also contains pledges to legalise the sex industry and drug use , decriminalise al qaeda membership and impose new taxes on pop music performances and large gifts . miss bennett said the greens were now the third-largest party in england and wales . ' we 're larger than ukip , we 're larger than the lib dems , ' she said . ' we have more than 58,000 members . ' we 've had one green mp , caroline lucas , who 's had a huge impact as one mp in the parliament . ' and we 're now working very hard to send a strong group of green mps to parliament who - given the electoral arithmetic - have a very good chance of being in a strong position to really influence the whole direction of the next government . ' she said her party offered a ' genuine alternative' to the conservatives ' ' austerity heavy' and the labour 's 'austerity-lite' . the green' manifesto will be launched next week .

Human Summary: green party leader flounders again in interview about her own policies . set out plans for ' citizens income ' paid to every adult in the country . forced to admit it would take ' decades ' to implement expensive plan . embarrassment at end of interview when she is called ' caroline lucas ' .

Baseline Summary: natalie bennett said the so-called ' citizens income' – which critics say would cost £280billion – would take longer than five years to bring in , and could take even longer . every adult in britain would be paid £72 a week under a commitment in the green manifesto , but the party leader admitted her pledges could take decades to implement . the revelations came in another disappointing performance by miss bennett , when she was interviewed on the bbc radio 4 today programme .
ROUGE-1 Score: 0.37

Fintuned Summary: natalie bennett said the so-called ' citizens income' – which critics say would cost £280billion – would take longer than five years to bring in , and could take even longer . every adult in britain would be paid £72 a week under a commitment in the green manifesto . but the party leader admitted her pledges could take decades to implement .
ROUGE-1 Score: 0.38

5.6 Quantitative Evaluation

Below we perform a comparison of ROUGE-1 and ROUGE-2 scores between the baseline model and our finetuned model.

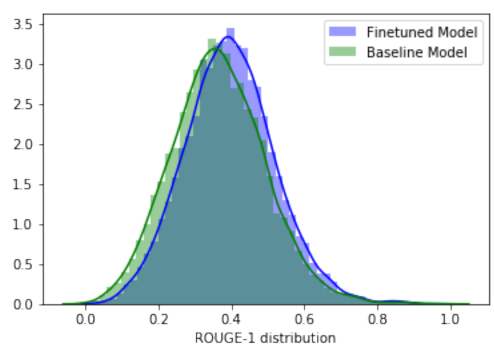

Figure 5.1: Distribution of ROUGE-1 scores between baseline and finetuned model

Model	ROUGE-1	ROUGE-2
Baseline	0.37	0.08
Finetuned	0.40	0.1

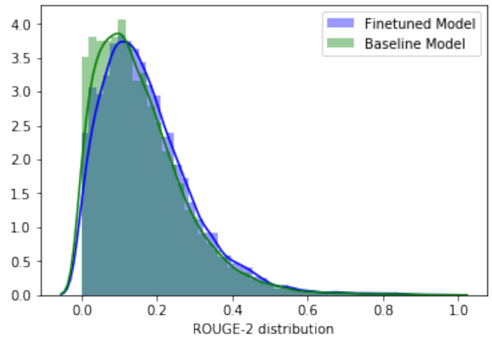

Figure 5.2: Distribution of ROUGE-2 scores between baseline and finetuned model

5.7 Qualitative Evaluation

As mentioned previously, one of the key objective of this thesis is to improve quality of text produced by generative models. The hypothesis was that by learning to identify human-written text from model-generated text, the model could eventually produce summaries that resemble human-written text. Quality of text is hard to quantify and evaluate using formula-driven metrics. Hence to assess the qualitative performance of our model and validate the hypothesis, human evaluation was used.

The human evaluation can be set up in many forms. In this study, the inspiration was a combination of the evaluations by Semeniuta, Severyn, and Gelly [76] and Fedus, Goodfellow, and Dai [69].

From each model, 100 samples are uniformly selected and rated by 5 human expert judges. 50 articles, along with the two summaries (baseline model and our fine-tuned model) were randomly selected from the dataset. These were shown to 5 human judges, to evaluate. The order of summaries were jumbled in order to prevent biases creeping in. The judges were not informed on the algorithm behind the summaries. They were asked to judge each summary individually on a scale of 1-10 on the following criteria:

- Readability - (1 representing a poor quality of text ; 10 near human-lik equality of text)
- Factual Accuracy - (1 representing relevant facts from the article are not captured in the summary ; 10 representing perfect reproduction of important facts from the article in the summary)

  Finally, the judges were instructed to pick the overall better summary.

For ease of reference, we refer to our model as 'Finetuned' in the following parts. Table below shows the mean scores and 95% confidence interval for each model and each task.

On average, out of all the summary sets, in 84% of the cases the judges picked the summary produced by our *finetuned* model as the overall better summary.

Model	Readability	Factual Accuracy
Baseline	6.2 ± 0.26	7.4 ± 0.31
Finetuned	7.5 ± 0.24	7.6 ± 0.27

Table 5.1: Human evaluation scores performed on 50 examples randomly selected from the test set.

5.8 Discussion

The baseline model chosen for this work performs competitively on ROUGE and was a state-of-the-art model on the NLP leaderboard until recently. It was the highest ranked model with both open-sourced code published and final weights of the fully trained neural network publicly available. However, as evident from the results, it has common problems ranging from factual details being reproduced incorrectly, abrupt termination of sentences, grammatical errors and at times producing repetitive phrases.

We have shown in the previous two chapters that an alternative training approach to finetune the baseline yields positive results in improving the quality of summary produced. In this work, the baseline generator was trained on linear combination of two loss signals - the baseline model's native maximum likelihood loss + the discriminator's classification loss. This led to significant improvement in the quality of outputs. The discriminator learnt to pick on aspects of language which render the text poor quality. By being able to distinguish between realistic and fake text, the discriminator was effective in guiding the generator to produce more realistic text.

Our results were backed up both qualitatively and quantitatively. Quantitatively, our model had a higher mean score on both the ROUGE-1 and ROUGE-2. There was an improvement of 0.03 in the mean ROUGE-1 score and 0.02 in the mean ROUGE-2 score. Although this is an improvement on the baseline, ROUGE scores are limited in making inferences on the quality of a summary. It is more relevant to observe the outcomes of qualitative evaluation.

The finetuned model were given higher scores by human judges on all three aspects - readability, factual accuracy and overall. On readability, our model was judged to be 1.3 points higher and on factual accuracy 0.2 points higher. This supports our initial hypothesis that the baseline model was able to identify relevant parts of the source fairly well but can struggle to reproduce them in a realistic way. Furthermore, our finetuned model, is only trained to identify qualitative shortcomings and only improve quality. The discriminator part of our model is not to trained to distinguish accurate content from inaccurate content and thus the training signal from it should not lead to much change in the content that is reproduced.

The results shared in this section only illustrate some qualitative aspects of linguistics that our model learnt to identify and finetune. The reader is referred to published results on Github for further reference.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

Being one of the most successful applications of sequence models, neural abstractive text summarization has become a prominent research topic that has gained a lot of attention from both industry and academia. Abstractive word summarization has proven to be a difficult NLP challenge as the task involves a good understanding of a language model to reproduce both structurally correct and meaningful sentences. Automatic summarization evaluation using quantitative metrics have seen to pose severe limitations in the quality of summaries generated. Despite the difficulty of the task, in this thesis we focus on improving the quality of the decoder to produce better summaries. In this thesis, we finetuned a highly-ranked baseline model to produce summaries that are more human looking. This work primarily focused on the challenges of evaluating summaries using automatic quantitative metrics such as ROUGE and proposed an evaluation metric, on the basis of linguistic acceptability. While our results were positive for the research question posed, we identify that there are number of aspects this work could be built on to potentially yield breakthrough results.

6.2 Future Work

The work undertaken during this thesis can be considered as a step in the direction of generating text that is more human-like and abstractive in nature.

6.2.1 Interesting Aspects to Study

Although the methods used here were simple in relation to many advanced neural network models that exist today, qualitative evaluation showed that the levels of abstraction of summaries generated from our model were better. At the same time, there is still a substantial gap between the quality of the summaries generated from neural network models and human-written ones. Below are potential aspects to study for future work. For ease of model comparison as well as time constraints, hyperparameter search was not conducted. All hyperparameters used by this model were based on previous research. A better model could maybe found by varying one hyperparameter at a time. Another potential future direction, is having a metric that can capture the accuracy of factual information reproduced by the generated summaries. Neither ROUGE nor the linguistic acceptability score can be a good measure for knowing if the relevant parts of the source text is reproduced. It is worth exploring how the our model performs on datasets from other domains. As mentioned previously, CNN/Daily Mail dataset is the standard de facto dataset for text summarization. Good summarization models must be able to generalize well to datasets from other domains without loss of performance. In [77] a novelty metric is proposed that measures the novelty of generated sequences in comparison to the summarized articled and thus the abstraction of the trained model. This idea could be used to evaluate the abstraction levels achieved by our model.

6.2.2 Possible Improvements to the Model

Recently *transformers* [78] have begun outperforming RNNs/LSTMs on sequence to sequence tasks like machine translation. However, early attempts to apply transformers to generate summaries, have resulted in repetitive and factually incorrect outcomes. As transformer models improve and evolve, it could be worth exploring hybrid approaches to summarization using transformers and pointer-generator/coverage mechanisms.

Just as how not all words contribute equally to represent a sentence's meaning, not all sentences in an article contribute equally to the article's content. Sentence attention uses attention mechanism to introduce a sentence level context vector to measure the importance of sentences. *Hierarchical Attention Networks* [79] adopt a combination of word attention and sentence attention. This approach has the potential to improve the results of our model, especially with regards to factual accuracy.

In a typical GAN model, the generator and the discriminator are trained in alternate fashion. One's weights are frozen while the other trained for finite number of epochs after which the training happens viceversa. This process continues until convergence i.e. the discriminator can not distinguish the outputs being real from fake of the generator anymore. Our model was not trained back and forth all the way to convergence, as this highly computationally expensive and requires significant manual hyperparameter tuning. However, if the model could be trained such, the performance of the model would very likely be improved.

Bibliography

[1] Gregory J. Chaitin. "The limits of reason." In: *Scientific American* 294 3 (2006), pp. 74–81.

[2] Udo Hahn and Inderjeet Mani. "The Challenges of Automatic Summarization". In: *Computer* 33.11 (Nov. 2000), pp. 29–36. ISSN: 0018-9162. DOI: 10.1109/2.881692. URL: http://dx.doi.org/10.1109/2.881692.

[3] Tian Shi et al. "Neural Abstractive Text Summarization with Sequence-to-Sequence Models". In: *ArXiv* abs/1812.02303 (2018).

[4] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *Association for Computational Linguistics*. 2017.

[5] Lantao Yu et al. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. 2016. eprint: arXiv:1609.05473.

[6] Nikhil Buduma and Nicholas Locascio. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms 1st*. O'Reilly Media, Inc, 2017.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] Ramesh Nallapati et al. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. DOI: 10.18653/v1/K16-1028. URL: https://www.aclweb.org/anthology/K16-1028.

[9] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN: 0130950696.

[10] Jonathan J. Webster and Chunyu Kit. "Tokenization as the initial phase in NLP". In: *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*. 1992. URL: https://www.aclweb.org/anthology/C92-4173.

[11] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[12] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[13] Warren Sturgis McCulloch and Walter H. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *Bulletin of Mathematical Biology* 52 (1988), pp. 99–115.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Neurocomputing: Foundations of Research”. In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. Learning Representations by Backpropagating Errors, pp. 696–699. ISBN: 0-262-01097-6. URL: http://dl.acm.org/citation.cfm?id=65669.104451.

[15] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *Ann. Math. Statist.* 22.3 (Sept. 1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[16] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (2014). eprint: arXiv:1412.6980.

[17] Yoav Goldberg and Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017. ISBN: 1627052984, 9781627052986.

[18] Yann LeCun et al. “Object Recognition with Gradient-Based Learning”. In: *Shape, Contour and Grouping in Computer Vision*. London, UK, UK: Springer-Verlag, 1999, pp. 319–. ISBN: 3-540-66722-9. URL: http://dl.acm.org/citation.cfm?id=646469.691875.

[19] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[20] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[21] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. URL: http://doi.acm.org/10.1145/1390156.1390177.

[22] Y-Lan Boureau, Jean Ponce, and Yann LeCun. "A Theoretical Analysis of Feature Pooling in Visual Recognition". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 111–118. ISBN: 978-1-60558-907-7. URL: http://dl.acm.org/citation.cfm?id=3104322.3104338.

[23] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[24] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks". In: (2015). eprint: arXiv:1505.00387.

[25] Jiaxian Guo et al. *Long Text Generation via Adversarial Training with Leaked Information*. 2017. eprint: arXiv:1709.08624.

[26] Andrej Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. [Online; posted 21-May-2015]. May 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[27] Darlene Olah. *Understanding LSTM Networks*. [Online; posted 27-August-2015]. Aug. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[28] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.0473.

[29] Mark F. Medress et al. "Speech Understanding Systems". In: *Artif. Intell.* 9 (1976), pp. 307–316.

[30] Alexander M. Rush, Sumit Chopra, and Jason Weston. "A Neural Attention Model for Abstractive Sentence Summarization". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 379–389. DOI: 10.18653/v1/D15-1044. URL: https://www.aclweb.org/anthology/D15-1044.

[31] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. *Pointer Networks*. 2015. eprint: arXiv:1506.03134.

[32] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3 (May 1992), pp. 229–256. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

[33] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. eprint: arXiv:1406.2661.

[34] Yizhe Zhang et al. "Adversarial Feature Matching for Text Generation". In: *ICML*. 2017.

[35] Lijun Wu et al. *Adversarial Neural Machine Translation*. 2017. eprint: arXiv:1704.06933.

[36] Luke Metz et al. *Unrolled Generative Adversarial Networks*. 2016. eprint: arXiv:1611.02163.

[37] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. eprint: arXiv:1701.04862.

[38] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[39] Natalie Schluter. "The limits of automatic summarisation according to ROUGE". In: *EACL*. 2017.

[40] H. P. Luhn. "The Automatic Creation of Literature Abstracts". In: *IBM J. Res. Dev.* 2.2 (Apr. 1958), pp. 159–165. ISSN: 0018-8646. DOI: 10.1147/rd.22.0159. URL: http://dx.doi.org/10.1147/rd.22.0159.

[41] H. P. Edmundson. "New Methods in Automatic Extracting". In: *J. ACM* 16.2 (Apr. 1969), pp. 264–285. ISSN: 0004-5411. DOI: 10.1145/321510.321519. URL: http://doi.acm.org/10.1145/321510.321519.

[42] Dipanjan Das and André F. T. Martins. *A Survey on Automatic Text Summarization*. 2007.

[43] Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. "Headline Generation Based on Statistical Translation". In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. ACL '00. Hong Kong: Association for Computational Linguistics, 2000, pp. 318–325. DOI: 10.3115/1075218.1075259. URL: https://doi.org/10.3115/1075218.1075259.

[44] Sander Wubben, Antal van den Bosch, and Emiel Krahmer. "Sentence Simplification by Monolingual Machine Translation". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. ACL '12. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 1015–1024. URL: http://dl.acm.org/citation.cfm?id=2390524.2390660.

[45] Kristian Woodsend, Yansong Feng, and Mirella Lapata. "Title Generation with Quasi-Synchronous Grammar". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, Oct. 2010, pp. 513–523. URL: https://www.aclweb.org/anthology/D10-1050.

[46] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 3104–3112. URL: http://dl.acm.org/citation.cfm?id=2969033.2969173.

[47] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. eprint: arXiv:1406.1078.

[48] Sumit Chopra, Michael Auli, and Alexander M. Rush. "Abstractive Sentence Summarization with Attentive Recurrent Neural Networks". In: *HLT-NAACL*. 2016.

[49] Sho Takase et al. "Neural Headline Generation on Abstract Meaning Representation". In: *EMNLP*. 2016.

[50] Yishu Miao and Phil Blunsom. "Language as a Latent Variable: Discrete Generative Models for Sentence Compression". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 319–328. DOI: 10.18653/v1/D16-1031. URL: https://www.aclweb.org/anthology/D16-1031.

[51] Marc'Aurelio Ranzato et al. *Sequence Level Training with Recurrent Neural Networks*. 2015. eprint: arXiv:1511.06732.

[52] Karl Moritz Hermann et al. "Teaching Machines to Read and Comprehend". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1693–1701. URL: http://dl.acm.org/citation.cfm?id=2969239.2969428.

[53] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. "SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents". In: *AAAI*. 2016.

[54] Hongyan Jing. "Sentence Reduction for Automatic Text Summarization". In: *Proceedings of the Sixth Conference on Applied Natural Language Processing*. ANLC '00. Seattle, Washington: Association for Computational Linguistics, 2000, pp. 310–315. DOI: 10.3115/974147.974190. URL: https://doi.org/10.3115/974147.974190.

[55] Jackie Chi Kit Cheung and Gerald Penn. "Unsupervised Sentence Enhancement for Automatic Summarization". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 775–786. DOI: 10.3115/v1/D14-1085. URL: https://www.aclweb.org/anthology/D14-1085.

[56] Samy Bengio et al. "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1171–1179. URL: http://dl.acm.org/citation.cfm?id=2969239.2969370.

[57] Alex Lamb et al. *Professor Forcing: A New Algorithm for Training Recurrent Networks*. 2016. eprint: arXiv:1610.09038.

[58] Ferenc Husz'ar. "How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?" In: 2015.

[59] Sam Wiseman and Alexander M. Rush. "Sequence-to-Sequence Learning as Beam-Search Optimization". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1296–1306. DOI: 10.18653/v1/D16-1137. URL: https://www.aclweb.org/anthology/D16-1137.

[60] Kishore Papineni et al. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[61] Michael Denkowski and Alon Lavie. "Meteor Universal: Language Specific Translation Evaluation for Any Target Language". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 376–380. DOI: 10.3115/v1/W14-3348. URL: https://www.aclweb.org/anthology/W14-3348.

[62] Caglar Gulcehre et al. *Pointing the Unknown Words*. 2016. eprint: arXiv:1603.08148.

[63] Stephen Merity et al. *Pointer Sentinel Mixture Models*. 2016. eprint: arXiv:1609.07843.

[64] Jiatao Gu et al. "Incorporating Copying Mechanism in Sequence-to-Sequence Learning". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1631–1640. DOI: 10.18653/v1/P16-1154. URL: https://www.aclweb.org/anthology/P16-1154.

[65] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. eprint: arXiv:1409.0473.

[66] Dzmitry Bahdanau et al. "An Actor-Critic Algorithm for Sequence Prediction". In: *ArXiv* abs/1607.07086 (2016).

[67] Richard S. Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Denver, CO: MIT Press, 1999, pp. 1057–1063. URL: http://dl.acm.org/citation.cfm?id=3009657.3009806.

[68] Kevin Lin et al. *Adversarial Ranking for Language Generation*. 2017. eprint: arXiv:1705.11001.

[69] William Fedus, Ian Goodfellow, and Andrew M. Dai. *MaskGAN: Better Text Generation via Filling in the* . 2018. eprint: arXiv:1801.07736.

[70] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://www.aclweb.org/anthology/D14-1181.

[71] Yoshua Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *Neural Networks: Tricks of the Trade*. 2012.

[72] Dani Yogatama and Noah A. Smith. "Bayesian Optimization of Text Representations". In: *CoRR* abs/1503.00693 (2015). arXiv: 1503.00693. URL: http://arxiv.org/abs/1503.00693.

[73] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[74] Yoav Goldberg. "A Primer on Neural Network Models for Natural Language Processing". In: *J. Artif. Int. Res.* 57.1 (Sept. 2016), pp. 345–420. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=3176748.3176757.

[75] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. "Understanding Convolutional Neural Networks for Text Classification". In: *BlackboxNLP@EMNLP*. 2018.

[76] Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. *On Accurate Evaluation of GANs for Language Generation*. 2018. eprint: arXiv:1806.04936.

[77] WOJCIECH KRYSCINSKI. *Training Neural Models for Abstractive Text Summarization*. 2018.

[78] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[79] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://www.aclweb.org/anthology/N16-1174.

[80] Joshua Goodman. "A Bit of Progress in Language Modeling". In: *CoRR* cs.CL/0108005 (2001). URL: http://arxiv.org/abs/cs.CL/0108005.

[81] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944966.

Appendix A

Appendix

A.1 Highway Layer

Given a plain feed-forward neural network with L layers, where each layer has a nonlinear transfer function H_l parametrised by the weights W_h, the output from each layer is given by

$$y_l^{\text{plain}} = H_l\left(x_l, W_{H,l}\right) \quad l \in \{1, \ldots, L\}$$

given input data x_l. The highway layer architecture extends this model by introducing a transform gate $T_l\left(x_l, W_{H,l}\right)$ and a carry gate $C_l\left(x_l, W_{C,l}\right)$, which are both nonlinear functions. The gates define the output from each layer as

$$y_l = H_l\left(x_l, W_{H,l}\right) \cdot T_l\left(x_l, W_{H,l}\right) + x_l \cdot C_l\left(x_l, W_{C,l}\right)$$

A simplification made by Srivatava et al. is to set

$$C_l = 1 - T_l, \quad l \in \{1, \ldots, L\}$$

.

As a consequence of introducing the highway architecture, the derivative of each layer l in the network $H\left(x, W_h\right)$ becomes

$$\frac{\mathrm{d}y_l}{\mathrm{d}x_l} = \begin{cases} 1, & \text{if } T_l\left(x_l, W_{H,l}\right) = 0 \\ H_l'\left(x_l, W_H\right), & \text{if } \quad T_l\left(x_l, W_{H,l}\right) = 1 \end{cases}$$

A.2 Development of Statistical Language Modeling

Noam Chomsky published several influential papers in the 1950s and 1960s. He argued that the finite-state Markov processes, such as n-grams, could never fully model the complexity of grammar in natural languages [9]. Following this, many linguists and computational linguists explored alternative methods. N-gram models gained popularity once again in the 80s, computational power and text data availability began to increase rapidly.

N-gram models, in all their simplicity, have been successful in the past, and Goodman [80] showed that the advantages of sophisticated models vanished compared to n-gram models using smoothing and a cache memory, if they are given enough training data.

While n-gram models in theory can improve indefinitely by increasing the amount of training data and computational power, the Markov assumption means that they will never be able to truly capture long term information dependencies. Further, as the n-gram order increases, the number of parameters needed to describe a model increases exponentially while the model needs even more training data, which is why models larger than 5-gram models are typically not used [9].

To conquer the sparsity that arises when modeling natural languages, Bengio et al.[81] suggested a neural network approach based on distributed word representations, with remarkable results.

A.3 Bilingual Evaluation Understudy (BLEU)

In the domain of machine translation, a widely used metric is Bilingual Evaluation Understudy (BLEU) [60], which is based on the assumption that the closer a text resembles a human reference sequence, the better it is. The BLEU metric is used by both SeqGAN and LeakGAN and is based on n-gram overlaps of tokens between candidate and reference sequences and how well the length of the generated text matches the length of the references. The resulting BLEU score ranges from 0 to 1, where 1 corresponds to a perfect match with a reference sequence and 0 corresponds to a non-existing overlap. In the original paper on BLEU evaluation of machine translation by Papineni et al. [60], the BLEU score was compared to text evaluations done by humans with very good results, which motivates it as a good metric for text quality.

A.4 Special Tokens

Generated sequences have a fixed, predefined length. Real life sequences, though have no such restriction. In order for the generator to be able to generate genuine sequences, it therefore must have freedom in deciding how long a sequence should be. Specifically, the generator must be able to unambiguously indicate when a text ends. Following Buduma [6], this is done by adding a special *end-of-sequence* (EOS) token to the vocabulary and appending each real summary with an EOS token. To indicate that a text ends, the generator generates the EOS token. Similarly the generator starts with a special *start-of-sequence* (SOS) token and all real summaries are prepended with SOS.

In the general case, the encoder and decoder vocabularies produced from a corpus do not include all of the tokens in the corpus. This means that real sequences include tokens which are unknown to the encoder and decoder, respectively. Instead of discarding all samples which include such unknown tokens, the common practice of introducing a general *unknown-token* token (UNK) is used. In the pre-processing stage, tokens that are not included in the corresponding vocabulary are simply replaced with the UNK token.

TRITA -SCI-GRU 2019:397