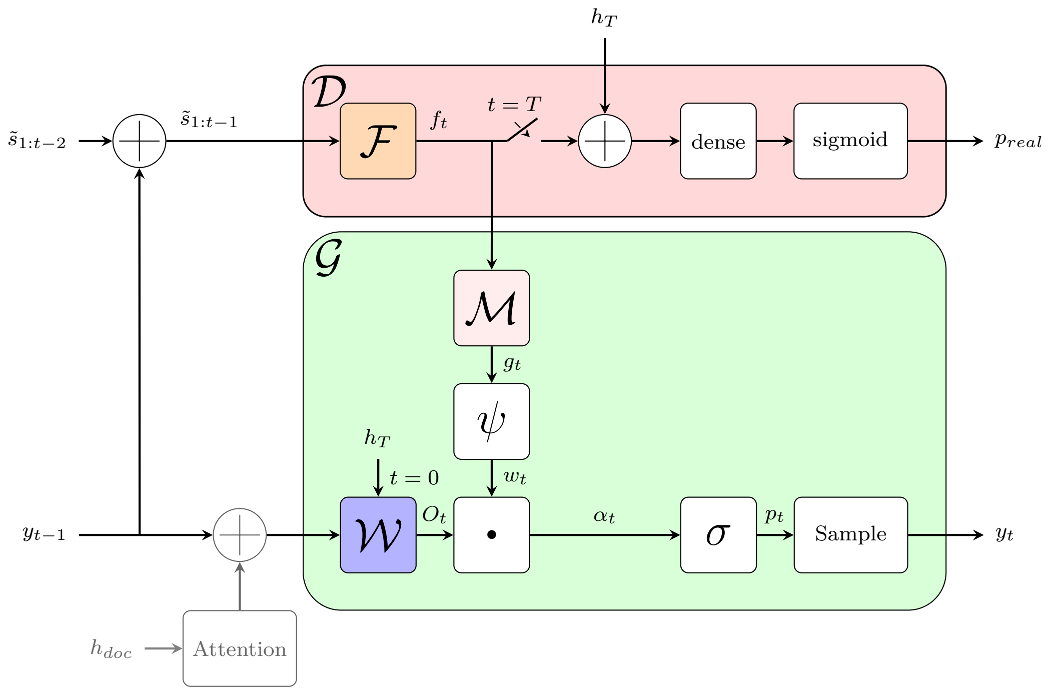
CHALMERS
UNIVERSITY OF TECHNOLOGY



Abstractive Document Summarisation using Generative Adversarial Networks

Master's thesis in Engineering Mathematics and Computational Science

JOHAN BJÖRK
KARL SVENSSON

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Abstractive Document Summarisation using Generative Adversarial Networks

JOHAN BJÖRK
KARL SVENSSON

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2018

Abstractive Document Summarisation using Generative Adversarial Networks
JOHAN BJÖRK
KARL SVENSSON

Supervisor: Rebecka Jörnsten, Department of Mathematical Sciences, Chalmers University of Technology
Examiner: Johan Jonasson, Department of Mathematical Sciences, Chalmers University of Technology

Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A schematic overview of the model proposed by the thesis – SumGAN.

Gothenburg, Sweden 2018

Abstractive Document Summarisation using Generative Adversarial Networks
JOHAN BJÖRK
KARL SVENSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The use of automatically generated summaries for long texts is commonly used in digital services. In this thesis, one method for such *document summarisation* is created by combining existing techniques for abstractive document summarisation with *LeakGAN* – a successful approach at text generation using generative adversarial networks (GAN). The resulting model is tested on two different datasets originating from conventional newspapers and the world's largest online community: Reddit. The datasets are examined and several important differences are highlighted. The evaluations show that the summaries generated by the model do not correlate with the corresponding documents. Possible reasons are discussed and several suggestions for future research are presented.

Keywords: abstractive, adversarial, document, gan, generative, networks, summarisation.

Acknowledgements

First, we would like to thank our advisors from Findwise AB: Niklas Helmertz, Johan Hasselqvist and Jonatan Bengtsson. Their collected knowledge has been a large asset during the work. We would also like to thank Fredrik Axelsson at Findwise for his administrative support throughout the thesis process.
Second, we thank our academic advisor Rebecka Jörnsten for her aid in deciding what aspects of the model to study.
Third, we thank the remaining staff at Findwise AB for making our thesis work an enjoyable experience.

Johan Björk and Karl Svensson, Gothenburg, June 2018

Contents

List of Figures

List of Tables

1 Introduction

In the field of natural language processing (NLP), one interesting problem is that of text summarisation. Most of existing summarisation techniques are of an *extractive* kind as they summarise documents by extracting the most relevant sentences from the documents themselves. Extractive document summarisation (EDS) is ubiquitous in search engines such as Google, Bing, and others, in which links to search results are accompanied by short extractive summaries of the content behind the links. The technique is robust, but also limited to the text within the content.

A more sophisticated technique is *abstractive* document summarisation (ADS), in which text is *generated* based on the content of the document. Because text is generated, there are infinite possible summaries to each document, which means that summaries have the potential to be very good, but also to be absolute nonsense.

An illustrating analogy to compare EDS to ADS can be made with a pair of scissors and a pen: EDS is done with the scissors by cutting the document to pieces and choosing the best pieces as the summary of the document; ADS 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 wielder of the pen.

One of the limiting factors of ADS is that the results of natural text generation is far from perfect. Previous work within ADS has shown the ability to generate accurate summaries, but suffers from inconsistency in the generation [2, 3].

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. [4] in 2014, introduces the concept of using two adversarial networks for generative tasks. The original approach has subsequently been improved to enable text generation. Examples of such augmented approaches are *SeqGAN* [5], *TextGAN* [6] and *LeakGAN* [7].

In this thesis, existing attempts at abstractive document summarisation, such as a thesis by Helmertz and Hasselqvist [8] are combined with LeakGAN in order to examine if the generative power of GAN can be used to improve the quality of generated summaries.

2

Related Work

This chapter highlights related work within the fields of document summarisation and text generation with GAN.

2.1 Document Summarisation

The thesis *Query-Based Abstractive Summarization Using Neural Networks* by Helmertz and Hasselqvist [8], which was the inspiration for this thesis, presents an attempt at generating query dependent document summaries using an encoder-decoder architecture. The basic architecture is augmented by a *pointer* and an *attention* mechanism, which enable it to produce relevant summaries. The presented model is trained on a dataset consisting of news stories from CNN and Daily Mail.

In *Neural Machine Translation By Jointly Learning To Align And Translate*, Bahdanau et al. [9] propose the use of *attention* mechanisms for improved decoding performance in an encoder-decoder model. The attention mechanism enables the decoder to access a "memory" of the encoded document, which enables Bahdanau et al. to achieve state-of-the-art results at a translation task.

In the thesis *A Continuous Approach to Controllable Text Generation using Generative Adversarial Networks*, Helgøy and Lund [10] try to use SeqGAN to produce captions of images. Helgøy and Lund do several experiments with the intention to determine whether GAN can be used successfully for controlled text generation or not. Even though the thesis deals with translation from image to text domain, the basic methodology is the same for translation from document to summary domain.

2.2 Text Generation with GAN

Yu et al. [5] introduce a novel approach for text generation using GAN 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. The model, *SeqGAN*, is tested on real as well as synthetic data and achieves promising results on both.

In *Long Text Generation via Adversarial Training with Leaked Information* Guo et al. [7] 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. approach text generation with GAN in a different way than SeqGAN and LeakGAN. Their solution, *TextGAN*, is presented in *Adversarial Feature Matching for Text Generation* [6]. 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.

In the paper *Texygen: A Benchmarking Platform for Text Generation Models* Zhu et al. [11] present *Texygen*, a benchmarking platform for evaluation of text generation models. The benchmarking platform reached the authors' attention at a late stage of the thesis work, so the benchmark is not used in the thesis work. An interesting part of the paper, though, is when the proposed benchmark are used to compare established GAN methods for text generation, including SeqGAN, LeakGAN and TextGAN. The results show a clear difference in performance between the different approaches and is a good starting point for further research.

3

Background

This section presents the theory needed to explain the model that is proposed by this thesis, as well as the metrics that are used for evaluation. Since the reader is assumed to have basic prior knowledge of artificial neural networks (ANN), only some aspects of ANN are presented here. The reader that wishes to learn more about the deep learning parts of the text below is referred to the book *Fundamentals of Deep Learning* by Buduma [12], which was an important resource during the thesis work.

3.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is the field of treating human language from a computational perspective. There are many sub-fields within NLP, such as translation, entity extraction, and summarisation of texts. NLP is a vast field, and a complete review of it is beyond the scope of this thesis. Only parts of NLP that are relevant to this thesis are described in this section.

3.1.1 Tokenisation

For text processing purposes, it is of great importance to be able to identify significant units, so called *tokens*, of a given text. The act of dividing a text into tokens is called *tokenisation* [13]. These basic units, which can be defined on several levels, from morphemes to words and even phrases [14], 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 $\tilde{s}_{1:T}$ or $\tilde{s}$ denote machine generated sequences, where T is the number of tokens in the 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. Batches are formatted in bold font, so a batch of sequences is written as $\boldsymbol{s}$.

3.1.2 Word Embeddings

Once a text has been tokenised, it can be processed further by using the concept of *word embeddings* in order to map words into real valued vectors in a continuous vector space. 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

$$d = \frac{a \cdot b}{\|a\|_2 \, \|b\|_2}.$$

A common approach is to use already trained word embeddings, such as Word2Vec [15] and GloVe [16] as the embedded representation of the vocubulary of choice. An alternative approach would be to initialise the word embeddings randomly in a vector space, and then optimise the distribution of word embeddings as the training of the model progresses. This method is adopted by LeakGAN, see Section 3.5.3 and since our model extends the LeakGAN model, we chose to train our word embeddings without any pre-trained vectors.

3.2 Artificial Neural Networks (ANN)

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* [12].

The capability of the human brain does not come from a single neuron, but from an enormous network thereof; according to Buduma [12], a piece of brain the size of a grain of rice contains 10,000 neurons.

In 1943, McCulloch and Pitts introduced a model [17] set to mimic the neurons in the human brain. The so called *artificial neuron*

$$y = f(Wx + b),$$

which is shown in Figure 3.1, is a mathematical function that combines its inputs x in a matrix multiplication with weights W and, optionally, adds a constant bias b. The result is fed into a nonlinear *activation function* $f(\cdot)$ which typically ranges between $[0, 1]$ or $[-1, 1]$. By adjusting W and b properly, the neuron can be trained to output a signal y suitable for a specific set of inputs x. The process of tuning W and b is often referred to as *learning* or *training*.

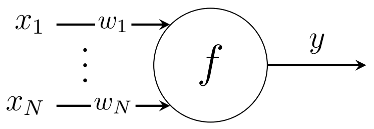


Figure 3.1: An artificial neuron with inputs $x_{1,\ldots,N}$, weights $w_{1,\ldots,N}$, activation function f and output y.

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. 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 3.2.

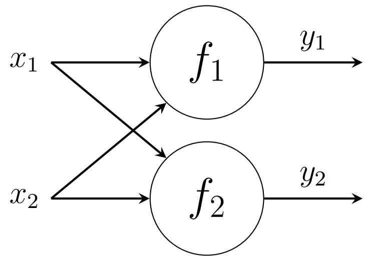


Figure 3.2: A simple artificial neural network with two inputs and one layer consisting of two neurons.

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. [18]. During training, a *sample* x with a corresponding correct output y is transmitted through the ANN, which outputs $\tilde{y}$. Unless the ANN is perfect, there will be an *error* $E = \tilde{y} - y$, which the network has to minimise in order to improve its performance. Backpropagation does this by differentiating E with respect to the weights w, which can then in turn be updated via various methods such as stochastic gradient descent (SGD) [19] or Adaptive Moment Estimation (Adam) [20].

3.2.1 Dropout

A common problem of training ANNs is *overfitting* which means that the network adjusts itself too well to fit the training data, with the effect that the performance for unseen data is bad. In other words: the model does not generalise well. An effective and simple method to counteract this is *Dropout* [21] in which, during training, neurons are active based on a probability. Thus, each training iteration features a unique combination of neurons, which forces the network to generalise.

3.2.2 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. The outputs $\alpha_i,\ i = 1, \dots, N$ (often called *logits*) from a layer of neurons, do, however, not always add up to one, which is a requirement if the logits should represent a probability distribution. A common way to assure that the logits satisfy the requirement is to use the softmax function

$$\sigma(\alpha_i) = \frac{\exp(\alpha_i)}{\sum_{j=1}^{N} \exp(\alpha_j)}. \tag{3.1}$$

The softmax function can be viewed as a distribution over logits, where $[\sigma(\alpha_1), \dots, \sigma(\alpha_N)]$ denotes the probabilities for selecting an outcome related to one of the logits $\alpha_1, \dots, \alpha$. 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

$$\sigma(\alpha_i) = \frac{\exp(\alpha_i/\tau)}{\sum_{j=1}^{N} \exp(\alpha_j/\tau)}.$$

If $\tau > 1$, it will result in a flatter distribution than (3.1), and if $0 < \tau < 1$, 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 α_1 and α_2 with large and small values, respectively, is smaller than the ratio $\sigma(\alpha_1) : \sigma(\alpha_2)$. A special case of the softmax is when $N = 2$ when the softmax function is referred to as a *logistic sigmoid.*

3.3 Discriminative Models

Within the field of machine learning, a common category of models is *discriminative models*, whose 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 $p(y \mid x)$ for the outcome y, given input data x.

One neural network architecture that has grown very popular when building discriminative models is the convolutional neural network, as presented below.

3.3.1 Convolutional Neural Networks (CNN)

A *Convolutional Neural Network* (CNN), is a type of feed forward network in which the hidden layers consist of convolutional layers (see Buduma [12] for a thorough explanation). An example of a CNN used for text classification in the paper on TextGAN by Zhang et al. [6] is presented in Figure 3.3. The convolutional layer consists of a set of filters of varying sizes that project the input data to a lower dimension.

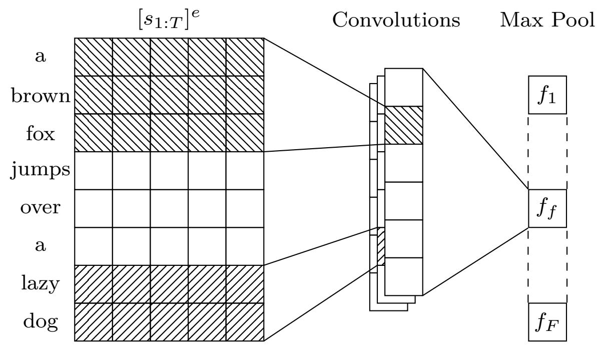


Figure 3.3: A CNN used to extract features $f_{1,\ldots,F}$ from a text.

A method for reducing the size of the data in between layers is a technique called *downsampling* 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 3.3, the max pool operation is performed on the entire output from each filter, but it could also have been performed on subsets of the outputs, in which case the dimensionality reduction would not have been as drastic.

3.3.2 Highway Layer

When training deep neural networks with many layers, a common challenge is *vanishing gradients* which means that the gradients used in the backpropagation algorithm become very small (vanish) when calculated through several layers [12].

A remedy for this, proposed by Srivastava et al. is the *Highway Layer* [22] 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.

Given a plain feedforward 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

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

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

$$y_l = H_l(x_l, W_{H,l}) \cdot T_l(x_l, W_{H,l}) + x_l \cdot C_l(x_l, W_{C,l}) \qquad l \in \{1, \ldots, L\}.$$

A simplification made by Srivastava et al. is to set $C_l = 1 - T_l, l \in \{1, \ldots, L\}$.

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

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

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 for deep architectures. Highway networks have been used with success in text generation, e.g. SeqGAN, Section 3.5.2 and LeakGAN, Section 3.5.3.

3.4 Generative Models

As opposed to discriminative models, *generative models* serves the purpose of generating 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 the set of parameters θ^{g*} which maximises 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 $\tilde{x}$ from $p_g(x; \theta^{g*})$.

There are several different neural network architectures that are designed for generative modelling, of which some common techniques used for text generation are presented below.

3.4.1 Recurrent Neural Networks (RNN)

Recurrent neural network (RNN) is a type of neural network that, in contrast to feed forward networks can have cyclic connections within layers, i.e. *recurrent* connections. This gives the network the ability to handle a data sequence of variable length by having 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 3.4 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 output values and h_t are *internal hidden states* that are the means with which information is passed on to other time steps.

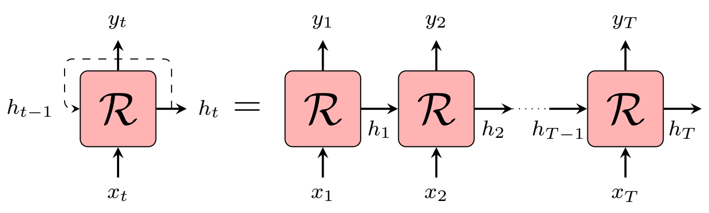


Figure 3.4: A recurrent neural network depicted in a compact format (left) and rolled out (right).

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

$$h_t^j = \begin{cases} 0 & \text{if } t = 0 \\ g(W(x_t \oplus h_{t-1}))^j & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, k \tag{3.2}$$

where g is an activation function, e.g. the logistic sigmoid function presented in Section 3.2.2, and W is a weight matrix associated with the input and the previous hidden state which are concatenated, denoted 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 3.4.2 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 [23].

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 conditional probability distributions for each token $\tilde{y}_{t-1}$ given the previous tokens generated in the sequence as

$$p(y_t \mid y_{t-1}, \ldots, y_1) = g(h_t), \tag{3.3}$$

with g being an activation function and $h_t = [h_t^1, \ldots, h_t^k]$ defined in (3.2).

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, (3.3) 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 highest probability in (3.3), 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 consisting of t tokens is then calculated as

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

3.4.2 Long Short Term Memory (LSTM) Units

A *Long Short Term Memory* (LSTM) network is a network containing *LSTM units*. One such unit is illustrated in Figure 3.5, which is inspired by an illustration in an essay by Olah [1]. As Olah describes, there are several different LSTM implementations [1], but below, the version used by LeakGAN is described.

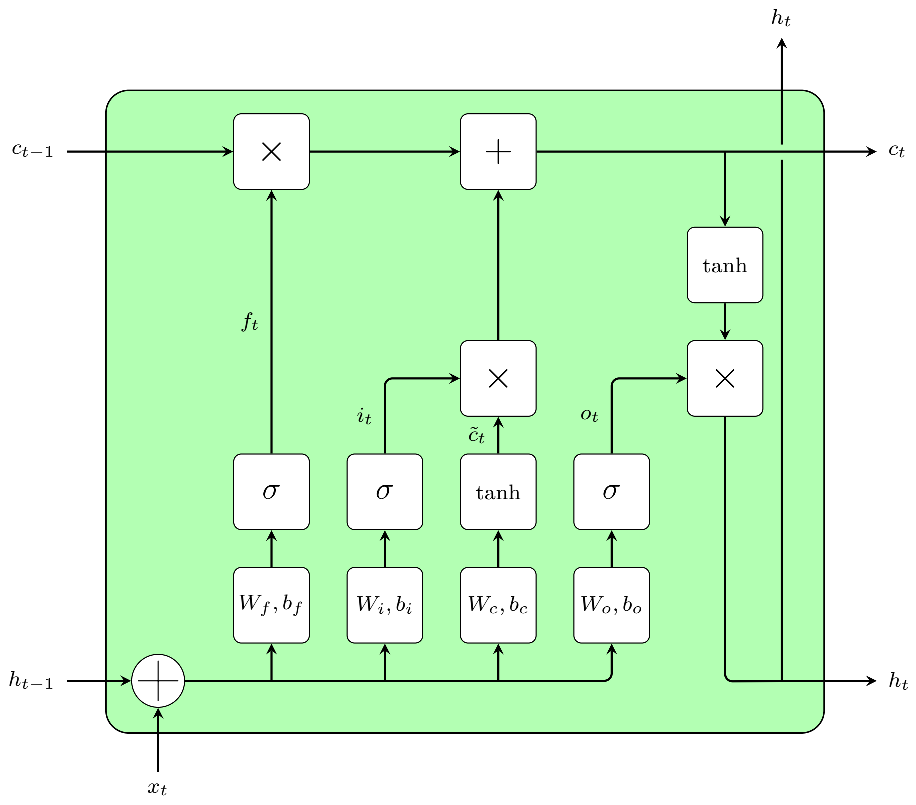


Figure 3.5: The inner workings of the LSTM unit used by LeakGAN. The illustration is inspired by an essay by Olah [1].

The concept behind LSTM units is to, in contrast to the recurrent units defined in Section 3.4.1, let each update of the j^{th} hidden state h_t^j at time t be computed as

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

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

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

with weight matrices W_o and bias b_o.

The variable c_t^j 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

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

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

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

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

3.4.3 Encoder-Decoder Model and Attention Mechanism

One common approach to generating sequences of text based on some input data is to use an *encoder-decoder* model, for which the general idea is to use an *encoder* $\mathcal{E}_n$ to encode the input data into a low dimensional embedding and to use a *decoder* $\mathcal{D}_e$ to decode the embedding to an output. This is illustrated in Figure 3.6 where 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* denote *start of sentence* and *end of sentence*, see Section 5.1.2.1.

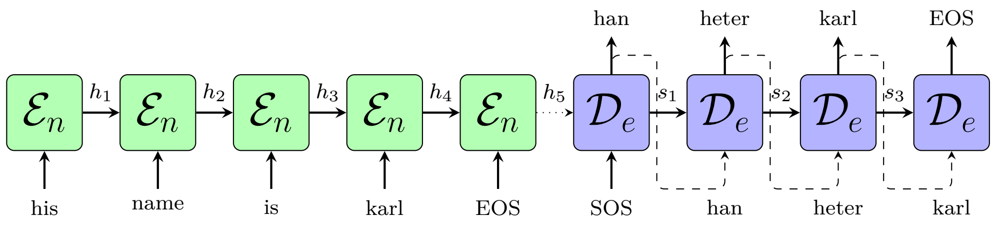


Figure 3.6: A simple encoder-decoder model consisting of an encoder $\mathcal{E}_n$ and a decoder $\mathcal{D}_e$.

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 the illustration) or document summarisation, which motivates the use of encoder-decoders in this thesis. For more information on encoder-decoder models, see Buduma [12].

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 texts such as "his name is karl", this is not a problem, but for longer sequences, or in more data intense applications such as sound to sequence, it quickly becomes an issue.

To deal with this issue, an *attention mechanism* can be utilised. 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 3.7, 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 3.6, but rather a concatenation of the previously generated token and the output from the attention mechanism.

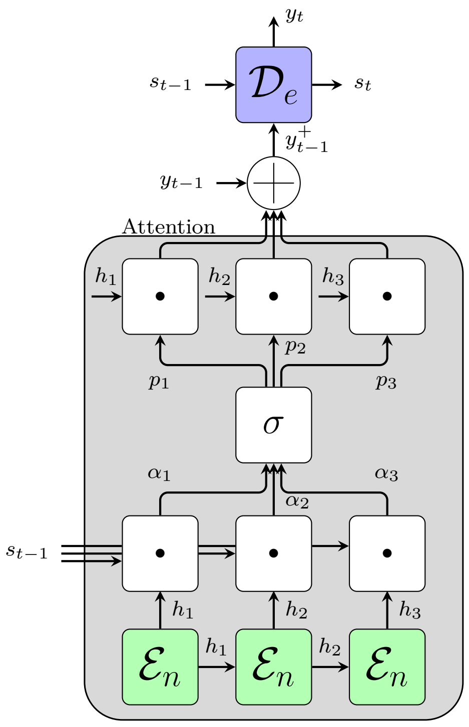


Figure 3.7: An encoder-decoder model with an attention mechanism.

The inner workings of the attention mechanism varies between implementations, but a common practice presented by Bahdanau et al. [9] 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 3.7. This is done by scoring the importance of each encoder output h_j as the dot product between h_j and s_{t-1}, normalising the scores with a softmax, and then weighting h_j with the resulting, normalised score. As defined in Figure 3.7, the output of the previous recurrence y_{t-1}, which is used as input to $\mathcal{D}_e$ in the simple encoder-decoder model presented in Figure 3.6 is concatenated with the outcome of the attention mechanism, forming y^+_{t-1}, which is instead used as an input to $\mathcal{D}_e$.

In mathematical terms

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

3.4.4 Reinforcement Learning

Some recent advancements in the field of text generation [5, 7] 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 objective of the agent is to, based on its *state* σ, take actions that maximises 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.

One RL method that has been used within text generation using GAN is the REINFORCE algorithm [24] which focuses on maximising 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 $\mathcal{L}_{\pi_\theta}(\theta)$ to be minimised with respect to the parameters θ of the policy $\pi_\theta(\sigma, a)$ can be formulated as the negative expected reward r:

$$\mathcal{L}_{\pi_\theta}(\theta) = -\mathbb{E}_{\pi_\theta}[r] = -\sum_{\sigma\in\mathcal{S}} d(\sigma) \sum_{a\in\mathcal{A}} \pi_\theta(\sigma, a) r_{\sigma,a} \tag{3.4}$$

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

In order to be able to update θ, an expression for $\nabla_\theta \mathcal{L}_{\pi_\theta}(\theta)$ is needed. By using the fact that $\nabla_\theta \pi_\theta(\sigma, a) = \pi_\theta(\sigma, a)\frac{\nabla_\theta \pi_\theta(\sigma,a)}{\pi_\theta(\sigma,a)}$ 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 in equation (3.4) can be defined as

$$\begin{aligned} \nabla_\theta \mathcal{L}_\pi(\theta) &= \sum_{\sigma\in\mathcal{S}} d(\sigma) \sum_{a\in\mathcal{A}} \pi_\theta(\sigma, a) \nabla_\theta \log \pi_\theta(\sigma, a) r_{\sigma,a} \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\sigma, a) r]. \end{aligned} \tag{3.5}$$

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 Tree Search*, in which a subsequence of actions $a_{1:t},\ t < T$ is generated at first. In order to obtain a complete action sequence $a_{1:T}$, the rest of the action sequence $a_{t+1:T}$ is then sampled using the policy $\pi_\theta(\sigma, a \mid \sigma_t, a_t)$ 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.

3.5 Generative Adversarial Networks (GAN)

A specific strategy for training generative models that has received a lot of attention since its introduction by Goodfellow et al. in 2014 [4] is *Generative Adversarial Networks*, commonly known as GAN.

GAN can be pictured as a game between two networks, a *generator* $\mathcal{G}$ and a *discriminator* $\mathcal{D}$ as illustrated in Figure 3.8. 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 [25]. 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.

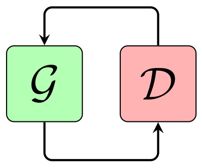


Figure 3.8: A schematic representation of the GAN architecture.

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 p_r or p_g, which are the distributions of real and generated data, respectively. The aim of $\mathcal{D}$ is thus to learn enough about the features of p_r 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 $\mathcal{G}(\boldsymbol{z})$ is a cost function $\mathcal{L}_{\mathcal{G}}(\theta^{\mathcal{G}})$ which the generator wishes to minimise.

$\mathcal{D}$ 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 $\mathcal{D}$ and $\mathcal{G}$ can be viewed as a zero sum game, or a *minimax game*, where a value function $\mathcal{L}_{GAN} = \mathcal{L}_{\mathcal{G}} = -\mathcal{L}_{\mathcal{D}}$ can be defined to describe the total payoff of the game [25]. Solving (i.e. finding the Nash equilibrium of) the minimax game, corresponds to finding the optimal generator parameters

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

There are various ways of defining the loss functions for $\mathcal{D}$ and $\mathcal{G}$. Goodfellow [25] introduces the *cross entropy* as the loss function

$$\mathcal{L}_{GAN} = \mathbb{E}_{x\sim p_r} \log \mathcal{D}(x) + \mathbb{E}_{z\sim p_g} \log\left[1 - \mathcal{D}(\mathcal{G}(z))\right]. \tag{3.6}$$

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

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

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

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

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, among others, Arjovsky and Bottou [26]. 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 [27].

3.5.1 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 [4] 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 3.5.

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 [26]. The implication is that the contribution of $\mathcal{G}$ to the learning signal vanishes as $\mathcal{D}$ approaches a local optimum [6].

3.5.2 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 3.5.1, by treating the generator as an agent in reinforcement learning, which was presented in Section 3.4.4. SeqGAN is also the foundation the more advanced attempt, LeakGAN, which is described further in Section 3.5.3 below, and which in turn is the basis for SumGAN. Therefore, in this section, only the parts of SeqGAN that are also used by LeakGAN are described.

The sequence of generated tokens is viewed as the state σ of the system and the next token to be generated is the action a 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) $\mathcal{G}_\theta\left(y_t \mid \tilde{s}_{1:t-1}\right)$ is to generate a sequence $\tilde{s}_{1:T} = \left[y_1, \ldots, y_t, \ldots, y_T\right],\ y_t \in V$, 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

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

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

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

which means that the reward for an entire generated sequence is the verdict of the discriminator.

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 ${}^{MC}\tilde{s}^n_{1:T}$ as described in Section 3.4.4. 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

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

$\mathcal{D}_\phi$ is updated by minimising the cross entropy cost function (3.6).

To update the generator with gradient descent, $\nabla_\theta \mathcal{L}_G(\theta)$ has to be approximated. Yu et al. [5] do this in accordance with the REINFORCE algorithm described in Section 3.4.4 as

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

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

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

with learning rate α.

3.5.3 LeakGAN

An alternative approach to text generation using GAN that has previously shown promising results is *LeakGAN* [7] developed by Zhang et al.. The basic structure of LeakGAN is quite similar to the one of SeqGAN in the sense that they both model the text generation as a sequential decision making process, and the training is done using policy gradient, see Section 3.4.4. The main difference between SeqGAN and LeakGAN is that the latter uses recent advancements in hierarchical reinforcement learning, proposed by Vezhnevets et al. [28], to divide the task of text generation into two parts: a manager $\mathcal{M}$ and a worker $\mathcal{W}$, as shown in Figure 3.9. In essence, this lets the discriminator $\mathcal{D}$ *leak* extracted features f to the generator $\mathcal{G}$ in order to help the generator produce more realistic texts.

The schematic overview of LeakGAN in Figure 3.9 is described further in the sections that follow below.

3.5.3.1 Discriminator

As shown in the upper part of Figure 3.9, the discriminator $\mathcal{D}$ of LeakGAN is composed by a *feature extractor* $\mathcal{F}(s; \phi_f)$ which is a CNN with weights ϕ_f, followed by a dense layer and a sigmoid layer.

The feature extractor, which is illustrated in Figure 3.10, utilises three components in order to extract features $f = \mathcal{F}(s_{1:T};\ \phi_f)$ from $[s_{1:T}]^i$, which, as the superscript i indicates, is defined as one-hot indices. After embedding each sequence in a batch, using token embeddings as in Section 3.1.2, the embedded sequence ($[s_{1:T}]^e$ in the figure) is sent through numerous filters, each of which consist of a convolution, an activation and a max pool operation, as detailed in the enlargement in the figure.

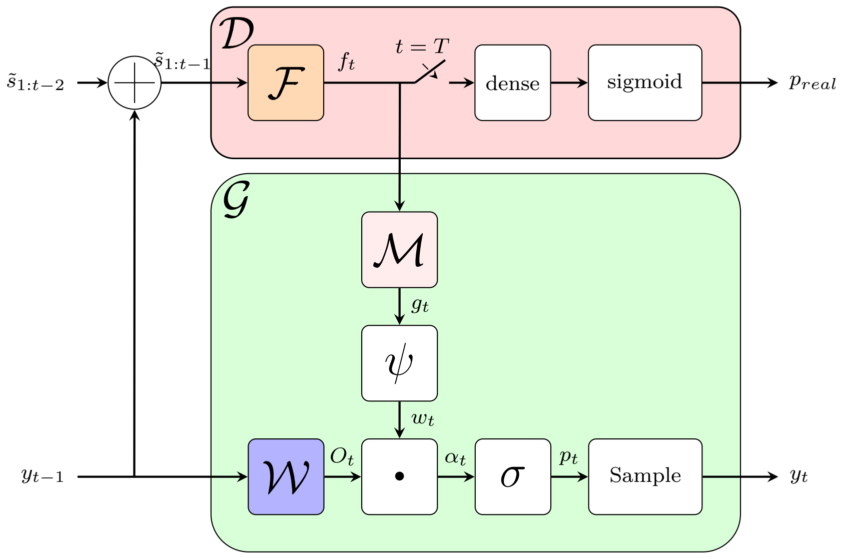


Figure 3.9: A schematic overview of LeakGAN. The generated token from the previous time step y_{t-1} is appended to the subsequence $\tilde{s}_{1:t-2}$, forming the subsequence $\tilde{s}_{1:t-1}$, which is sent to the discriminator $\mathcal{D}$. In $\mathcal{D}$, features f_t are extracted from $\tilde{s}_{1:t-1}$ by the feature extractor $\mathcal{F}$, and then sent to the manager $\mathcal{M}$. The manager output g_t is sent to the linear projection operator ϕ to produce a goal embedding w_t. The worker $\sqsupseteq$ produces an action embedding O_t which is dot multiplied with w_t to produce logits α_t. The logits are sent through a softmax function that outputs probabilities p_t for each token in the vocabulary, that are used in order to sample the next token y_t. Note that the features f_t are only used in the dense layer of $\mathcal{D}$ once the generated sequence is complete, and thereby ready for classification.

The outputs from the filters are concatenated and passed through, first, a highway layer (see Section 3.3.2) and, second, a dropout layer (see Section 3.2.1). The resulting values are the extracted features.

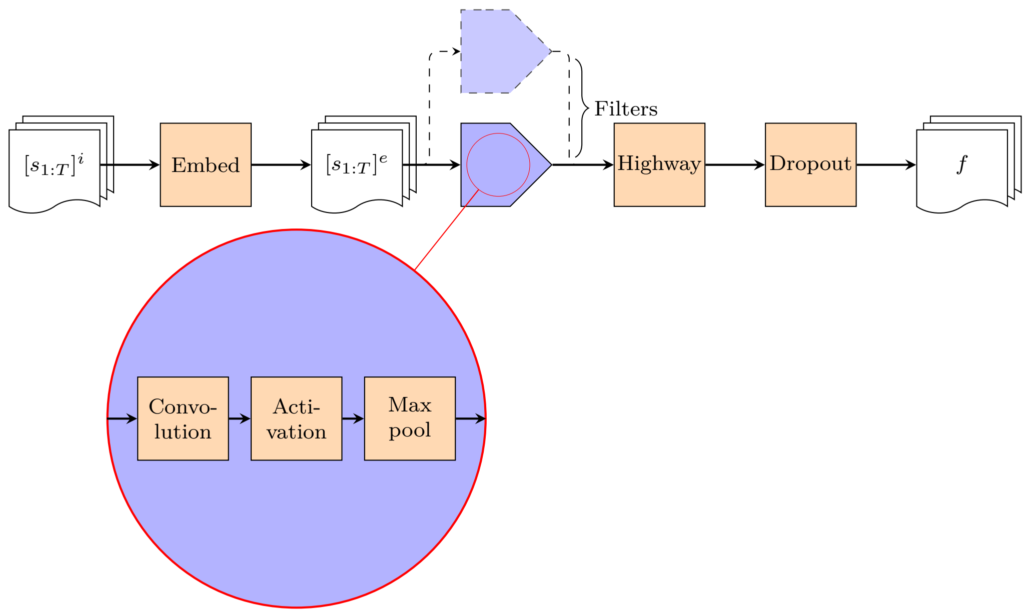


Figure 3.10: The feature extractor used in LeakGAN.

The discriminator also consists of a dense layer with weight matrix ϕ_l and bias b_l that takes the features f and produces logits α as

$$\alpha = \phi_l f + b_l.$$

The probabilities that the sequence is real or fake are then given as

$$\begin{aligned} p_{real} &= \text{sigmoid}(\alpha) = \frac{1}{1+e^{-\alpha}}, \\ p_{gen} &= 1 - p_{real}. \end{aligned}$$

These probabilities are then used to calculate the discriminator loss, which is a cross entropy defined as

$$\mathcal{L}(p_{real}, p_{gen}) = -(p_{real} \log p_{gen} + (1-p_{real}) \log(1-p_{gen})) \tag{3.7}$$

which is used to update the parameters ϕ_f, ϕ_l, b_l via backpropagation.

3.5.3.2 Manager

In order to control the transmission of leaked text features to the generation process, a *manager* module $\mathcal{M}$ is introduced, which consists of an LSTM network. The purpose of $\mathcal{M}$ is to guide the worker in an advantageous direction through its parameter space. It does so by first, at each time step t, processing features f_t from the feature extractor and outputting a *raw sub-goal* $\hat{g}_t$ of dimension $d_{manager}$ and the hidden state $h_t^{\mathcal{M}}$ of the current time step

$$\begin{aligned} \hat{g}_t, h_t^{\mathcal{M}} &= \mathcal{M}(f_t, h_{t-1}^{\mathcal{M}}; \theta_{\mathcal{M}}) \\ h_0^{\mathcal{M}} &= 0. \end{aligned}$$

The guiding signal for the worker (see Section 3.5.3.3), the d_{goal}-dimensional *goal embedding* vector w_t, is then obtained as

$$\begin{aligned} w_t &= \psi\left(\sum_{i=t-x}^{t} g_i\right) = W_\psi\left(\sum_{i=t-x}^{t} g_i\right), \qquad (3.8)\\ x &= \min(t, c)\\ g_0 &\sim \mathcal{N}(0,\ 0.01)\\ g_t &= \frac{\hat{g}_t}{\|\hat{g}_t\|} \end{aligned}$$

where the matrix of trainable weights W_ψ, with dimension $d_{goal} \times d_{manager}$, performs a linear transformation of the goal at time t added element-wise to goals from the c previous time steps. Following Guo et al. [7] the *goal horizon* constant $c = 4$. As described below in Section 3.5.3.3, the goal embedding vectors $w_t,\ t = 1, \dots T$ are used by the worker to produce logits which are later used to sample the next token.

Given a sequence $s_{1:T}$, the loss function for the manager during adversarial training, also referred to as the *goal loss* is defined as

$$\begin{aligned} \mathcal{L}_{goal,adv}(\theta_\mathcal{M}) &= -\frac{c}{T}\sum_{i=0}^{\lfloor \frac{T}{c} \rfloor - 1} r_{ic} \cdot d_{cos}(f_{(i+1)c} - f_{ic},\ g_{ic}(\theta_\mathcal{M})) \qquad (3.9)\\ f_{ic} &= \mathcal{F}(s_{1:ic}; \phi_f) \end{aligned}$$

where $d_{cos}(f_{(i+1)c} - f_{ic},\ g_{ic}(\theta_\mathcal{M}))$ is the cosine similarity (see Section 3.1.2) between the *sub feature* $f_{(i+1)c} - f_{ic}$ and the produced sub goal vector $g_{ic}(\theta_\mathcal{M})$, and r_{ic} is the *generator reward*, which is presented in greater detail in Section 3.5.3.5. The sub feature $f_{(i+1)c} - f_{ic}$ is the difference between features extracted by the feature extractor at time step $(i + 1)c$ and ic, which is equivalent to the features arising from the subsequence $s_{ic:(i+1)c}$.

The loss function $\mathcal{L}_{goal,adv}$ can be understood better by recalling the purpose of the manager; to guide the worker in latent space by pointing out directions that lead to a high reward. The manager is thus successful if a) the goal vector g_{ic} is similar to the actual transition in feature space $f_{(i+1)c} - f_{ic}$, i.e. $d_{cos}(\cdot, \cdot)$ is small, and b) the reward r_{ic} is high.

After a sequence generation round, the manager parameters $\theta_\mathcal{M}$ are updated using backpropagation with the aim of minimising (3.9)

$$\theta_\mathcal{M} \leftarrow \theta_\mathcal{M} + \eta \nabla_{\theta_\mathcal{M}} g_t(\theta_\mathcal{M})$$

with learning rate η and the gradient

$$\nabla_{\theta_\mathcal{M}} g_t(\theta_\mathcal{M}) = -r_t \nabla_{\theta_\mathcal{M}} d_{cos}(f_{t+c} - f_t,\ g_t(\theta_\mathcal{M})).$$

3.5.3.3 Worker

The second part of the generator in the LeakGAN model is the *worker* module $\mathcal{W}$, which, like the manager, is represented by an LSTM network. The worker uses its input $[y_{t-1}]^e$ (the word embedding of the previously generated token y_{t-1}) to output an *action embedding* O_t as

$$O_t, h_t^{\mathcal{W}} = \mathcal{W}([y_t]^e, h_{t-1}^{\mathcal{W}}; \theta_{\mathcal{W}}). \tag{3.10}$$

where $h_t^{\mathcal{W}}$ is the recurrent hidden vector of the LSTM network and $\theta_{\mathcal{W}}$ are the parameters of the network.

The action embedding O_t is then combined with the goal embedding vector w_t from $\mathcal{M}$ (3.8) via a matrix multiplication to produce a vector $\alpha_t = [\alpha_1, \ldots, \alpha_{|V|}]_t$ containing logits for each token in the vocabulary, as

$$\alpha_t = O_t w_t.$$

These logits are then fed into a softmax function $\sigma(\cdot)$ in order to compute a probability distribution across the vocabulary for the next token which is to be generated y_{t+1}. The next token y_{t+1} is then sampled from a discrete distribution based on the computed probabilities $\{\sigma(\alpha_t^1), \ldots, \sigma(\alpha_t^{|V|})\}$, with $|V|$ being the total number of tokens in the vocabulary.

A time step t in the text generation process, a generated subsequence $\tilde{s}_{1:t} = [y_0, \ldots, y_t],\ t < T$ is fed into the feature extractor $\mathcal{F}$, in which the features f_t of the text are extracted, and passed on to $\mathcal{M}$ which produces a new action sub goal g_t which is projected into a goal embedding vector w_t as described in (3.8).

The previously generated token y_{t-1} is fed into $\mathcal{W}$, which produces the output matrix O_t. A new probability distribution is computed by feeding the logits $\alpha_t = O_t w_t$ to a softmax $\sigma(\alpha_t)$, and sampling the next token from the resulting distribution. During training, a temperature parameter $\tau = 1.2$ is used, and to decrease the diversity during text generation, the temperature is then decreased to $\tau = \frac{2}{3}$. This procedure of generating tokens is repeated until an entire sequence is generated, i.e. when $t = T$, at which point the worker loss

$$\mathcal{L}_{worker,adv}(\theta_w) = \mathbb{E}_{\tilde{s}_{1:t-1} \sim G}\left[\sum_{y_t} r_t^I(\mathcal{W}(y_t \mid \tilde{s}_{1:t-1}; \theta_{\mathcal{W}}))\right] \tag{3.11}$$

can be computed. The parameter r_t^I is the intrinsic worker reward, which is defined as the average cosine similarity between the sequence features and the guiding goal from the manager

$$r_t^I = \frac{1}{c}\sum_{i=1}^{c} d_{cos}(\mathcal{F}(\tilde{s}_{1:t}) - \mathcal{F}(\tilde{s}_{1:t-i}), g_{t-i}). \tag{3.12}$$

The gradient of the worker can be calculated using policy gradient described in

Section 3.4.4 as

$$\begin{aligned}\nabla_{\theta_W}\mathcal{L}_{worker,adv}(\theta_w) &= \nabla_{\theta_W}\mathbb{E}_{\tilde{s}_{1:t-1}\sim G}\left[\sum_{y_t} r_t^I(\mathcal{W}(y_t \mid \tilde{s}_{1:t-1};\theta_w))\right] \\ &= \mathbb{E}_{\tilde{s}_{1:t-1}\sim G, y_t\sim\mathcal{W}(y_t|\tilde{s}_{1:t-1})}\left[r_t^I \nabla_{\theta_W}\log(\mathcal{W}(y_t \mid \tilde{s}_{1:t-1};\theta_w))\right].\end{aligned}$$

3.5.3.4 Discriminator Training

When training the discriminator, a batch of labelled samples $\tilde{\boldsymbol{s}}$ are generated by the generator according to its current policy, which is then paired with a batch of labelled samples from the ground truth data set. The sequences are then used in order to compute the probabilities p_{real}, p_{fake} of the text being real or fake as described in Section 3.5.3.1, which are fed into (3.7) for loss computation. Weights are then updated using backpropagation. The pre-training procedure of the discriminator does not differ from the adversarial training process; the same loss function applies.

3.5.3.5 Generator Training

At the start of the process of training the generator, a batch of samples $\tilde{\boldsymbol{s}} = [\tilde{s}^1, \dots, \tilde{s}^n, \dots, \tilde{s}^N]$ are generated according to the generator policy. With these samples, the generator policy is evaluated using rollout as described in Section 3.4.4, with a step size $c = 4$ on a total sequence length $T = 32$. Thus, in each rollout iteration $k = 1, \dots, \frac{T}{c}$, the subsequence $\tilde{s}^n_{1:kc}$ is copied from each sample $\tilde{s}^n$ and succeeded by a rolled-out subsequence $\tilde{s}_{kc+1:T} = [y^{RO}_{kc+1}, \dots, y^{RO}_T]$ that are generated with the generator policy. The result is a batch of partly rolled out samples $\tilde{\boldsymbol{s}}_k$ where each sample $\tilde{s}^n_k = [y_1, \dots, y_{kc}, y^{RO}_{kc+1}, \dots, y^{RO}_T]_k$, $n = 1, \dots, N$

The sequences generated by the rollout procedure are passed on to the discriminator, which returns probabilities $(\boldsymbol{p}_{real})_k = [p^1_{real}, \dots, p^N_{real}]_k$ of the samples being real sequences. These probabilities are used as a basis for the reward of the generator.

In order to prevent vanishing gradient due to the reward becoming too small, the rewards $\boldsymbol{r}_k$ for each rollout iteration are calculated by re-scaling the probabilities using *bootstrapped rescaled activation*, which was proposed by Zhang et al. [7]. It is a computationally efficient method which involves ranking the partly rolled out samples $\tilde{s}^n_k$, $k = 1, \dots, \frac{T}{c}$ originating from sample $\tilde{s}^n$ such that the partly rolled out sample that correspond to the highest probability corresponds to a rank of 1, the second highest probability to 2, etc. up to the lowest probability which has a rank of N.

The rewards are then calculated as

$$r^i_t = \sigma\left(\delta \cdot \left(\frac{1}{2} - \frac{\text{rank}(\tilde{s}^n_k)}{N}\right)\right),$$

where σ is the sigmoid function, $\delta = 16.0$ a scale factor introduced by Guo et al. [7] and $\text{rank}(\tilde{s}_k^n)$ denotes the high to low ranking of the samples as described above. r_t^i are used in (3.9) in order to compute the goal loss.

During pre-training, features $\hat{f}$ are extracted from a real input sequence s. The goal loss function used in pre-training is defined by setting $r_t^i = 1$ for all the rewards, since the real input sequences are assumed to obtain a perfect score from the discriminator. For one sample, the pre-train loss is defined as

$$\mathcal{L}_{goal,pre}(\theta_m) = -\frac{1}{c}\sum_{k=1}^{c} d_{cos}(\hat{f}_{t+c} - \hat{f}_t, g(\theta_{\mathcal{M}})). \tag{3.13}$$

Since the input text during pre-training is real text, the cosine similarity between input features and real features will equal 1. Thus, the intrinsic worker reward as defined in (3.12) will be set to 1, and the pre-training worker loss function reduces to

$$\mathcal{L}_{worker,pre}(\theta_w) = \mathbb{E}_{s_{t-1}\sim G}\left[\sum_{y_t} \log(\mathcal{W}(y_t \mid s_{1:t-1}; \theta_w))\right]. \tag{3.14}$$

Once complete sequences have been generated, the weights $\theta_{\mathcal{M}}$ and $\theta_{\mathcal{W}}$ can be updated using backpropagation.

In order to prevent mode collapse during the adversarial training, the LeakGAN model introduces one epoch of *interleaved training* every 15th epoch, which is equivalent to one epoch of pre-training with the maximum likelihood approach. The interleaved training also prevent the model from getting stuck in a bad local minimum, and not diverging too much from the sequences generated by pre-training.

3.6 Evaluation Methods

An important aspect of 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. Both metrics are used for text evaluation in this thesis and are 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 5.1.2.1. If the special tokens happen to appear in the middle of a sequence, though, they are during evaluation.

3.6.1 Bilingual Evaluation Understudy (BLEU)

In the domain of Machine Translation, a widely used metric is *Bilingual Evaluation Understudy* (BLEU) [29], 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. [29], 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.

3.6.2 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

The metric *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) introduced by Lin [30] is commonly used for evaluation of text summarisations and is used by, among others, [31] and [32]. As for the BLEU metric, 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 [30]: picture the reference sequence $s = [ABCDEFG]$ and the two generated candidates $\tilde{s}_1 = [ABCDHIJ]$ and $\tilde{s}_2 = [AHBICJD]$ 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, and are the ones that are used in this thesis.

Even though BLEU and ROUGE are the de facto standards in evaluating machine generated texts, there are some flaws to both metrics. Since both BLEU and ROUGE measure overlap of tokens between reference and the generated sequence, none of the introduced metrics take synonyms into account. It is therefore possible to have a generated sentence that captures the semantic meaning of the reference but with different words, and thus being judged with poor BLEU/ROUGE scores.

4

Model – SumGAN

This chapter describes the model that is the outcome of the thesis. The model is given the name SumGAN since it uses the GAN principle with the intention to summarise documents.

Prior to developing SumGAN, text generation experiments were performed using SeqGAN [5], TextGAN [6] and LeakGAN [7] with parts of Webis-TLDR-17 as training data. The generated texts from these experiments only had to be evaluated by ocular inspection in order to select LeakGAN as the winning model for text generation with GAN. This result is also further supported by the results of the paper introducing the Texygen benchmark for models that generate texts [11], in which LeakGAN outperforms the other text generation models using GAN.

With these experiments as a starting point, LeakGAN was chosen as the foundation of SumGAN, which was then extended with the intention to do enable summarisation. The result is illustrated in Figure 4.1 and is described in detail in the forthcoming sections in this chapter.

4.1 Encoder

In order to obtain an informative representation of the documents, an encoder network which encodes each document is used. The encoder consists of a bidirectional LSTM that produces an encoding of the document by letting one LSTM network process the document from start to end, and a second LSTM network instead processes the document in the reverse order. Given a document, the encoder outputs state vectors $c_{doc} = [c_{doc}^0, \ldots, c_{doc}^T]$ along with hidden state vectors $h_{doc} = [h_{doc}^0, \ldots, h_{doc}^T]$ (see Section 3.4.2).

As described in Section 3.4.3, the hidden states h_{doc} contain the information of the document and are intended to tell the generator what to generate. In most applications, only the last of the hidden states, h_T, is used. The remaining states, $h_{1,\ldots,T-1}$ are only used for the attention mechanism, which was described in Section 3.6

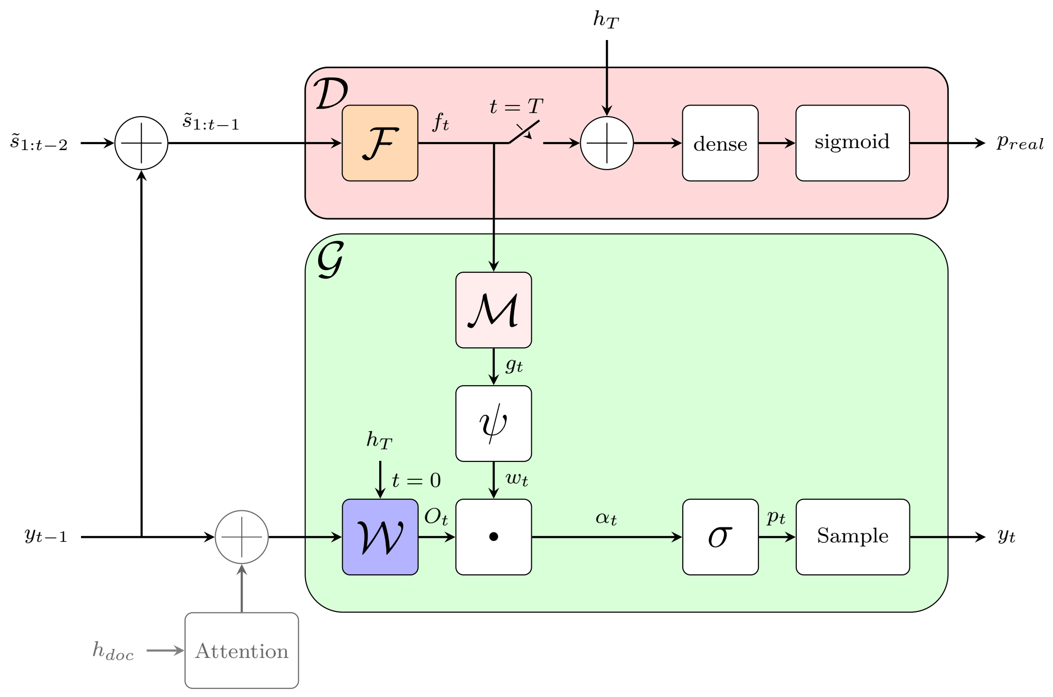


Figure 4.1: A schematic overview of SumGAN. Note the three differences compared to LeakGAN: i) In the discriminator, the final hidden state h_T of an encoded document is concatenated with the extracted features f_t. ii) h_T is used as the initial hidden state of the worker $\mathcal{W}$ at $t = 0$. iii) All hidden states of the encoded document h_{doc} are used as input to an attention mechanism, whose output vectors are concatenated with the generated token y_{t-1} from the previous time step. The reason why the attention mechanism is faded is that experiments were performed both with and without it.

4.2 Discriminator

Our implementation of the discriminator follows the structure of the one used in LeakGAN, see Section 3.5.3.1, with the difference that instead of just taking the sequence features f from the feature extractor as input, the dense layer takes the concatenated input $f \oplus h_T$ where h_T is the final hidden state of the document provided by the encoder. The discriminator classifies $\tilde{s}$ by examining p_{real} and p_{fake}; if $p_{real} > p_{fake}$, then $\tilde{s}$ is classified as a real sequence, otherwise as a synthetic one.

4.2.1 Adding Falsely Paired Data

Since the original version of GAN [4] only proposes feeding real samples along with generated ones, the discriminator only has to distinguish real text from fake in order to get the right answer, i.e. recognising the connection between a document and a summary is not considered using this training approach. An attempt to force the discriminator to check whether a summary is relevant to a given document is therefore done by feeding document-summary pairs to the discriminator in which the summaries are drawn from the real world dataset, but paired with documents other than the ones they summarise. In this way, the connections between the documents and the summaries are removed. In addition, generated summaries are paired with irrelevant documents. These document-summary pairs are labelled as “false” summaries and are fed to the discriminator for training, along with the original “true” and “false” Doc-Sum pairs.

4.3 Generator

The generator has two different processes for generating text, depending on whether the model is in pre-training or adversarial training. Both of these training processes initialise the worker hidden state $h_0^{\mathcal{W}}$ with the encoded document h_T as shown in Figure 4.1.

4.3.1 Pre-training

The purpose of the pre-training is to make the generator learn a policy for generating human like text via maximum likelihood training. This is done with rollout as described in Section 3.5.3.5 where a real sequence $s_{1:T}$ is used as input.

4.3.2 Adversarial Training

Once the pre-training is done, the adversarial training starts. It begins by using the pre-trained policy in order to generate a full sequence $\tilde{s}_{1:T}$ given an encoded document which has initialised the hidden states of the worker. The generated sequence is then used in the rollout procedure described in Section 3.5.2 for evaluating the policy from which $\tilde{s}_{1:T}$ is generated, and obtaining the rewards $[r_1, \ldots, r_{\lfloor \frac{T}{c} \rfloor}]$ needed for computing the goal loss as in (3.9).

A complete schematic illustration of the adversarial training process can be viewed in Figure 4.2 which shows the high level interactions of the different components of the generator.

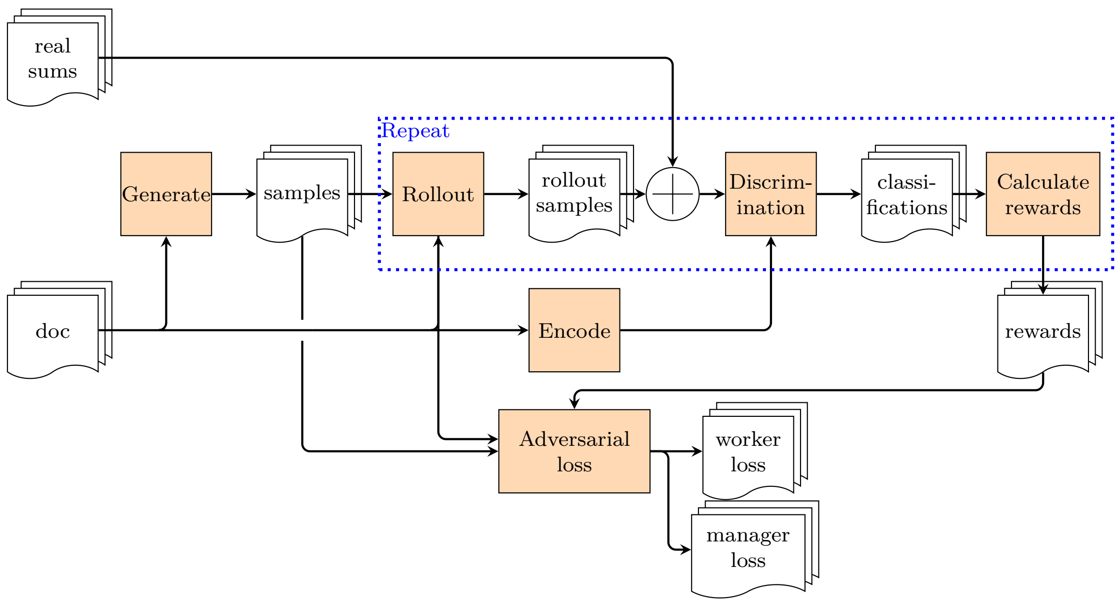


Figure 4.2: The adversarial training process for the generator.

4.3.3 Extending Model with Attention Mechanism

With the purpose of making the generator recognise important parts of the document to a greater extent, SumGAN is extended with an attention mechanism, as described in Section 3.4.3 and illustrated in Figure 3.7.

Since attention is a computationally demanding operation, the model was trained both with and without the attention mechanism and hence, the attention mechanism in Figure 4.1 is greyed out.

4.4 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* [33]. In the survey, which was completed by 101,592 developers worldwide, Python ranked as the seventh most used programming language and was described as *"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 papers on which the thesis work is based, used Tensorflow. Of these, some even published their code online [5, 6, 7], which made for a good opportunity of code re-use. Second, Tensorflow is one of the most used frameworks overall in the developer community [33], which means that the knowledge gained during the work would be valuable in the future for the authors of this report.

One issue with code development across different computers is dependency handling. A system which runs on one machine may not run on another machine with different hardware and/or software specifications. A tool which is commonly used to circumvent this issue is the virtualization software *Docker*, which lets independent "containers" bundle with an operating system to run on virtual machines. When building such a Docker container, specific versions of Python, Tensorflow etc. can be specified explicitly by the developer who thereby can be sure that the code will run regardless of the software configuration of the machine itself.

In order to evaluate the quality of the text as described in Section 3.6, the Python package `pyrouge` using the original Perl script `ROUGE-1.5.5.pl` was used for computing ROUGE scores. To compute BLEU scores for the generated summaries, the built in BLEU method `nltk.translate.bleu_score` from the Python package NLTK 3.2.5 was used [34].

As a service to the reader who wishes to re-produce the results presented in this report, important hardware and software specifications are listed in Table 4.1. Worth noting is that the Python binaries that are used are downloaded by Docker from Google Container Registry (GCR). This so called *Docker image* includes Python modules such as Tensorflow, which means they do not need to be manually configured.

Because of the many matrix operations, machine learning in general and deep machine learning in particular 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 4.1 along with the operating system of the computer.

Table 4.1: Hardware and software specifications of the machine which was used for training of the model.

Part	**Specification**
Operating system	Ubuntu 16.04.4 LTS
GPU	NVIDIA GeForce GTX 1080 Ti 12GB
CPU	Intel Core i7-7700 CPU @ 3.60GHz
Memory	16GB
Cuda	7.5.17
Nvidia-Docker	2.0.3
Tensorflow docker image	gcr.io/Tensorflow/Tensorflow:1.4.0-gpu-py3
ROUGE	1.5.5
NLTK	3.2.5

5
Data

In order to obtain successful results with deep learning, a large set of training data is needed. This chapter first describes the pre-processing steps taken in order to prepare data for training and then presents two datasets that have been used for experiments.

5.1 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.

5.1.1 Tokenisation

The first step in the pre-processing process is tokenisation, which was introduced in Section 3.1.1. Tokenisation can be performed in different ways, so choosing a good tokenisation procedure is important to the outcome of a model.

A naïve tokenisation approach is to split a sequence on spaces and newlines. This approach is simple to grasp, but limited in that it does not handle punctuation, contractions and capital words well. To illustrate this, consider the sequence: *You're not worried, are you?* and the tokenised version in Table 5.1. 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 naïve approach with lower-casing is also presented in Table 5.1.

With the naïve approach combined with lower-casing, the contracted word "you're" and the final word with punctuation "you?" are treated as individual tokens. This is a problem since the common word "you" has been separated into several tokens. A more sophisticated approach is to use the Natural Language Toolkit (NLTK) [34], which is a Python toolkit for various NLP tasks. There are many options for tokenisation, but NLTK was chosen since it is widely used. According to the documentation of NLTK[1], the tokeniser performs the following steps:

- split standard contractions, e.g. don't → do n't and they'll → they 'll
- treat most punctuation characters as separate tokens
- split off commas and single quotes, when followed by whitespace
- separate periods that appear at the end of line.

With the NLTK approach, "You're" and "you?" are properly split, as shown in Table 5.1. Still, though, the capitalisation results in the two words "You" and "you" being separated. In order to cope with this, an extension to the NLTK approach, introduced by Helmertz and Hasselqvist [8], is used. The augmented NLTK approach lower cases all words and handles certain edge cases such as "'90s" appropriately. The resulting tokenised version of the example sequence is shown in the final row of Table 5.1.

Table 5.1: Results of various tokenisation approaches on the sequence *You're not worried, are you?*.

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", "?"]

It is worth mentioning that the approach to lower case all tokens brings the issue that some tokens that should be separate are treated as the same. One example is *"New York"* which, when lower cased *"new york"*, could mean something different. The reason why lower case is still used is that there are more cases where lower case is justified than where it introduces ambiguity.

5.1.2 Vocabulary Selection

The text data that is fed to, processed by, and produced by SumGAN 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 [12].

As described in Section 4.3, 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 vocabulary $|V|$. A large vocabulary results in a large output dimension of the final dense layer of the generator, which

[1]http://www.nltk.org/api/nltk.tokenize.html

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.

In the case of SeqGAN and LeakGAN, the vocabulary size $|V|_{decoder}$ for the decoder is 4,839 and 5,000, respectively, which means that a decoder vocabulary size of approximately 5,000 tokens is reasonable.

Since neither SeqGAN nor LeakGAN use an encoder, inspiration for the size of the encoder, $|V|_{encoder}$, is instead retrieved from Hasselqvist and Helmertz [8] who use 173,256 tokens in the encoder vocabulary. For the datasets *News Summary* and *TLDR-Submission-Relationship* described in Section 5.2.4 below, the sizes of the encoder vocabularies were 23,032 and 92,998, respectively. The reason why the vocabulary of the *News Summary* dataset is much smaller than the other is that the dataset contains a lot fewer samples than the other.

The process of extracting a vocabulary from a corpus is straightforward: first all sequences in the corpus are tokenised according to the description in Section 5.1.1. 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.

Because of this procedure, if the total number of unique tokens in the corpus $|V|$ is less than the predefined $|V|_{encoder}$, $|V|_{encoder} = |V|_{decoder} = |V|$.

5.1.2.1 Special Tokens

As described in Section 4.3, 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 [12], 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.

Also described in Section 4.3 is that the generation process starts with a predefined token each time. In the original paper on LeakGAN by Guo et al. [7], each generated sequence starts with the token “A”, which is a reasonable choice since “A” is one of the most common tokens in the English language. It does, however, imply that all sequences start with “A”, which severely limits the kind of grammatically correct sequences that can be generated. Thus, in this thesis work 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 documents and 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.

An example of the usage of EOS, SOS and UNK is shown in Table 5.2. In the sample summary *"mr abramovich recently aquired chelsea ."*, the surname *abramovich* is uncommon, so it is not included in the vocabulary and is hence replaced with UNK. The pre-processed summary sequence also starts and ends with SOS and EOS, respectively. The table also shows that the EOS token is followed by a number of PAD tokens, which are described below.

Table 5.2: Example of a sequence with and without special tokens.

Original sequence	"mr abramovich recently aquired chelsea."
With special tokens	"SOS mr UNK recently aquired chelsea . EOS PAD . . . PAD"

5.1.3 Padding and Bucketing

As indicated above, pre-processed summaries end with an EOS token followed by a number of PAD tokens. The reason is that in the model presented in Chapter 4, 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 [12], 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 $L \in \{N, N+1, \ldots, N+B\}$ 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 [12], bucketing reduces computation time significantly during the training phase, which is the reason why it is used in this thesis.

5.2 Data Set

In the field of text generation, a few datasets are commonly used for model evaluation. Nallapati et al. [2] and Helmertz and Hasselqvist [8] used a dataset consisting of news articles and summaries from the news publishers CNN and Daily Mail[2]. Rush et al. [3] used the DUC-2004[3] and Gigaword[4] datasets. After deeper investigations, neither of these datasets were deemed suitable for the model developed in this thesis.

The CNN/Daily Mail dataset consists of documents and summaries with average token counts of approximately 773 and 14, respectively [8]. Although the dataset consists of an adequate amount of data (approximately 1 million document-summary pairs), training the proposed model with this data set would be too computationally heavy in regard to the resources available for this thesis,

The DUC-2004 dataset, along with the similar DUC-2005, DUC-2006, and DUC-2007 datasets, all published by the *Document Understanding Conferences*, consists of only 500 document-summary pairs, and is cumbersome to come by. Therefore it was not used.

Gigaword consists of almost ten million documents from news publishers such as Associated Press, Los Angeles Times, and New York Times. Of these documents, a reasonably sized subset of document-summary pairs with adequate lengths could have been extracted. Unfortunately, though, the licensing fee of the dataset was $3,000, which meant that it could not be accessed.

Instead of the above data sets, two lesser known corpora were used. They are both easy to access, free to use and have characteristics that are suitable to SumGAN. In order to gain insight into them, in-depth reviews are presented below.

[2]https://cs.nyu.edu/ kcho/DMQA/
[3]https://duc.nist.gov
[4]https://catalog.ldc.upenn.edu/LDC2011T07

5.2.1 Webis-TLDR-17

An interesting dataset is *Webis-TLDR-17* [35], produced by Völske et al. [36], which is constructed of data from the World's largest online community, *Reddit* [5]. The dataset utilises the common practise of Reddit's users to submit a brief summary, a so called *TL;DR* (*too long; didn't read*), along with long posts.

Reddit is structured such that users can create *submissions*, to which other users can post *comments*. Since TL;DR:s are posted both in submissions and in comments, the dataset can be divided into *submission type* samples and *comment type* samples. The distinction is natural since, in general, users who create submissions want to convey some information to other users, whereas users who comment on other user's submissions and comments want to convey an opinion. Thus, in the forthcoming analysis of the dataset, the submission and comment style samples are treated as two separate data sets which are denoted *TLDR-Comment* and *TLDR-Submission*.

Table 5.3 shows that the $3,848,194$ samples in the entire Webis-TLDR-17 dataset are divided somewhat equally between TLDR-Comment and TLDR-Submission, but that there are more samples in TLDR-Comment than TLDR-Submission. The table also shows the number of unique topics, so called *subreddits*[6] that the dataset consists of. Even though both TLDR-Comment and TLDR-Submission are spread over thousands of subreddits, both are concentrated to a small number of subreddits, as indicated by Figure 5.1 and 5.2. For both sets, about one third of the samples originate in ten subreddits, of which one is dominant.

It is interesting to note that the most common subreddit in TLDR-Submission, `r/relationships`, is not even one of the ten most common subreddits in TLDR-Comment. Conversely, `r/AskReddit`, which is the by far most common subreddit in TLDR-Comment holds substantially fewer samples in TLDR-Submission. This signal a difference in how users behave, depending on the subreddit in which they operate.

Table 5.3: The Webis-TLDR-17 dataset in numbers, divided into submissions and comments.

Dataset	Number of samples	Number of subreddits
TLDR-Submissions	1,762,893	24,377
TLDR-Comments	2,085,301	16,929
Webis-TLDR-17	**3,848,194**	**29,650**

[5]https://www.reddit.com

[6]By convention, subreddits are typeset as `r/SomeSubreddit` in this report.

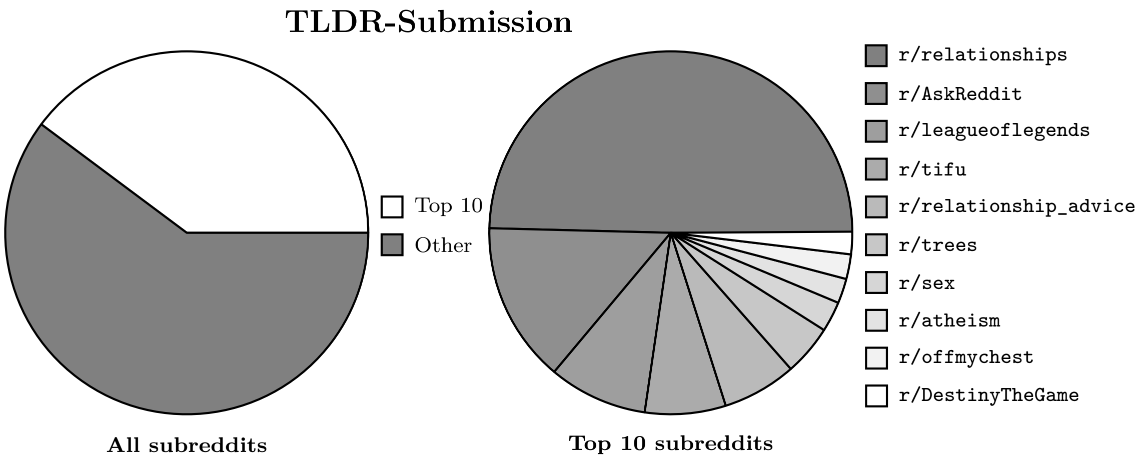


Figure 5.1: The distribution of submission type over subreddits in the Webis-TLDR-17 dataset. The left chart shows that the ten subreddits with the largest amount of documents in the dataset, comprise approximately one third of the entire dataset. The right chart shows the distribution between the ten largest subreddits. Apparently, a large amount of the dataset is isolated to a few subreddits.

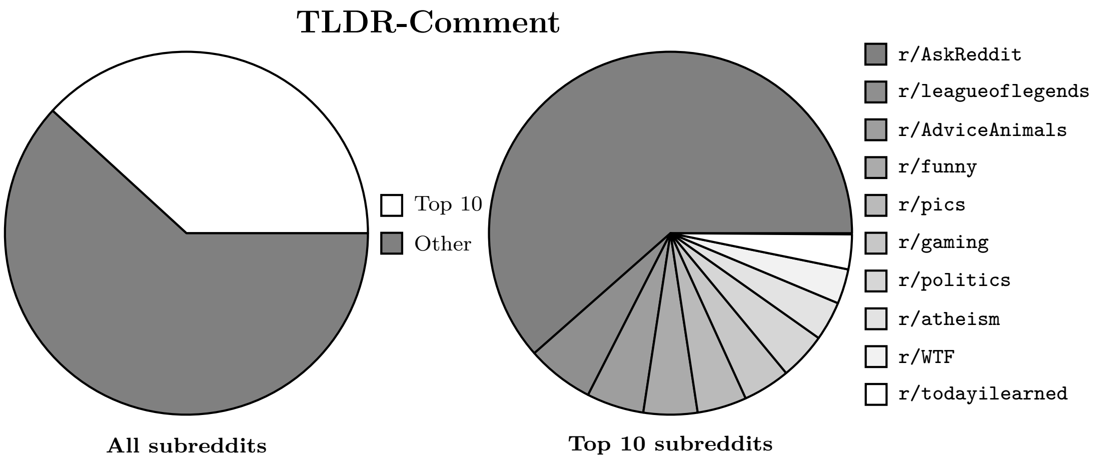


Figure 5.2: The distribution of comment type samples over subreddits in the Webis-TLDR-17 dataset. The left chart shows that the ten subreddits with the largest amount of documents in the dataset, comprise approximately one third of the entire dataset. The right chart shows the distribution between the ten largest subreddits.

Doc-Sum 5.1 and Doc-Sum 5.2 show one example from TLDR-Submission and TLDR-Comment, respectively. The samples were chosen at random and stem from the subreddits `r/loseit` and `r/WildStar`. By reading the documents and summaries in the examples, traces of the difference in user behaviour can be discerned. In Doc-Sum 5.1, the summary written by the user is a fairly good representation of the document. In Doc-Sum 5.2, on the other hand, the user does not use the

TL;DR convention to summarise the document, but rather to provide some advice to another user.

Document
Hi r/loseit! I only recently found this community, but I am so glad I did. I'm a 19 year old female that is 5' 8" tall and has gone from 200 pounds to 185 in about 4 months. I was so extremely excited with this progress when I was initially saw it through the scale measurements. Now, 4 months in, however, I'm having a hard time staying motivated. I know that I have lost 15 pounds, but I can't really see a huge difference in my appearance. I can however see an improvement in my ability to remain active and my overall mood. Is this normal? What are some things I could do to help myself see my own progress?
Thanks!

Summary
I have lost 15 pounds but still feel that I look the same as I did before I started my weight loss journey. This is making it hard to stay motivated.

DocSum 5.1: A sample submission from the Webis-TLDR-17 dataset [35]. The sample is from the subreddit `r/loseit`.

Document
Dude, the compression of that trailer is showing, and it's pretty terrible. The effects are overdone in such a way that's it a clusterfuck of patterns, colors and shapes and I can't even tell what's going on in this image anymore. The white neon tubes coiling around her fingers don't make any sense, and I don't think you even used a proper blending mode for those. The summoning circle-esque brushes in the background don't the image any justice either.

Summary
it's not good, or even remotely decent. P.S. less is more, unless you can justify it. In this case, you can't.

DocSum 5.2: A sample comment from the Webis-TLDR-17 dataset [35]. The sample is from the subreddit `r/WildStar`.

In order to gain a deeper insight into the nature of the texts in the datasets, the distribution of tokens in the summaries and documents of a subset of the entire dataset is illustrated in Figure 5.3. The histograms in the figure show that the majority of the summaries and documents have lengths in the range of 0-40 and 0-400 tokens, respectively. Also, the histograms for both documents and summaries have long tails of infrequent document/summary lengths.

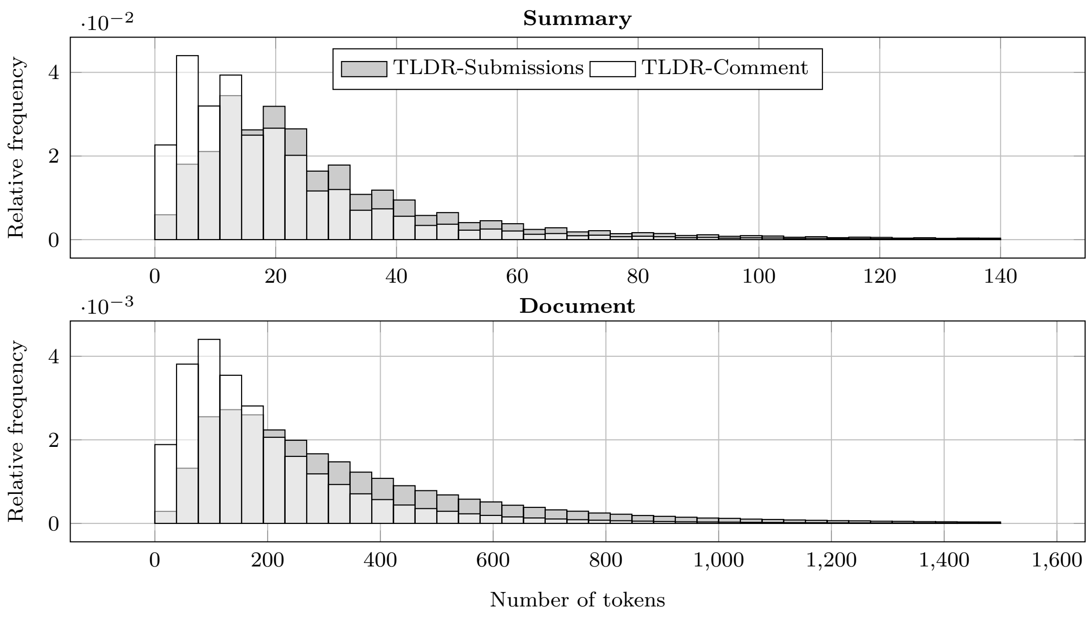


Figure 5.3: Number of tokens in the summaries and documents in the Webis-TLDR-17 dataset.

Because of the skewed distribution of samples over subreddits and the user's varying behaviour depending on the subreddit, we cannot be select random subreddits from the dataset if a large, coherent corpus is to be obtained. If, for instance, a corpus of scientific nature is desired, none of the top ten largest subreddits would be an appropriate choice.

5.2.2 News Summary

Another interesting dataset is the *News Summary dataset* [37] published on the website Kaggle[7] by K. Vonteru.

The News Summary dataset consists of 4,513 news articles that have been scraped from the newspapers *The Hindu*, *Indian Times* and *The Guardian*. As the pie chart in Figure 5.4 reveals, the news articles are not evenly distributed between the three sources.

[7]https://www.kaggle.com/sunnysai12345/news-summary

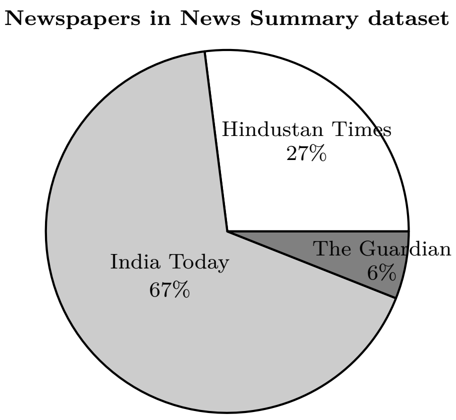


Figure 5.4: The distribution of news sources in the News Summary dataset.

Each of the samples in the dataset contain the following fields:

- The name of the author
- An url to online version of the article
- The headline of the article
- A shortened version of the article
- The complete article

Of these, the headline, the shortened version, and the full article can be used for document summarisation in three different ways:

1. Treat the headline as a summary of the shortened version
2. Treat the headline as a summary of the complete article
3. Treat the shortened version as a summary of the complete article

To aid in deciding which approach is most suitable for SumGAN, the number of tokens in each headline, shortened article and full article are presented in histograms in Figure 5.5. The histograms reveal that the full articles has the same characteristics as the summaries and documents in the Webis-TLDR-17 dataset, as shown in Figure 5.3; the distribution is asymmetric and has a long tail. The headlines and shortened versions, however, are normally distributed, which is emphasised by Gaussian curves with the same mean values and variances as the token counts.

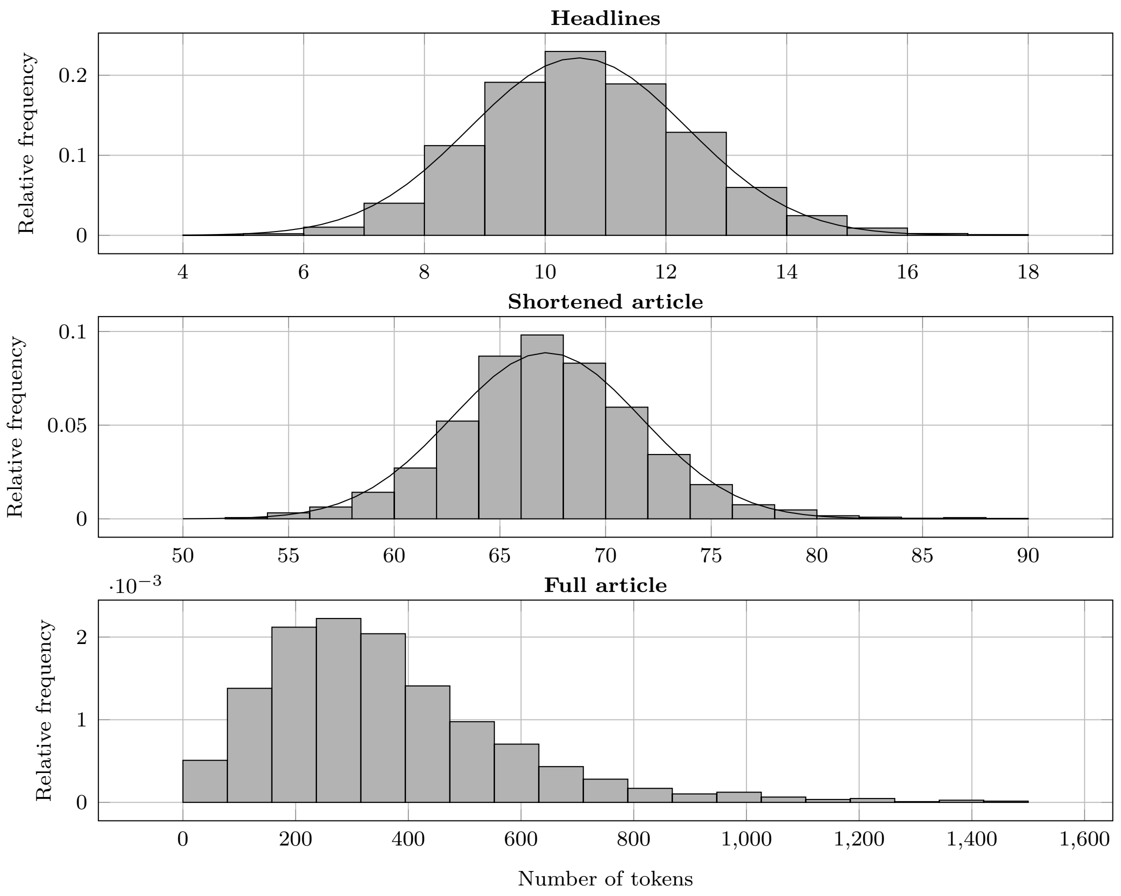


Figure 5.5: Number of tokens in the headlines, shortened articles and full articles in the News Summary dataset. The distributions of the headlines and the shortened articles are accompanied by Gaussian curves with the same mean and variance as the distributions. Evidently, the distributions agree well with the Gaussian curves.

An example from the dataset is shown in Doc-Sum 5.3. Compared to the sample comments and submissions from TLDR-Submission and TLDR-Comment, Doc-Sum 5.1 and Doc-Sum 5.2, the style of the language of Doc-Sum 5.3 is clearly different; the author writes in third person and has no jargon. It is of course not possible to represent an entire dataset with one sample, but a manual review of the News Summary dataset suggests that the presented sample represents the entire dataset well.

5.2.3 Comparison of the Data Sets

The three data sets presented above are apparently different. In order to gain a better insight into the differences, a more in-depth analysis of some aspects is presented below.

Document
A 60-year old Dalit woman was allegedly lynched in Agra after villagers thought that she was behind the recent cases of chopping hair of sleeping women. The family members of the woman who left home around 4 am on Wednesday said, "She pleaded that she had lost her way but they took her as one responsible for chopping women hair."

Summary
60-yr-old lynched over rumours she was cutting people's hair

DocSum 5.3: A sample document-summary pair from the News Summary dataset [37].

5.2.3.1 Token Counts

The two datasets presented above are different in many ways. Webis-TLDR-17 has a larger amount of documents than News Summary, but the language of the documents is less coherent. In order to extract a coherent corpus from it, specific subreddits have to be selected. Because of the imbalance between number of samples in each subreddit, such a coherent corpus will have a lot fewer articles than the entire dataset.

The model used for document summarisation imposes an upper limit to the length of both the document and summary, as described in Chapter 4. Because of the asymmetric distribution of tokens in the summaries, a noticeable portion of the Doc-Sum pairs have to be discarded, which further reduces the size of the corpus from Webis-TLDR-17.

News Summary, on the other hand, contains headlines and shortened articles with normally distributed token counts, which means that fewer Doc-Sum pairs, if any, have to be discarded.

As discussed in Section 5.1.3, the model divides the documents into batches and pads each batch up to the length of the longest document in the batch, which is known as *bucketing*. The bucketing procedure is used in order to use less memory than if the documents of the entire dataset were padded up to the length of the longest document of the entire dataset, which decreases execution time and thus enables the network to train faster.

The benefit of bucketing decreases if the length of the documents has a large variance since that increases the probability for a batch to include a relatively long document and thus a lot of pads. This is an argument in favour of using the shortened articles in the News Summary dataset as documents since the lengths of its samples are gathered more closely around its mean value than both the full articles of News Summary and the documents of Webis-TLDR-17.

5.2.3.2 Summary Precision

Another important difference between the data sets is the connection between the document and the summary, i.e. how much of the document that is represented in the summary.

To illustrate the document-summary connection, two randomly selected samples from the News Summary dataset and from Webis-TLDR-17, Doc-Sum 5.4 and 5.5, respectively, are presented. In the two samples, non-stop word tokens are highlighted with bold fonts and those that appear both in the summary and in the document and numbered. Stop words are words that are common in a language and carry little or no valuable information [38]. Two examples in the English language are *"the"* and *"a"*.

In Doc-Sum 5.4, all of the important tokens in the summary appear in the document as well, which is in stark contrast to Doc-Sum 5.5 where only one of the important tokens in the summary appears also in the document.

To describe this in mathematical terms, the *precision* metric can be used. Precision P is defined as $P = \frac{tp}{tp+fp}$ where tp are true positives and fp are false positives [38]. For a Doc-Sum, the true positives are important tokens that occur in the summary and the document, and false positives are important tokens that occur in the summaries only. For Doc-Sum 5.4 $P = \frac{7}{7+0} = 1$ and for Doc-Sum 5.5 $P = \frac{1}{1+3} = 0.25$, which means that, according to the precision metric, Doc-Sum 5.4 has a larger connection between the summary and the document than Doc-Sum 5.5.

Document
Hotels^1 in Maharashtra will **train**^{3} their **staff**^{2} to **spot**^{4} **signs**^{5} of **sex**^{6} **trafficking**^{7}, including frequent requests for bed linen changes and 'Do not disturb' signs left on room doors for days. A mobile phone app called Rescue Me, which will allow staff to alert police of suspicious behaviour, will be developed. The initiative has been backed by the Maharashtra government.

Summary
Hotel^1 **staff**^{2} to get **training**^{3} to **spot**^{4} **signs**^{5} of **sex**^{6} **trafficking**^{7}

DocSum 5.4: A sample document-summary pair from the News Summary dataset [37].

Based on the two samples above, it seems like the two data sets have different levels of connection. In order to quantify the suspicion, the following method was developed:

1. Tokenize the document and the summary as described in Section 5.1.1.

2. Remove stop words and punctuation from the document and summary. As with the stop words, punctuation such as *"."* and *"!"* carry little or no information about the correlation, which is why they are also removed. For the implementation, a list of punctuation symbols was retrieved from Python's built in

Document
It's not **racism**[1], stop using words you do not understand.
Yes Islam will be the most dominant religion in Europe (because atheism is not a religion), and that has nothing to do with **racism**[1]. It actually already is the dominant religion in French youth in Paris.
Racism[1] is would be saying "due to their genetic inability to think by themselves, muslim are more religious than good white people". But staying the fact is not **racism**[1], you can even state the real origin (white European don't care about religion because they've seen all the bullshit it comes with, muslims fall in religion to get an identity and because their parents raised them that way) without it being racist.

Summary
go **check** "**racism**[1]" in a **dictionary please**

DocSum 5.5: A sample document-summary pair from the Webis-TLDR-17 dataset [35]. The sample is from the subreddit `r/politics`.

`string.punctuation`[8] and a list of stop words from NLTK's `nltk.corpus.stopwords`[9].

3. Apply *stemming* to the tokens in the document and summary. Stemming is the process of stripping a word to its base/stem form by removing ending characters, e.g. the stem of "running" is "run". Of the many existing stemming algorithms, Porter's algorithm [39] was chosen since it is the most common algorithm for stemming according to Manning et al. [38]. In the implementation, NLTK's `nltk.stem.porter`[10] was used.

4. Calculate the precision score, as described above, for the stemmed summary and document.

The method was applied to 1,000 randomly selected samples in the News Summary, TLDR-Comments, and TLDR-Submission data sets. As the results in Table 5.4 show, the News Summary dataset has substantially higher precision scores than any of the TLDR data sets, which consolidates the suspicion from above.

Table 5.4: Results from a comparison of the connection between summaries and documents in three different data sets. A higher value indicates a higher connection.

Dataset	Mean	Std
News Summary	0.80	0.16
TLDR-Comments	0.55	0.23
TLDR-Submission	0.42	0.27

Table 5.4 also shows that TLDR-Comments has a higher precision score than TLDR-Submission, which indicates that there is a difference in the two datasets. As suggested in Section 5.2.1, this might be because of a difference in behaviour of Reddit's users depending on whether they create a submission or comment on existing content.

[8]https://docs.python.org/3/library/string.html#format-string-syntax
[9]https://www.nltk.org/api/nltk.corpus.html
[10]http://www.nltk.org/api/nltk.stem.html

5.2.4 Selection of Data Sets for Experiments

SumGAN is a fairly complex model consisting of many different networks that together gives the model many degrees of freedom, wherefore training the network is time-consuming. Because of that, using the entire Webis-TLDR-17 dataset is not feasible with the experimental setup presented in Section 4.4.

From a deep learning perspective, the News Summary dataset is small, which means that training SumGAN with it is feasible, but which also implies a risk of overfitting during training. The analysis above shows that the dataset contains data with a high connection between the summary and the document, and that the documents and summaries in the dataset have appropriate lengths. Therefore, the entire News Summary dataset was used for training of SumGAN.

For a selected number of especially interesting settings, training with a relatively large amount of data was performed. To get a large amount of data, a subset of the Webis-TLDR-17 dataset was constructed. According to the analysis in Section 5.2.3.2 the TLDR-Submission dataset has a larger connection between the summary and the document than the TLDR-Comment dataset, which suggests that the training data is better suited for training of SumGAN. Therefore, only samples from TLDR-Submission were selected. As the pie chart in Figure 5.1 indicates, many of the samples in TLDR-Submission are concentrated to a few selected subreddits of which `r/relationships` is dominant. In order to use a set of samples with coherent content and language, it was decided to select samples belonging to one subreddit. Since `r/relationships` has the largest amount of samples (98,475), the second dataset was selected as all samples belonging to `r/relationships` in TLDR-Submission. Henceforth, in the report, the dataset is called *TLDR-Submission-Relationship*.

6
Experiments and Results

As shown in several independent papers, GAN can be used to generate text arbitrarily. The main goal of this thesis, however, is to examine the prospect of using GAN for generation of summaries to documents, which means that text should be generated in a *controlled* manner, not arbitrarily. The experiment outlined below was constructed and performed in order to examine the ability of SumGAN to generate relevant summaries.

6.1 Influence of Document on Summarisation

As described in Section 4.3, SumGAN is based on LeakGAN and improved with a few additions, which are intended to enable it to generate relevant summaries. The two main additions are: i) initialising the worker with an encoded document and ii) using an attention mechanism. To verify the impact of either of these, SumGAN was trained in three different settings:

1 No augmentation to the generator, i.e. with no connection between the document and the generator.

2 The worker initialised with an encoded document.

3 The worker initialised with an encoded document as well as an attention mechanism enabled.

In all three cases, the discriminator was augmented as described in Section 4.2, i.e. to not only use extracted features, but to also use an encoded document. Following LeakGAN, the discriminator was handed batches of data consisting of an equal number of real and synthetic summaries, along with corresponding documents.

The tests were performed on the TLDR-Submission-Relationship and News Summary datasets, both of which were described in detail in Section 5.2. The reason for performing tests on both data sets was to enable training with a large amount of data, numerous epochs, and with different text characteristics. Ideally, a model should be trained with as much data as possible and for many epochs. This is however not always feasible because of the long computation times involved in training. The difference in computation times is illustrated well by comparing the computation time for 20 epochs of pre-training with the two data sets and with identical hyperparameters: with the News Summary dataset, which consists of 4,020 training samples, 20 epochs took 1 hour and 23 minutes; with the TLDR-Submission-Relationship dataset, which consists of 93,552 samples, 20 epochs took 37 hours and 10 minutes on the computer specified in Section 4.4. Since some papers such as SeqGAN [5] have highlighted the importance of an adequate number of pre-training epochs, achieving enough pre-training with the larger dataset might be problematic. The News Summary dataset also has a different characteristic as discussed in Section 5.2; both the document and the summaries are on general shorter than the ones from TLDR-Submission-Relationship and the language is more coherent, which makes it a different task for the generator to solve.

During training, sample summaries were generated after every training epoch, both during pre-training and adversarial training. The generated samples were then used to calculate various BLEU-2 and ROUGE-2-F scores for each iteration. In this section, plotted values only for BLEU-2 and ROUGE-2-F are shown since they are commonly used metrics and displaying all plots in this chapter would consume a lot of space. The reader interested in other scores is referred to Appendix A.

6.2 Model Parameters

Table 6.1 displays the parameter values used in the model. Except for the document encoder dimension, the parameter settings are equivalent to the ones of LeakGAN [7].

Table 6.1: Parameter settings for SumGAN.

Parameter	Value
Minimum summary length	8
Maximum summary length	32
Embedding dimension	128
Worker hidden state dimension	128
Manager hidden state dimension	128
Document encoder hidden state dimension	64
Goal vector dimension	16
Feature extractor filter dimension	1, 2, 3, 4, 5, 6, 7 8, 9, 10, 15, 20, 32
Feature extractor number of filters	100, 200, 200, 200, 200, 100, 100, 100, 100, 100, 100, 160, 160, 160
Discriminator hidden dimension	256
Dropout probability	0.2
Batch size	64

6.3 Discrimination Target

In models with GAN, the role of the discriminator is to classify samples as belonging to one of two categories. In some applications, such as LeakGAN, SeqGAN and the experiments outlined above in Section 6.1, the two categories are *real* and *generated.*

An alternative way to categorise DocSum:s is as either *valid* or *invalid*, where in a valid DocSum the summary belongs to the document and in an invalid the summary has no relation to the document, i.e. the summary was not composed with the document in mind. The idea for choosing those two categories instead of real/generated is that if the discriminator is given the task of separating generated text from real, as in the original implementation of LeakGAN, the encoded document might be overlooked by the discriminator. The effect would be that the generator would only learn to generate real-looking text, not text influenced by a document.

To test the result of this alternative categorisation, SumGAN was trained in the same way as in Section 6.1, but with different data given to the discriminator. Instead of batches with real and synthetic samples, four different types of data were used, as shown in Table 6.2. Only real samples corresponding to its document were put in category 1, which corresponds to the desired result. The remaining three were put in category 0. This was implemented by gathering one batch from each of the four kinds and from the selected set of samples choose half a batch belonging to “1” and “0”, respectively.

Table 6.2: A description of the data that was handed to the discriminator in experiments with an alternative approach at providing data to the discriminator.

	Valid	Invalid
Real	1	0
Generated	0	0

6.4 Baseline

In order to evaluate SumGAN:s performance, various BLEU and ROUGE metrics are used. In order to place the values returned by the metrics in a context, it is common to use a baseline. The baseline can be chosen in various different ways.

For ROUGE and BLEU, which measure how well a text relates to another text by counting token overlaps two interesting baselines are to either generate a sequence consisting of only one token, or to generate sequences with randomly sampled tokens. The reason why those two are interesting is that they both pose plausible and undesirable behaviours of the generator. By comparing the values from the text generated by the manager with the baselines, the generated texts can be distinguished from the baseline behaviours.

In the plots of BLEU-2 and ROUGE-2-F below, the baseline of comparing to a sequence with only one token is included. Via manual inspection of real samples, it was found that summaries from the News Summary dataset contain a large ratio of UNK tokens. Therefore, it was decided to compare the generator's results with a sequence consisting only of UNK:s. Since the length of the sequence matter in both BLEU and ROUGE, the mock sequences for the News Summary and TLDR-Submission-Relationship were set to have lengths 11 and 20, respectively. Those specific lengths were chosen based on the analysis of token counts presented in Section 5.2, which showed that 11 and 20 are typical summary lengths for the two datasets. In the legends, the baseline is referred to as

6.5 News Summary

This section presents and discusses the results from the three experiments *1*, *2* and *3*, as described in Section 6.1 and with the news summary dataset. The training times for the experiments were between one and two days for each execution. The categories used were *real* and *generated*.

Numerical results from different metrics are presented and discussed first, followed by generated samples. When numerical results are presented, the break point between pre-training and adversarial training is marked with a distinct, red, vertical line at $t = 200$. The epochs where interleaved training takes place are highlighted with red, dashdotted, vertical lines. The baseline is presented as a solid, blue line.

6.5.1 Evaluation Metrics

Figure 6.1 shows the BLEU-2 and ROUGE-2-F scores for training and validation data and for experiments *1*, *2* and *3*. The plots show that the values resulting from BLEU-2 and ROUGE-2-F in this case are very similar for all three experiments as well as for training and validation data, respectively. This is disappointing since it means that the additions made to LeakGAN did not infer a noticeable improvement in performance. In addition, all three models are below the baseline for the validation data.

An interesting phenomenon is the distinct peak at the start of the adversarial training, which occurs for all three experiments on the training data. The experiments differ, however, in how long it takes for the performance to decay thereafter. Another interesting thing to note is that the scores start to oscillate in phase with the interleaved training, which manages to temporarily improve the performance on both training and validation data.

A comparison of the vertical scales reveals that the scores on the validation data never reach high levels compared to the training data. This indicates that the model

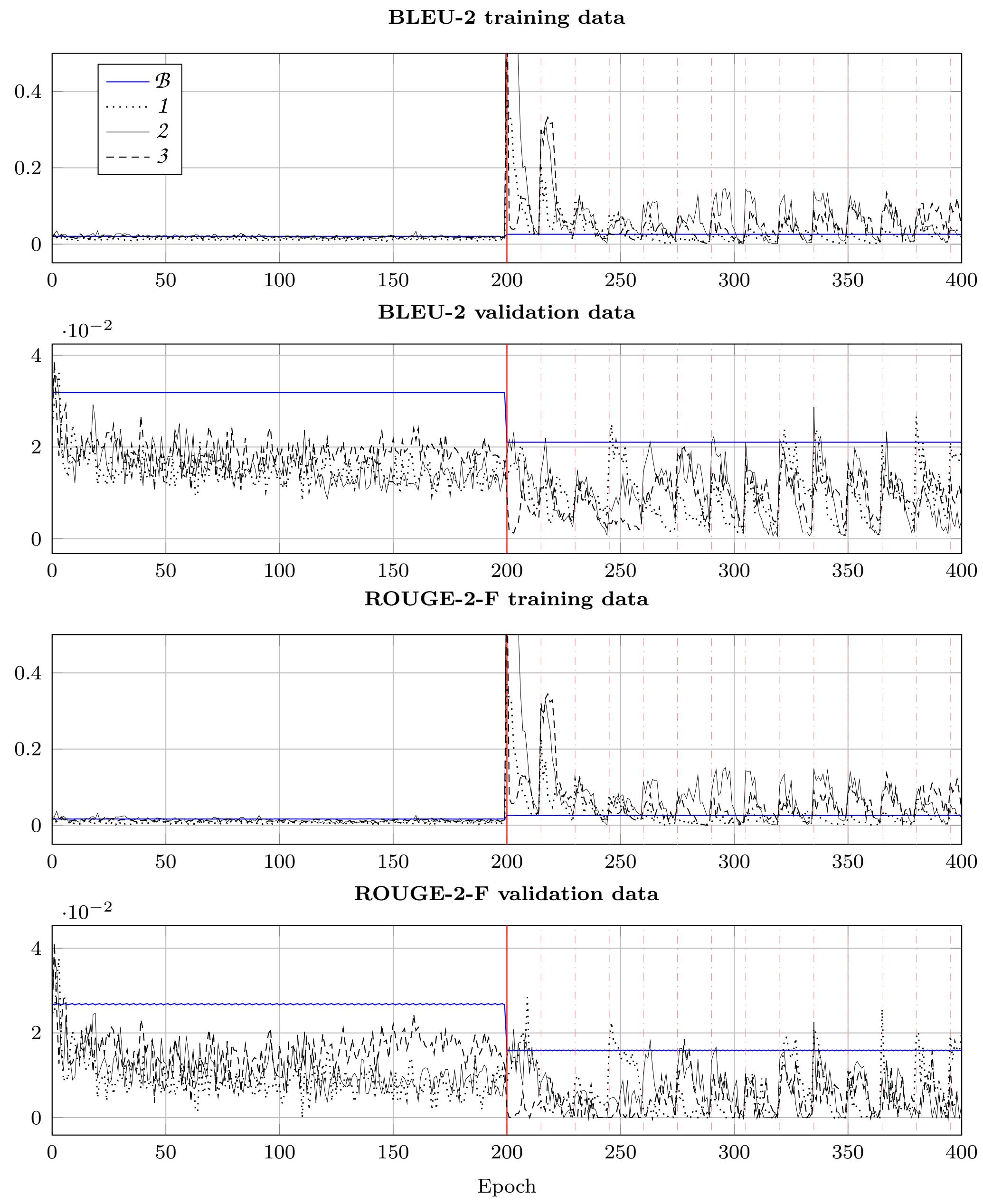


Figure 6.1: Selected BLEU and ROUGE scores for training and validation data in the experiments using the News Summary dataset and text classification as in LeakGAN.

never manages to generalise to the dataset.

6.5.2 Loss and Accuracy

The worker loss presented in Figure 6.2 show the loss functions used during pre-training and adversarial training, which is the reason why the characteristics is suddenly changed. The enlargement in the plot show that the worker manages well to minimise its loss function to the training data during pre-training as desired, but that the validation loss increases, which means that it overfits to the data.

During the adversarial training, the loss increases for both the training and validation data, but is brought back down by the interleaved training, which is shown clearly in the plots.

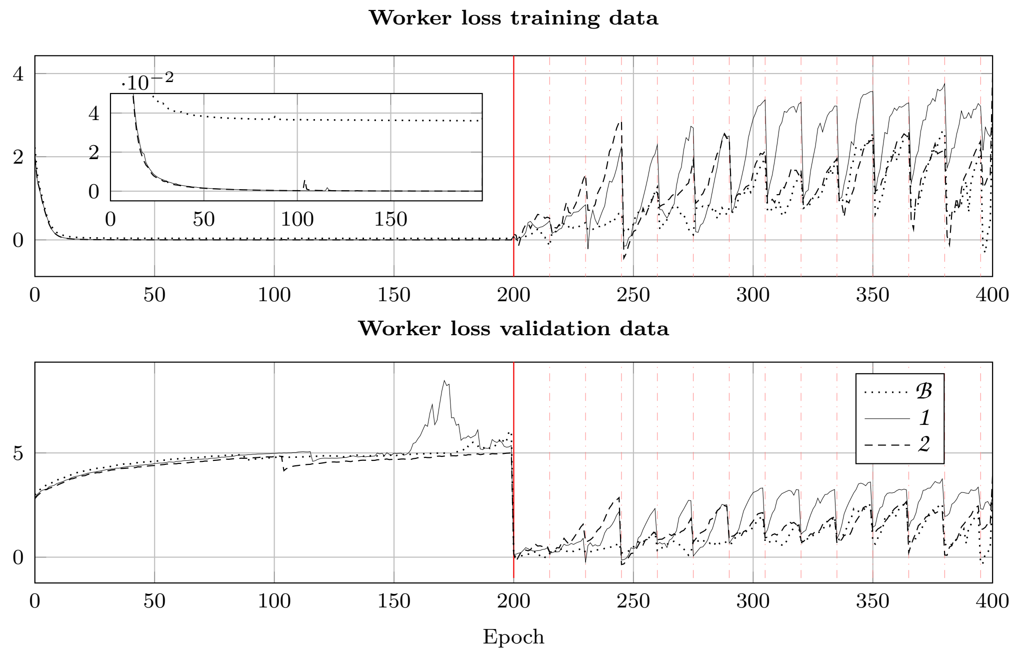


Figure 6.2: Worker loss for training and validation data in the experiments using the News Summary dataset and categorisation as in LeakGAN.

The manager loss is shown in the plots in Figure 6.3. During pre-training, the manager minimises its loss for the training data during the entire training, but starts to increase its loss for the validation data after around 50 epochs. This is a clear indication that the manager overfits. When adversarial training starts, the loss increases for all three models and at the end of the training, the loss is very high. Another interesting difference compared to the worker loss is that the interleaved training actually enlarges the loss, whereas the adversarial training brings it back towards zero.

For the training data, we see in Figure 6.2 and 6.3 that for the training data, both the worker loss and manager loss go to 0 during the pre-training stage, and when the adversarial training starts, the worker loss rises, manager loss decreases and both of them starts fluctuating. The fluctuating behaviour also occurs for the discriminator accuracy in Figure6.4, which periodically reaches values near 1, i.e. it is almost a perfect classifier.

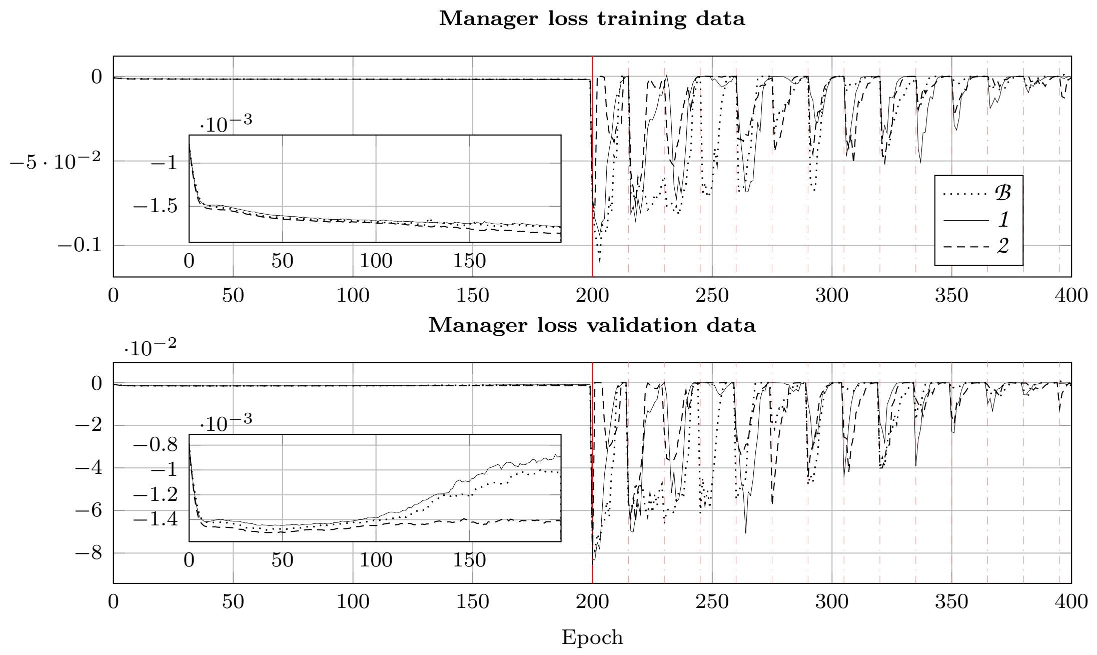


Figure 6.3: Manager loss for training and validation data in the experiments using the News Summary dataset and categorisation as in LeakGAN.

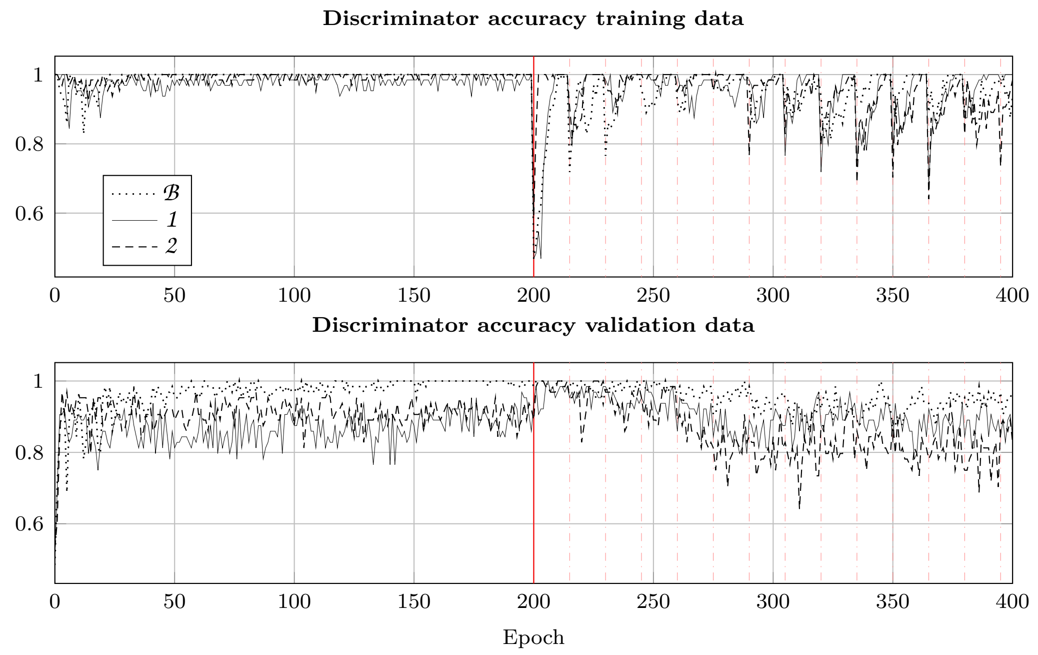


Figure 6.4: Discriminator accuracy for training and validation data in the experiments using the News Summary dataset and categorisation as in LeakGAN.

6.5.3 Text Quality

In this section, the quality of the generated summaries throughout the training process is showcased. Summaries generated for the documents in the training set are presented first, followed by summaries generated for documents in the validation set.

6.5.3.1 Training Data

When surveying the texts produced by *1*, *2*, and *3*, a pattern that explains the graphs in Figure 6.1 could be distinguished. In the report, samples from pre-training epochs 0 and 99, and adversarial epochs 0, 99 and 199 (corresponding to epochs 200, 300 and 400 in the figures) are presented since they portray the model's performance throughout the training process. As the plots in Figure 6.1 indicate, the texts produced by *1*, *2*, and *3* share the same characteristics. In addition, model *2* showed the most interesting results of the three for the TLDR-Submission-Relationships data. In the interest of keeping the report reasonably short, only samples from *2* are therefore shown.

At the start of the training process, at pre-training epoch 0, summaries such as DocSum 6.1 were generated. As the sample reveals, the texts are not coherent and the structure is not correct since two EOS tokens appear inside the sequence.

Document
maharashtra minister girish mahajan has asked protesting doctors to resume work by wednesday or risk losing six months ' worth of salary . the minister further promised to increase security guards in all government hospitals to provide security to doctors . notably , over 4,000 resident doctors across maharashtra have been on a mass leave protesting against dangerous working conditions .

Reference
resume work or lose 6 months ' salary : govt to maha doctors

Generated summaries
why elected of : with bjp into network _UNK _EOS _EOS airport

DocSum 6.1: Document, ground truth summary and generated summaries with the News Summary training data for pre-training epoch 0.

At epoch 99 of the pre-training, texts such as the one presented in DocSum 6.2 were generated. The structure is correct since there are no misplaced EOS or other tokens and the text has a reasonable meaning. The text has, however, no connection to the document, which is the reason why the BLEU-2 and ROUGE-2-F scores are low during the entire pre-training.

When the adversarial training starts, the characteristics of the text suddenly changes, as indicated both by the plots in Figure 6.1 and by DocSum 6.3. Many of the generated texts are exact copies of the summaries, which explains why the BLEU-2 and ROUGE-2-F scores are high.

As the adversarial training progresses, the quality of the text degrades substantially, as shown by DocSum 6.4 and 6.5. After 99 epochs, the text is no longer

Document
nawazuddin siddiqui 's first look from the upcoming short film ' carbon ' has been released . the film also stars jackky bhagnani and prachi desai and focuses on environmental issues like global warming and climate change . written and directed by maitrey bajpai , the film will feature nawazuddin in the role of a man from the planet mars .

Reference
nawazuddin 's first look from short film _UNK ' unveiled

Generated summaries
maharashtra _UNK ban on _UNK _UNK

DocSum 6.2: Document, ground truth summary and generated summaries with the News Summary training data for pre-training epoch 99.

Document
delhi 's department of environment is reportedly drawing a standard operating procedure to dispose the road dust collected by more than a dozen mechanical sweepers in order to control the air pollution level in the city . additionally , the public works department has floated a tender to procure six more sweeping machines this year .

Reference
delhi government to fix rules for road _UNK _UNK

Generated summaries
delhi government to fix rules for road _UNK _UNK

DocSum 6.3: Document, ground truth summary and generated summaries with the News Summary training data for adversarial epoch 0.

coherent, but of reasonable length. After 199 epochs, however, the model starts to generate coherent text but quickly changes to generate random samples instead.

Document
the up government on sunday approved a ? 47-crore package to ensure immediate availability of drinking water in the parched bundelkhand region . bundelkhand was continuously ignored in the last 15 years and no steps had been taken to ensure the holistic development of the region , said up cm yogi adityanath .

Reference
? _UNK package approved for drinking water in _UNK

Generated summaries
sand asks mandatory for ? _UNK postponed in greater noida

DocSum 6.4: Document, ground truth summary and generated summaries with the News Summary training data for adversarial epoch 99.

6.5.3.2 Validation Data

From DocSum 6.6, we can see some tendencies in how the text generation progresses over time. First of all, none of the generated summaries that are presented are either

Document
former sri lankan spinner muttiah muralitharan , who has the highest number of wickets in international cricket , has praised indian spinner ravichandran ashwin and called him a smart cricketer . muralitharan further said that he is looking forward to watching ashwin in action during the indian cricket team 's upcoming tour to sri lanka .

Reference
r ashwin is a smart cricketer : sri lanka 's _UNK

Generated summaries
pakistan awards _UNK ' to top military shares exit leader listed blames million formula mothers sindhu dubai chopped national associate fielding neither islamic related crimes sanon tenders children malaysia co-founder bhatt

DocSum 6.5: Document, ground truth summary and generated summaries with the News Summary training data for adversarial epoch 199.

relevant to the document, or resemble the content of the reference summary. The quality of the text varies in between epochs though, from not having a coherent sentence structure at pre-training epochs 0 and 99, to obtaining more grammatically correct sentences at adversarial epochs 0 and 99. At adversarial epoch 199, SumGAN has collapsed and only noise is generated.

The implication of this is that SumGAN learns to accurately copy texts from the training summaries, but does not learn to generalise for unknown summaries.

Document
the united states of america cricket association (usaca) is facing expulsion from the international cricket council (icc) , the sport 's global governing body revealed on monday . icc had suspended the board in 2015 as it claimed that the body did not have unity and did not hold the widespread authority over the country 's cricket activities .

Reference
usa cricket governing body faces _UNK from icc

Generated summaries

Pre-training epoch 0: control _UNK _UNK _UNK ' to 's , _EOS in

Pre-training epoch 99: _UNK _UNK , _UNK in pakistan _UNK cross 10 lankan security

Adversarial epoch 0: _UNK mumbai _UNK breeding grounds destroyed in 6 months

Adversarial epoch 99: iaf officer gets _UNK oscar-winning found hero tn posters

Adversarial epoch 199: mumbai company for aap first _UNK in india _EOS _PAD _PAD laptop collaboration expressway guest jurisdiction shetty democratic trip refused users advani wrestling slogans game services collecting forcing near student notably

DocSum 6.6: Document, ground truth summary and generated summaries with the News Summary validation data.

6.5.4 Model with Attention Mechanism

DocSum 6.7 presents a summary that is generated by model *3*. It is included to show that the language of the summary is not well-structured and has no connection to the document, when attention is used. This behaviour is similar to the one of model *2*, as described in the previous section. Hence, judging from the evaluation metrics along with the presented example, an attention mechanism does not seem to improve the model when trained on the News Summary dataset.

Document
the supreme court-appointed committee of administrators has said to the state associations that the indian cricket team 's withdrawal from the upcoming icc champions trophy is not in the best interests of indian cricket . coa head vinod rai on tuesday had said that bcci officials are not mandated to take any decision on india 's champions trophy participation without the coa 's approval .

Reference
withdrawal from icc ct not in india 's best interests : _UNK

Generated summaries
gujarat rajya sabha polls to fix rules for molestation considers visa polls

DocSum 6.7: Document, ground truth summary and generated summaries with the News Summary data from model *3* with an attention mechanism at epoch 400.

6.5.5 Summary

The results clearly show that SumGAN does not perform well on the News Summary dataset. The results obtained from validation show no traces of improvement, regardless of whether the worker is initialised with an encoded document or if an attention mechanism is used.

As for the training data, SumGAN manages to copy entire summaries, but not pair the summary to the right document, when the attention mechanism is applied. This changes when the adversarial training is initiated, but after some adversarial epochs, the generator policy breaks down, and at the end of the adversarial training, random tokens are generated.

6.6 TLDR-Submission-Relationship

The larger dataset TLDR-Submission-Relationship was substantially more time-consuming to train than the smaller News Summary dataset. Because of that, it could not be trained as many epochs. Even though the total number of epochs were reduced from 400 to 40, training each of *1*, *2*, and *3* took between three and four days. This meant that no hyperparameter tuning could be performed. Instead, all parameters were based on suggestions from previous works.

As in Section 6.5, plots of the various metrics are presented first and followed by sample summaries from different epochs during the training process.

6.6.1 Evaluation Metrics

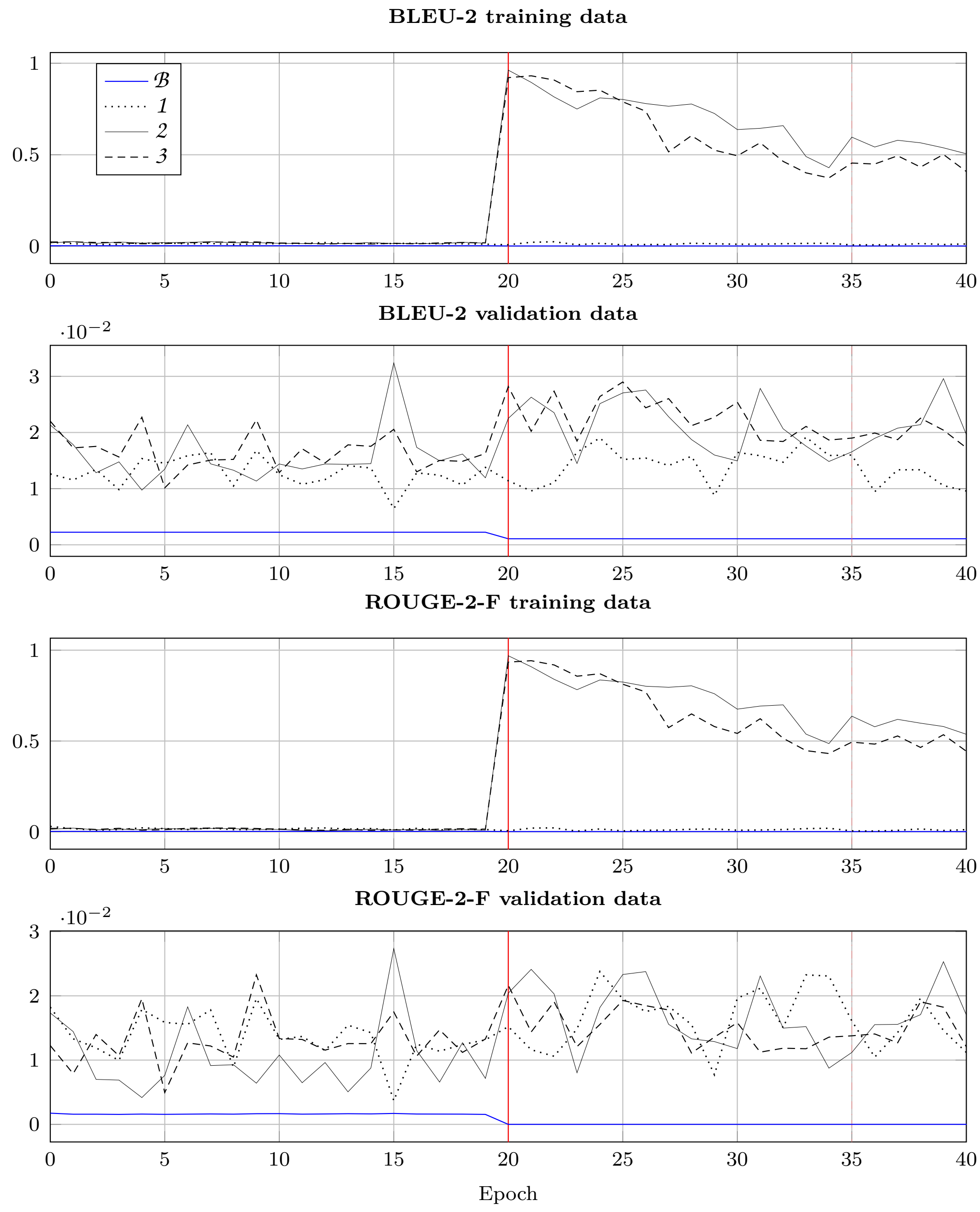


Figure 6.5: BLEU-2 and ROUGE-2-F scores for the training and validation parts of the TLDR-Submission-Relationship dataset. Note the different scales for the training and validation data on the y-axis.

For the training set, the BLEU-2 and ROUGE-2-F scores are flat near zero and the baseline during pre-training, which can be seen in Figure 6.5. When SumGAN goes into adversarial training though, the BLEU-2 and ROUGE-2-F values rise to almost 1.0, and then slowly decrease for the remaining epochs for models *2* and *3*, while model *1* does not improve at all. For model *2* and *3*, this behaviour is similar to the one reported on the News Summary dataset, with the difference that the decrease in

BLEU-2 and ROUGE-2-F scores is slower for the TLDR-Submission-Relationship dataset. For model *1*, the behaviour is entirely different since it remains at the same level as during pre-training.

For the experiment with the News Summary dataset, the BLEU-2 and ROUGE-2-F scores dropped down to the levels obtained during pre-training after 15 epochs, whereas the levels of the evaluation metrics are kept at a higher level for the remaining experiment, with values around 0.5 after 20 adversarial epochs. It is also noteworthy that the BLEU-2 and ROUGE-2-F values are significantly higher than the baseline, which was not the case for the News Summary dataset.

We also see that the behaviour of the training data is not present for the validation data, in which the evaluation metrics are fluctuating in a noise like manner, just above the baseline, and is not improving during training. Also, there is no model that excels the others in performance. To conclude, none of the models are able to generalise to unseen data even though models *2* and *3* perform promising results on the training data. This indicates that the model is exposed to overfitting.

6.6.2 Loss and Accuracy

The manager and worker losses are presented in Figure 6.6 and 6.7.

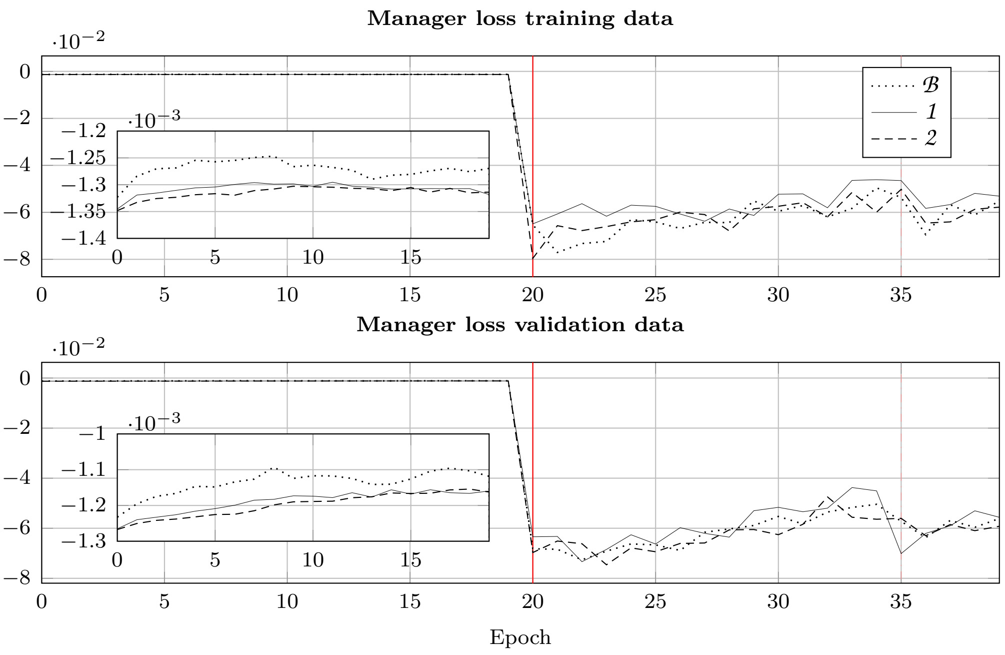


Figure 6.6: Manager loss for training and validation data of the TLDR-Submission-Relationship dataset.

When comparing the manager loss curves in Figure 6.3 and 6.6, we see some differences between the manager loss of the News Summary data and the manager loss obtained for TLDR-Submission-Relationship for the training and validation

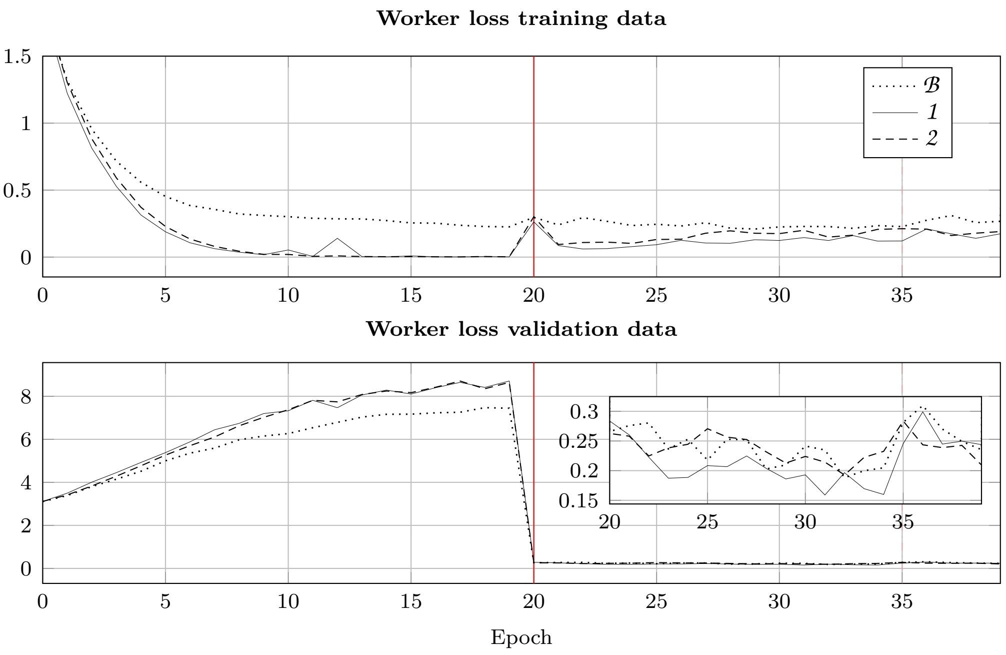


Figure 6.7: Worker loss for training and validation data of the TLDR-Submission-Relationship dataset.

datasets. For the former, the manager loss shows a clear case of overfitting where the training loss decreases during the entire pre-training and the validation loss decreases at first and then suddenly increases. For the validation part of TLDR-Submission-Relationship, though, both manager and worker loss increase during the pre-training. During the adversarial training, both the validation and training loss behave in similar ways and do not display the oscillations present in the News Summary plots. The reason for the absence of oscillations is probably that only one interleaved epoch takes place during the training of TLDR-Submission-Relationship. An interesting observation is that the interleaved training epoch has no distinct effect on the manager loss in any of the cases.

Contrary to the manager loss, the worker loss of the training and validation data, shown in Figure 6.7 do not resemble each other. The worker loss for the training set decreases to 0 and the validation curve increases up to around 8 for models *2* and *3* and somewhat lower for model *1*. When adversarial training starts, neither the training data nor the validation data loss seems to increase as they did for the News Summary dataset. As for the manager loss, there are no oscillations and the interleaved epoch has no apparent effect.

For the discriminator, the first observation from the data displayed in Figure 6.8 is that the accuracy hardly ever reaches 1.0, which is the case for a perfect discriminator. As the adversarial training starts, the accuracy even falls to 0.5 for the training data, which practically means that the discriminator is clueless since it has a 50 % chance at placing the label correctly. This supports the performance presented by the BLEU-2 and ROUGE-2-F scores in Figure 6.5. For the validation data, though, the accuracy increases after adversarial data starts and thereafter remains steady at around 0.9.

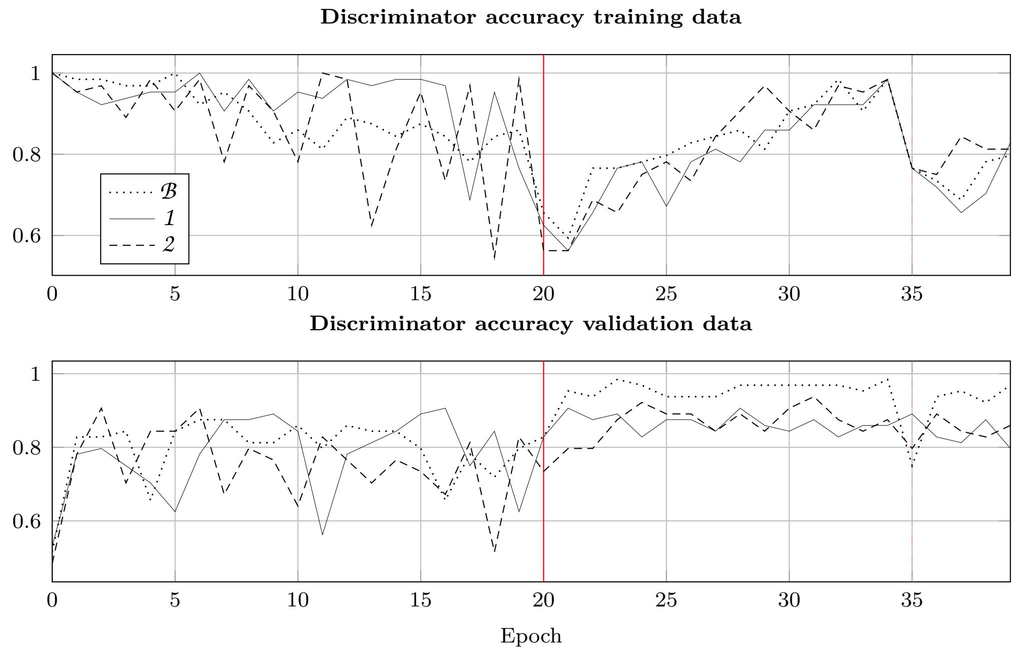


Figure 6.8: Discriminator accuracy for training and validation data of the TLDR-Submission-Relationship dataset.

6.6.3 Text Quality

In contrast to the News Summary dataset, the TLDR-Submission-Relationship data set is significantly larger, with the consequence that there were no DocSum pairs that occurred on the selected epochs for which results are presented. Instead, five different DocSums are presented, with generated summaries from the training and validation data for pre-training epochs 0 and 9, and for adversarial epochs 0, 9 and 19.

As for the experiments with the News Summary dataset, samples generated with model *2* are presented. The results from model *3* to the ones from model *2* which is supported by the similarity between the plots shown in the previous section. Model *1*, however, has no distinct progression from the level achieved during pre-training.

6.6.3.1 Training Data

From DocSums 6.8 and 6.9, sampled at pre-training epochs 0 and 9, we see that the generated summaries have a proper language structure and are relevant to the dataset. However, they are by no means connected to the document or the summary.

Document
i 'll try to keep this short , we 've been on about 6 dates so this is very recent . we have great conversations , we like to do a lot of the same things , i 've met her friends and they 're all great and we always have a good time but when it comes down to it somethings just not there . so i guess what my question is why do you think sometimes things just do n't click and am i just psyching myself out ?

Reference
new relationship everything should be great but does n't feel that way , am i _UNK my own happiness ?

Generated summaries
i 'm always been talking to my girlfriend and i 'm not ready to be uncomfortable . please help me .

DocSum 6.8: Document, ground truth summary and generated summary with the TLDR-Submission-Relationship training data for pre-training epoch 0.

Document
i am a [19m] and i work at a grocery store . i have a crush on a [24-25f] coworker who is a cashier there . she and i are always nice to each other . and she usually starts small conversations with me . and if she needs me at the register for something she calls me by my name not by the department i work in like most everyone else does . is it weird to like an older female coworker ? thanks . **

Reference
just want to know if anyone thinks it 's weird ? and how i should _UNK the situation ?

Generated summaries
i work with a guy with a friend and she 's still friends with me . what should i do ?

DocSum 6.9: Document, ground truth summary and generated summary with the TLDR-Submission-Relationship training data for pre-training epoch 9.

When examining sampled DocSums from the generated summaries obtained from the adversarial training with the training data with model *2*, we see that the ground truth summaries are perfectly replicated at epoch 0. This result is showcased in DocSum 6.10. By reading DocSum 6.11 and 6.12 from epoch 9 and 19, it becomes evident that the copying mechanism is somewhat kept throughout the adversarial epochs, but that the text starts diverging from ground truth after a while. This is probably the reason for the maintained performance shown by the metrics in the former section.

Document
so i 've known this girl for around 3 months and after recent events of us hanging out things have got awkward between us , we both have feelings for each other but it didnt work out , is there anything i can do or say to bring the relationship back to friendship ?

Reference
awkward with girl , how to get back to good friendship

Generated summaries
awkward with girl , how to get back to good friendship

DocSum 6.10: Document, ground truth summary and generated summary with the TLDR-Submission-Relationship training data for adversarial epoch 0.

Document
i had the time to myself and the crappy car and apartment , but i still wish for it again sometimes . it 's just hard having to take care of kids and try to be everything for everyone all the time and not be able to have time to just be yourself .

Reference
i agree with this person and sometimes even if you did get it out of your system , you still want it .

Generated summaries
i agree with this person and sometimes i get back with your _UNK , does it mean anything i do ?

DocSum 6.11: Document, ground truth summary and generated summary with the TLDR-Submission-Relationship training data for adversarial epoch 9.

Document
well my ex girlfriend was acting odd for the past three days . yesterday she said some awful things about me and kind of said we cant go further and she wanted to end it . i felt terrible and still kind of do.what are ways to feel better and not be in the dumps ?

Reference
ex gf said bad things ... called it _UNK feel like shit . help !

Generated summaries
ex gf said bad things ... called it 's not feel romantic !

DocSum 6.12: Document, ground truth summary and generated summary with the TLDR-Submission-Relationship training data for adversarial epoch 19.

6.6.3.2 Validation Data

By examining DocSums 6.13-6.17, we see a similar pattern to the one seen for the News Summary data: during pre-training, DocSum 6.13 (epoch 0) and 6.14 (epoch 9), the language quality is mediocre, and there is no direct connection to the document.

Document
so me and my bf have been together for two years now . he has kids with another woman . i have met the mother and the kids several times and they are great . me and my bf just got a place together . i 'm scared that something might happen between them while i 'm not here . should i be concerned ? am i overreacting ? all advice is greatly appreciated .

Reference
bf is bringing ex and kids to our place without me there . should i be concerned ?

Generated summaries
what are some short time to tell my girlfriend of me healthy ?

DocSum 6.13: Document, ground truth summary and generated summary for the TLDR-Submission-Relationship validation data at pre-training epoch 0.

For adversarial epoch 0 though, we see that the generated summary in DocSum 6.15 has a better language quality in terms of sentence structure and grammar,

Document
she wants to buy us tickets to hawaii so i can meet her aunt and hang out with her family on vacation for a couple days . i do n't feel comfortable with her buying an expensive ticket for me , especially after talking about it a few weeks earlier with her saying i should save up so we could go together soon . (knowing i 'm a broke college student)

Reference
my girlfriend is offering to buy expensive tickets for us to vacation with her family in _UNK , but i do n't want her to .

Generated summaries
i 'd like your _UNK move on the one of one .

DocSum 6.14: Document, ground truth summary and generated summary for the TLDR-Submission-Relationship validation data after 9 epochs.

but is still not relevant to the document. The generated summary in DocSum 6.15 could still be relevant to some other document in the dataset, since it is still poses a relationship oriented question.

Document
so the title is the gist of it . we are in similar stages in our education (she transfers 1 semester ahead of me) . so there is no real concern there . but i am concerned on the social differences . she can go out to bars and i cant unless i get in illegally . she has a job and i 'm still searching . plus other less important things . any advice is appreciated

Reference
is this age gap too big for a practical relationship in the us ?

Generated summaries
so i 'm in love with her but it is n't to be , but i do n't know how to move on

DocSum 6.15: Document, ground truth summary and generated summary for the TLDR-Submission-Relationship validation data at adversarial epoch 0.

For DocSums 6.16 (adversarial epoch 9) and 6.17 (adversarial epoch 19), the quality of the language decreases, and there is still no connection to the document.

Document
so there 's this cute girl in my anthropology class that i see looking over at me until i catch her . the problem is that i do n't know how to just approach her out of the blue and talk/pysche myself out of doing it when i have the chance when we leave class . i 'm fairly certain she has some interest in me but i guess i 'm too afraid of the results from bad experiences .

Reference
how do i approach a girl out of the blue from class ?

Generated summaries
_UNK on date with dating friend for a month after we can we date someone who you want to have a relationship , but when your so is n't _UNK .

DocSum 6.16: Document, ground truth summary and generated summary for the TLDR-Submission-Relationship validation data after 9 adversarial epochs.

Document
so i have n't seen or really spoken to my father for 7 almost 8 years . i have seen him twice once at my grandmas house and second when he got into a motorcycle accident almost a year ago . i have sent him letters and text messages but i do n't get a response . i do n't know if i should even bother sending him one , but i do want to show him i did it without him .

Reference
my does n't talk to me should i send him a _UNK of my graduation ?

Generated summaries
what do you prevent , girls that are going through even dates ?

DocSum 6.17: Document, ground truth summary and generated summary for the TLDR-Submission-Relationship validation data after 19 adversarial epochs.

6.6.3.3 Model with Attention Mechanism

In DocSum 6.18 one of the summaries is presented that is generated by model *3*, i.e. with attention. From this model, we see that the characteristics of the generated summary is quite similar to the ones of summaries generated by model *2*: it is somewhat grammatically correct, but has no connection to the document or to the ground truth summary. Combining ocular inspection of summaries from model *3* and examining the evaluation metrics, we can conclude that adding an attention mechanism does not improve the generated summaries' connections to the document.

Document
i 'm 5'0 " and a little over 200 pounds . i 'm taking a swimming class , and the gym at my school offers numerous things to do to stay in shape , but i 'm nervous the social/dating scene will be different because of my weight . i have a good personality , and i 'm pretty confident , but i never thought much about it . opinions ? tips ? any advice you can give me would be good .

Reference
i 'm scared being overweight will fuck up my _UNK life .

Generated summaries
i have a huge crush on this girl but she is with me cheating .

DocSum 6.18: Document, ground truth summary and generated summaries with the TLDR-Submission-Relationship data from model *3* with an attention mechanism at epoch 40.

6.6.4 Summary

The results from the TLDR-Submissions-Relationships data show that the model is able to generate reasonable text during the adversarial training but without connection to the document. We also see that the training process is somewhat stable, even though the model gets about half of the summaries right at the end of the adversarial training.

An interesting observation is that the BLEU-2 and ROUGE-2-F scores for model *1* remains the same as the training progresses from pre-training to adversarial training while model *2* and *3* perform better. This might indicate some improvement caused by giving the generator information of the document (model *2* and *3*), but it cannot be stated with confidence.

6.7 News Summary with Differently Paired Data

Experiments with data labelled as *valid/invalid*, as described in Section 4.2.1, were run on the News Summary dataset in the same way as with the *real/generated* labelled samples. Therefore, the simulation times were roughly the same: between one and two days per experiment.

6.7.1 Evaluation Metrics

We did not observe any significant improvements compared to the results presented in Section 6.5. The BLEU-2 and ROUGE-2-F scores in Figure 6.9 are quite similar to the ones presented in Figure 6.1 in the sense that the evaluation metrics are stable during pre-training at relatively low values, but then increases with a sudden spike as the adversarial training starts, and then starts rising at the interleaved epochs, and then drops down again.

However, we see peaks in the BLEU-2 and ROUGE-2-F scores even for the validation data, albeit small. This could be an indicator of the method of adding falsely paired data giving improvements in the document summarisation ability, but the effect is far too small in order to draw any conclusions. As for the results presented in Section 6.5, the BLEU-2 and ROUGE-2-F values are below the baseline of sequences with UNK tokens.

6.7.2 Loss and Accuracy

The worker loss seen in Figure 6.10 is similar to the one presented in Figure 6.2, and there are thus no further remarks to be made. On the manager loss in Figure 6.11 though, we see that the model overfits during pre-training which is characterised by loss decreasing and then increasing in the beginning of the pre-training. During the adversarial training, we see in Figure 6.10 that the values are oscillating and follows the interleaved training schedule. However, the graphs for model *1* and *3* diverge from model *2*, which in turn is following the behaviour observed for all three models in Figure 6.3. The values for model *1* and *3* are lower than for model *2*, which indicated that the manager performs better for these two models.

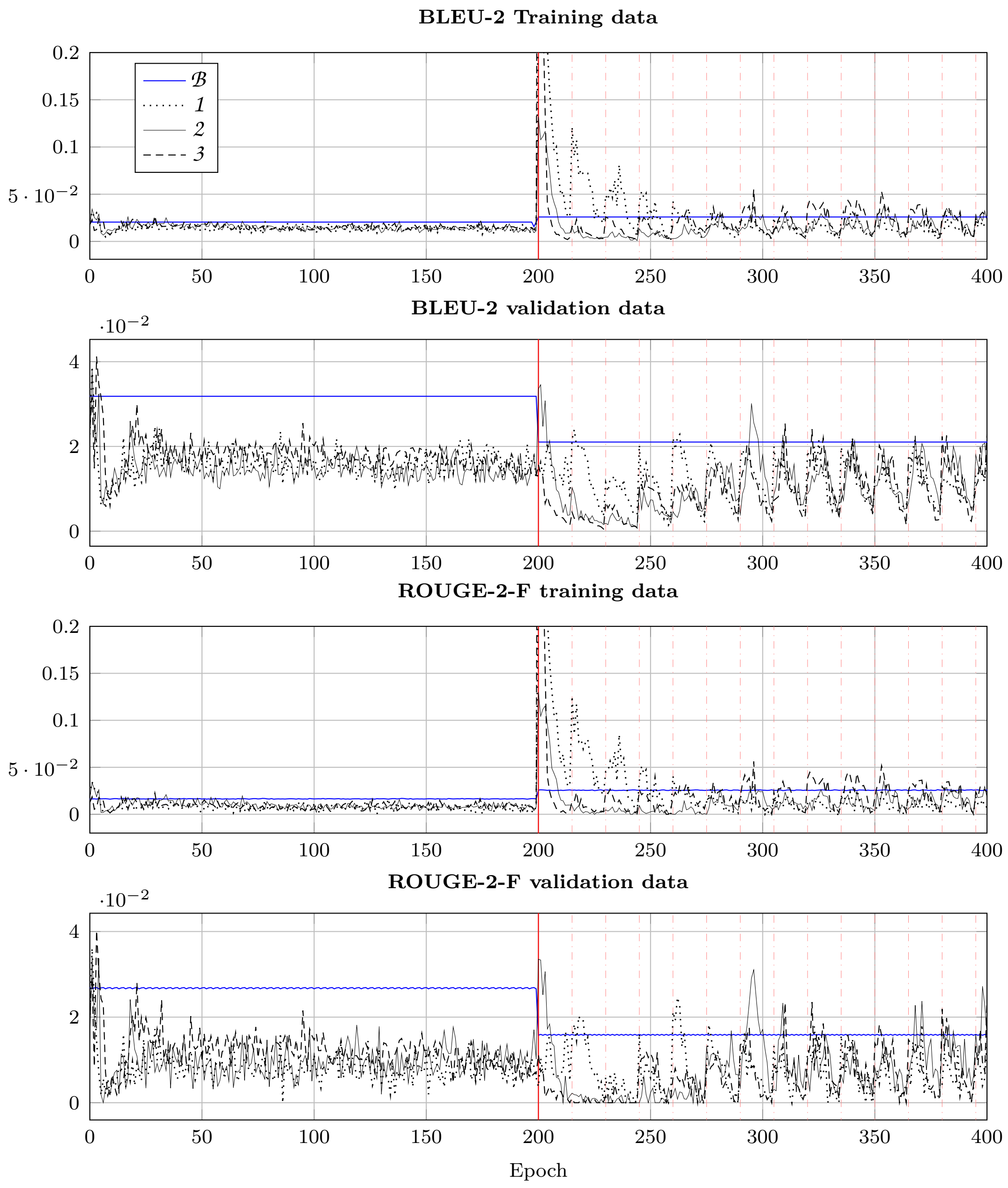


Figure 6.9: BLEU-2 and ROUGE-2-F scores for training and validation parts of the News Summary dataset, where some of the documents have been assigned other summaries as described in Section 4.2.1.

The discriminator accuracy in Figure 6.12 shows no signs of improvement during the training process, and it is noise like for all three models, with mean around 0.8 and 0.7 for the training and validation data respectively.

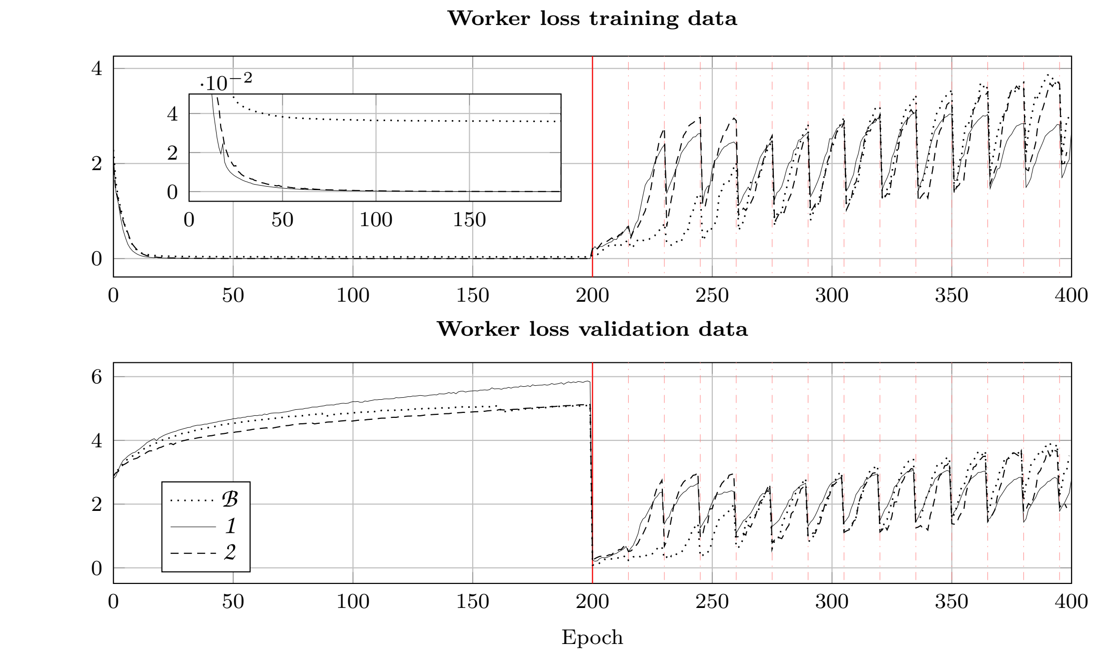


Figure 6.10: Worker loss for training and validation parts of the News Summary dataset, where some of the documents have been assigned other summaries as described in Section 4.2.1.

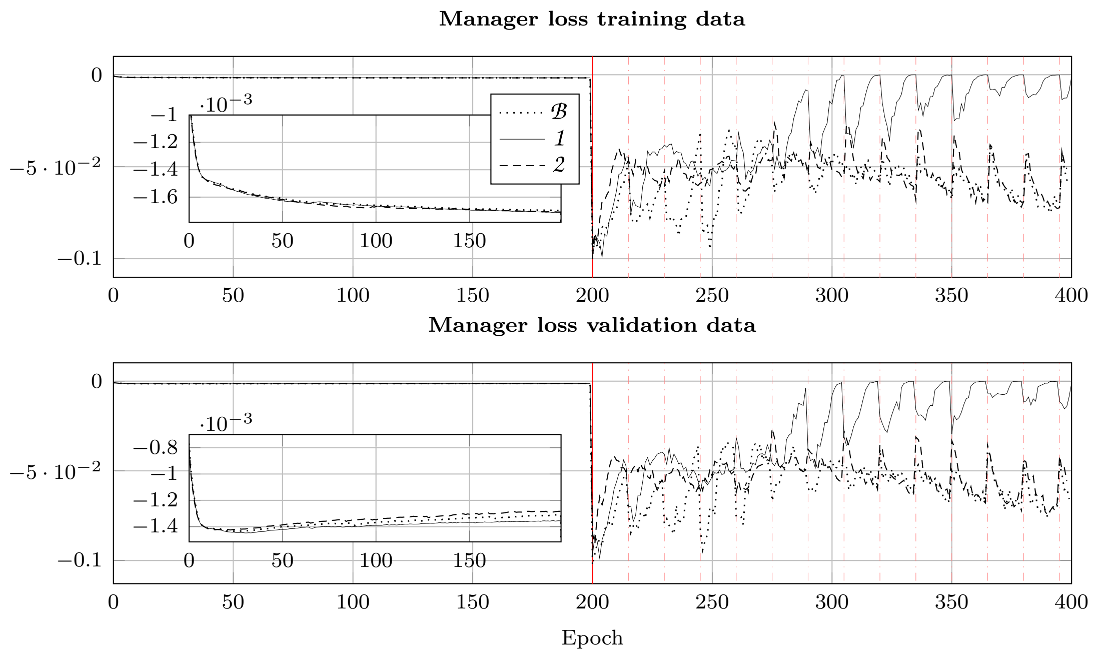


Figure 6.11: Manager loss for training and validation parts of the News Summary dataset, where some of the documents have been assigned other summaries as described in Section 4.2.1.

6.7.3 Text Quality

With the falsely paired data, no significant improvements were seen. At the beginning of the adversarial training, model *2* produces summaries with many UNK

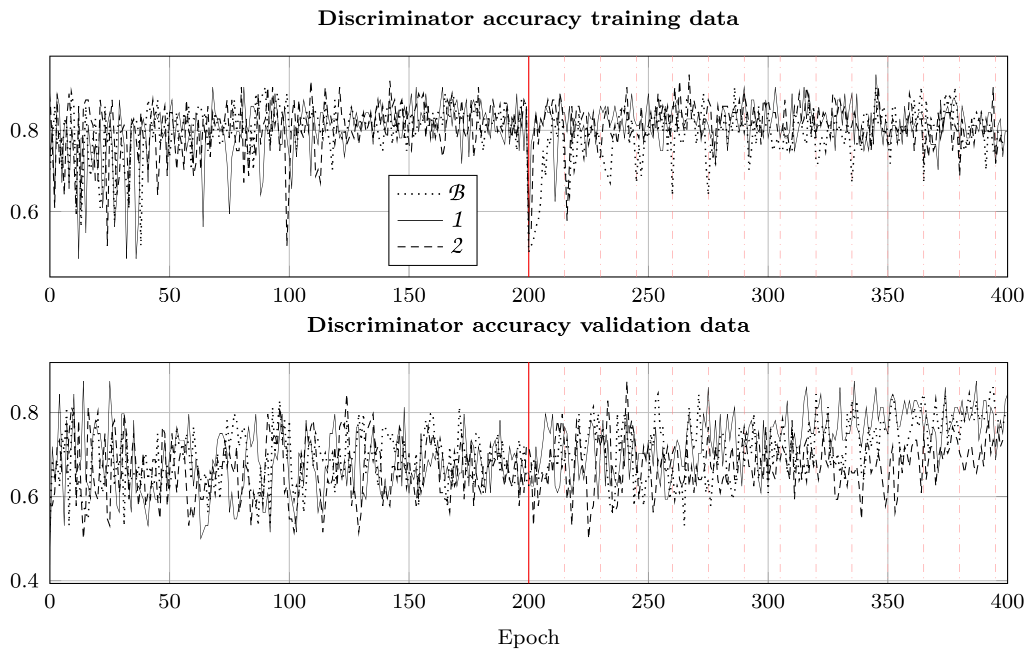


Figure 6.12: Discriminator accuracy for training and validation parts of the News Summary dataset, where some of the documents have been assigned other summaries as described in Section 4.2.1.

tokens, and not very coherent language. They are not relevant to the document, as exemplified by DocSum 6.19. In addition, the model quickly performs worse, and at epoch 9, only noise is generated as revealed by DocSum 6.20.

Document
tesla has fired engineer aj vandermeyden who accused the automaker of ignoring her complaints of sexual harassment and paying her less than her male counterparts . vandermeyden had claimed she was taunted and catcalled by male employees . the company argued that she received special treatment at the expense of others , yet chose to " pursue a miscarriage of justice by suing tesla . "

Reference
_UNK fires female engineer who alleged sexual harassment

Generated summaries
_UNK _UNK , _UNK placed next _UNK with from us ? _UNK

DocSum 6.19: Document, ground truth summary and generated summary at adversarial epoch 0 for News Summary training data, where some of the summaries have been falsely paired when feeding to the discriminator.

6.7.4 Summary

By examining the loss curves and some hand picked samples, it turns out that the strategy of adding falsely paired DocSum pairs to the discriminator did not

Document
samajwadi party supremo mulayam singh yadav on monday said the party still belongs to him and he enjoys people 's support . he further added he led a spot-free life so far , and the supreme court gave him a clean chit when corruption charges were levelled against him . mulayam has approached the election commission to take up battle of party symbol ' cycle ' .

Reference
samajwadi party still belongs to me : mulayam singh

Generated summaries
26 pm modi _UNK must _UNK must _UNK chopra memorial memorial we cop loan after polls to sc to govt polls to govt off friend

DocSum 6.20: Document, ground truth summary and generated summary at adversarial epoch 9 for News Summary training data, where some of the summaries have been falsely paired when feeding to the discriminator.

improve any version of SumGAN. Instead, the model became even more instable when generating summaries.

7

Conclusion and Future Work

7.1 Conclusion

In this thesis we have investigated the possibilities of doing ADS, i.e. controlled text generation using generative adversarial networks. We have created a model named SumGAN which is extending LeakGAN, one of the best text generation models using GAN up to date, by adding encoded documents as guiding signals to the model. In addition, the performance of a model extended with an attention mechanism is also investigated. We have applied SumGAN on a small news dataset, as well as a larger dataset with Reddit submissions from `r/relationships`.

Our results conclude that using our model, we are not able to achieve high performing results. For the TLDR-Submission-Relationship dataset, we are able to generate texts with proper language that are relationships oriented, with the text generation being stable over training epochs. However, the produced summaries are nowhere near relevant to the given documents, nor to the ground truth summaries. For the News Summary data, we are not able to obtain a lasting generation of text with good quality. Instead the model collapses and produces noise. We also found that adding an attention mechanism to the model did not improve the document summarisation ability for neither News Summary nor TLDR-Submission-Relationship.

An important factor in designing and evaluating the model was the long training times. The experiments that produced the data shown in Chapter 6 were run over the course of two and a half weeks. For a thesis work which lasts about 20 weeks, that is a substantial amount of time given that constructing and implementing the model also was time-consuming. It is also limiting in the sense that there was no time to further investigate the many interesting aspects that were discovered during the data analysis. We therefore advise the reader interested in performing a similar project to keep in mind that even with a good hardware setup, experiments take a long time to complete.

Even though the results are far from satisfactory, we still see good potential in the idea to incorporate the adversarial method to ADS. Text generation with GAN is a field that is being actively researched and where rapid progression has been made in the short time that it has existed. With each research project, the collective knowledge of the inner workings of generative models is increased and more efficient methods are developed. In order to advance the performance further and gain more knowledge about how to perform controlled text generation, more research is needed. Based on the experience gained from this thesis work, the following sections present suggestions for aspects of the model to study further, as well as possible improvements.

When doing document summarisation in general, one should note that there are several ethical aspects to keep in mind. To let a fully automated computer program summarise a document could make the summary skewed towards a certain standpoint, especially if the content in question is of a political nature. Such skewing could occur if, for instance, a model for generating news summaries is trained solely with news articles with a specific political agenda, or so called "fake news". It is also important to choose the generative vocabulary carefully in order to avoid summaries containing hate speech.

7.2 Future Work

As concluded above, SumGAN in its current configuration does not produce satisfactory results. There exist, however, many ways in which the model could possibly be improved and aspects of the model that could be examined further. Some of these are discussed below.

7.2.1 Interesting Aspects to Study

The analysis presented in Chapter 6 examine only some of the many interesting aspects of the model. The reason is the long training times required on the hardware available during the thesis. Given more time, more experiments could have been performed. Below is a list of experiments that the authors suggest as starting points for future research

- **Perform a grid search on the hyperparameters of the model.** Because of time constraints, all hyperparameters were set to values suggested by previous research. By varying one hyperparameter at a time, a better configuration could probably be found.
- **Train with more data.** In deep learning, training on more data usually implies better model performance. The datasets presented in Chapter 5 are rather small compared to previous attempts. Given more data, the model might learn to generalise to unseen data. To do this, future research might turn towards the field of headline generation, which was not covered in this report, but that is closely related to document summarisation. Within the

field, a wide array of large datasets are available, which might be utilised.

- **Examine the performance peak at the border between pre-training and adversarial training.** As discussed in Chapter 6, all experiments showed a distinct peak in BLEU-2 and ROUGE-2-F score on the training data. Manual sampling of the generated texts show that the generator manages to create perfect copies of the ground truth summaries, so the BLEU-2 and ROUGE-2-F scores are accurate. As the reason behind the peak remains unclear, the phenomenon should be examined further. A way to do that would be to examine the size of the probabilities from which tokens are sampled.
- **Monitor the scale of the gradients during training.** A common problem in training of deep networks is vanishing gradients. During the experimentation with SumGAN, the gradients were not monitored, and thus there is a possibility that vanishing gradients occurred, even though methods such as bootstrapped rescaled activation was employed.
- **Examine the interplay between the manager and the worker.** The manager and the worker should work in unison to collectively improve the result of the text. The worker receives information about the document from previous worker states and the attention mechanism. The manager receives only information about the generated subsequence and has the ability to control what of the output from the worker that is used in the sampling. Therefore, if the manager is destructive, it might block out much of the influence of the document. By monitoring the interplay between them, relevant discoveries might be made, which could result in improvements.
- **Study the effectiveness of the attention mechanism.** All of the experiments presented in Chapter 6 indicated that including an attention mechanism did not improve the performance of the model. It would therefore be interesting to investigate the workings of the attention mechanism during training, similar to previous research, such as the thesis by Helmertz and Hasselqvist [8].

7.2.2 Possible Improvements to the Model

As described in Chapter 4, SumGAN constitutes of LeakGAN, an attention mechanism and a document encoder, where the latter two are picked from other attempts at ADS. There are, however, other techniques that have been used in successful attempts at ADS. Therefore, the insufficient performance of SumGAN should not be interpreted as proof that ADS with GAN is not feasible, but rather as an example of how the two topics could be merged. The list below presents relevant techniques that could be brought into SumGAN in order to improve its performance.

- **Use a pointer mechanism.** In texts with a large portion of entities such as names of people, places and organisations, many of the words are uncommon, wherefore the probability of them being generated is small. A possible remedy is to use a *pointer mechanism* which in effect is a binary switch that determines if the next token should be generated by the generative neural networks, or by copying the token in the document which draws the most attention. This way, the generator can output rare as well as common tokens. Another potential benefit comes from the fact that LeakGAN treats text generation as reinforcement learning problem where an agent takes actions based on its state. By copying tokens from the document, the state of the agent would be affected such that it was drawn closer to tokens relevant for the document.

- **Employ beam search.** In its current implementation, SumGAN generates new tokens by sampling from a distribution. This method is common in content creation tasks where diverse texts are to be generated. In controlled text generation, however, precision is more important than diversity. Therefore, hard sampling could be used in combination with beam search to replace the sampling in SumGAN. With beam search, a set number of sequences would be generated in parallel and the best among them selected. In the current implementation of LeakGAN, and thereby SumGAN, the diversity of the sampling of tokens is lowered by lowering the temperature in the softmax, see Section 3.5.3.3. To experiment with different temperatures, or even taking the argmax token would probably yield more precision in the summary generation.

- **Modify the loss functions.** As shown by Arjovsky and Bottou [26], the choice of the loss function plays a large role in the performance of the model. Arjovsky and Bottou, among others, suggest the use of the Wasserstein metric instead of the original metric proposed by Goodfellow [4], which offers more stability. Therefore, it would be interesting to experiment with the Wasserstein metric.

Bibliography

[1] C. Olah. (2015, aug) http://colah.github.io/posts/2015-08-understanding-lstms/. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] R. Nallapati, B. Xiang, and B. Zhou, "Sequence-to-sequence rnns for text summarization," *CoRR*, vol. abs/1602.06023, 2016. [Online]. Available: http://arxiv.org/abs/1602.06023

[3] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *CoRR*, vol. abs/1509.00685, 2015. [Online]. Available: http://arxiv.org/abs/1509.00685

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. [Online]. Available: https://arxiv.org/pdf/1406.2661.pdf

[5] J. W. Y. Y. Lantao Yu, Weinan Zhang, "Seqgan: Sequence generative adversarial nets with policy gradient," *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 2852–2858, 2017. [Online]. Available: https://arxiv.org/abs/1609.05473v6

[6] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," *arXiv preprint arXiv:1706.03850*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03850v3

[7] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," *arXiv preprint arXiv:1709.08624*, 2017. [Online]. Available: https://arxiv.org/abs/1709.08624v2

[8] J. Hasselqvist and N. Helmertz, "Query-based abstractive summarization using neural networks," Master's thesis, Chalmers University of Technology, University of Gothenburg, 2017.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by

jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. [Online]. Available: https://arxiv.org/pdf/1409.0473.pdf

[10] D. I. Helgøy and M. Lund, "A continuous approach to controllable text generation using generative adversarial networks," Master's thesis, Norwegian University of Science and Technology, 2017.

[11] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A benchmarking platform for text generation models," *arXiv preprint arXiv:1802.01886*, 2018.

[12] N. Buduma and N. Locascio, *Fundamentals of deep learning : designing next-generation machine intelligence algorithms*, first edition. ed. Sebastopol, CA: O'Reilly Media, 2017.

[13] IBM. The art of tokenization. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en

[14] J. J. Webster and C. Kit, "Tokenization as the initial phase in nlp," in *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, ser. COLING '92. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 1106–1110. [Online]. Available: https://doi.org/10.3115/992424.992434

[15] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[17] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[19] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. [Online]. Available: http://www.jstor.org/stable/2236626

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov,

"Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015. [Online]. Available: http://arxiv.org/abs/1505.00387

[23] A. Karpathy. (2015, may) The unreasonable effectiveness of recurrent neural networks. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[25] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016. [Online]. Available: https://arxiv.org/abs/1701.00160v4

[26] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017. [Online]. Available: https://arxiv.org/pdf/1701.04862.pdf

[27] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016. [Online]. Available: https://arxiv.org/pdf/1611.02163.pdf

[28] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1703.01161*, 2017.

[29] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics.* Association for Computational Linguistics, 2002, pp. 311–318. [Online]. Available: https://www.aclweb.org/anthology/P02-1040.pdf

[30] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8. Barcelona, Spain, 2004. [Online]. Available: http://www.aclweb.org/anthology/W04-1013

[31] P. Nema, M. M. Khapra, A. Laha, and B. Ravindran, "Diversity driven attention model for query-based abstractive summarization," *CoRR*, vol. abs/1704.08300, 2017. [Online]. Available: http://arxiv.org/abs/1704.08300

[32] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017. [Online].

Available: https://arxiv.org/pdf/1704.04368.pdf

[33] S. Overflow. (2018, jan) Developer survey results 2018. [Online]. Available: https://insights.stackoverflow.com/survey/2018/#top

[34] N. Project. (2017, sep) Natural language toolkit. [Online]. Available: https://www.nltk.org

[35] S. Syed, M. Voelske, M. Potthast, and B. Stein, “Webis-tldr-17 corpus,” Nov. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1043504

[36] M. Völske, M. Potthast, S. Syed, and B. Stein, “Tl; dr: Mining reddit to learn automatic summarization,” in *Proceedings of the Workshop on New Frontiers in Summarization*, 2017, pp. 59–63. [Online]. Available: http://www.aclweb.org/anthology/W17-4508

[37] K. Vonteru. (2017, aug) News summary generating short length descriptions of news articles. [Online]. Available: https://www.kaggle.com/sunnysai12345/news-summary

[38] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval.* Cambridge: Cambridge University Press, 2008.

[39] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

A

Appendix 1

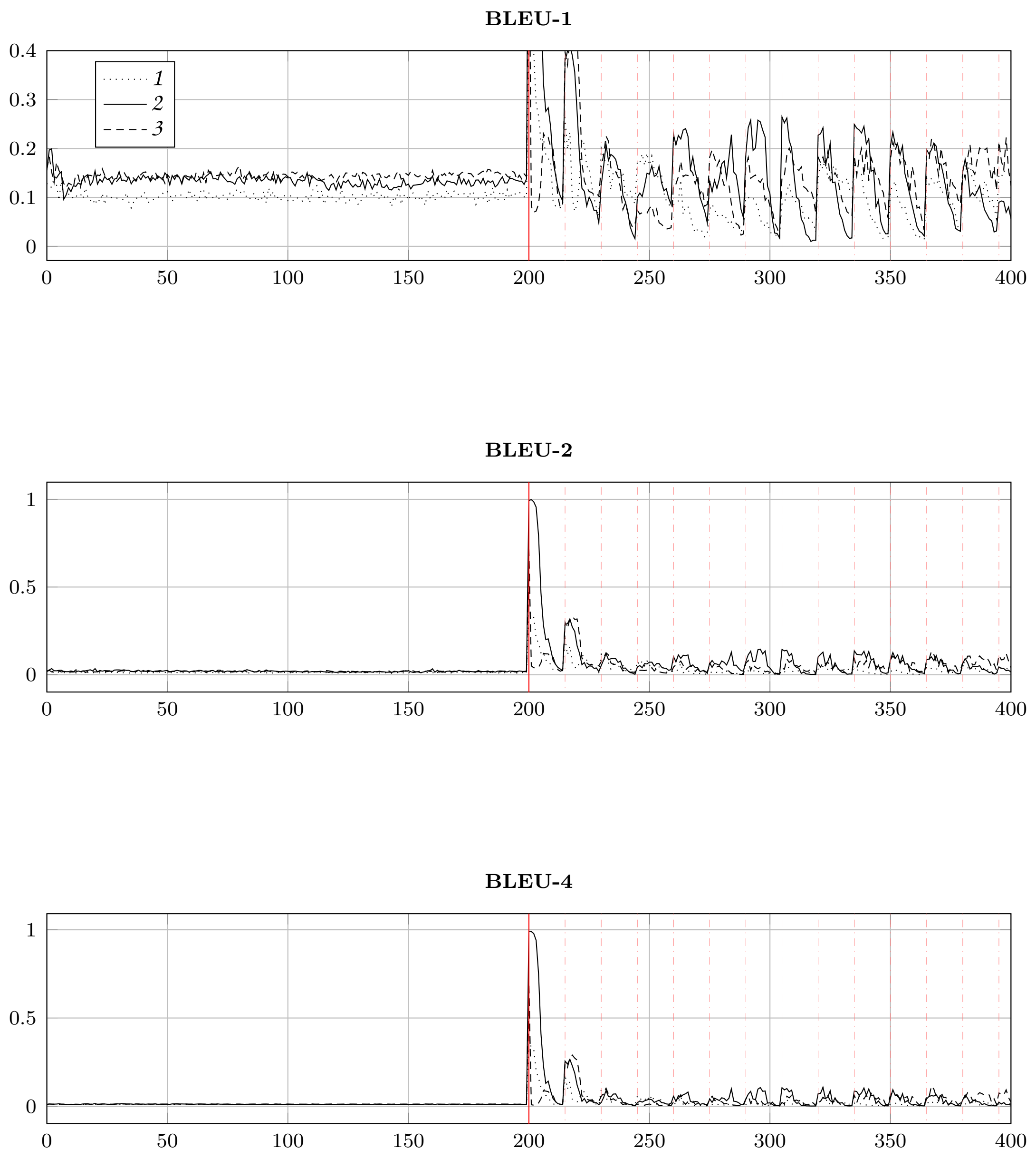


Figure A.1: BLEU scores for News Summary training data.

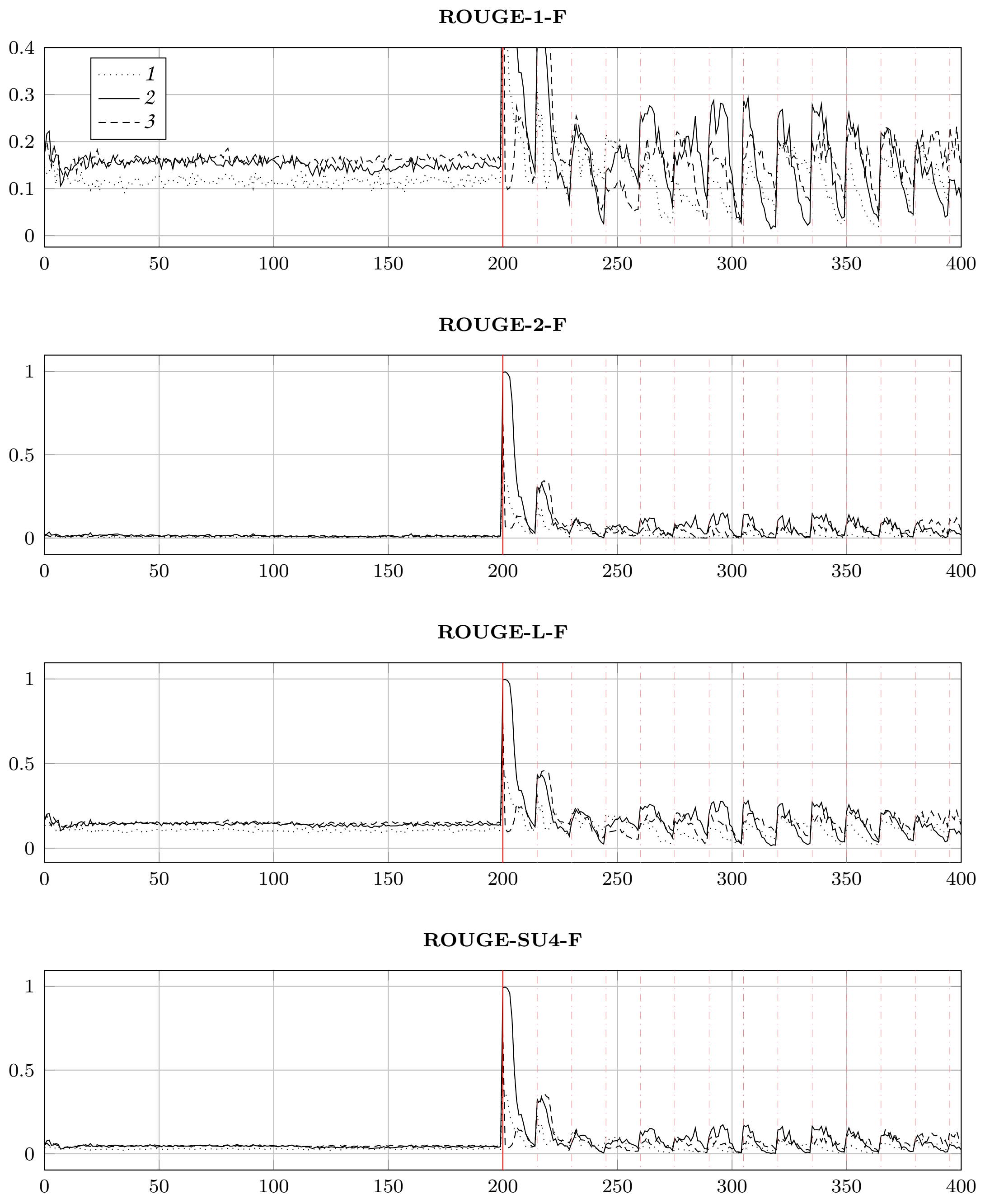
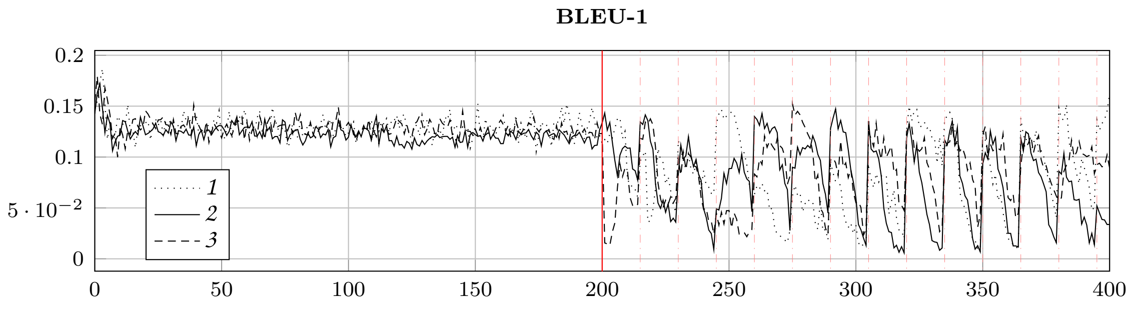
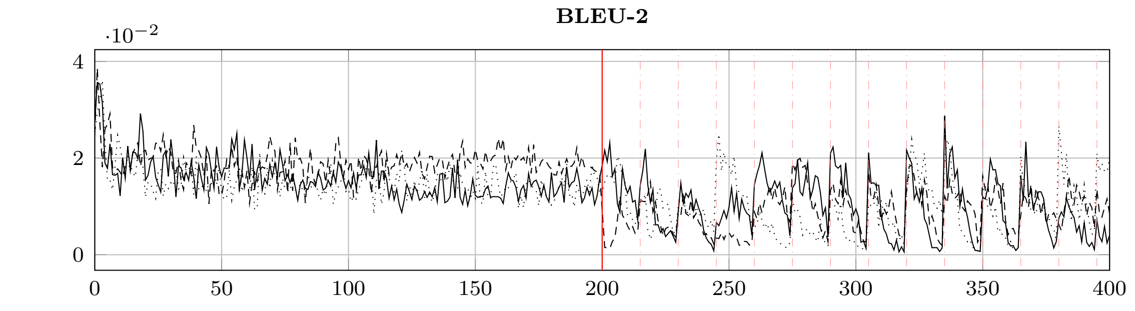


Figure A.2: ROUGE scores for News Summary training data.





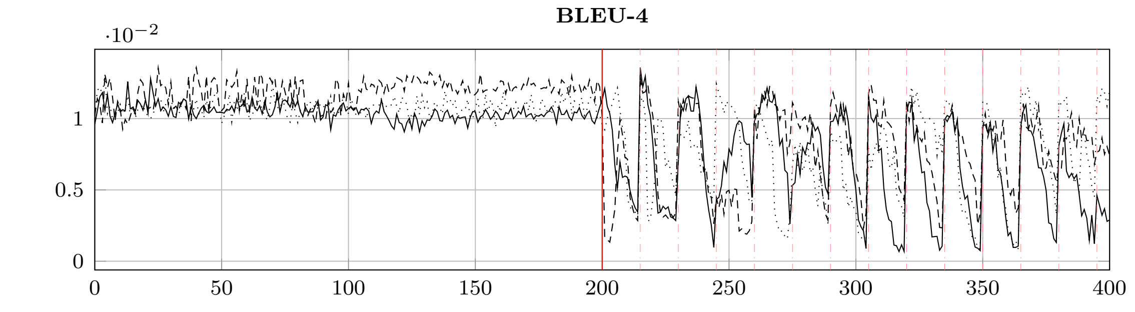


Figure A.3: BLEU scores for News Summary validation data.

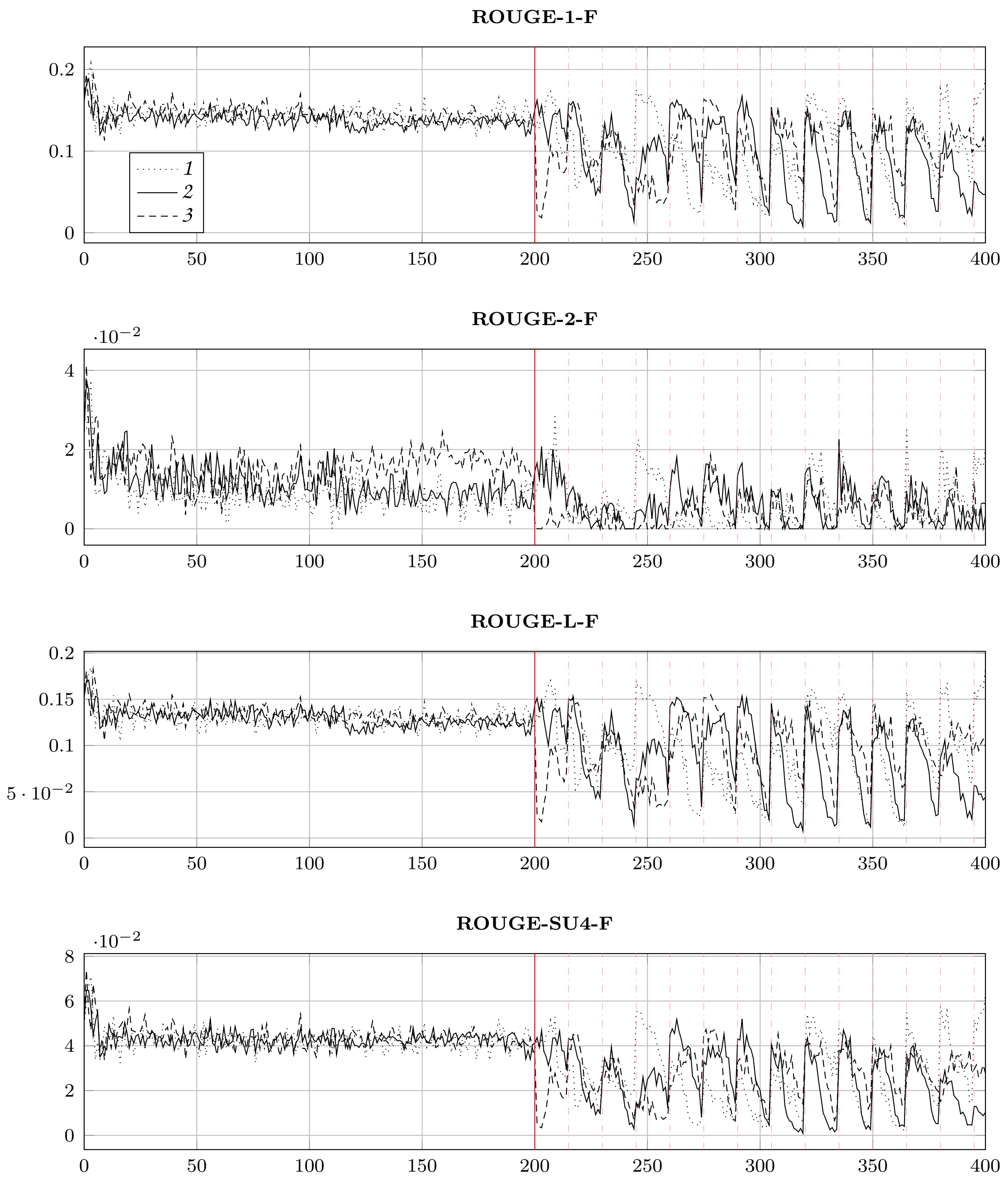


Figure A.4: ROUGE scores for News Summary validation data.

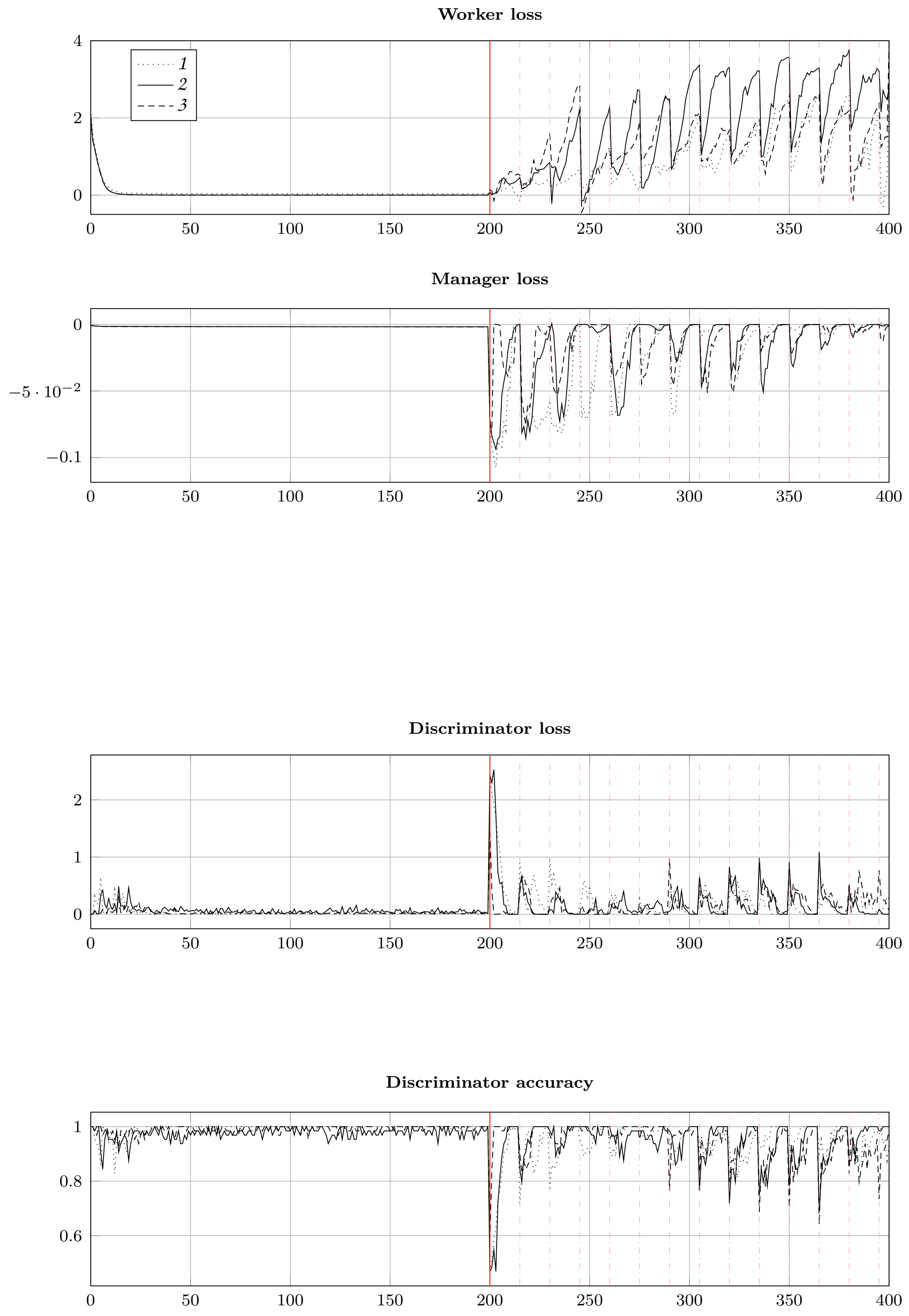
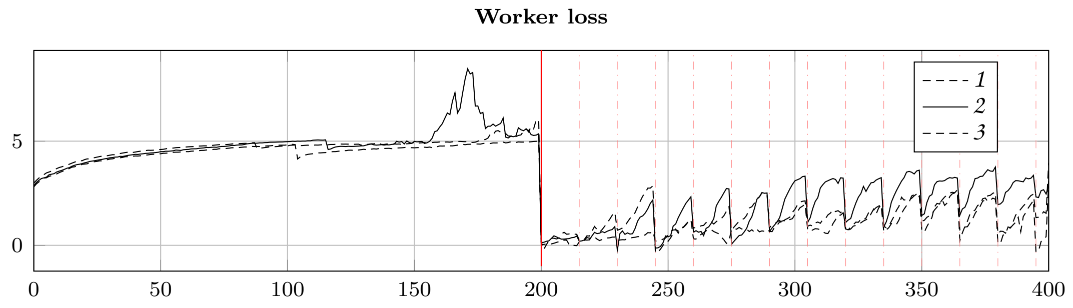
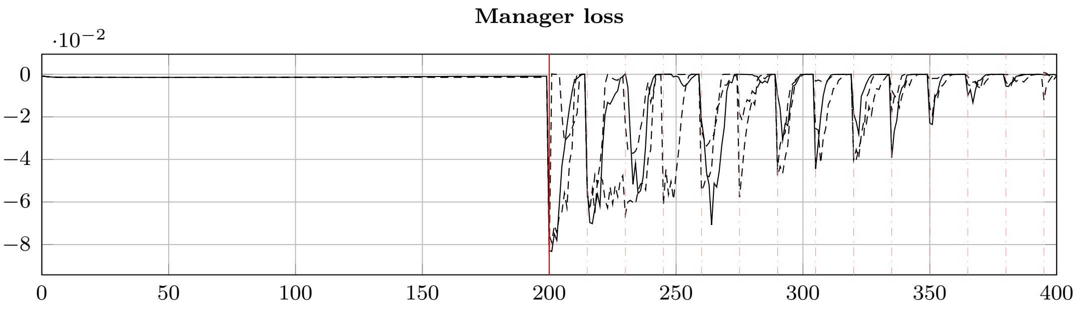
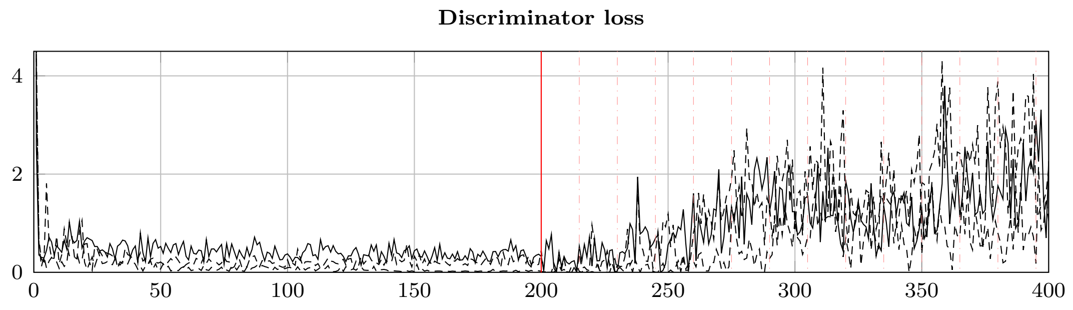


Figure A.5: Losses and accuracy for News Summary training data.







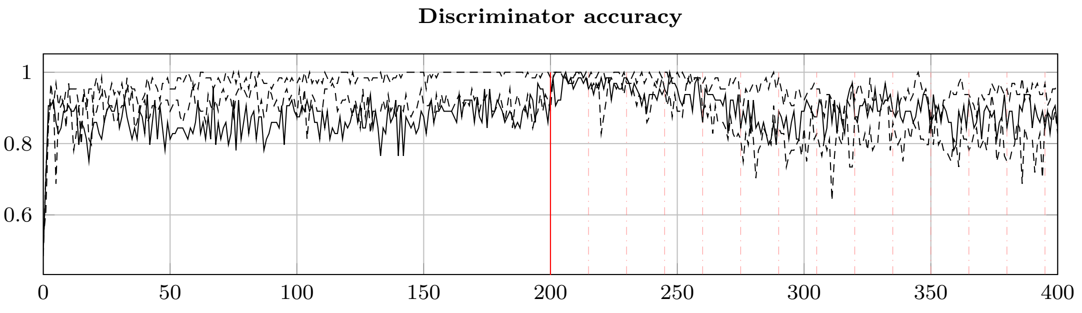


Figure A.6: Losses and accuracy for News Summary validation data.

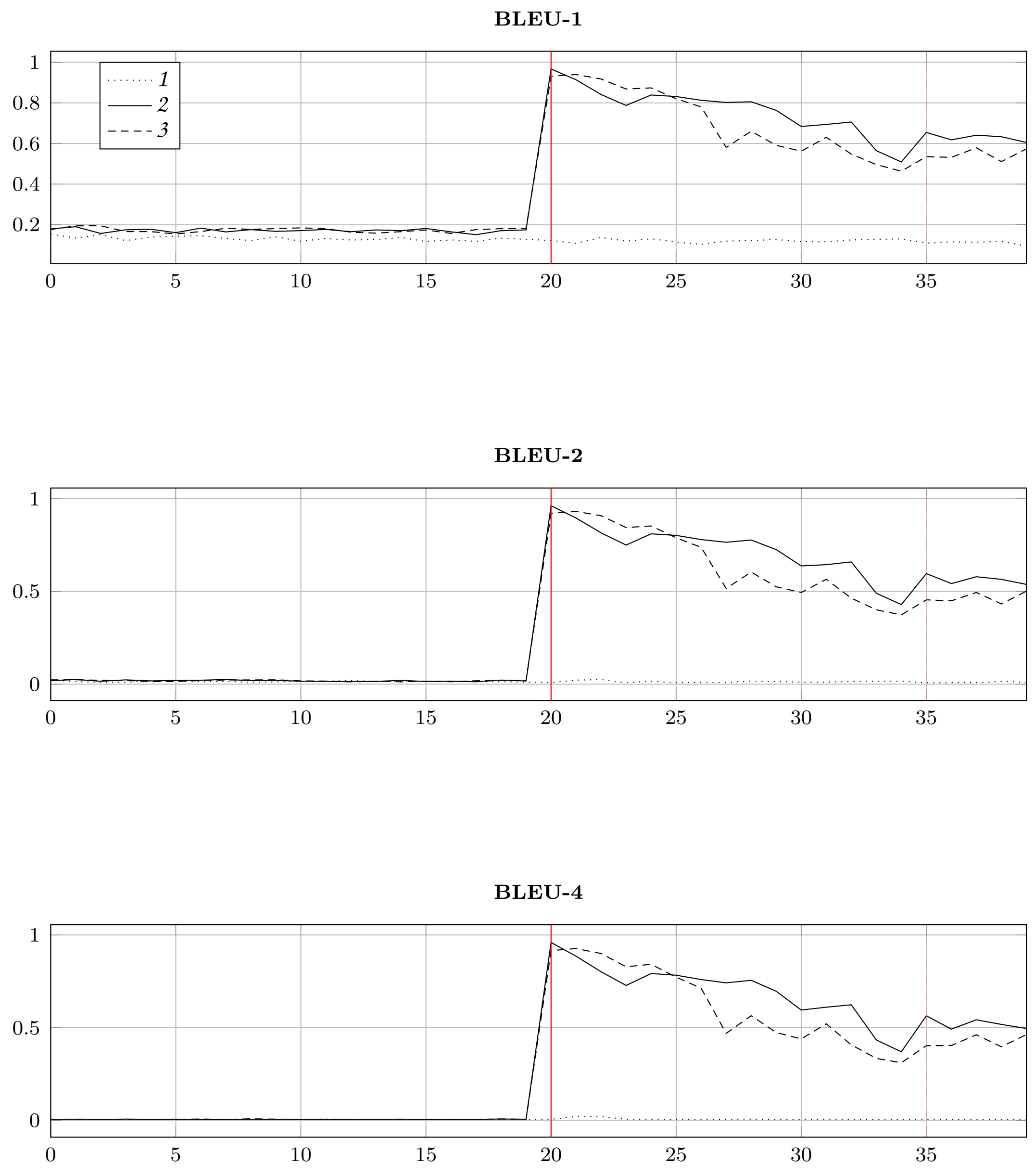


Figure A.7: BLEU scores for TLDR-Submission-Relationship training data.

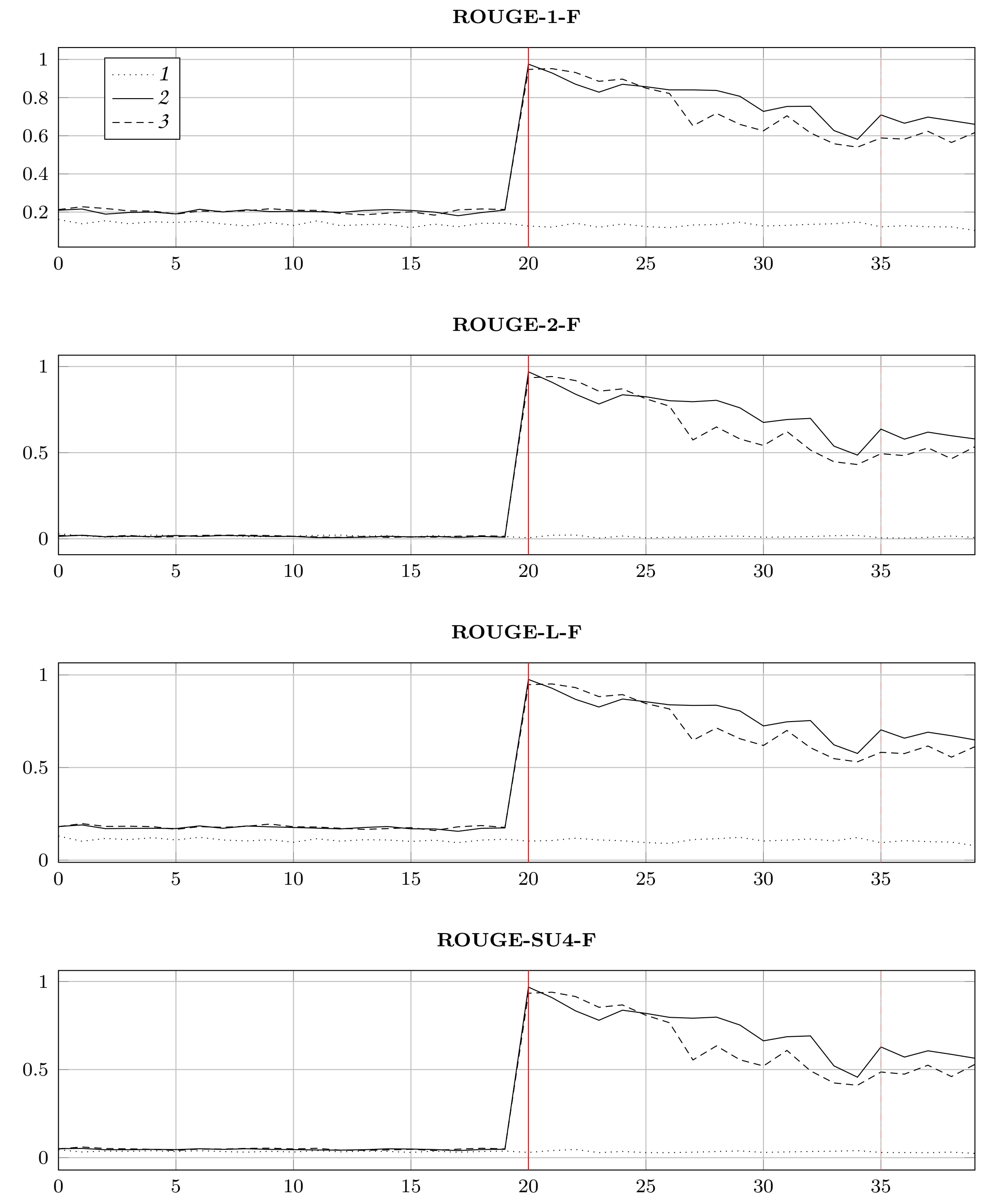


Figure A.8: ROUGE scores for TLDR-Submission-Relationship training data.

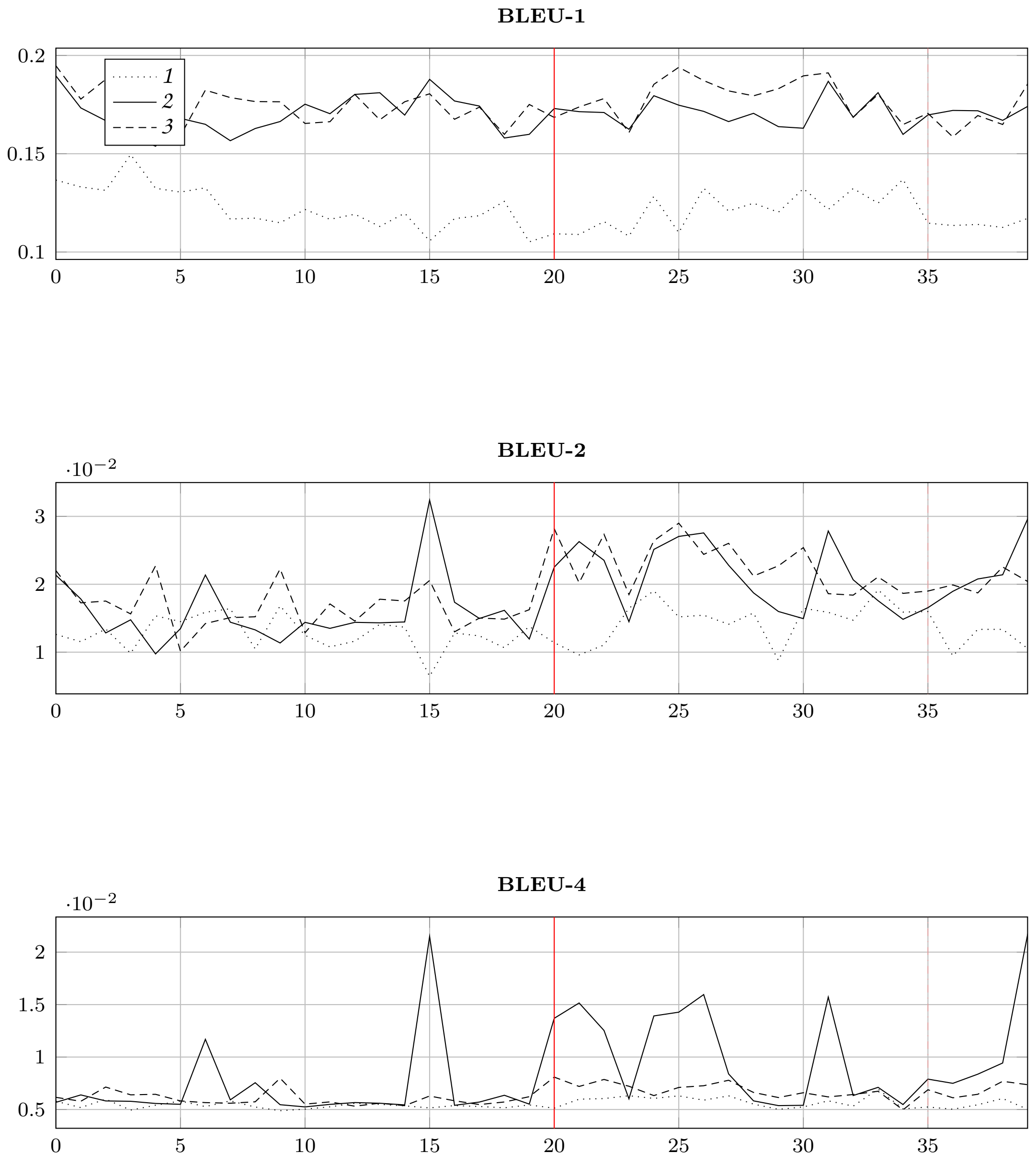


Figure A.9: BLEU scores for TLDR-Submission-Relationship validation data.

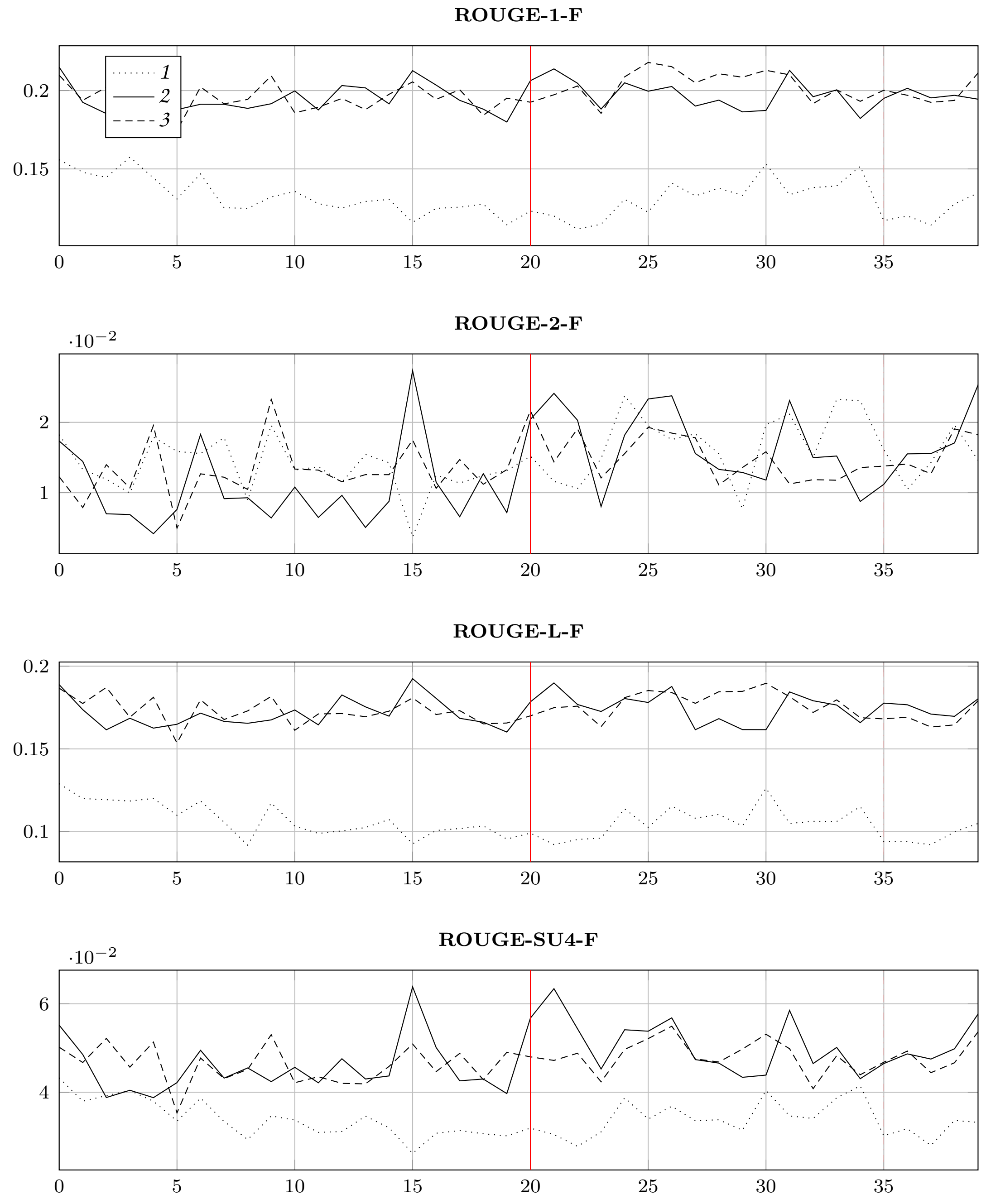


Figure A.10: ROUGE scores for TLDR-Submission-Relationship validation data.

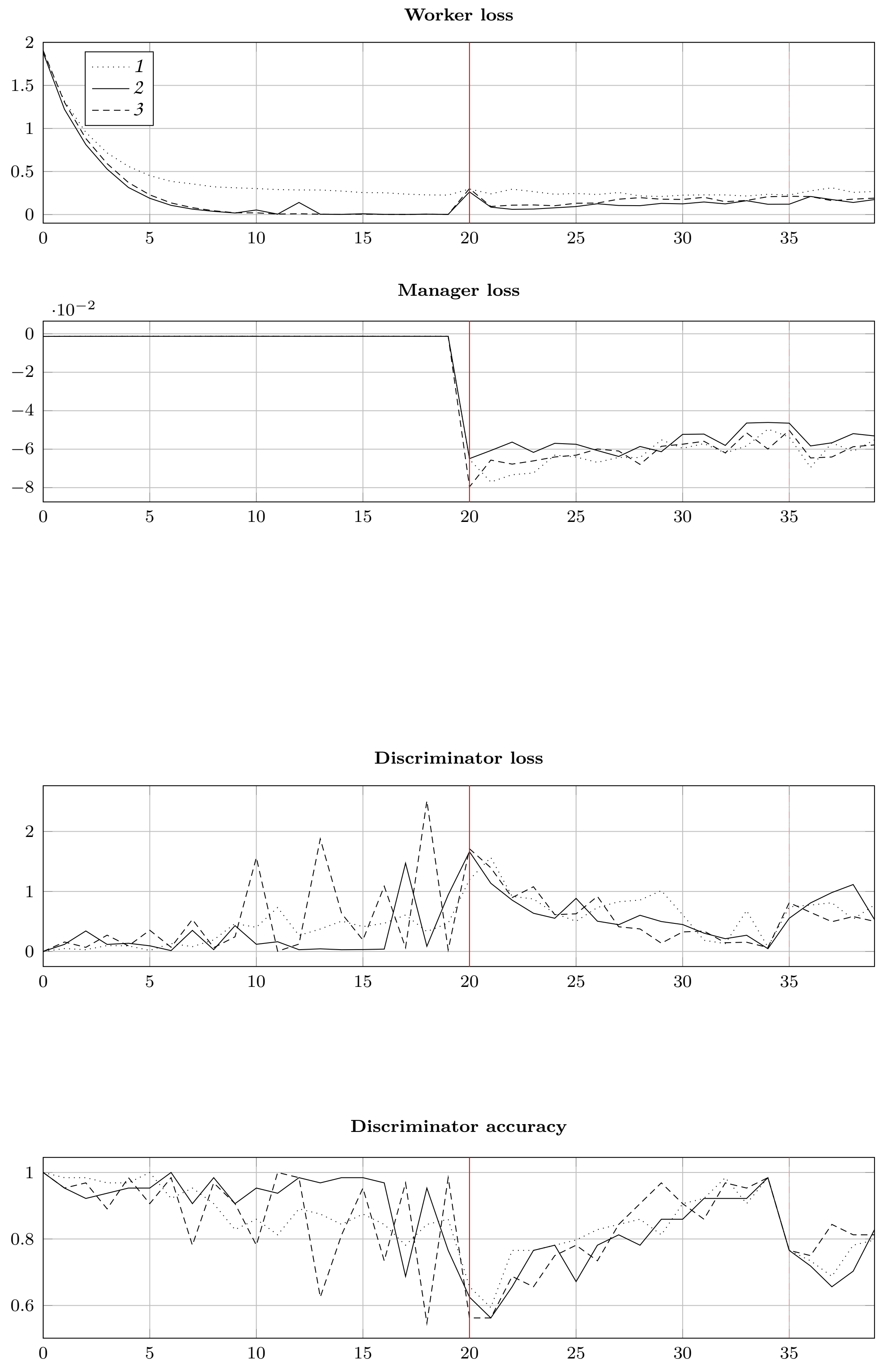


Figure A.11: Losses and accuracy for TLDR-Submission-Relationship training data.

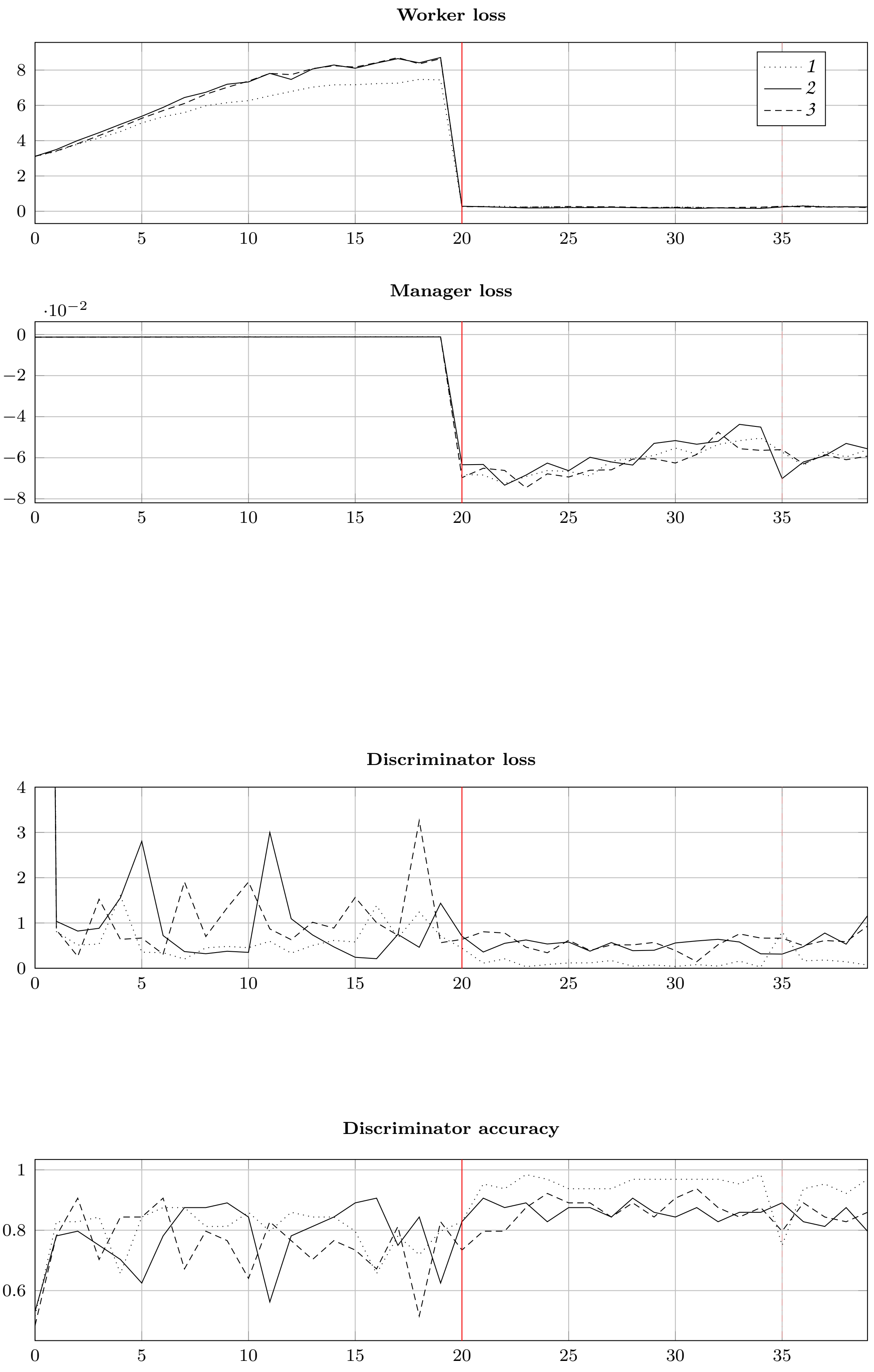


Figure A.12: Losses and accuracy for TLDR-Submission-Relationship validation data.

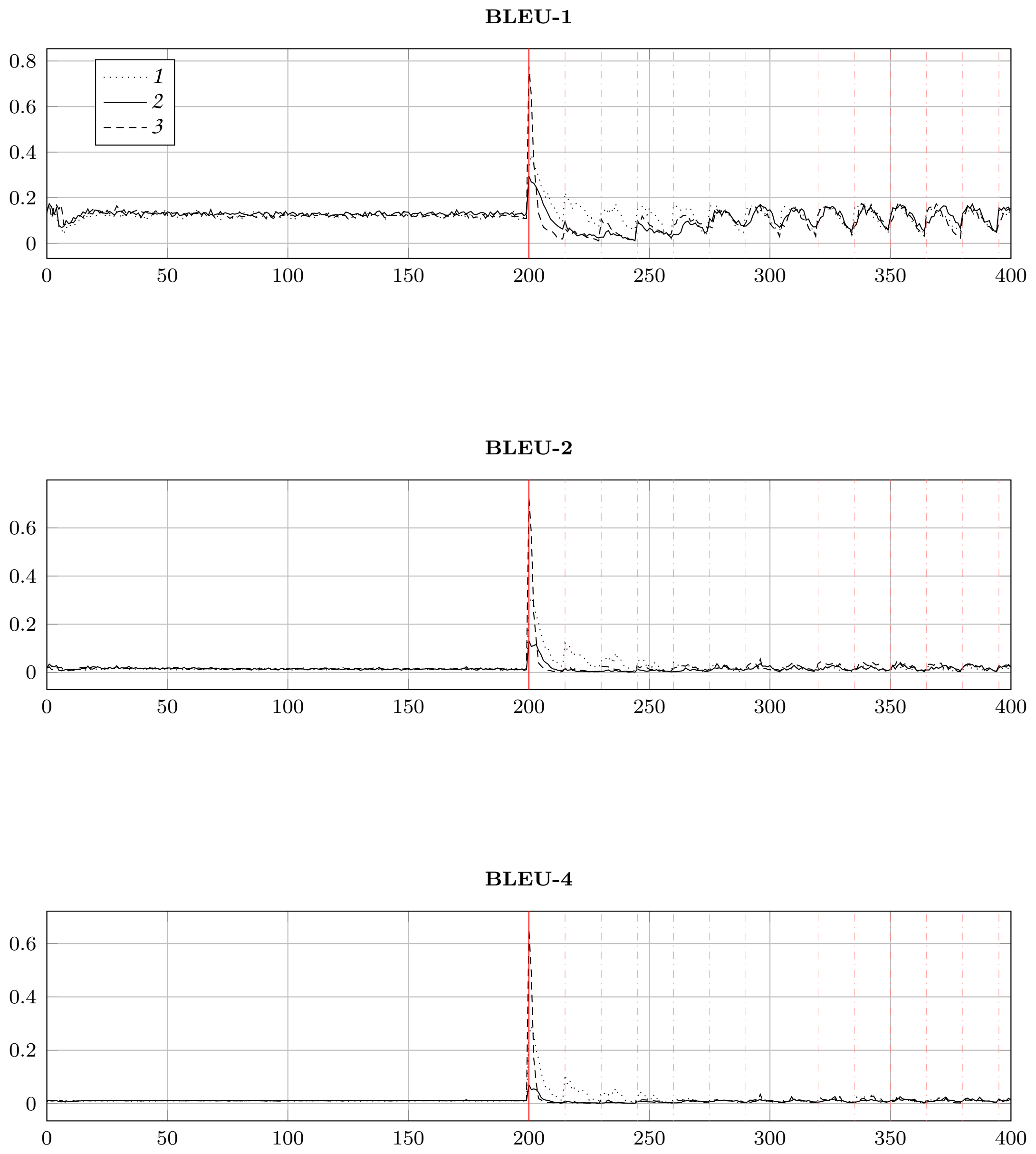


Figure A.13: BLEU scores for News Summary training data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.

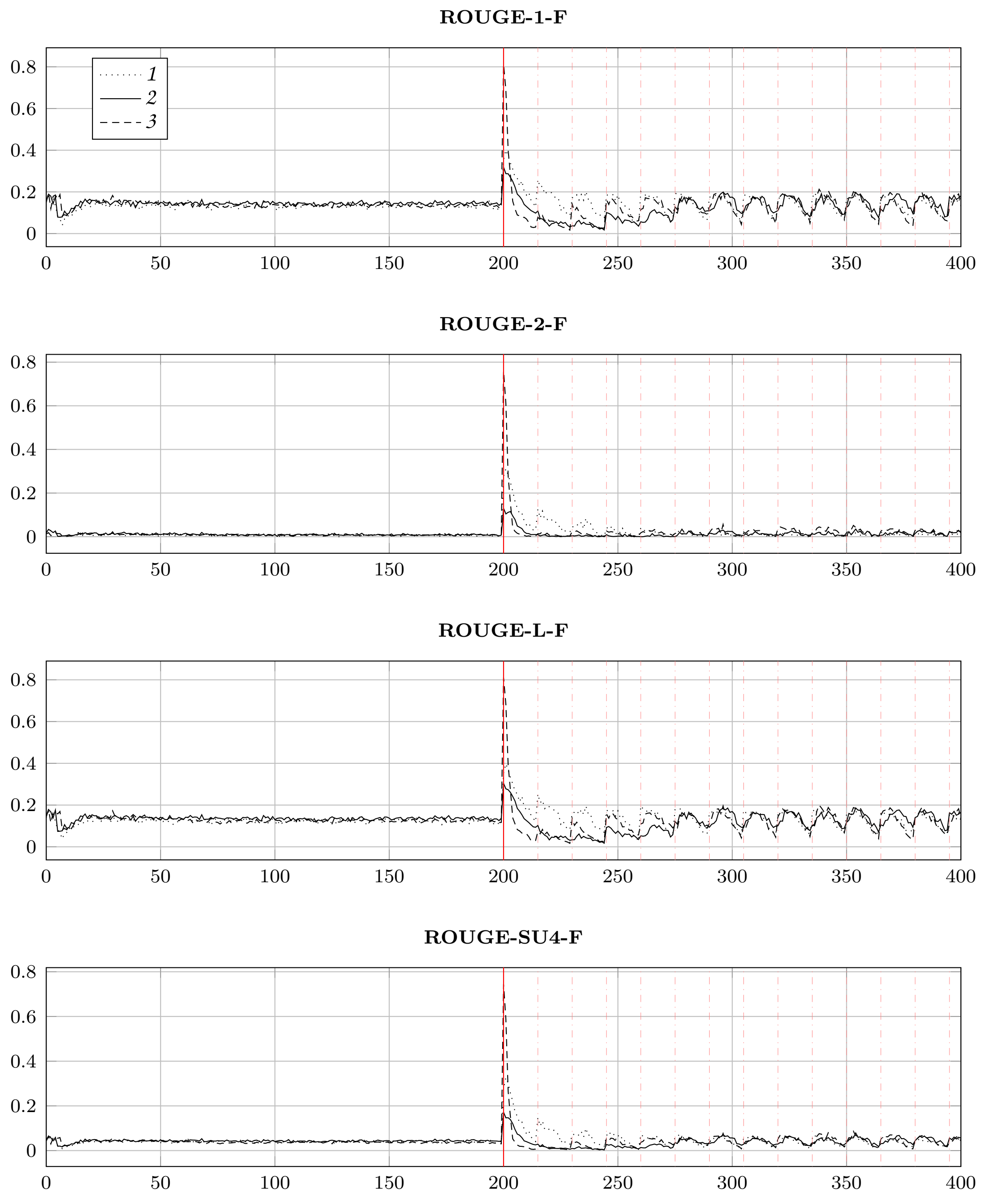
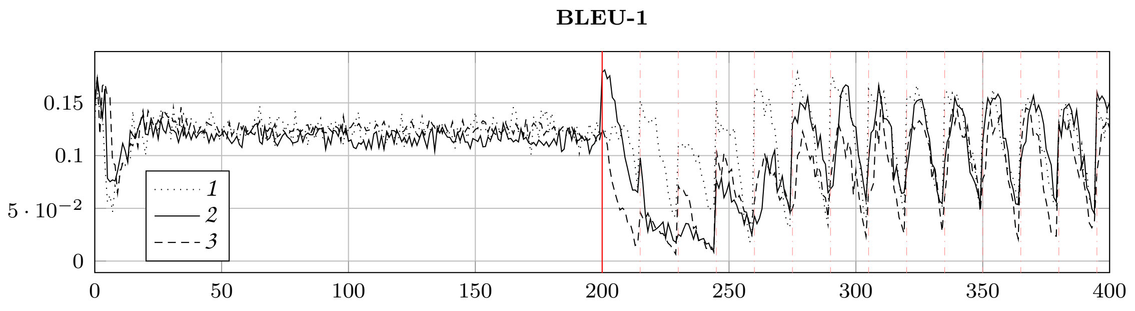
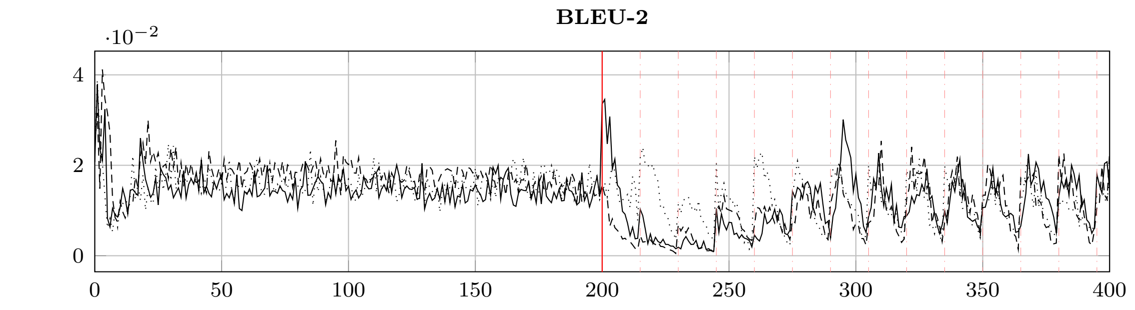


Figure A.14: ROUGE scores for News Summary training data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.





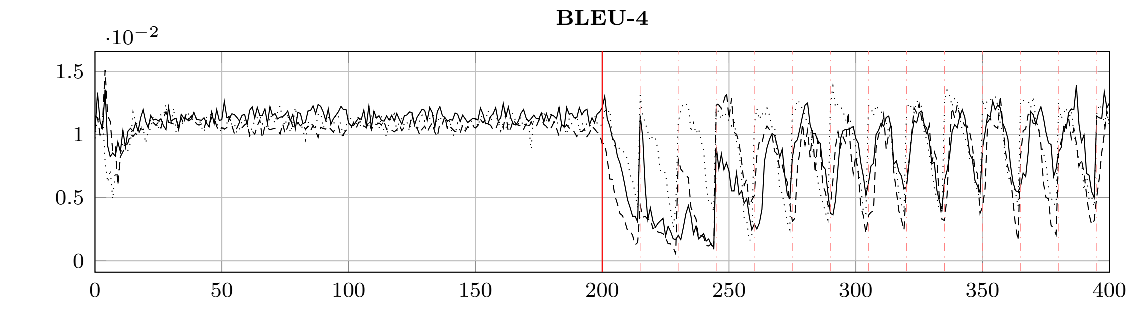


Figure A.15: BLEU scores for News Summary validation data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.

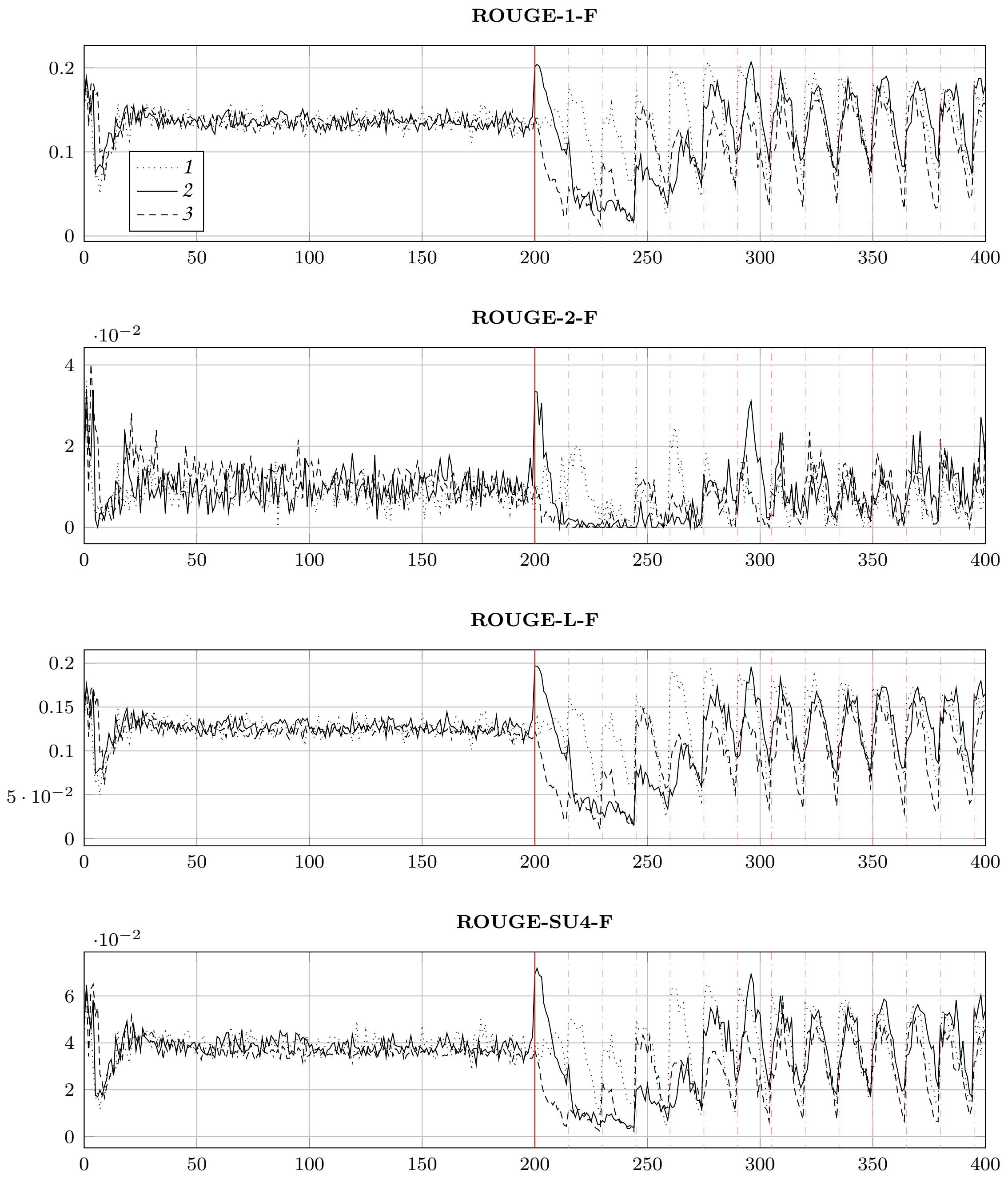


Figure A.16: ROUGE scores for News Summary validation data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.

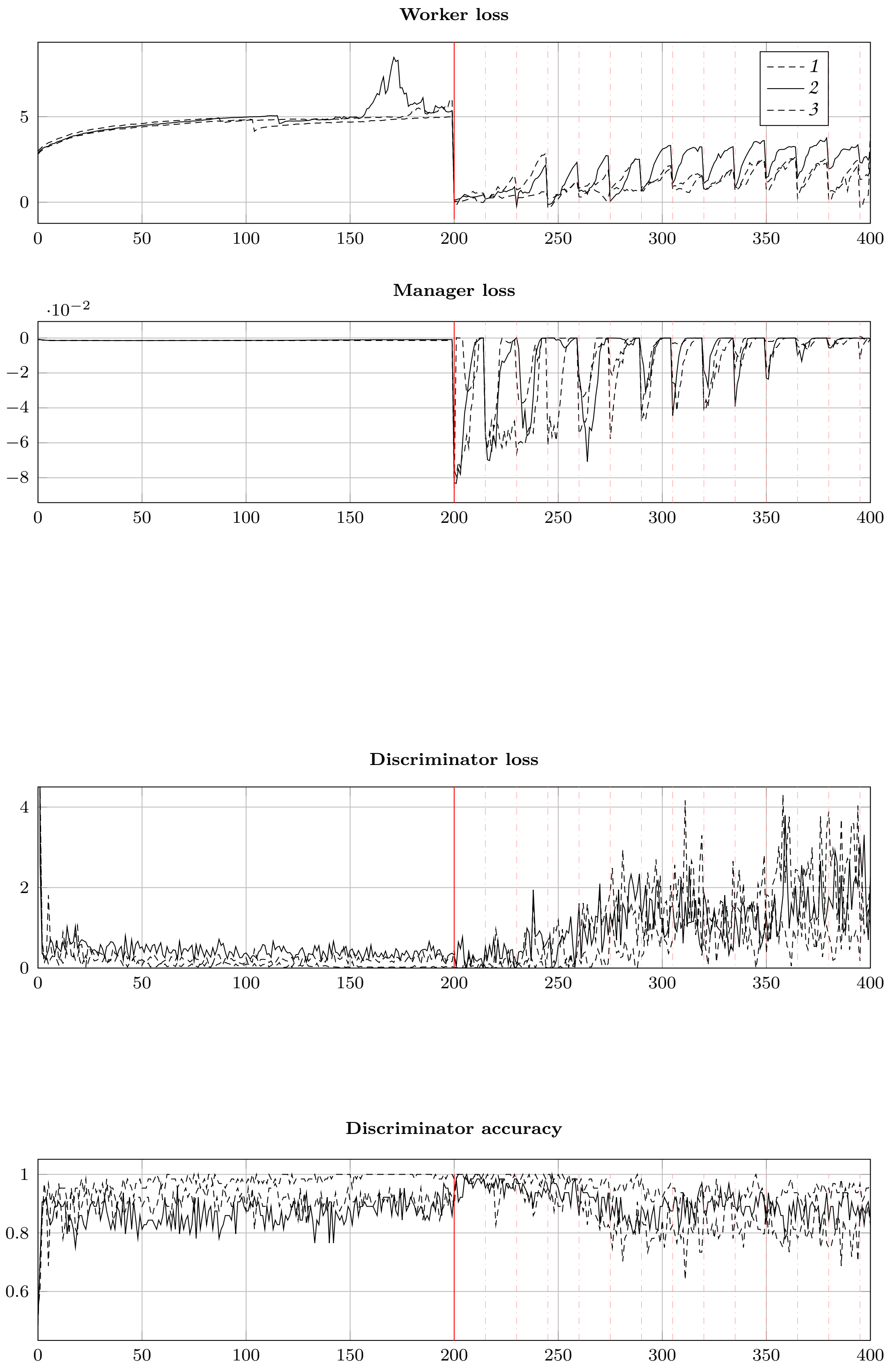


Figure A.17: Losses and accuracy for News Summary validation data.

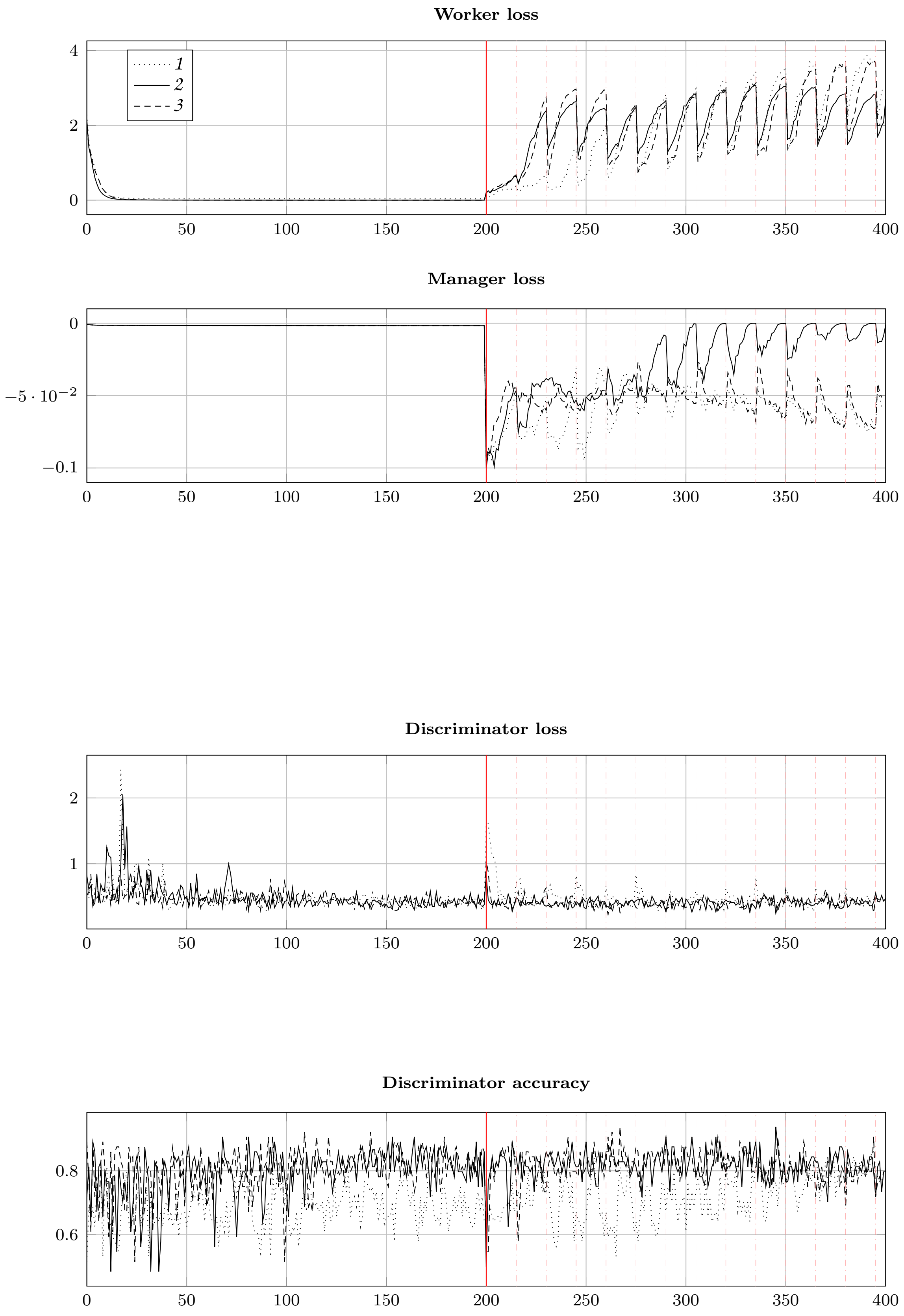


Figure A.18: Losses and accuracy for News Summary training data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.

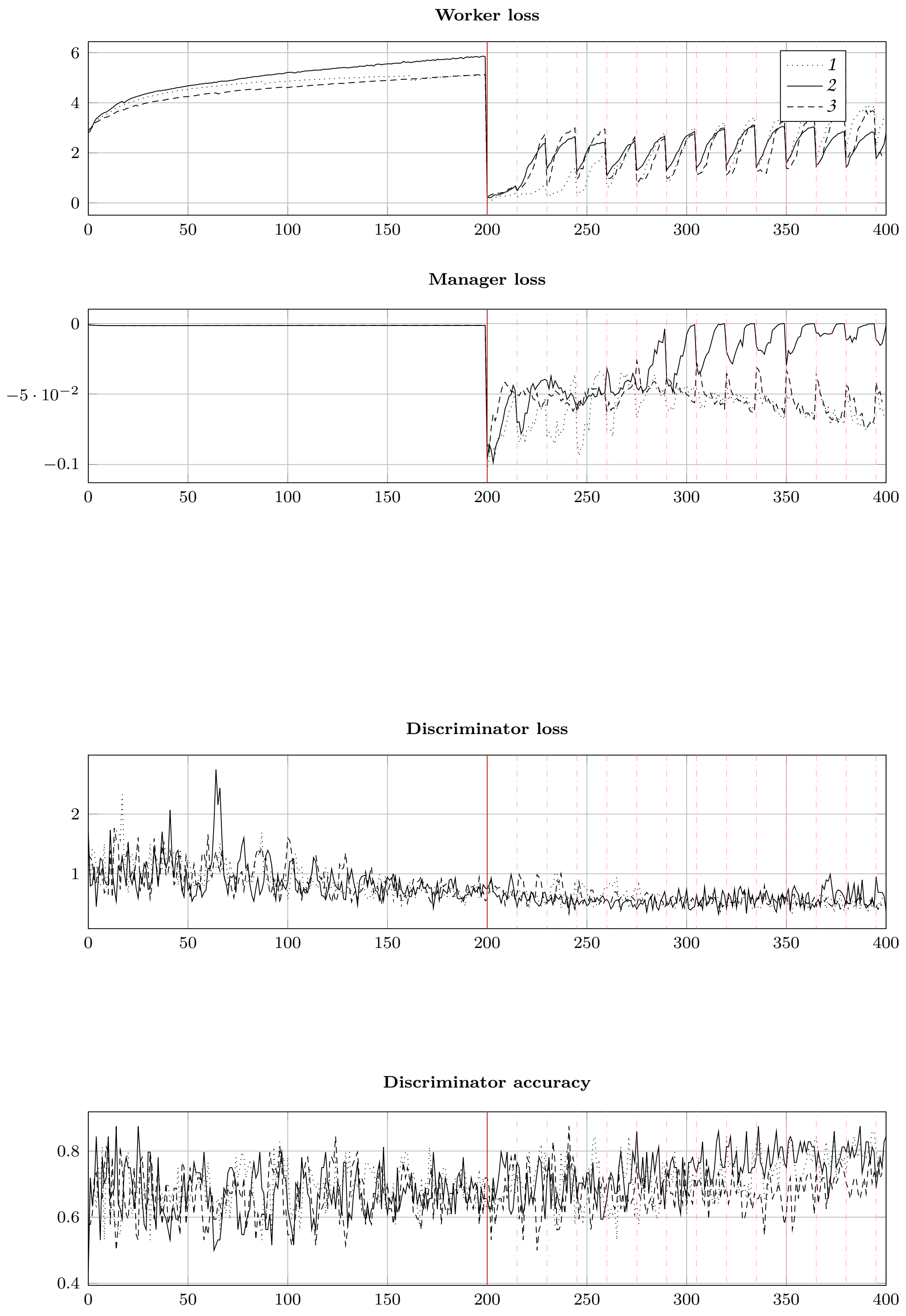


Figure A.19: Losses and accuracy for News Summary validation data, where the discriminator has also received documents with falsely paired summaries in addition to the generated summaries.