People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of 20 août 1955 - Skikda

Faculty of Sciences

Department of Computer Science

Master's Thesis

Arabic Abstractive Text Summarization via Supervised Transfer and Deep Reinforcement Learning

Submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

Specialization: **Artificial Intelligence**

By

Abdelbasset Djamai

Thesis committee:

Dr. Mawloud Mosbah	Chair
Dr. Khaira Tazir	Examiner
Dr. Fateh Bougamouza	Supervisor

July, 2024

This page was intentionally left blank.

Dedication

To my parents, for nurturing curiosity and inspiring perseverance. This journey began with your love and guidance.

Acknowledgements

I owe my deepest gratitude to my supervisor, **Dr. Fateh Bougamouza**, whose support, encouragement, and kindness have been a constant source of motivation. I do appreciate the liberty I was given to pursue my own research ideas while providing invaluable direction and feedback.

I am honored by the distinguished jury members who have agreed to review and judge this thesis. I thank each of the members for their time and valuable insights throughout this process.

I also extend my thanks to the instructors of the Computer Science department for all of their mentorship throughout my journey there.

Special and heartfelt gratitude goes to my parents, whose unwavering encouragement and support have been the bedrock of my journey. Their love and guidance have been my all-time companions, and their belief in me has been a source of strength.

To my friends and colleagues, thank you for the discussions and emotional sustenance that helped me along the way.

Abstract

The rapid growth of digital content has created a pressing need for efficient text summarization techniques, particularly for languages with complex features like Arabic. Although progress in Arabic abstractive text summarization has been limited compared to languages like English, recent advancements leveraging transfer learning with pretrained transformer-based language models (PTLMs) have shown promise. These approaches have primarily relied on supervised learning techniques for finetuning, which typically focus on maximizing next-word prediction likelihood rather than summary quality. They also suffer from inconsistencies between training and inference conditions, as well as exposure bias. Notably, reinforcement learning (RL), which offers potential solutions to these issues, especially its ability to directly optimize non-differentiable objectives, remains largely unexplored in Arabic ATS. This thesis proposes a novel approach combining transfer learning and RL to address these limitations and advance Arabic ATS. We introduce a novel two-phase framework: (1) supervised finetuning (SFT) of a PTLM, followed by (2) RL-based optimization via Proximal Policy Optimization. Our approach is evaluated on the XL-Sum and AraSum datasets, using reward functions derived from different automatic evaluation metrics and textual entailment. Experimental results demonstrate that our RL-finetuned models outperform our supervised baseline across multiple metrics, while demonstrating enhanced factual consistency. Comparison with prior work shows that our models achieve new state-of-the-art performance in terms of BERTScore, while achieving competitive and more balanced ROUGE scores on both datasets. Moreover, a test-only transfer evaluation on a new dataset reveals that our RL-optimized models exhibit superior generalization capabilities compared to the supervised baseline. We also conduct ablation studies to analyze the contributions of some critical components in our approach.

Keywords: Arabic abstractive text summarization, automatic text summarization, deep reinforcement learning, transfer learning, pretrained models, transformers.

RÉSUMÉ

La croissance rapide du contenu numérique a créé un besoin pressant de techniques efficaces de résumé de texte, en particulier pour les langues aux caractéristiques complexes comme l'arabe. Bien que les progrès dans le résumé de texte abstractif en arabe aient été limités par rapport aux langues comme l'anglais, de récentes avancées exploitant l'apprentissage par transfert avec des modèles de langage transformers pré-entraînés (MLTP) se sont avérées prometteuses. Ces approches se sont principalement appuyées sur des techniques d'apprentissage supervisé pour l'ajustement fin, qui se concentrent généralement sur la maximisation de la probabilité de prédiction du mot suivant plutôt que sur la qualité du résumé. Elles souffrent également d'incohérences entre les conditions d'entraînement et d'inférence, ainsi que de biais d'exposition. Notamment, l'apprentissage par renforcement (AR), qui offre des solutions potentielles à ces problèmes, en particulier sa capacité à optimiser directement des objectifs non différentiables, reste largement inexploré dans le résumé de texte abstractif en arabe. Cette thèse propose une approche novatrice combinant l'apprentissage par transfert et l'AR pour répondre à ces limitations et faire progresser le résumé de texte abstractif en arabe. Nous introduisons un nouveau cadre en deux phases : (1) l'ajustement fin supervisé d'un MLTP, suivi par (2) une optimisation basée sur l'AR via l'algorithme Proximal Policy Optimization. Notre approche est évaluée sur les ensembles de données XL-Sum et AraSum, en utilisant des fonctions de récompense dérivées de différentes métriques d'évaluation automatiques et d'implications textuelles. Les résultats expérimentaux démontrent que nos modèles affinés par AR surpassent notre base de référence supervisée sur plusieurs métriques, tout en démontrant une meilleure cohérence factuelle. Les comparaisons avec les travaux antérieurs montrent que nos modèles atteignent de nouvelles performances de pointe en termes de BERTScore, tout en obtenant des scores ROUGE compétitifs et plus équilibrés sur les deux ensembles de données. De plus, une évaluation de transfert uniquement en test sur un nouvel ensemble de données révèle que nos modèles optimisés par AR présentent des capacités de généralisation supérieures par rapport au modèle supervisé. Nous menons également des études d'ablation pour analyser les contributions de certaines composantes critiques de notre approche.

Mots clés: Résumé de texte abstractif en arabe, résumé automatique, apprentissage par renforcement profond, apprentissage par transfert, modèles pré-entraînés, transformers.

ملخص

لقد أدى النمو السريع للمحتوى الرقمي إلى خلق حاجة ملحة لتقنيات تلخيص النصوص بكفاءة، خاصة للغات ذات السمات المعقدة مثل العربية. وعلى الرغم من أن التقدم في مجال التلخيص التجريدي للنص العربي كان محدودًا مقارنة بلغات مثل الإنجليزية، إلا أن التطورات الأخيرة التي تستفيد من التعلم بالنقل باستخدام نماذج اللغة المعتمدة على المحولات (الترانسفورمرز) المدربة مسبقًا (PTLMs) قد أظهرت إمكانات واعدة. اعتمدت هذه الأساليب بشكل أساسي على تقنيات التعلم الخاضع للإشراف للضبط الدقيق، والتي تركز عادة على تعظيم احتمالية التنبؤ بالكلمة التالية بدلاً من جودة الملخص. كما أنها تعاني من عدم الاتساق بين ظروف التدريب والاستدلال، بالإضافة إلى تحيز التعرض. ومن الجدير بالذكر أن التعلم المعزز (RL)، الذي يقدم حلولاً محتملة لهذه المشكلات، وخاصة قدرته على تحسين الأهداف غير القابلة للتفاضل بشكل مباشر، لا يزال غير مستكشف إلى حد كبير في مجال التلخيص التجريدي للنص العربي. تقترح هذه الأطروحة نهجًا جديدًا يجمع بين التعلم بالنقل والتعلم المعزز لمعالجة هذه القيود وتطوير التلخيص التجريدي للنص العربي. نقدم إطار عمل جديد من مرحلتين: (1) الضبط الدقيق الخاضع للإشراف (SFT) لنموذج لغة مدرب مسبقًا، يليه (2) التحسين القائم على التعلم المعزز عبر تحسين السياسة التقريبي (Proximal Policy Optimization). يتم تقييم نهجنا على مجموعتي البيانات XL-Sum وAraSumg، باستخدام دوال المكافأة المشتقة من مقاييس التقييم التلقائي المختلفة والاستلزام النصي. تُظهر النتائج التجريبية أن نماذجنا المدربة بالتعلم المعزز تتفوق على النموذج المرجعي الخاضع للإشراف عبر مقاييس متعددة، مع إظهار اتساق واقعي محسن. تتجاوز نتائجنا تلك الخاصة بالأعمال السابقة كلها على مقياس BERTScore، مع تحقيق قيم ROUGE تنافسية وأكثر توازنًا على كلتا مجموعتي البيانات. علاوة على ذلك، يكشف تقييم النقل للاختبار فقط على مجموعة بيانات جديدة أن نماذجنا المحسنة بالتعلم المعزز تُظهر قدرات تعميم متفوقة مقارنة بالنموذج المرجعي الخاضع للإشراف. كما نجري دراسات استئصال لتحليل مساهمات بعض المكونات المهمة في نهجنا.

كلمات مفتاحية: التلخيص التجريدي للنص باللغة العربية، التلخيص الآلي للنص، التعلم المعزز العميق، التعلم بالنقل، النماذج المدربة مسبقا، المحولات.

CONTENTS

This page was intentionally left blank.

List of Figures

List of Tables

This page was intentionally left blank.

1

INTRODUCTION

1.1 Context & Background

In recent years, the exponential growth of digital content has transformed how we consume information. The proliferation of digital content across multiple platforms has led to an unprecedented surge in the volume of textual data produced daily. From news articles and scientific publications to social media posts and corporate reports, the sheer magnitude of available information has become overwhelming for human consumption. This information overload presents a significant challenge to individuals and organizations alike, as they struggle to efficiently extract relevant and meaningful insights from vast amounts of text.

To address this challenge, the field of automatic text summarization has emerged within the broader domain of natural language processing (NLP). By condensing lengthy texts into shorter, more digestible formats, summarization systems serve as invaluable tools in various applications across multiple industries and sectors. Effective summarization not only saves time and cognitive resources but also enhances decision-making processes by providing clear, concise, and relevant information.

Text summarization approaches can be broadly divided into two categories: extractive and abstractive (Nenkova, 2011). Extractive methods select and arrange existing sentences from the source text, while abstractive techniques generate new sentences that capture the essence of the content. Abstractive summarization has the potential to produce more fluent, contextually rich, and innovative summaries that are not constrained by the exact wording of the original text. This approach, mimicking human summarization more closely, requires a deeper semantic understanding and the ability to paraphrase and synthesize information.

As we explore the field of abstractive summarization, it is important to consider the unique

challenges and opportunities presented by different languages and linguistic contexts. In this regard, the Arabic language stands out as an interesting subject for research in abstractive summarization.

Arabic, the fifth most spoken language globally with over 420 million speakers, presents unique challenges for NLP tasks. The language's rich morphological structure, diverse dialectal variations, and right-to-left script pose many complexities worthy of scientific exploration. These linguistic features, combined with the growing volume of Arabic digital content, highlight the importance of developing effective abstractive summarization tools adapted to the Arabic language.

1.2 Motivation

Despite progress in abstractive text summarization (ATS) for languages like English, Arabic abstractive summarization remains relatively less explored due to its unique linguistic features and challenging nature (Bani-Almarjeh & Kurdy, 2023). Nevertheless, there have recently been some notable advancements in Arabic ATS, mainly driven by neural sequence-to-sequence models, and, more recently, those leveraging transfer learning with pretrained transformer-based language models (PTLMs).

However, these approaches, which often use purely supervised techniques, face limitations. They typically focus on maximizing next-word prediction likelihood rather than overall summary quality, risking overfitting to training data patterns. Additionally, they suffer from exposure bias, where discrepancies between training and inference can lead to error accumulation in generated summaries (Uc-Cetina et al., 2023).

Reinforcement learning (RL), a machine learning technique inspired by behavioral psychology, emerges as a natural solution to address these issues and potentially enhance Arabic abstractive summarization. RL offers several key advantages that make it particularly suitable for this task (Alomari et al., 2022), including:

- Optimization of non-differentiable objectives: RL allows for the direct optimization of evaluation metrics like BERTScore, which are not differentiable and thus cannot be used as loss functions in traditional supervised learning approaches.
- Long-term reward consideration: Unlike traditional methods that make decisions based on immediate next-word predictions, RL enables models to consider the quality of the entire generated summary, leading to more coherent and contextually appropriate outputs.
- Mitigation of exposure bias: By allowing the model to learn from its own generated sequences during training, RL can help reduce the inconsistency between training and inference.
- Exploration of diverse summarization strategies: RL's exploration mechanisms can help the model discover novel summary styles and structures.

Despite its benefits, RL remains largely unexplored in Arabic ATS. This thesis aims to bridge this gap by investigating the applicability of RL in combination with supervised transfer techniques for the task of Arabic abstractive text summarization. By leveraging RL's strengths with the linguistic knowledge captured by PTLMs via transfer learning, we seek to advance the state of Arabic ATS and address current limitations in the field. As far as we are aware, such a combination has not been explored before in Arabic ATS.

1.3 Objective & Contributions

The main objective of this thesis is to explore reinforcement learning as a finetuning paradigm on top of transfer learning for the task of Arabic abstractive text summarization. To achieve this, we propose a two-phase approach:

1. Supervised finetuning (SFT) of pretrained transformer-based language models.
2. Application of deep reinforcement learning to further optimize the supervised finetuned model.

Our contributions can be distilled in the following points:

- ◇ We propose a novel two-phase framework for Arabic ATS: (1) supervised finetuning and (2) reinforcement learning-based finetuning. The first phase leverages PTLMs through finetuning with supervised learning. The second phase further optimizes the supervised model via RL using PPO.
- ◇ We perform an extensive evaluation of our proposed approach on two benchmark datasets: XL-Sum and AraSum. We use different reward functions within our RL framework derived from automatic evaluation metrics, including ROUGE, METEOR, and BERTScore. We also experiment with textual entailment as a reward signal to encourage factual consistency. Combinations of rewards were also explored.
- ◇ We show that the models trained using our proposed method outperform the supervised baseline on multiple evaluation metrics and scenarios. Moreover, we show that our approach leads to improved faithfulness, as measured by a natural language inference (NLI) classifier. Within our knowledge, this is the first work utilizing NLI as a factual accuracy metric in Arabic ATS research.
- ◇ Comparison with prior work shows that our RL-finetuned models surpass all existing methods on BERTScore, achieving a new state-of-the-art on the XL-Sum dataset. Our models also demonstrate competitive and even more balanced performance in terms of ROUGE, on both XL-Sum and AraSum.
- ◇ We publicly release our models to the research community.

1.4 Thesis Organization

Besides the Introduction, this thesis is structured into the following chapters:

- **Chapter 2—Background:** Provides essential context on the most pertinent concepts to our thesis topic. It covers deep learning in NLP, including state-of-the-art neural methods (sequence-to-sequence models, attention, transformer architecture), and their fundamentals. It provides an overview of RL and the PPO algorithm. It also introduces the domain of automatic text summarization and sheds light on the Arabic language and its challenges.
- **Chapter 3—Related Work:** Reviews relevant literature on Arabic ATS, with particular emphasis on techniques using RNN-based and transformer-based methods. It also identifies existing research gaps and introduces our proposed approach that aims to address them.
- **Chapter 4—Methods & Experiments:** Outlines our proposed hybrid approach. It provides a thorough formal description of the framework along with a detailed experimental design.
- **Chapter 5—Results & Discussion:** Presents and discusses our experimental results, comparing our models' performance against baselines and state-of-the-art methods. An ablative analysis is also covered.
- **Chapter 6—Conclusion:** Concludes the thesis by summarizing the key contributions, highlighting the novel aspects of our approach and its impact on Arabic abstractive summarization. It also proposes promising research directions for future work.

2

BACKGROUND

This chapter provides an overview of relevant concepts related to the thesis topic. This includes a comprehensive introduction to deep learning for natural language processing (NLP), starting with an overview of the NLP domain and introducing language models. It also covers the fundamentals of deep learning, from the basics to more advanced neural architectures.

2.1 Deep Learning in Natural Language Processing

2.1.1 Natural Language Processing

Natural Language Processing (NLP) stands at the intersection of linguistics, computer science, and artificial intelligence, aiming to bridge the gap between human communication and machine understanding (Khurana et al., 2023). This section provides an overview of key NLP concepts, techniques, and models that form the backbone of modern language technologies.

2.1.1.1 Overview of NLP

NLP encompasses a wide range of tasks and applications, from basic text processing to complex language understanding and generation. The field has evolved dramatically, transitioning from rule-based systems to data-driven approaches, and now to advanced neural network architectures (Khurana et al., 2023). These foundational tasks support higher-level applications such as machine translation, question answering, and text summarization.

2.1.1.2 Language Models

Language models (LMs) are fundamental to many NLP tasks, providing a probability distribution over sequences of words (Bengio et al., 2003). Traditional n-gram models estimate the probability of a word given its history:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-n+1}^{k-1}) \tag{2.1}$$

Neural language models have significantly improved upon n-gram models. A basic recurrent neural network language model (RNNLM) computes the probability of a sequence as:

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{t-1}) = \prod_{t=1}^{T} P(w_t | h_t) \tag{2.2}$$

where h_t is the hidden state at time t.

More advanced models like transformer-based LMs (e.g., AraT5 (Nagoudi et al., 2022)) have pushed the boundaries of language modeling, enabling the generation of coherent and contextually appropriate text across a wide range of domains and tasks (Naveed et al., 2024).

2.1.2 Machine Learning and Deep Learning

2.1.2.1 Machine Learning

Machine Learning (ML) is a field of study that focuses on algorithms and statistical models that computer systems use to perform tasks without explicit instructions, relying on patterns and inference instead (Alzubi et al., 2018). It is a method of data analysis that automates analytical model building, allowing systems to learn from data, identify patterns, and make decisions with minimal human intervention.

The main paradigms of Machine Learning are:

- **Supervised Learning:** The algorithm learns from labeled training data, trying to predict outcomes for unseen data. Common tasks include classification and regression.
- **Unsupervised Learning:** The algorithm works on unlabeled data, attempting to find structure or patterns. Typical applications include clustering and dimensionality reduction.
- **Semi-Supervised Learning:** This approach combines a small amount of labeled data with a large amount of unlabeled data during training. It's particularly useful when labeling data is expensive or time-consuming.

- **Reinforcement Learning:** The algorithm learns to make decisions by interacting with an environment. It receives feedback in the form of rewards or penalties to improve its performance over time.

2.1.2.2 Deep Learning

Deep Learning (DL) is a subset of Machine Learning that uses artificial neural networks with multiple layers. These deep neural networks are capable of learning complex patterns in large amounts of data (Goodfellow et al., 2016). Deep Learning models progressively learn more abstract and complex features of the data through successive layers, allowing them to perform human-like tasks with remarkable accuracy.

Deep Learning has significantly advanced fields such as computer vision, natural language processing, and speech recognition. Its power lies in its ability to automatically discover the representations needed for detection or classification from raw data, reducing the need for manual feature engineering (LeCun et al., 2015; Goodfellow et al., 2016).

2.1.3 Artificial Neural Networks

2.1.3.1 General Concept and Definition

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks. They are designed to recognize patterns and learn from data. Formally, an ANN can be defined as a directed graph consisting of nodes (neurons) and edges (connections) with associated weights (Goodfellow et al., 2016).

Mathematically, a single neuron can be represented as:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.3}$$

where x_i are inputs, w_i are weights, b is a bias term, f is an activation function, and y is the output.

2.1.3.2 Perceptron

The perceptron, introduced by Rosenblatt (1958), is the simplest form of a neural network (illustrated in Figure 2.1). It consists of a single neuron with binary output:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The perceptron learning rule for updating weights is:

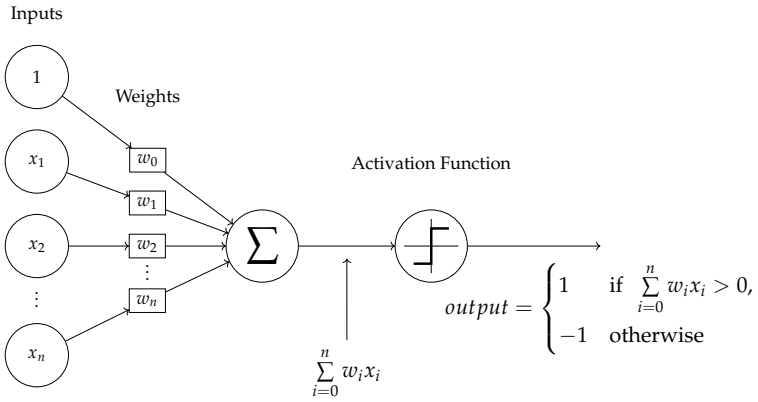


Figure 2.1: The schema of the perceptron.

$$w_i \leftarrow w_i + \eta(y^* - y)x_i \tag{2.5}$$

where η is the learning rate, y^* is the true label, and y is the predicted output.

2.1.3.3 Multilayer Perceptron (MLP)

An MLP (Figure 2.2) consists of multiple layers of neurons: an input layer, one or more hidden layers, and an output layer. Each neuron in one layer is connected to every neuron in the next layer.

For a neuron j in layer l:

$$a_j^l = f(\sum_{i=1}^{n_{l-1}} w_{ij}^l a_i^{l-1} + b_j^l) \tag{2.6}$$

where a_j^l is the activation of neuron j in layer l, w_{ij}^l is the weight connecting neuron i in layer $l-1$ to neuron j in layer l, and b_j^l is the bias of neuron j in layer l.

2.1.3.4 Activation Functions

Activation functions introduce non-linearity into the network, allowing it to learn complex patterns. Common activation functions include:

- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic Tangent (tanh): $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$

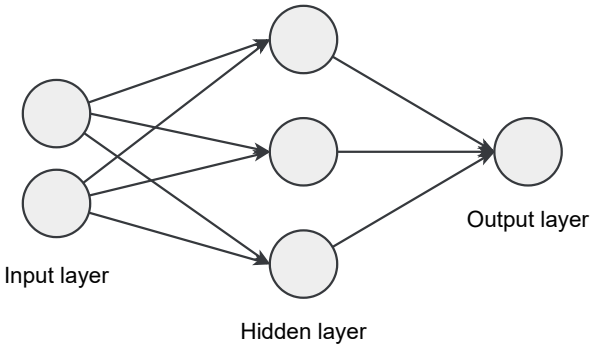


Figure 2.2: The MLP architecture.

2.1.3.5 Loss Functions

Loss functions measure the discrepancy between predicted and true values. Common loss functions include:

- Mean Squared Error (MSE): $L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Cross-Entropy: $L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$

where y_i are true values and $\hat{y}_i$ are predicted values.

2.1.3.6 Backpropagation

Backpropagation is the algorithm used to train ANNs. It computes the gradient of the loss function with respect to each weight by the chain rule, iterating backwards from the output layer to the input layer. ANNs are typically trained using gradient descent or one of its variants (Goodfellow et al., 2016).

For a weight w_{ij}^l, the update rule is:

$$w_{ij}^l \leftarrow w_{ij}^l - \eta \frac{\partial L}{\partial w_{ij}^l} \tag{2.7}$$

where η is the learning rate and $\frac{\partial L}{\partial w_{ij}^l}$ is computed using the chain rule:

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} \tag{2.8}$$

Here, $z_j^l = \sum_{i=1}^{n_{l-1}} w_{ij}^l a_i^{l-1} + b_j^l$ is the weighted sum input to neuron j in layer l.

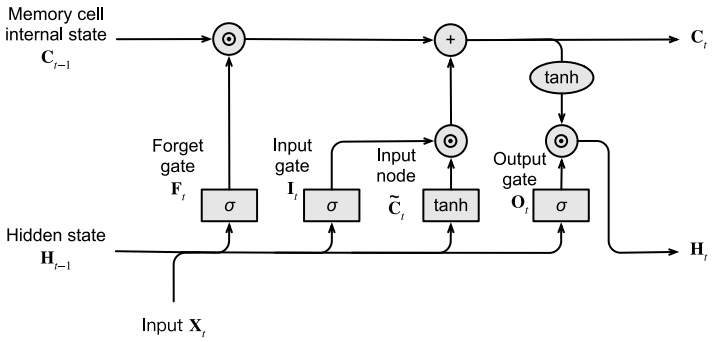


Figure 2.3: LSTM cell architecture.

The error term $\delta_j^l = \frac{\partial L}{\partial z_j^l}$ is computed recursively:

$$\delta_j^l = \begin{cases} f'(z_j^l) \frac{\partial L}{\partial a_j^l} & \text{for output layer} \\ f'(z_j^l) \sum_k w_{jk}^{l+1} \delta_k^{l+1} & \text{for hidden layers} \end{cases} \tag{2.9}$$

This recursive computation of error terms allows efficient calculation of gradients for all weights in the network.

2.1.4 Recurrent Neural Networks

Recurrent Neural Networks are neural networks designed to process sequential data by maintaining a hidden state that can capture information from previous time steps (Schmidt, 2019). The basic RNN architecture can be described by the following equations:

$$h_t = \tanh(W_{hx} x_t + W_{hh} h_{t-1} + b_h) \tag{2.10}$$

$$y_t = W_{yh} h_t + b_y \tag{2.11}$$

where x_t is the input at time t, h_t is the hidden state, y_t is the output, and W and b are learnable parameters.

While RNNs can theoretically capture long-term dependencies, they often struggle with this in practice due to the vanishing gradient problem during training.

2.1.4.1 Long Short-Term Memory (LSTM)

LSTMs (Hochreiter & Schmidhuber, 1997) address the limitations of standard RNNs by introducing a more complex cell structure with gating mechanisms. The LSTM architecture (see Figure 2.3) includes an input gate, a forget gate, an output gate, and a memory cell. The key equations are:

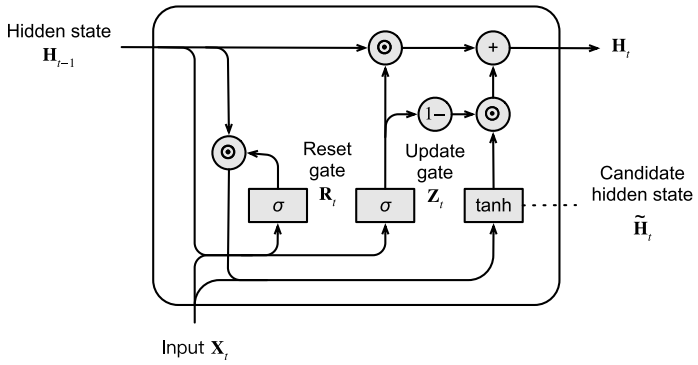


Figure 2.4: GRU cell architecture.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.12}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.13}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.14}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.15}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.16}$$

$$h_t = o_t * \tanh(C_t) \tag{2.17}$$

where σ is the sigmoid function, $*$ denotes element-wise multiplication, C_t is the cell state, and f_t, i_t, and o_t are the forget, input, and output gates respectively.

2.1.4.2 Gated Recurrent Unit (GRU)

GRUs are a simplified version of LSTMs, combining the forget and input gates into a single "update gate" and merging the cell state and hidden state (Goodfellow et al., 2016), as depicted in Figure 2.4. The GRU is defined by the following equations:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2.18}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2.19}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{2.20}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.21}$$

where z_t is the update gate and r_t is the reset gate.

GRUs have fewer parameters than LSTMs, which can lead to faster training and better performance on smaller datasets (Chung et al., 2014).

2.1.5 Sequence-to-Sequence Models

Sequence-to-Sequence (Seq2Seq) models, introduced by Sutskever et al. (2014), have become a fundamental architecture in NLP for tasks involving variable-length input and output sequences. The core idea behind Seq2Seq models is to transform an input sequence into an output sequence of potentially different length. This is achieved through two main components: an encoder and a decoder. An example of that is presented in Figure 2.5.

The encoder processes the input sequence $\mathbf{X} = (x_1, \ldots, x_n)$ and compresses it into a fixed-dimensional context vector $\mathbf{c}$. Formally, the encoder can be described as a function:

$$\mathbf{c} = f_{\text{enc}}(\mathbf{X}) \tag{2.22}$$

where f_{enc} is typically implemented as a recurrent neural network, such as LSTM or GRU.

The decoder generates the output sequence $\mathbf{Y} = (y_1, \ldots, y_m)$ based on the context vector $\mathbf{c}$ produced by the encoder. At each time step t, the decoder computes:

$$p(y_t | y_1, \ldots, y_{t-1}, \mathbf{c}) = g(y_{t-1}, s_t, \mathbf{c}) \tag{2.23}$$

where s_t is the hidden state of the decoder at time t, and g is a nonlinear function that outputs the probability distribution over the output vocabulary.

The decoder's hidden state is updated recurrently:

$$s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}) \tag{2.24}$$

where f is typically another RNN.

Seq2Seq models are typically trained end-to-end to maximize the likelihood of the target sequence given the input sequence (Ranzato et al., 2016):

$$\max_{\theta} \sum_{(\mathbf{X},\mathbf{Y}) \in \mathcal{D}} \log p(\mathbf{Y} | \mathbf{X}; \theta) \tag{2.25}$$

where $\mathcal{D}$ is the training dataset and θ represents the model parameters.

2.1.6 Attention Mechanism

Building upon the Sequence-to-Sequence (Seq2Seq) models discussed in the previous section, the attention mechanism, initially introduced by Bahdanau et al. (2016), has emerged

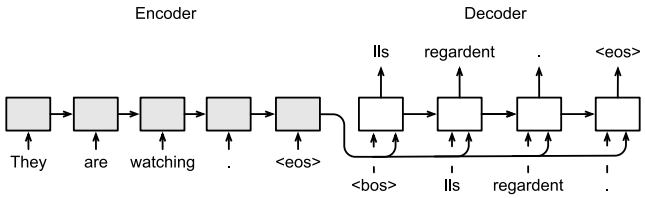


Figure 2.5: An example of an RNN-based encoder-decoder architecture for English-to-French translation.

as a critical component in many state-of-the-art natural language processing architectures. It addresses the limitations of traditional Seq2Seq models, which encode the entire input sequence into a fixed-length vector, often failing to capture long-range dependencies effectively. The mechanism computes a weighted sum of the input sequence, where the weights are determined by the relevance of each input element to the current output being computed. This allows the model to selectively focus on the most relevant parts of the input sequence when generating the output, alleviating the burden of encoding all the information into a fixed-length vector.

Mathematically, the attention weights α_{ij} are calculated as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.26}$$

where e_{ij} is the relevance score between the current output element j and the input element i, and T_x is the length of the input sequence. The relevance score e_{ij} is typically computed using a compatibility function, such as dot product or a neural network.

The attended output vector c_j is then computed as the weighted sum of the input sequence:

$$c_j = \sum_{i=1}^{T_x} \alpha_{ij} h_i \tag{2.27}$$

where h_i is the hidden state of the input element i.

Over the years, various attention mechanisms have been proposed to address different challenges in NLP tasks. Table 2.1 summarizes the most important attention types, their key characteristics, and mathematical formulations (Weng, 2018).

2.1.7 Transformer

The transformer architecture, introduced by Vaswani et al. (2017), has profoundly reshaped NLP, significantly altering traditional approaches and establishing itself as the foundation

Table 2.1: Summary of the most popular attention types.

Attention mechanism	Score function
Content-based attention	$\text{score}(a, h_i) = \text{cosine}(a, h_i)$
Additive	$\text{score}(a, h_i) = v_i^\top \tanh(W_1 a + W_2 h_i)$
Location-base	$\alpha_{i,t} = \text{softmax}(W_a s_t)$
General	$\text{score}(a, h_i) = a^\top W_a h_i$
Dot-product	$\text{score}(a, h_i) = a^\top h_i$
Scaled dot-product	$\text{score}(a_i, h_i) = \frac{a_i^\top h_i}{\sqrt{d_k}}$

for state-of-the-art models in various language tasks (Wolf et al., 2020). At its core, the transformer is a sequence-to-sequence model that entirely relies on attention mechanisms, dispensing with recurrence and convolutions that were prevalent in previous architectures. The model's primary innovation lies in its ability to process all elements of a sequence in parallel, capturing long-range dependencies more effectively than its predecessors.

The transformer architecture (Figure 2.6) consists of an encoder and a decoder, each composed of a stack of identical layers. The encoder processes the input sequence, while the decoder generates the output sequence. Both components leverage self-attention mechanisms to compute representations of their respective inputs.

The encoder consists of N identical layers (typically 6 in the original paper) (Vaswani et al., 2017), each containing two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Each sub-layer is followed by layer normalization and employs a residual connection. Formally, if we denote the input to a layer as x, the output of the self-attention sub-layer can be expressed as:

$$\text{SubLayer1}(x) = \text{LayerNorm}(x + \text{MultiHeadAttention}(x)) \tag{2.28}$$

The output of the feed-forward sub-layer is then:

$$\text{SubLayer2}(x) = \text{LayerNorm}(x + \text{FFN}(x)) \tag{2.29}$$

where FFN is a position-wise feed-forward network consisting of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.30}$$

The decoder also consists of N identical layers, but each layer has an additional sub-layer

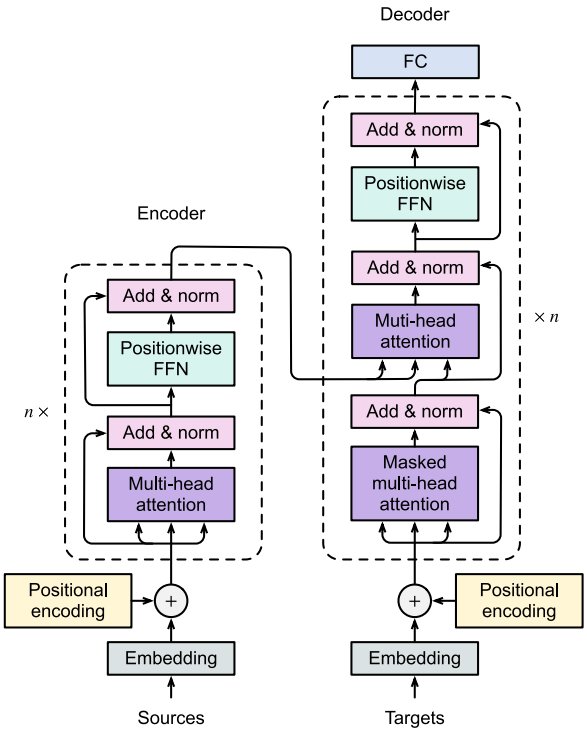


Figure 2.6: The original transformer architecture.

that performs multi-head attention over the encoder's output (Vaswani et al., 2017). The self-attention sub-layer in the decoder is modified to prevent positions from attending to subsequent positions, ensuring that predictions for position i can only depend on known outputs at positions less than i.

The key innovation in the transformer is the multi-head attention mechanism. The attention function can be described as mapping a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Specifically, the authors use scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.31}$$

where Q, K, and V are matrices representing queries, keys, and values respectively, and d_k is the dimension of the keys. The scaling factor $\sqrt{d_k}$ is introduced to counteract the effect of the dot product growing large in magnitude for large values of d_k, which could push the softmax function into regions with extremely small gradients.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. It is computed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.32}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.33}$$

Here, W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices. The authors use $h = 8$ parallel attention layers or heads.

Since the transformer contains no recurrence or convolution, it must inject information about the relative or absolute position of the tokens in the sequence (Vaswani et al., 2017). This is achieved through positional encodings that are added to the input embeddings. The authors use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{2.34}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{2.35}$$

where pos is the position and i is the dimension. This choice of positional encodings allows the model to easily learn to attend by relative positions, as for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

The transformer uses layer normalization, which normalizes the inputs across features, instead of batch normalization. This choice was made to facilitate training and improve generalization. The authors also employ residual connections around each sub-layer, which helps in training deeper models.

The transformer architecture has several advantages over previous sequence-to-sequence models (Vaswani et al., 2017). It allows for more parallelization during training, as all positions in the sequence can be processed simultaneously. It captures long-range dependencies more effectively due to the direct connections between all positions provided by the self-attention mechanism. The model is also more interpretable, as the attention weights provide insight into which parts of the input the model is focusing on for each output.

2.1.8 Transfer Learning & Pretrained Language Models

Transfer learning is a machine learning technique where knowledge gained from solving one problem is applied to a different but related problem (Ruder, 2019). The main idea behind transfer learning in NLP is to leverage general language understanding acquired from large-scale pretraining to improve performance on specific downstream tasks (Zhuang et al., 2020). This approach is based on two key principles:

1. **Feature reuse:** Lower layers of the model capture generic linguistic features that are broadly applicable across tasks.
2. **Task adaptation:** Higher layers of the model can be finetuned to specialize in specific NLP tasks.

In the context of NLP, transfer learning has become particularly powerful with the advent of Pretrained Language Models (PLMs), which have revolutionized NLP by leveraging large amounts of unlabeled text data. The process typically involves two main stages (Naveed et al., 2024):

1. **Pretraining:** The model is trained on a large corpus of text using self-supervised learning objectives. Common pretraining tasks include masked language modeling and next sentence prediction.
2. **Finetuning:** The pretrained model is then adapted to specific downstream tasks using task-specific labeled data. This process allows the model to transfer its general language understanding to particular NLP applications, using relatively small amounts of data.

2.1.8.1 Types of Pretrained Language Models

Pretrained language models can be categorized based on their architecture and the way they process input and generate output (Naveed et al., 2024):

- **Encoder-only models:** These models process the entire input sequence to create contextual representations. They are typically used for tasks that require understanding of the input, such as text classification or named entity recognition. A prominent example of this type is BERT (Devlin et al., 2019).
- **Encoder-Decoder models:** These models have separate components for processing the input (encoder) and generating the output (decoder). They are well-suited for sequence-to-sequence tasks like machine translation or text summarization. T5 (Raffel et al., 2020) and its Arabic version AraT5 (Nagoudi et al., 2022) are examples of encoder-decoder PLMs.
- **Decoder-only models:** These models generate text autoregressively, predicting one token at a time. They are often used for text generation tasks and can be adapted for various NLP applications. The GPT family is a popular example of this model type (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020).

2.2 Reinforcement Learning

Reinforcement learning (RL) is a computational approach to learning whereby an agent interacts with an environment to achieve a goal. The agent makes decisions by taking actions

in different states of the environment, aiming to maximize cumulative rewards (Sutton & Barto, 2020), as shown in Figure 2.7. Unlike supervised learning, where the model is trained on a given dataset, RL relies on the agent learning from the consequences of its actions, making it suitable for tasks where outcomes are not immediately apparent.

2.2.1 Markov Decision Processes (MDPs)

At the core of RL is the concept of a Markov Decision Process (MDP), which provides a formal framework for modeling decision-making scenarios where outcomes are partly random and partly under the control of the agent (Sutton & Barto, 2020).

2.2.1.1 Components of an MDP

An MDP is defined by the tuple (S, A, P, R, γ):

- **States (S)**: A finite set of states representing different situations in the environment.
- **Actions (A)**: A finite set of actions available to the agent.
- **State Transition Probabilities ($P(s'|s,a)$)**: The probability of transitioning to state s' from state s when action a is taken.
- **Reward Function ($R(s,a)$)**: The reward received after taking action a in state s.
- **Discount Factor (γ)**: A factor $\gamma \in [0,1)$ representing the importance of future rewards.

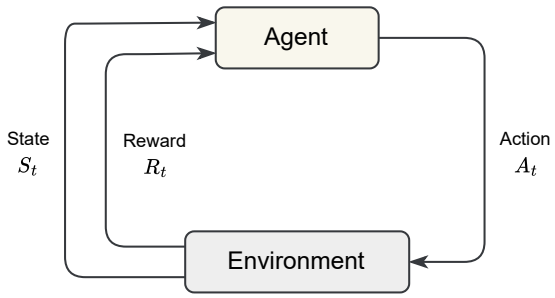


Figure 2.7: RL block diagram.

2.2.1.2 Value Functions

The value function estimates the expected cumulative reward from a given state, which is central to finding the optimal policy (Lu et al., 2023).

State Value Function ($V(s)$) The state value function $V^\pi(s)$ under policy π is defined as:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right] \tag{2.36}$$

Action Value Function ($Q(s,a)$) The action value function $Q^\pi(s,a)$ under policy π is defined as:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right] \tag{2.37}$$

Bellman Equations For an optimal policy π^*, the value functions satisfy the Bellman optimality equations:

$$V^*(s) = \max_a \mathbb{E} \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right] \tag{2.38}$$

$$Q^*(s,a) = \mathbb{E} \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a') \right] \tag{2.39}$$

2.2.2 Reinforcement Learning Paradigms

RL algorithms can be broadly categorized based on how they learn and utilize the value functions and policies (Dong et al., 2020). Due to the extensive nature of model RL, we focus on the main learning paradigms without going into specific algorithms. For a comprehensive and in-depth exploration of RL, the reader is referred to the seminal work of Sutton & Barto (2020).

2.2.2.1 Model-Free vs. Model-Based RL

Model-Free RL In model-free RL, the agent learns to act directly from interactions with the environment without building a model of it (AlMahamid & Grolinger, 2021).

Policy-Based Methods Policy-based methods directly optimize the policy $\pi(a|s)$ (AlMahamid & Grolinger, 2021). The goal is to maximize the expected return $J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$. The policy is updated using gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{2.40}$$

Value-Based Methods Value-based methods estimate the value function $V(s)$ or the action-value function $Q(s,a)$ and derive the policy from these estimates (Lu et al., 2023). The action-value function is updated as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2.41}$$

Actor-Critic Methods Actor-critic methods combine policy-based and value-based approaches (AlMahamid & Grolinger, 2021). The actor updates the policy, and the critic updates the value function. The updates are:

$$\delta = R(s,a) + \gamma V_w(s') - V_w(s) \tag{2.42}$$

$$w \leftarrow w + \alpha_c \delta \nabla_w V_w(s) \tag{2.43}$$

$$\theta \leftarrow \theta + \alpha_a \delta \nabla_\theta \log \pi_\theta(a|s) \tag{2.44}$$

Model-Based RL In model-based RL, the agent builds a model of the environment, including state transition probabilities and reward functions, and uses this model for planning. The agent predicts future states and rewards using the learned model and optimizes the policy through planning algorithms like dynamic programming or Monte Carlo tree search (Sutton & Barto, 2020).

2.2.3 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a state-of-the-art policy gradient algorithm introduced by Schulman et al. (2017). It has gained significant popularity due to its simplicity, robustness, and sample efficiency. PPO addresses the issue of catastrophic performance collapse often observed in traditional policy gradient methods by carefully constraining the updates to the policy, ensuring stability and reliable performance improvements.

The core idea behind PPO is to optimize a surrogate objective function that trades off policy improvement and policy stability (Schulman et al., 2017). The surrogate objective is designed to approximate the true performance objective while preventing the new policy from deviating too much from the old policy.

Let π_θ be the current policy parameterized by θ, and $\pi_{\theta_{old}}$ be the old policy. The PPO objective is defined as:

$$J^{PPO}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2.45}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is an estimator of the advantage function, and ϵ is a clipping hyperparameter.

The advantage function A_t is defined as:

$$A_t = Q^{\pi_{\theta_{old}}}(s_t, a_t) - V^{\pi_{\theta_{old}}}(s_t) \tag{2.46}$$

and can be estimated using various techniques, such as Monte Carlo returns, temporal difference methods, or learned value functions.

The 'clip' function in the PPO objective clips the probability ratio within the interval $[1-\epsilon, 1+\epsilon]$. This clipping operation ensures stability and mitigates the risk of catastrophic performance collapse.

The PPO objective can be interpreted as a lower bound on the true performance objective, with the 'min' operation ensuring that the surrogate objective is a pessimistic estimate of the true objective. Optimizing this surrogate objective encourages policy updates that improve performance while maintaining a trust region around the old policy.

In practice, PPO is often implemented using a clipped surrogate objective for the value function as well:

$$J^{PPO}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) + \lambda \min \left(V^{\pi_\theta}(s_t) - c_t, \text{clip}(V^{\pi_\theta}(s_t) - c_t, -\epsilon, \epsilon) \right) \right. \tag{2.47}$$

where c_t is a constant baseline, and λ is a coefficient that determines the relative importance of the value function term.

PPO can be combined with various policy architectures, such as deep neural networks, and can be applied to continuous and discrete action spaces. It has been successfully used in various RL tasks, including natural language generation tasks (Stiennon et al., 2020; Gunasekara et al., 2021; Ouyang et al., 2022).

In the context of NLP tasks like text summarization, PPO can be used to finetune pretrained language models or to directly optimize task-specific objectives (Cao et al., 2023). The policy network in this case would be a language model that generates text, and the reward signal could be based on evaluation metrics or human feedback.

One advantage of PPO is its sample efficiency compared to other policy gradient methods. By constraining the updates to the policy, PPO can make effective use of limited data and achieve stable performance improvements with fewer iterations. However, PPO still suffers from some limitations inherent to policy gradient methods, such as high variance in the gradient estimates and the potential for reward hacking or misalignment with the true objective. Careful reward design and techniques like reward shaping may be necessary to mitigate these issues (Zheng et al., 2023).

2.3 Automatic Text Summarization

Automatic text summarization is a field within natural language processing (NLP) that focuses on developing computational methods to generate shortened and concise versions of lengthy text documents (Nenkova, 2011). The primary goal is to distill the essential information from the source material while preserving the most salient details and overall meaning. This field has significant practical applications, as it enables efficient information access and consumption in various domains, such as news articles, scientific literature, legal documents, and more (Khan et al., 2023).

2.3.1 Classification of Automatic Text Summarization Systems

ATS systems can be categorized based on numerous aspects, including the following (Khan et al., 2023):

- **Input text type**: ATS systems can handle various types of input texts, such as:

  - News articles
  - Scientific papers and research literature
  - Web pages and online content
  - Multimedia content (e.g., transcripts, subtitles)
- **Summarization approach**:
  - **Extractive**: Created by selecting and concatenating important sentences or phrases from the original text.
  - **Abstractive**: Generated by producing new sentences that capture the essence of the content.
- **Summary length**: ATS systems can produce summaries of varying lengths, ranging from:
  - Short one-sentence summaries
  - Multi-paragraph summaries

  The appropriate length depends on the application and user requirements.
- **Language**: ATS can be developed for different languages, each presenting unique challenges due to linguistic variations and requirements.
- **Single-document or multi-document summarization**:
  - **Single-document summarization**: Focuses on summarizing a single text.
  - **Multi-document summarization**: Involves combining information from multiple sources to create a comprehensive summary.
- **Domain-specific or general-purpose**:
  - **Domain-specific**: ATS systems tailored for specific domains, such as legal, medical, or financial texts.
  - **General-purpose**: ATS systems designed to be applicable to a wide range of text genres.
- **Query-based or generic summaries**:
  - **Query-based summarization**: Generates summaries focused on specific topics or questions.
  - **Generic summaries**: Aims to capture the overall gist of the text without a specific query.

2.3.2 Summarization Approaches

There exists two main approaches to automatic text summarization: extractive and abstractive. Each approach has its own advantages and challenges, and the choice depends on the specific requirements and constraints of the task.

2.3.2.1 Extractive Summarization

Extractive summarization involves selecting and concatenating the most important sentences or phrases from the original text to form the summary (Elsaid et al., 2022). This approach is relatively straightforward and does not require generating new text. The advantages of extractive summarization include:

- **Simplicity**: Extractive summarization is computationally less complex and easier to implement compared to abstractive methods.
- **Preservation of information**: Since the summary is composed of sentences from the original text, it preserves the original wording and factual information.
- **Coherence**: By selecting complete sentences, extractive summaries often maintain coherence and readability.

However, extractive summarization has some limitations (Khan et al., 2023):

- **Lack of abstraction**: Extractive summaries may lack abstraction and fail to capture the overall meaning or synthesize information from different parts of the text.
- **Redundancy**: Selected sentences may contain redundant or irrelevant information, resulting in verbose summaries.
- **Loss of context**: Extracted sentences may lose important context from their original surroundings, potentially altering their intended meaning.

2.3.2.2 Abstractive Summarization

The abstractive approach to text summarization takes a different approach than the extractive method. Instead of relying solely on the original text, abstractive summarization involves generating new sentences that capture the essence of the source content (Widyassari et al., 2022). This approach requires a deeper understanding of the text and the ability to paraphrase and synthesize information. Abstractive summarization techniques utilize natural language generation (NLG) models to paraphrase and rephrase information in a more concise and coherent manner.

The advantages of abstractive summarization include:

- **Conciseness**: Abstractive summaries can be more concise and focused, as they are not constrained by selecting complete sentences from the original text.
- **Abstraction and synthesis**: Abstractive methods can abstract and synthesize information from different parts of the text, providing a more coherent and comprehensive summary.
- **Natural language generation**: Abstractive summaries can produce fluent and natural-sounding sentences, mimicking human-written summaries.

Abstractive summarization also faces several challenges:

- **Complexity**: Abstractive summarization is a more complex task that requires advanced natural language processing techniques, such as semantic understanding, language generation, and content planning (Elsaid et al., 2022).
- **Factual consistency**: Generating new sentences increases the risk of introducing factual inconsistencies or hallucinations, where the summary includes information

not present in the original text (Roit et al., 2023).

- **Evaluation difficulty**: Evaluating the quality of abstractive summaries is more challenging than evaluating extractive summaries, as there is no direct mapping between the summary and the original text (Fabbri et al., 2021).

2.3.3 Arabic Language and its Challenges

Arabic is a Semitic language spoken by over 400 million people worldwide, making it one of the most widely spoken languages globally. It is the official language of 26 countries and the liturgical language of Islam. Arabic has a rich history dating back to the 6th century CE, with Classical Arabic being the language of the Quran and early Islamic literature (Al-Huri, 2016). Modern Standard Arabic serves as the standardized written form used in media, literature, and formal contexts across the Arab world. However, spoken Arabic varies widely, with numerous regional dialects that can differ significantly in vocabulary and pronunciation (Zitouni, 2014).

The Arabic script is written from right to left and consists of 28 consonants and three long vowels. Short vowels are typically omitted in writing but can be indicated by diacritical marks (see Table 2.2 for the most common diacritical marks). The script is known for its cursive style, with most letters connecting to the following letter within a word. Table 2.3 shows the standard Arabic alphabets.

Table 2.2: Most commonly used Arabic diacritical marks.

Diacritic	Name	Explanation
◌َ	Fatḥah (فَتْحَة)	A small diagonal line placed above a letter, indicating a short "a" sound.
◌ِ	Kasrah (كَسْرَة)	A small diagonal line placed below a letter, indicating a short "i" sound.
◌ُ	Ḍammah (ضَمَّة)	A small curl placed above a letter, indicating a short "u" sound.
◌ّ	Shaddah (شَدَّة)	A small "w"-shaped symbol above a letter, indicating the doubling of the consonant.
◌ْ	Sukūn (سُكُون)	A small circle placed above a letter, indicating the absence of a vowel (a consonant).
◌ً	Tanwīn Fatḥ (تَنْوِين فَتْح)	Two diagonal lines placed above a letter, indicating a nasalized "an" sound.
◌ٍ	Tanwīn Kasr (تَنْوِين كَسْر)	Two diagonal lines placed below a letter, indicating a nasalized "in" sound.
◌ٌ	Tanwīn Ḍamm (تَنْوِين ضَم)	Two curls placed above a letter, indicating a nasalized "un" sound.

2.3.3.1 Challenges of the Arabic Language

Arabic poses numerous challenges for language processing systems due to its unique linguistic features. These complexities span various aspects of the language, from its writing system to its grammatical structure and sociolinguistic variation (Huang et al., 2022).

Key challenges in the Arabic language include:

Table 2.3: The Arabic alphabet with approximate English pronunciation.

خ	ح	ج	ث	ت	ب	أ
Khaa	*Haa*	*Jiim*	*Thaa*	*Taa*	*Baa*	*Alif*
ص	ش	س	ز	ر	ذ	د
Saad	*Shiin*	*Siin*	*Zay*	*Raa*	*Dhaal*	*Daal*
ق	ف	غ	ع	ظ	ط	ض
Qaaf	*Faa*	*Ghayn*	*Ayn*	*Zaa*	*Taa*	*Daad*
ي	و	ه	ن	م	ل	ك
Yaa	*Waaw*	*Haa*	*Nun*	*Miim*	*Laam*	*Kaaf*

1. **Complex morphology:** Arabic has a rich and complex morphological system that relies heavily on root-based word formation. A single root can generate numerous words with related meanings by applying different patterns. See Table 2.4 for examples on some morphological features.
2. **Importance of vocalization (diacritics):** Diacritics ("harakat") are crucial for correct pronunciation and meaning in Arabic, but they are often omitted in everyday writing, leading to ambiguity. For example:

   عَلِمَ ("he knew") vs. عُلِّمَ ("he was taught")
3. **Agglutinative nature:** Arabic often uses prefixes, suffixes, and infixes to convey grammatical relationships, creating long words from shorter roots and affixes. For example:

   كتبته ("I wrote it")
4. **Different sentence structures:** Arabic allows for flexible word order due to its rich case system, which indicates the grammatical function of words in a sentence. Common structures include Verb-Subject-Object (VSO) and Subject-Verb-Object (SVO). For example:

   ذهبتْ فاطمة إلى السوق ("Fatimah went to the market") - VSO

   فاطمة ذهبتْ إلى السوق ("Fatimah went to the market") - SVO

2.3.4 Datasets for Arabic Abstractive Text Summarization

As this thesis focuses on abstractive text summarization in Arabic and employs data-driven methods, this subsection provides an overview of the publicly released benchmark Arabic ATS datasets commonly used in the literature, which include:

- **Arabic Headline Summary (AHS):** This dataset was introduced by Al-Maleh & Desouki (2020) and contains approximately 300,000 documents with their corresponding headlines as summaries. The dataset was collected from the Arabic website Mawdoo3,

Table 2.4: Examples of Arabic morphological features.

Morphological Feature	Example
Root	كتب (k-t-b, meaning "write")
Pattern	فَعَلَ (fa'ala, past tense verb pattern) كَتَبَ (kataba, "he wrote")
Derivational Morphology	مُفَاعِلْ (mufa"il, active participle pattern) مُكَاتِب (mukaatib, "correspondent")
Inflectional Morphology (Subject Agreement)	كَتَبْتُ (katabtu, "I wrote") كَتَبْتِ (katabti, "you (female) wrote")
Inflectional Morphology (Tense/Aspect/Mood)	كَتَبَ (kataba, "he wrote") يَكْتُبُ (yaktubu, "he writes") سَيَكْتُبُ (sayaktubu, "he will write")
Clitics	لِـكِتَابِـها (likitaabiha, "to her book")

which covers various topics across different domains. The articles' introductions were retained as the source text, while the titles were used as summaries.

- **XL-Sum:** Hasan et al. (2021) addressed the scarcity of summarization datasets for mid- and low-resource languages by presenting the XL-Sum dataset. It was sourced from the BBC website and consists of around 1 million news article-summary pairs across 44 languages. The Arabic subset comprises nearly 50k pairs of articles and their summaries. XL-Sum is regarded as a high quality dataset with a high degree of abstractiveness.
- **SumArabic:** This dataset was used in the study by Bani-Almarjeh & Kurdy (2023) and was collected from two Arabic news websites (emaratalyoum.com and almam-lakatv.com). It has 84,764 pairs of high-quality text documents. Each pair includes the first paragraph of an article as the source text and the article's title as the summary.
- **AraSum:** This dataset was gathered from the Deutsche Welle (DW) news website by Kahla et al. (2023). It contains about 50k articles with their leads used as summaries. AraSum is diverse and covers numerous news domains. The dataset includes well-written summaries with high relevance to the source documents.
- **Arabic Mogalad_Ndeef (AMN)**: Having approximately 265k news articles, AMN (also referred to as Arabic News Articles—ANA) was curated and used by Zaki et al. (2019). It was compiled from different Arabic news websites and Saudi newspapers. It includes one summary for each article.
- **WikiLingua**: Ladhak et al. (2020) This dataset was collected from the WikiHow

Table 2.5: Recap of Arabic abstractive text summarization datasets.

Dataset	Size	Domain	Document Type
AHS (Al-Maleh & Desouki, 2020)	300,000	Diverse	Single document
XL-Sum (Hasan et al., 2021)	50,000	News	Single document
SumArabic (Bani-Almarjeh & Kurdy, 2023)	84,764	News	Single document
AraSum (Kahla et al., 2023)	50,000	News	Single document
AMN/ANA (Zaki et al., 2019)	265,000	News	Single document
WikiLingua (Ladhak et al., 2020)	29,229	Diverse	Single document
Gigaword (Ladhak et al., 2020)	2,716,995	News	Multi-document

website. It contains 29,229 textual documents and their summaries in the form of multi-sentence highlights.

- **Gigaword**: Unlike previous datasets, Gigaword (Parker, Robert et al., 2011) is a multi-document dataset with different subsets of articles collected from 9 sources: Asharq Al-Awsat, Agence France Presse, Al-Ahram, Assabah, Al Hayat, An Nahar, Al-Quds Al-Arabi, Ummah Press, and Xinhua News Agency. It contains 2,716,995 articles in total. Gigaword is also not freely available.

The presented datasets are summarized in Table 2.5.

2.3.5 Evaluation Methods

There are multiple methods for evaluating summarization systems. Automated metrics quantitatively measure the similarity between system-generated and reference summaries. They offer standardized and efficient evaluation, but may not capture all aspects of summary quality. Common automated metrics include ROUGE, METEOR, BLEU, and BERTScore. Manual evaluation methods involve subjective assessments by human raters or annotators. However, they are typically more time-consuming and resource-intensive. This subsection covers the most popular evaluation methods. Next, we describe in detail the most commonly used methods.

2.3.5.1 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) is a set of metrics widely used to automatically evaluate the quality of a summary generated by a text summarization system. It measures the overlap between the system-generated summary and one or more reference summaries created by humans. The metric has several variants, with ROUGE-N and ROUGE-L being the most commonly used.

ROUGE-N measures the overlap of n-grams between the generated and reference texts. For a given n-gram length n, ROUGE-N is computed as:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{References}\}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \{\text{References}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \tag{2.48}$$

Here, $\text{Count}_{\text{match}}(\text{gram}_n)$ is the maximum number of n-grams co-occurring in the generated text and the reference text, and $\text{Count}(\text{gram}_n)$ is the number of n-grams in the reference text. ROUGE-1, ROUGE-2, and ROUGE-3 are commonly used variants, corresponding to unigram, bigram, and trigram overlap respectively. These variants capture different aspects of the text similarity, with ROUGE-1 focusing on word overlap and higher order ROUGE-N capturing phrase-level matches.

ROUGE-L, on the other hand, measures the longest common subsequence (LCS) between the generated and reference texts. It captures in-sequence matches that reflect sentence-level word order, allowing for gaps in the matching sequence. For a generated text X of length m and a reference text Y of length n, ROUGE-L is computed using the following equations:

$$R_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{n} \tag{2.49}$$

$$P_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{m} \tag{2.50}$$

$$F_{\text{lcs}} = \frac{(1 + \beta^2) R_{\text{lcs}} P_{\text{lcs}}}{R_{\text{lcs}} + \beta^2 P_{\text{lcs}}} \tag{2.51}$$

In these equations, $\text{LCS}(X, Y)$ is the length of the longest common subsequence between X and Y, and β is a parameter usually set to favor recall ($\beta > 1$). ROUGE-L is defined as the F-measure: ROUGE-L $= F_{\text{lcs}}$. This variant is particularly useful for capturing sentence-level structure similarities.

ROUGE has limitations in capturing semantic similarity and can be insensitive to meaning-preserving word reorderings. It may also struggle with paraphrases or synonyms that

convey the same meaning with different words (Aksenov et al., 2020). These limitations have led to the development of more reliable metrics that can capture nuance beyond superficial exact matches.

2.3.5.2 BLEU

BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) is a widely used metric for evaluating the quality of machine-generated text, including automatic text summarization. Originally developed for machine translation, BLEU measures the similarity between the system-generated text and one or more reference texts provided by human annotators.

The BLEU score is calculated based on the n-gram precision of the generated text compared to the reference texts, with a penalty for shorter outputs. The formula for computing the BLEU score is as follows:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.52}$$

where:

- N is the maximum n-gram order (typically 4)
- w_n is the weight for each n-gram order (uniform weights are commonly used)
- p_n is the precision score for n-grams of order n
- BP is the brevity penalty, which penalizes shorter outputs compared to the reference texts

The n-gram precision score, p_n, is calculated as:

$$p_n = \frac{\sum_{c \in C_n} \min\left(\text{Count}_c(C), \max_{r \in R} \text{Count}_c(r)\right)}{\sum_{c' \in C} \text{Count}_{c'}(C)} \tag{2.53}$$

where C_n is the set of n-grams in the generated text, C is the entire generated text, R is the set of reference texts, and $\text{Count}_c(x)$ is the count of the n-gram c in the text x.

The brevity penalty, BP, is calculated as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - r/c) & \text{otherwise} \end{cases} \tag{2.54}$$

where c is the length of the generated text, and r is the effective reference corpus length, which is typically the closest reference text length to the generated text length.

Similarly to ROUGE, BLEU has some weaknesses which include:

- It relies solely on n-gram precision and does not consider semantic similarity or contextual information.
- It can be easily exploited by systems that generate fluent but uninformative outputs.
- It may not perform well for evaluating abstractive summaries, where the generated text may use different words or paraphrasing compared to the references.

2.3.5.3 METEOR

METEOR (Metric for Evaluation of Translation with Explicit ORdering) (Banerjee & Lavie, 2005) is an automatic metric for evaluating machine translation output, but can also be used to evaluate other language generation tasks, including text summarization. METEOR addresses some limitations of BLEU by considering synonyms and paraphrases, and emphasizing recall. The computation of METEOR begins with creating an alignment between candidate and reference translations based on exact, stemmed, synonym, and paraphrase matches. It then calculates precision (P) and recall (R) using the formulas:

$$P = \frac{m}{t}, \quad R = \frac{m}{r} \tag{2.55}$$

where m is the number of matched unigrams, and t and r are the total unigrams in candidate and reference translations respectively.

METEOR then computes a parameterized harmonic mean (F-mean) of precision and recall:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R} \tag{2.56}$$

where α (typically 0.9) determines the relative weights of precision and recall. To account for word order, METEOR introduces a fragmentation penalty:

$$Penalty = \gamma \cdot (\frac{c}{u_m})^{\beta} \tag{2.57}$$

where c is the number of chunks, u_m is the number of matched unigrams, and γ and β are parameters (typically 0.5 and 3). The final METEOR score is calculated as:

$$METEOR = F_{mean} \cdot (1 - Penalty) \tag{2.58}$$

2.3.5.4 BERTScore

BERTScore (T. Zhang et al., 2020) is a recent evaluation metric proposed for text generation tasks, including automatic text summarization. It leverages pretrained language models,

originally based on BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), to measure the semantic similarity between the generated text and the reference text.

Unlike traditional metrics like ROUGE, which rely solely on lexical overlap, BERTScore captures semantic similarities by considering the contextual representations of words and phrases. This makes it more suitable for evaluating abstractive summaries, where the generated text may use different words or paraphrasing compared to the reference.

Let $x = (x_1, ..., x_k)$ be a candidate sentence and $y = (y_1, ..., y_l)$ be a reference sentence. For each token x_i in the candidate sentence, BERTScore finds the most similar token in the reference sentence using cosine similarity:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{y_j \in y} x_i^T y_j$$

Similarly, for each token in the reference sentence, it finds the most similar token in the candidate sentence:

$$P_{\text{BERT}} = \frac{1}{|y|} \sum_{y_j \in y} \max_{x_i \in x} x_i^T y_j$$

These two scores are then combined to compute the F1 score:

$$F_{\text{BERT}} = 2 \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}$$

To account for the varying importance of different words, BERTScore incorporates inverse document frequency (idf) weights. The weighted versions of the precision and recall scores are:

$$R_{\text{BERT}} = \frac{\sum_{x_i \in x} \text{idf}(x_i) \cdot \max_{y_j \in y} x_i^T y_j}{\sum_{x_i \in x} \text{idf}(x_i)}$$

$$P_{\text{BERT}} = \frac{\sum_{y_j \in y} \text{idf}(y_j) \cdot \max_{x_i \in x} x_i^T y_j}{\sum_{y_j \in y} \text{idf}(y_j)}$$

The idf weights are computed using a large corpus, typically the training data for the BERT model. This weighting scheme gives more importance to content words and less to function words or common tokens.

BERTScore has shown strong correlation with human judgments across various text generation tasks, including machine translation, image captioning, and summarization. Its

ability to capture semantic similarity beyond exact matches makes it particularly useful for evaluating abstractive summaries, paraphrases, and translations that convey the same meaning using different words.

BERTScore also has some limitations. It can be computationally expensive, especially for large datasets, due to the need to compute the contextual embeddings. Additionally, like other learned metrics, its performance can be sensitive to the choice of the pretrained model and the domain of the text being evaluated.

2.3.5.5 Human Evaluation

In the human evaluation of text summarization systems, human annotators or judges are involved in assessing the summary generated by the model. Human evaluators are presented with specific tasks, such as rating the fluency, coherence, relevance, or overall quality of the generated text. They may also be asked to compare different outputs and select the most appropriate or preferred one. The evaluators provide subjective judgments and feedback based on their expertise, perception, and understanding of the task, offering insights that automated metrics may not capture accurately.

Human evaluation, while valuable, still faces some limitations and challenges, including bias, difficulty of scaling, and impossibility to reproduce.

2.4 Conclusion

This chapter presented a detailed exploration of key concepts pertinent to the thesis. It began with a thorough introduction to deep learning in the context of natural language processing (NLP), outlining the NLP domain and discussing language models. The chapter also covered the essential principles of deep learning, progressing from foundational concepts to more sophisticated neural network architectures.

In the next chapter, we will provide a review the related literature on Arabic abstractive text summarization.

3

Related Work

The field of abstractive text summarization (ATS) has seen significant progress in recent years, driven by advancements in neural architectures and pretrained language models. In this chapter, we review a selection of literature on Arabic ATS, a domain with promising potential but also limitations. We focus on studies employing deep learning techniques within the sequence-to-sequence framework and those using transformer-based approaches.

3.1 Sequence-to-Sequence RNN-based Methods

The vast majority of studies using the sequence-to-sequence architecture based on RNNs utilized the attention mechanism to enhance the performance of their summarization models.

In their research, Suleiman & Awajan (2020) presented a sequence-to-sequence model based on recurrent neural networks with a global attention mechanism. The model incorporates a two-layer bidirectional LSTM encoder and a single-layer LSTM decoder. The first encoder layer captures word embeddings of the input text, while the second layer focuses on named entity embeddings. By adding an extra layer to the encoder, they observed enhanced performance relative to a single-layer baseline. Furthermore, they studied the impact of pretrained word embeddings on summary quality, finding the choice significantly affects overall performance. The best model scored 38.4% in terms of ROUGE-1. Subsequently, Suleiman & Awajan (2022) extended their prior work by exploring further architectural modifications, employing a three-layer bidirectional LSTM encoder and unidirectional LSTM decoder. The encoder's three layers generate representations of the input text, named entities, and keywords. The revised model demonstrated superior performance compared to multiple baseline variants, achieving a ROUGE-1 score of 38.4% and 75.9% in human evaluation. In both works, the authors used their own private dataset, which makes it impossible to directly compare their results with other works.

In another study using attention, Al-Maleh & Desouki (2020) adopted a similar methodology by training an LSTM attention-based sequence-to-sequence model on a dataset comprising around 295k articles and their summaries sourced from an Arabic website. The dataset, which they named Arabic Headline Summary (AHS), was made publicly available. To improve the quality of the generated summaries, they utilized the copy mechanism (Vinyals et al., 2015) and introduced a length penalty to discourage lengthy summaries. Moreover, a coverage penalty was implemented to mitigate redundancy in the generated output. The model using the copy mechanism achieved a ROUGE-1 score of 44.23%, surpassing the baseline model.

Wazery et al. (2022) conducted a study to compare various RNN-based architectures, including GRUs, standard LSTM networks, and bidirectional LSTMs, with different layer configurations. Additionally, they examined the impact of text preprocessing techniques and different word embedding models on the summarization results. The models were evaluated on two datasets: AHS and AMN. The study revealed that a sequence-to-sequence framework employing a 3-layer bidirectional LSTM encoder and a single-layer unidirectional LSTM decoder with attention achieved the highest performance among the different model variations considered, outperforming the best model by Al-Maleh & Desouki (2020) on the AHS dataset. Specifically, on the AHS dataset, the model scored ROUGE-1, ROUGE-2, and ROUGE-L values of 51.49%, 12.27%, and 34.37%, respectively. On AMN, the corresponding scores were 44.28%, 18.35%, 32.46% for the same metrics.

In an effort to enhance the RNN-based sequence-to-sequence architecture, Alahmadi et al. (2022) introduced topic-awareness into the framework. Their model consists mainly of an RNN topic classifier, which assists the decoder in generating topic-relevant summaries by providing an additional guiding vector. The model was evaluated on the AMN dataset and achieved 71.6% in ROUGE-1, 58.6% in ROUGE-2, and 70.1% in ROUGE-L, surpassing the results by Wazery et al. (2022) on the same dataset.

Adopting a distinctive approach, Etaiwi & Awajan (2022) proposed a semantic graph embedding-based method for Arabic ATS. To embed the semantic graph, they used a custom random-walk based technique, and an LSTM sequence-to-sequence architecture with attention for generating the summaries. The proposed model, named SemG-TS, was evaluated on a dataset of 8385 documents collected from AlJazeera.net website, and was compared with models using Word2Vec embeddings. The results showed that SemG-TS achieved a ROUGE F-measure of 4.70%. Although this score was lower than in other similar studies, SemG-TS outperformed the models that used Word2Vec embeddings.

3.2 Transformer-based Methods

Owing to their proven effectiveness in various NLP tasks, there has recently been an increased adoption of transformer-based language models for the task of Arabic abstractive summarization.

Elmadani et al. (2020) and Kahla et al. (2021) were among the first to leverage pretrained transformer-based language models for the task of Arabic ATS. Elmadani et al. (2020) used BERT (Devlin et al., 2019) along with a BERTSUM encoder (Yang Liu, 2019) and trained their models on both English and Arabic corpora. To train on the Arabic data, they used the multilingual BERT variant called mBERT. Evaluation on the KALIMAT (El-Haj & Koulali, 2013) dataset yielded a ROUGE-1 score of 12.21%, a ROUGE-2 score of 4.36%, and a ROUGE-L score of 12.19%. Results also show that it is possible to summarize Arabic text in low-resource cases. It should be noted that the dataset used is more appropriate for extractive summarization, which also explains the generally low scores. Kahla et al. (2021) conducted finetuning experiments on three different models, namely mBERT, mBART-50 (Yinhan Liu et al., 2020), and AraBERT (an Arabic model based on BERT) (Antoun et al., 2020), using a dataset consisting of 21,508 Arabic news articles. Additionally, they employed an approach based on cross-lingual transfer resulting in three models, which include mBERT finetuned on English and Arabic, mBERT finetuned on Hungarian and Arabic, and mBART-50 finetuned on Russian and Arabic. Evaluation showed that the mBERT model, finetuned on both English and Arabic data, achieved the highest performance with ROUGE-1, ROUGE-2, and ROUGE-L scores of 12.61%, 2.11%, and 12.61%, respectively.

In a multilingual finetuning setup, the authors in (Hasan et al., 2021) evaluated mT5 (Xue et al., 2021) on their dataset, XL-Sum, which they introduced to address the issue of limited availability of datasets for low- and mid-resource languages. The dataset contains 1 million highly-abstractive article-summary pairs in 44 languages, including Arabic. They achieved a ROUGE-1 score of 33.23%, a ROUGE-2 score of 13.74%, and a ROUGE-L score of 27.84% on the Arabic subset.

Taking inspiration from the original BART model (Lewis et al., 2020), Kamal Eddine et al. (2022) introduced AraBART, a pretrained sequence-to-sequence language model based on the BART architecture, specifically designed for Arabic ATS. They evaluated their model on the XL-Sum and Gigaword datasets. They also compared AraBART to baseline models, including mBART25, CAMeLBERT (Inoue et al., 2021) using a BERT2BERT architecture, mT5, and AraT5 (Nagoudi et al., 2022). Results showed that AraBART outperformed the baselines on most of the datasets it was evaluated on. The highest scores were obtained on the Gigaword XIN subset, with ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore values of 66.0%, 53.9%, 65.1%, and 84.4%, respectively. On XL-Sum, AraBART surpassed the model by Hasan et al. (2021) and yielded a ROUGE-1 score of 34.5%, a ROUGE-2 score of 14.6%, a ROUGE-L score of 30.5%, and a BERTScore value of 67.0%. The latter is a semantic metric that has seen very limited use in Arabic ATS, although studies have demonstrated that it has better correlation with human judgements than traditional lexical metrics (T. Zhang et al., 2020). In our work, we adopt both lexical metrics and semantic metrics to evaluate our models.

Following a hybrid approach, Reda et al. (2022) used AraBERT for the extractive summarization phase to extract the most important sentences from the articles. In the abstractive

phase, an mT5 model was finetuned on the output of the extractive module to produce the final summary and enhance its quality. Their approach was evaluated on the EASC dataset (El-Haj et al., 2010), which is designed for extractive summarization tasks and contains only 153 articles with 765 summaries. The type and size of this dataset present limitations to their model generalizability, which, based on user satisfaction, scored 3 out of 5.

To compare the performance of some pretrained transformer-based language models for the task of Arabic ATS, Chouikhi & Alsuhaibani (2022) finetuned five pretrained models including mBART, mT5, PEGASUS-Large, PEGASUS-XSUM, and AraBART on three datasets: AHS, AMN, and WikiLingua. They also compared them against a baseline BiLSTM model. The experimental findings demonstrated the superiority of the transformer-based models over the baseline model. Particularly, the PEGASUS (J. Zhang et al., 2020) models were the best performers, achieving ROUGE-1, ROUGE-2, ROUGE-L scores of 88.89%, 75.75%, 84.57% on the AMN dataset, 66.70%, 58.41%, 66.50% on AHS, and 94.62%, 88.72%, 94.58% on WikiLingua, respectively. It is worth mentioning that the PEGASUS models do not support the Arabic language as they were pretrained using English data. The study did not mention how the models were adapted for Arabic, making it impossible to reproduce or verify the obtained results. Moreover, the specific test splits used for evaluation were not disclosed.

In another comparative study, Bani-Almarjeh & Kurdy (2023) aimed to analyze the performance of various model architectures and pretrained language models for Arabic abstractive summarization. They employed both RNN-based sequence-to-sequence models as baselines and transformer-based models, including AraT5, mBERT and AraBERT using a BERT2BERT architecture, and AraGPT2 (Antoun et al., 2021). To finetune the models, they built a high-quality dataset, called SumArabic, which consists of 84,764 news articles and their summaries. The proposed models were assessed using ROUGE metrics, manual human evaluation, and performance on out-of-domain data. The results indicated that AraT5 outperformed all of the other models, scoring 49.06%, 30.81%, and 46.87% in terms of ROUGE-1, ROUGE-2, and ROUGE-L, respectively. Furthermore, both AraT5 and AraGPT2 were the best at summarizing out-of-domain text. The researchers made their models and dataset publicly available.

Kahla et al. (2023) extended their previous work (Kahla et al., 2021) by pretraining and finetuning multiple transformer-based models. The new models developed in their study included a BART model that underwent pretraining specifically for Arabic. Subsequently, this model was finetuned using a dataset of approximately 50,000 article-summary pairs curated from the Deutsche Welle (DW) news website, and referred to as AraSum. Additionally, a trilingual BART model was trained on Arabic, English, and Hungarian text sourced from Wikipedia. This trilingual model was then finetuned on the AraSum dataset. The study also involved the use of mT5 models, with one variant being exclusively finetuned on the AraSum dataset, while the other one was finetuned on both AraSum and XL-SUM. The latter was the best-performing on both datasets, attaining scores of 29.12%, 11.04%, and

24.07% for ROUGE-1, ROUGE-2, and ROUGE-L on XL-SUM, respectively. On AraSum, it scored a ROUGE-1 of 33.17%, a ROUGE-2 of 13.91%, and a ROUGE-L of 24.78%. The dataset was released to the research community.

To explore its effectiveness for Arabic ATS, Alqahtani & Al-Yahya (2023) conducted a series of experiments to finetune the pretrained transformer-based language model AraBART. Multiple datasets, including XL-Sum (the Arabic subset), AHS, and WikiLingua were used to evaluate their model. The model finetuned on the AHS dataset emerged as the top performer, achieving a ROUGE-1 score of 55%, a ROUGE-2 score of 40.15%, a ROUGE-L score of 54.55%, and 88.06% for BERTScore. These results surpass those in previous works on the same dataset when compared to models of similar size and type. On the XL-Sum dataset, the model yielded slightly better results than those by Kahla et al. (2023), with ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore values of 31.77%, 18.81%, 29.63%, and 78.84%, respectively. On the other hand, the scores on the WikiLingua dataset were comparatively lower, with values of 24.06%, 10.06%, 23.81%, and 78.13% for the same metrics.

A recap of the reviewed studies is presented in Table 3.1.

Table 3.1: Summary of the reviewed Arabic abstractive text summarization works. Only ROUGE metric is considered as it is the most commonly used (R-1: ROUGE-1, R-2: ROUGE-2, R-L: ROUGE-L). The scores represent the highest achieved on each dataset. For detailed results, the reader is referred to the respective works.

Reference	Method (highlight)	Dataset(s)	ROUGE (%)		
			R-1	R-2	R-L
Sequence-to-sequence RNN-based methods					
Suleiman & Awajan (2020)	BiLSTM (2-layer encoder) + attention	Private	38.4	-	-
Suleiman & Awajan (2022)	BiLSTM (3-layer encoder) + attention	Private	38.4	-	-
Al-Maleh & Desouki (2020)	LSTM + attention + copy mechanism	AHS	44.28	-	-
Wazery et al. (2022)	LSTM, BiLSTM, GRU + attention	AHS	51.49	12.27	34.37
		AMN	44.28	18.35	32.46
Alahmadi et al. (2022)	LSTM + topic-aware module + attention	AMN	71.6	58.6	70.1
Etaiwi & Awajan (2022)	Semantic graph embedding + LSTM + attention	Private	4.70*	-	-
Transformer-based methods					
Elmadani et al. (2020)	Finetuning multilingual BERT (mBERT)	KALIMAT	12.21	4.36	12.19
Kahla et al. (2021)	Finetuning mBERT, mBART-50, mBERT + cross-lingual transfer	Private	12.61	2.11	12.61
Hasan et al. (2021)	Multilingual finetuning using mT5	XL-Sum	33.23	13.74	27.84
Kamal Eddine et al. (2022)	Introducing and finetuning AraBART	XL-Sum	34.50	14.60	30.50
		Gigaword (XIN)	66.00	53.90	65.10
Reda et al. (2022)	Finetuning AraBERT (extractive) + mT5 (abstractive)	EASC	-	-	-
Chouikhi & Alsuhaibani (2022)	Finetuning mBART, mT5, PEGASUS-Large, PEGASUS-XSUM, AraBART	AHS	66.70	58.41	66.50
		AMN	88.89	75.75	84.57
		WikiLingua	94.62	88.72	94.58
Bani-Almarjeh & Kurdy (2023)	Finetuning AraT5, mBERT, AraBERT, AraGPT2	SumArabic	49.06	30.81	46.87
Kahla et al. (2023)	Pretraining (multilingual) and finetuning of BART-based models, mT5	XL-Sum	29.12	11.04	24.07
		AraSum	33.17	13.91	24.78
Alqahtani & Al-Yahya (2023)	Finetuning AraBART	AHS	55.00	40.15	54.55
		XL-Sum	31.77	18.81	29.63
		WikiLingua	24.06	10.06	23.81

* Reported as ROUGE F-measure

3.3 Research Gap & Motivation

While the literature on Arabic ATS has shown promising results, the vast majority of existing works have focused solely on supervised learning techniques, which may be limited by the objective functions used during training. Other problems include exposure bias and inconsistency between training and inference (Uc-Cetina et al., 2023). Building upon prior works that harness transfer learning with pretrained transformer-based LMs (PTLMs), we further introduce an additional optimization step using deep reinforcement learning (DRL), an area that remains large unexplored in the context of Arabic ATS. We propose to combine the language understanding and generation capabilities of PTLMs with RL techniques, specifically the proximal policy optimization (PPO) algorithm. We leverage RL as a finetuning paradigm benefiting from its ability to directly optimize non-differentiable objectives such as evaluation metrics (e.g., ROUGE, BERTScore) (Paulus et al., 2018; Alomari et al., 2022). By incorporating reward signals within our RL framework, our method aims to generate enhanced summaries that better align with desired evaluation criteria. To our knowledge, this is the first work exploring the combination of PTLMs and deep RL for Arabic abstractive summarization, potentially addressing a research gap.

3.4 Conclusion

We provided a literature review on the research in Arabic abstractive summarization. We covered both RNN-based sequence-to-sequence approaches and those utilizing the transformer architecture and pretrained language models. This review led us to identify gaps in existing research which this thesis aims to address. In the next chapter, we will provide a detailed description of our proposed approach to Arabic ATS, along with our experimental design.

4

Methods & Experiments

In this chapter, we present our proposed approach to Arabic abstractive text summarization. We begin by describing our methods in detail to establish the formal foundation for our framework. Subsequently, we cover the experimental setup, including the used datasets, applied data preprocessing techniques, the foundation (pretrained) models employed, as well as the training and implementation details and the experimental environment, including the software and computational resources used.

4.1 Methods

The proposed framework, illustrated in Figure 4.1, combines transfer learning through supervised finetuning of pretrained transformer-based language models (PTLMs) and deep reinforcement learning for Arabic ATS.

On a high level, our approach can be summarized as follows:

1. Supervised Finetuning (SFT): Finetune a PTLM on an Arabic ATS dataset, treating it as a sequence-to-sequence task, where the input is the article text, and the output is the target summary.
2. Reinforcement Learning-based Finetuning (RLFT):
    - Initialize the RL agent with the supervised finetuned model (from step 1).
    - Define reward functions (ROUGE, METEOR, BERTScore, textual entailment, other combinations).
    - Use proximal policy optimization (PPO) to finetune the RL policy by optimizing the reward signal through interaction with the environment.

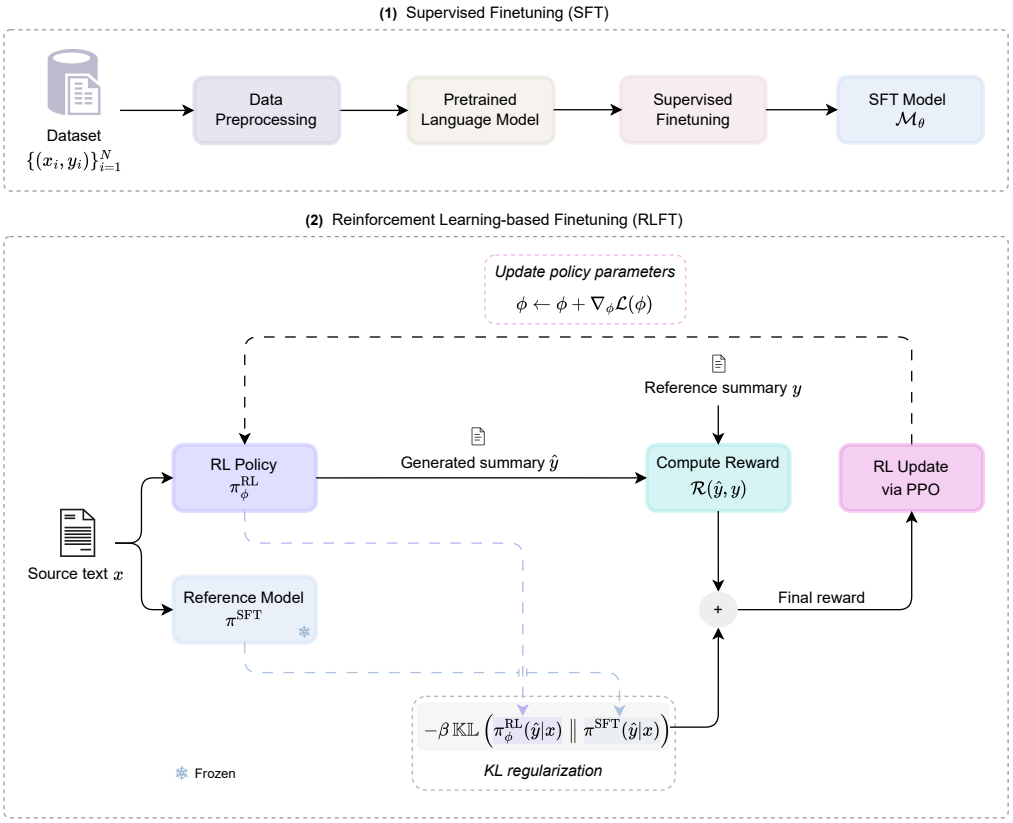


Figure 4.1: A high-level schema of our proposed framework. It starts with a supervised finetuning phase (SFT) where a pretrained LM is finetuned on an Arabic ATS dataset. Next, the SFT model is further optimized via RL using PPO with KL-regularized rewards.

4.1.1 Supervised Finetuning (SFT)

As a first step, we finetune a pretrained language model on a parallel corpus of Arabic texts and their corresponding summaries using supervised learning.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a dataset of document-summary pairs, where x_i represents the input text and y_i is the reference summary. We denote our supervised finetuned model as $\mathcal{M}_\theta$, where θ represents the model parameters. The objective during supervised finetuning is to maximize the likelihood of the reference summary given the input text. This is equivalent to minimizing the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}}(\theta) = -\sum_{t=1}^{T_i} \log p(y_t \mid y_{i,<t}, x_i; \theta) \tag{4.1}$$

where:

- $p(y_{i,t} \mid y_{i,<t}, x_i; \theta)$ is the conditional probability of the t-th token in the reference summary given the input text x_i and the previously generated tokens $y_{i,<t}$, parameterized by θ.
- T_i is the length of the i-th reference summary.

The models are trained using the teacher-forcing technique. This means that at each time step, the model is provided with the ground-truth token from the reference summary as the previous token, rather than its own generated token. This helps stabilize training and allows the model to learn the correct dependencies between tokens more effectively.

This phase aims to establish a strong baseline for use in the subsequent reinforcement learning phase.

4.1.2 Reinforcement Learning-based Finetuning (RLFT)

Following supervised finetuning, we use reinforcement learning to further optimize our summarization model. The RL stage involves initializing the policy, denoted π_ϕ^{RL} and parameterized by ϕ, with the SFT model: $\pi_\phi^{\text{RL}} = \mathcal{M}_\theta$.

In our RL framework, the agent (language model) interacts with environment and performs actions (autoregressive token generation) in a given state (the input text) in order to maximize the expected reward $\mathcal{R}$. The RL objective can be formally expressed as:

$$\mathcal{L}_{\text{RL}}(\phi) = \mathbb{E}_{(x,y)\sim D,\, \hat{y}\sim\pi_\phi^{\text{RL}}(\cdot|x)} \left[\mathcal{R}(\hat{y}, y)\right] \tag{4.2}$$

where $\mathcal{R}(\hat{y}, y)$ is the reward function that evaluates the quality of the generated summary $\hat{y}$ given the reference summary y.

Following previous efforts (Stiennon et al., 2020; Ziegler et al., 2020), we also add a Kullback-Leibler (KL) divergence[1] term to the reward in order to prevent drastic deviation from the initial supervised policy. This term is defined as:

$$\mathbb{KL}\left(\pi_\phi^{\text{RL}}(\hat{y}|x)) \parallel \pi^{\text{SFT}}(\hat{y}|x))\right) = \log\left(\frac{\pi_\phi^{\text{RL}}(\hat{y}|x)}{\pi^{\text{SFT}}(\hat{y}|x)}\right) \tag{4.3}$$

where $\mathbb{KL}(\cdot\|\cdot)$ is the KL divergence between the (active) RL policy π_ϕ^{RL} and the supervised finetuned model π^{SFT}. This latter remains "frozen" (i.e., no update of parameters) during the RL training and serves as a reference model to compute the KL-penalized reward $\hat{\mathcal{R}}$:

[1] Kullback-Leibler (KL) divergence measures the statistical distance between two probability distributions.

$$\hat{\mathcal{R}}(\hat{y}, y) = \mathcal{R}(\hat{y}, y) - \beta \, \mathbb{KL}\left(\pi_{\phi}^{\mathrm{RL}}(\hat{y}|x)) \parallel \pi^{\mathrm{SFT}}(\hat{y}|x)\right) \tag{4.4}$$

This leads to the final objective below:

$$\begin{aligned} \mathcal{L}_{\mathrm{RL}}(\phi) &= \mathbb{E}_{(x,y)\sim\mathcal{D},\, \hat{y}\sim\pi_{\phi}^{\mathrm{RL}}(\cdot|x)}\left[\mathcal{R}(\hat{y}, y) - \beta \, \mathbb{KL}\left(\pi_{\phi}^{\mathrm{RL}}(\hat{y}|x)) \parallel \pi^{\mathrm{SFT}}(\hat{y}|x)\right)\right] \\ &= \mathbb{E}_{(x,y)\sim\mathcal{D},\, \hat{y}\sim\pi_{\phi}^{\mathrm{RL}}(\cdot|x)}\left[\mathcal{R}(\hat{y}, y) - \beta \log\left(\frac{\pi_{\phi}^{\mathrm{RL}}(\hat{y}|x)}{\pi^{\mathrm{SFT}}(\hat{y}|x)}\right)\right] \end{aligned} \tag{4.5}$$

where β is the KL penalty coefficient, which is a hyperparameter that controls the strength of the regularization.

Additionally, this regularization serves the following purposes:

1. Mitigating the risk of *reward hacking*, where the model generates unintelligible sequences (e.g., repetitions) that produce high rewards (i.e., gaming the system).
2. Preventing *catastrophic forgetting*, which is when the model loses its ability to summarize, or even its general language modeling skills.
3. Allowing further exploration by encouraging additional entropy, and thus stopping the model from collapsing to a single mode.

Reward Functions The reward function $\mathcal{R}$ is an essential element of the RL phase, as it guides the optimization process toward the desired criteria. We propose multiple rewards derived from automatic metrics: (a) ROUGE, (b) METEOR, (c) BERTScore, as well as a (d) natural language inference (NLI) signal (also known as textual entailment). We also experiment with reward combinations: (e) reward average and (f) adaptive reward. We explain next in more detail how each reward is calculated.

a. ROUGE reward: We use the average of the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L variants to have a balanced reward signal:

$$\mathcal{R}^{\mathrm{ROUGE}}(\hat{y}, y) = \text{ROUGE-Avg}(\hat{y}, y) \tag{4.6}$$

where:

$$\text{ROUGE-Avg}(\hat{y}, y) = \frac{\text{ROUGE-1}(\hat{y}, y) + \text{ROUGE-2}(\hat{y}, y) + \text{ROUGE-L}(\hat{y}, y)}{3} \tag{4.7}$$

b. METEOR reward: We use the score obtained from the METEOR metric as is:

$$\mathcal{R}^{\mathrm{METEOR}}(\hat{y}, y) = \text{METEOR}(\hat{y}, y) \tag{4.8}$$

c. BERTScore reward: We employ the F1 score of the BERTScore metric:

$$\mathcal{R}^{\mathrm{BERTSCORE}}(\hat{y}, y) = \text{BERTScore-F1}(\hat{y}, y) \tag{4.9}$$

d. Textual entailment (NLI) reward: We leverage this reward signal to encourage the model to generate summaries that are more factually consistent. We assume that the ground-truth summary is logically entailed by the source text and so use the reference summary y as the premise and the generated summary $\hat{y}$ as the hypothesis. We use the probability of an "entailment" decision as a reward:

$$\mathcal{R}^{\text{NLI}}(\hat{y}, y) = \text{NLI}(\hat{y}, y) = p_{\text{NLI}}(\text{entailment} \mid \hat{y}, y) \tag{4.10}$$

We employ a readily-available NLI classifier[2] based on the mDeBERTa-V3 model (He et al., 2023) and finetuned on a mix of the XNLI and multilingual-NLI-26lang-2mil7 datasets (Laurer et al., 2024). The total size of the training data is over 3M premise-hypothesis pairs in 27 languages, including Arabic.

e. Reward average (AVG): We use the average of the previous rewards:

$$\mathcal{R}^{\text{AVG}}(\hat{y}, y) = \frac{\mathcal{R}^{\text{ROUGE}}(\hat{y}, y) + \mathcal{R}^{\text{METEOR}}(\hat{y}, y) + \mathcal{R}^{\text{BERTSCORE}}(\hat{y}, y) + \mathcal{R}^{\text{NLI}}(\hat{y}, y)}{4} \tag{4.11}$$

f. Adaptive reward: We employ a linear combination of rewards with adaptive weights that are adjusted dynamically during training based on performance. We update the weights at 100-step intervals. This reward is given by:

$$\mathcal{R}^{\text{ADAPTIVE}}(\hat{y}, y) = \sum_{i=1}^{4} w_i \cdot \mathcal{R}^i(\hat{y}, y) \tag{4.12}$$

where $\mathcal{R}^i$ represents individual rewards (ROUGE, METEOR, BERTSCORE, NLI), and w_i are the adaptive weights, with $\sum_{i=1}^{4} w_i(t) = 1$. Since we observed some level of instability when optimizing for the BERTScore reward (perhaps due to its overestimated score), we set its initial weight to 0.1 and used 0.3 for the other rewards.

A high-level outline of the RL finetuning process is described in Algorithm 1.

Training Algorithm We use the proximal policy optimization (PPO) algorithm to train our summarization models. Our choice was motivated by its success in Reinforcement Learning From Human Feedback (RLHF) and its advantages (Ouyang et al., 2022; Zheng et al., 2023). PPO provides a good balance between sample efficiency and ease of implementation. It also works well with the continuous action spaces in language generation tasks. Its relative robustness to hyperparameter choices also simplified our training process.

[2] https://huggingface.co/MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7

Algorithm 1 RL Finetuning Process

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ (x_i: source text, y_i: reference summary)
Require: RL policy π_ϕ^{RL} initialized with the SFT model
Require: Reference model (a "frozen" copy of the SFT model)
Require: Reward function $\mathcal{R}(\hat{y}, y)$
1: **for** step $k = 1, 2, \ldots, K$ **do**
2: Sample a batch of source texts $\{x_i\}_{i=1}^{M}$ from $\mathcal{D}$
3: **for** each x_i in the batch **do**
4: Generate a summary $\hat{y}_i \sim \pi_\phi^{\text{RL}}(\cdot|x_i)$
5: Compute the KL-regularized reward $\hat{\mathcal{R}}_i(\hat{y}, y)$ ▷ Eq. 4.4
6: **end for**
7: Run a PPO optimization step and update policy parameters:
8:

$$\phi^{\text{k+1}} = \phi^{\text{k}} + \nabla_\phi \mathcal{L}(\phi)$$

9: **end for**
10: **return** Optimized policy $\pi_\phi{}^*$

4.2 Experimental Design

4.2.1 Datasets

We evaluate our approach on two Arabic abstractive text summarization benchmark datasets: XL-Sum and AraSum.

XL-Sum We used the Arabic subset of this dataset which was introduced by Hasan et al. (2021). It contains 46,897 article-summary pairs in total, collected from the BBC news website. The training split contains 39,863 (we used 50% of the validation as extra data), the validation split consists of 2,345, while the test split remains unchanged, comprising 4,689 text documents with their corresponding summaries. Table 4.2 shows an an example entry from this dataset.

AraSum This dataset, developed by Kahla et al. (2023), contains 49,604 text documents with their summaries sourced from the Deutsche Well (DW) Arabic news website. The dataset was split into training, validation, and test sets that contain 44,703, 932, and 3,969 article-summary pairs, respectively. An example from this dataset is shown in Table 4.3

Table 4.1 provides a summary of the split sizes and average lengths for each datasets.

Dataset	Split sizes			Avg. # tokens	
	Training	Validation	Test	Source text	Ref. summary
XL-Sum	39,863	2,345	4,689	429.00	25.00
AraSum	44,703	932	3,969	423.64	35.88

Table 4.1: Datasets split sizes and average number of tokens per example.

We opted for these datasets for the following reasons. Firstly, we found that they are of highest quality among the existing datasets. Secondly, despite both belonging to the news domain, they still cover diverse news sub-genres (politics, technology, entertainment, etc.). These datasets are also some the most challenging ones among publicly available resources, which helps us test the robustness of our methods. The only available dataset covering non-news topics is AHS (Al-Maleh & Desouki, 2020), which was found to be of mediocre quality with many repeated entries (Bani-Almarjeh & Kurdy, 2023), while having very short summaries, making it inappropriate to evaluate our methods.

Table 4.2: An example entry from the XL-Sum dataset. The source text was truncated due to excessive length. English translations are provided for general readers.

Source text
استخدم العلماء خوارزمية ذكاء اصطناعي قوية لتحليل أكثر من 100 مليون مركبا كيميائيا خلال أيام وأشاد خبراء بهذا الإنجاز، واعتبروه تقدما كبيرا في مكافحة مشكلة مقاومة الأمراض للعقاقير الطبية، والتي أصبحت متفاقمة. واستخدم العلماء خوارزمية (عمليات حسابية آلية معقدة) قوية لتحليل أكثر من 100 مليون مركب كيميائي في غضون أيام. وقال الباحثون إن تركيبة المضادات الحيوية المكتشفة حديثا، تمكنت من القضاء على 35 نوعا من البكتيريا القاتلة. وارتفعت معدلات الإصابة بالأمراض المقاومة للمضادات الحيوية خلال السنوات الأخيرة بنسبة 9 في المئة في إنجلترا بين عامي 2017 و 2018، لتبلغ ما يقرب من 61 ألف حالة ...
Scientists used a powerful AI algorithm to analyze over 100 million chemical compounds in a matter of days. Experts praised this achievement, considering it a significant advancement in combating the issue of drug-resistant diseases, which has been escalating. The scientists utilized a robust algorithm (complex automated computations) to analyze more than 100 million chemical compounds within days. The researchers stated that the newly discovered antibiotic composition managed to eliminate 35 types of deadly bacteria. The rates of antibiotic-resistant diseases have increased by 9% in England between 2017 and 2018, reaching nearly 61,000 cases ...
Reference summary
تمكن علماء من اكتشاف نوع جديد من المضادات الحيوية باستخدام الذكاء الاصطناعي(AI) ، في إنجاز هو الأول من نوعه في العالم.
Scientists have discovered a new type of antibiotic using artificial intelligence (AI), in a world-first achievement.

Table 4.3: An example entry from the AraSum dataset. The source text was truncated due to excessive length. English translations are provided for general readers.

Source text
حتى الآن لا يوجد "رابح" في سباق حماية البيئة بين دول العالم، لأن أياً منها لم يحقق بعد الأهداف الملقاة على عاتقها بموجب اتفاق باريس لحماية المناخ، الذي ينصّ على إبقاء الزيادة في درجات الحرارة العالمية تحت مستوى الدرجتين المئويتين. ويتفق الباحثون على أنه، وبالنظر إلى أوضاع العالم اليوم وبالمقارنة مع فترة ما قبل الثورة الصناعية، فإن درجة الحرارة حول العالم سترتفع بنحو ثلاث درجات مئوية مع نهاية القرن الحالي، إلا لو تمكنت البشرية من خفض انبعاثات غاز ثاني أكسيد الكربون بسرعة وبنسبة كبيرة. لكن ذلك لا يبدو ممكناً، بحسب ما توصل له الفريق العلمي وراء مؤشر حماية المناخ، الصادر عن معهد "نيوكلايميت" ومنظمة "جيرمان ووتش" غير الحكومية، و "كلايميت آكشن نيتوورك." ...
So far, there is no “winner” in the race to protect the environment among the world's nations, as none have yet achieved the goals set out in the Paris Agreement to protect the climate, which aims to keep the increase in global temperatures below two degrees Celsius. Researchers agree that, given the current state of the world and compared to the pre-industrial era, the global temperature is expected to rise by about three degrees Celsius by the end of this century, unless humanity can rapidly and significantly reduce carbon dioxide emissions. However, this does not seem feasible, according to the scientific team behind the Climate Protection Index, issued by the “NewClimate” Institute, the non-governmental organization “Germanwatch”, and the “Climate Action Network” …
Reference summary
ملف التقلبات المناخية سباق مع الزمن، والغلبة فيه للدولة التي تضع حماية البيئة على أعلى سلم أولوياتها. مؤشر حماية المناخ يقيس مدى التزام دول العالم بمعايير الرفق بالبيئة، ويصدر تقييمه للدول الرابحة والخاسرة كل عام.
The issue of climate fluctuations is a race against time, with the advantage going to the country that places environmental protection at the top of its priorities. The Climate Protection Index measures the extent to which countries around the world adhere to environmentally friendly standards and releases its assessment of the winning and losing countries each year.

Table 4.4: An example entry from the Aljazeera.net Arabic News Articles dataset. The source text was truncated due to excessive length. English translations are provided for general readers.

Source text
قالت الأمم المتحدة إن أكثر من 235 ألف شخص نزحوا في غضون أسبوعين من شمال غرب سوريا جراء التصعيد العسكري هناك، في حين طالبت منظمات إنسانية إغاثية محلية المجتمع الدولي بتدخل فوري لحماية المدنيين. وأورد مكتب تنسيق الشؤون الإنسانية التابع للأمم المتحدة في بيان أنه بين 12 و25 ديسمبر/كانون الأول الجاري نزح أكثر من 235 ألف شخص من شمال غرب سوريا، مشيرا إلى أن كثيرين منهم فروا من مدينة معرة النعمان وقرى وبلدات في محيطها، وجميعها باتت "شبه خالية من المدنيين" ...
The United Nations stated that more than 235,000 people have been displaced within two weeks from northwest Syria due to the military escalation there. Local humanitarian relief organizations have called on the international community for immediate intervention to protect civilians. The UN Office for the Coordination of Humanitarian Affairs reported in a statement that between December 12 and 25, more than 235,000 people were displaced from northwest Syria, noting that many of them fled from the city of Maarrat al-Nu'man and surrounding villages and towns, which have all become "almost empty of civilians" …
Reference summary
قالت الأمم المتحدة إن أكثر من 235 ألف شخص نزحوا خلال أسبوعين من شمال غرب سوريا جراء التصعيد العسكري هناك، بينما طالبت منظمات إنسانية المجتمع الدولي بتدخل فوري لحماية المدنيين.
The United Nations stated that more than 235,000 people have been displaced within two weeks from northwest Syria due to the military escalation there, while humanitarian organizations have called on the international community for immediate intervention to protect civilians.

We also used the Aljazeera.net Arabic News Articles dataset (Rhouati, 2019) in a test-only transfer setup aimed at assessing further the generalizability of our models. This dataset

was collected from Aljazeera.net news website, and contains over 5000 examples. We use a random subset of 500 examples in our experiment. An example from this dataset is given in Table 4.4.

4.2.2 Data Preprocessing

The Arabic texts and summaries underwent standard preprocessing steps, including tokenization and sequence padding. For the XL-Sum dataset, the input source texts and target summaries were truncated to 512 and 64 tokens, respectively. For AraSum, the input text documents were truncated to 512 tokens and the summaries were truncated to 128 tokens. To alleviate some of the Arabic linguistic complexity, we used the open-source Arabic text preprocessing helper function `AraBERT Preprocess`[3] (Antoun et al., 2020). It was mainly used to perform the following operations:

- Removal of diacritics (Tashkeel) and stretching character (Tatweel);
- Conversion of Hindi digits to Arabic digits.

4.2.3 Foundation Models

We experimented with the encoder-decoder pretrained language models AraT5 (Nagoudi et al., 2022) and AraBART (Kamal Eddine et al., 2022), which have shown promising performance on abstractive summarization tasks.

AraT5 This is a text-to-text transformer-based model for Arabic language generation, based on the architecture of T5 (Raffel et al., 2020). AraT5 was pretrained on a large corpus of MSA and dialectal Arabic and was shown to outperform the multilingual mT5 model. We use the more recent and updated version `AraT5 v2`[4], which has a sequence length of 1024 tokens and 368M parameters.

AraBART This model is based on BART (Lewis et al., 2020), where the encoder and the decoder are trained end-to-end. It has 6 encoder and 6 decoder layers with 768 hidden dimensions, totaling 139M parameters. AraBART achieved the best performance on multiple Arabic abstractive text summarization datasets, surpassing strong baselines including mBART and mT5. We employ the official pretrained checkpoint[5] publicly available for finetuning.

4.2.4 Training Details

SFT We finetuned both models, AraT5 and AraBART, on the XL-Sum and AraSum training sets, separately.

[3] https://github.com/aub-mind/arabert/blob/master/preprocess.py
[4] https://huggingface.co/UBC-NLP/AraT5v2-base-1024
[5] https://huggingface.co/moussaKam/AraBART

For the XL-Sum dataset, we used a learning rate of 7×10^{-5} with a linear scheduler, while for AraSum, we used a learning rate of 3×10^{-5} decayed to 0 with a cosine schedule. For both datasets, we set an initial batch size of 32 with automatic batch size finding using exponential decay, and no gradient accumulation. The models were trained for 5 epochs in both cases, with warmup periods of 990 steps for XL-Sum and 1400 steps for AraSum.

The AdamW (Adam with decoupled weight decay) (Loshchilov & Hutter, 2019) optimizer was employed for all models. We also experimented with mixed precision training using FP16 (half-precision floating-point) but it did not result in better performance.

For each dataset, only the top-performing model between AraT5 and AraBART, based on test set evaluation, was selected to be used in the next RL phase.

RLFT We initialized the RL policy with the (best) SFT checkpoint and trained a separate RL-based model for each reward defined in Section 4.1.2. We used the original training sets for both datasets. The models were trained for a total of 500 steps (we found that training longer causes instability and performance degradation). We used the Adam optimizer with a learning rate of 2×10^{-6} and a batch size of 16, while the mini-batch size was set to 8. For the initial KL penalty coefficient, we set $\beta = 0.2$ while the the target KL value was set to 2.

For the other PPO hyperparameters, we set a clip range parameter $\epsilon = 0.2$, a discount factor $\gamma = 0.99$, a gamma[6] value of $\lambda = 0.95$, and a value function coefficient of 0.2.

4.2.5 Decoding (Generation) Strategies

SFT For summary generation at inference time (e.g., evaluation), we used beam search with a beam size of 6. For validation during training, we used the more efficient greedy search strategy.

RLFT For decoding from our RL-based policies during training, we employed a combination of beam search and top-k nucleus sampling with temperature to encourage exploration (beam size = 4, top-k = 80, top-p = 0.95, and τ = 0.5). For inference, we used beam search with a beam size of 6.

4.2.6 Model Evaluation

Automatic Metrics We evaluated the performance of our models using standard automatic evaluation metrics for text summarization, including the lexical n-gram-based metric

[6] This parameter is used in the calculation of the Generalized Advantage Estimation (GAE), which is used by PPO for estimating the advantage function. Refer to the original PPO paper (Schulman et al., 2017) for more details.

ROUGE, METEOR, and the distributional semantics-based BERTScore[7] metric. We also propose using the textual entailment (NLI) classifier, used in the RL phase as a reward, as a proxy to measure the factual consistency of generated summaries. We compute the average of the entailment probabilities (converted to percentages) across all examples in the test set, and use this as the final NLI score. While not a direct measure of factual consistency, this approach serves as an approximative indicator of the summaries' alignment with the source content.

Data Statistics To complement our evaluation, we report extractiveness and abstractiveness metrics[8] (Grusky et al., 2018; Fabbri et al., 2021). Extractiveness is measured by coverage (proportion of source words in the summary) and density (average length of copied word sequences). Abstractiveness is quantified by the percentage of novel n-grams, which are word sequences in the summary not found in the source. Additionally, we consider compression (ratio of summary to source length) and repeated n-grams (percentage of n-word sequences appearing multiple times in the summary). For more details on how these metrics are computed, refer to (Grusky et al., 2018) and (Fabbri et al., 2021).

4.2.7 Software and Computational Resources

All of our experiments used Python v3.12 as the primary programming language. To facilitate access to and finetuning of pretrained language models, we used the Hugging Face Transformers library (Wolf et al., 2020) with PyTorch as the backbone framework. For the PPO algorithm implementation, we used the open-source TRL (Transformer Reinforcement Learning) library (von Werra et al., 2020). The experiments were conducted in a cloud-based environment and ran on different devices, including NVIDIA P100/RTX5000 and A5000, having 16GB and 24GB of memory (VRAM), respectively.

4.3 Conclusion

This chapter provided a detailed description of our methods and experimental setup. We began with a formal explanation of our approach, followed by detailed documentation of our experimental and implementation settings. We concluded with a description of the employed evaluation methods and the software and hardware resources utilized to carry out the experiments. In the next chapter, we will present and thoroughly discuss our experimental results.

[7] We used the multilingual DistilBERT variant "distilbert-base-multilingual-cased" as a base model for BERTScore computation.

[8] We used the official implementation by Fabbri et al. (2021) available via https://github.com/Yale-LILY/SummEval

5

Results & Discussion

Our approach involved a two-stage framework: (1) supervised finetuning (SFT) and (2) a reinforcement learning-based optimization phase (RL). In this chapter, we present and discuss our experimental results starting with those of supervised finetuning phase then moving on to the reinforcement learning stage. We also cover additional experiments including test-only transfer on a new dataset as well as some ablation studies. Lastly, we compare our results with those of prior works.

5.1 Supervised Finetuning

The evaluation results of the supervised finetuned models, on both datasets, are summarized in Table 5.1.

Table 5.1: Evaluation results for SFT models on XL-Sum and AraSum test sets. Highest scores are marked in bold.

Test set	Model	ROUGE			METEOR	BERTScore	Fact. cons. (NLI)
		R-1	R-2	R-L			
XL-Sum	AraT5	**33.97**	**16.71**	**29.77**	**28.85**	**87.68**	**25.22**
	AraBART	31.49	15.04	27.56	26.22	82.59	22.10
AraSum	AraT5	**30.89**	**14.43**	**87.25**	**26.43**	**86.20**	**26.16**
	AraBART	29.62	13.82	24.17	24.91	86.04	24.66

The results show that the AraT5 model consistently outperforms AraBART on both datasets. It achieves superior scores across all metrics, with values comparable to those of AraBART in terms of BERTScore. This is likely due to the semantic nature of this metric in contrast to other metrics such as ROUGE, which rely on exact matches between the generated and

reference summaries. The AraT5 model also generates more factually consistent summaries compared to AraBART, as reflected in the NLI metric.

While the generally low NLI scores might suggest unfaithful summaries, these scores are influenced by our use of the "entailment" decision probability to compute them. Therefore, any "neutral" classification by the NLI model results in a lower score.

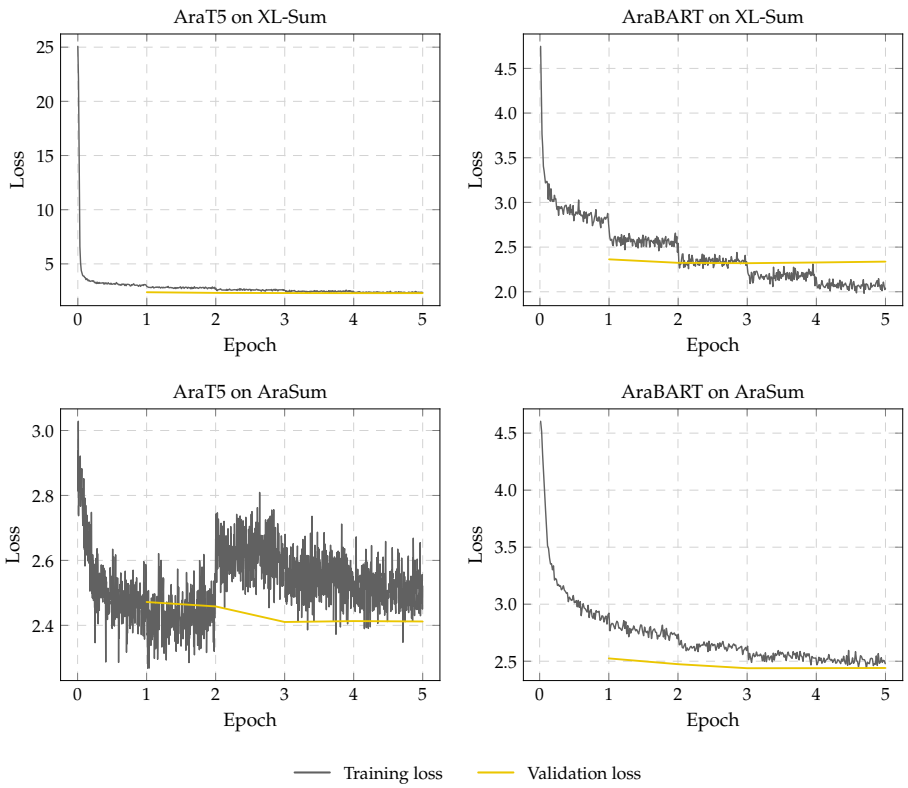


Figure 5.1: Loss curves for AraT5 and AraBART on XL-Sum (top) and AraSum (bottom).

Figures 5.1 shows stable training and validation loss curves for both models. The losses show steady decreasing over each epoch, with minimal fluctuations. The validation loss for all models indicate no apparent overfitting.

These results suggest that AraT5 is the optimal choice for our task; therefore, the model was selected for further optimization through reinforcement learning. Additionally, this model will serve as the SFT baseline for comparison with the RL-finetuned models.

5.2 Reinforcement Learning-based Finetuning

Table 5.2 presents the evaluation results of the RL-optimized (SFT + RL) models on the XL-Sum and AraSum test sets. The table compares the performance of the RL models,

trained using different rewards, against the supervised finetuned (SFT) baseline in terms of automatic evaluation metrics.

Table 5.2: Evaluation results for SFT + RL models on XL-Sum and AraSum test sets. Highest scores are marked in bold.

Test set	Method	Reward	ROUGE				METEOR	BERTScore	Fact. cons. (NLI)
			R-1	R-2	R-L	R-Avg			
XL-Sum	SFT	–	33.97	16.71	29.77	26.82	**28.85**	87.68	25.22
	SFT + RL	ROUGE	**34.22**	**17.00**	**30.11**	**27.11**	28.39	87.83	27.00
		METEOR	34.07	16.80	29.89	26.92	28.50	87.73	25.81
		BERTScore	33.80	16.91	29.93	26.88	27.42	**87.85**	28.77
		NLI	33.93	16.87	29.96	26.92	27.86	87.80	**29.79**
		AVG	33.97	16.86	29.88	26.90	28.17	87.77	27.76
		Adaptive	33.85	16.81	29.83	26.83	27.67	87.80	28.00
AraSum	SFT	–	30.89	14.43	25.19	23.50	**26.43**	86.20	26.16
	SFT + RL	ROUGE	**31.23**	**14.76**	**25.57**	**23.85**	26.13	86.30	26.87
		METEOR	31.13	14.56	25.43	23.71	26.37	86.25	26.08
		BERTScore	30.74	14.60	25.32	23.55	24.61	**86.35**	26.47
		NLI	31.04	14.64	25.55	23.74	26.10	86.27	**28.59**
		AVG	31.14	14.75	**25.57**	23.82	25.71	**86.35**	27.09
		Adaptive	31.05	14.56	25.41	23.67	26.08	86.26	27.27

We observe that, generally, the RL models surpass the supervised baseline in all metrics on both test sets. As expected, optimizing for a specific automatic metric-based reward results in the most significant improvement in the corresponding metric.

However, METEOR is an exception, as optimizing for any reward does not yield improvements in this metric. This might be due to its unique computation method, making it challenging to directly optimize for.

Notably, almost all rewards lead to enhanced factual consistency, as measured by the NLI classifier, with the NLI signal providing the most substantial increase. This highlights the effectiveness of our RL-based approach in producing more factually accurate summaries.

Interestingly, using textual entailment as a reward results in higher lexical overlap and semantic similarity, as reflected in the ROUGE and BERTScore metrics, respectively. This can be attributed to the fact that optimizing for entailment encourages the model to produce token sequences that are more closely aligned with the ground-truth summaries.

Moreover, we find that using an average of rewards yields balanced improvements. Besides, employing a linear reward combination with adaptive weights does not lead to better all-around performance compared to a simple reward average.

To examine the training dynamics of our RL models, we provide the plots in Figure 5.2.

These visualizations illustrate four critical aspects of the RL training process: the reward evolution, the return[1], the KL divergence (between the RL policy and the initial SFT model), and the loss.

Observing the charts in Figure 5.2, we see that the reward fluctuates throughout the optimization period, which is typical in RL-based training due to inherent exploration. The return plot shows a clear upward trend, indicating that our RL policies are learning effectively as training progresses.

The KL divergence subfigure reveals stable training for most rewards, though the BERTScore signal exhibits some instability, especially toward the end.

Despite not showing a decreasing pattern, the loss maintains a stable progression with minor fluctuations and no significant spikes. This is expected in RL-based training, where often the loss is not indicative of actual performance, unlike in supervised learning scenarios.

5.2.1 Extractiveness & Abstractiveness Analysis

To gain a more nuanced understanding of our summarization models' performance, we examine their extractiveness and abstractiveness properties. This analysis complements traditional evaluation metrics and provides information on other important summarization aspects.

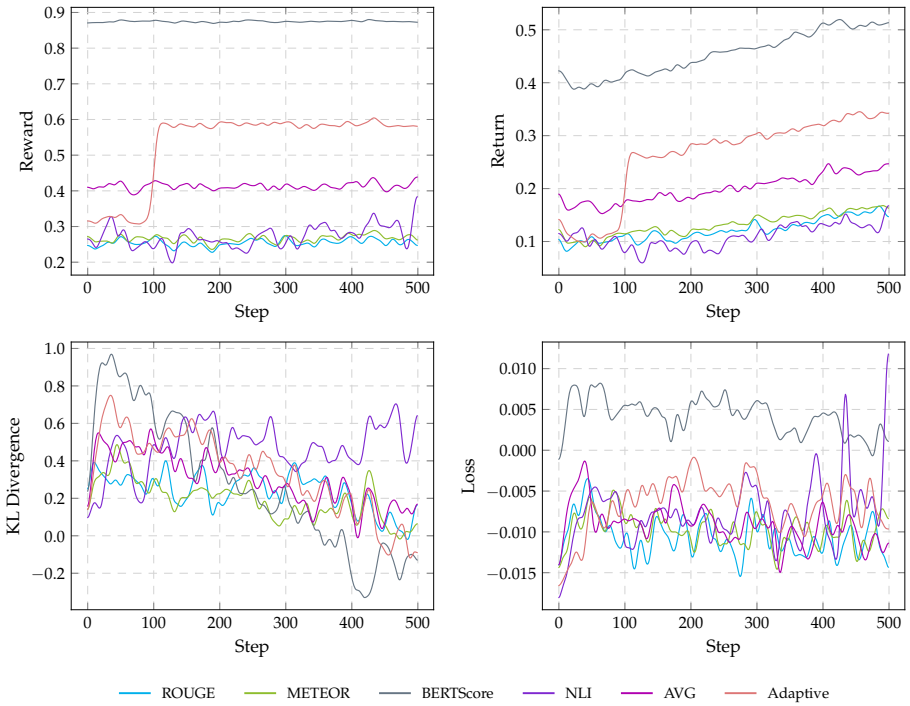
Extractiveness, measured by coverage and density, indicates the extent to which generated summaries directly incorporate content from the source text. High extractiveness suggests the model closely adheres to the original wording. Conversely, abstractiveness, represented by the percentage of novel n-grams in the summaries w.r.t the source text, reflects the model's ability to paraphrase and generate new language constructions.

Table 5.3 presents the extractiveness and abstractiveness statistics for our SFT and SFT + RL models. We also report the percentage of repeated n-grams within generated summaries as well as the compression ratio between the original text and the produced summary. Due to the computational intensity of these metrics, we used random subsets of 500 examples from each test set.

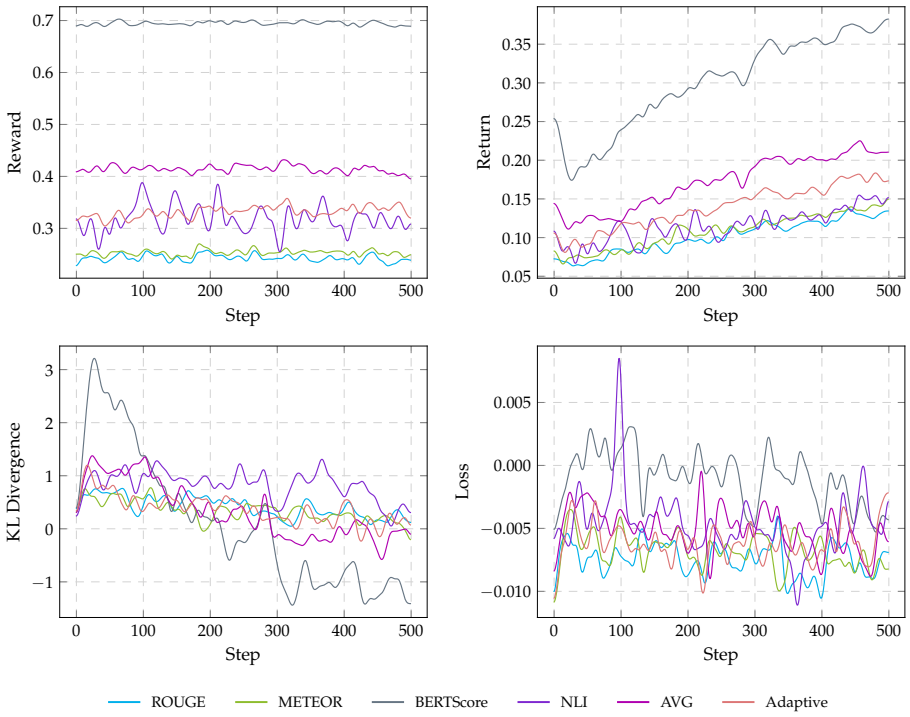
According to Table 5.3, the RL-generated summaries generally have lower extractiveness levels compared to those produced by the SFT model, on both datasets. The RL methods, especially those using BERTScore as reward, increase the novelty of bigrams and trigrams, indicating more abstract summaries along with achieving the lowest repetition rates. Additionally, the RL methods tend to have higher compression rates, suggesting more concise summaries. These trends are similar across both datasets.

On AraSum, the summaries are more extractive across all methods in comparison with

[1] The "return" here refers to the TD(λ) estimation. Please refer to (Huang et al., 2022) for more details.



(a) Training dynamics on XL-Sum.



(b) Training dynamics on AraSum.

Figure 5.2: Training dynamics of RL models illustrated by reward, return, KL divergence, and loss curves for (a) XL-Sum and (b) AraSum datasets.

Table 5.3: Extractiveness, abstractiveness, repeated n-grams percentage, and compression ratio statistics on random 500-example subsets from the XL-Sum and AraSum test sets. The best scores are marked in bold. ↓: *lowerisbetter*.

Test set	Method	Reward	Extractiveness		Novel n-grams		Repeated n-grams ↓		Compression
			Coverage	Density	$n=2$	$n=3$	$n=2$	$n=3$	
XL-Sum	SFT	–	72.93	2.26	63.98	80.16	1.54	0.92	14.75
	SFT + RL	ROUGE	72.40	2.18	**64.70**	80.82	1.14	0.69	15.90
		METEOR	72.70	2.26	63.90	80.22	1.26	0.72	15.59
		BERTScore	72.47	2.13	64.61	**80.87**	**0.80**	**0.42**	17.66
		NLI	73.00	2.23	64.05	80.58	1.12	0.70	16.67
		AVG	72.75	2.21	64.15	80.45	1.07	0.71	16.37
		Adaptive	72.76	2.20	64.21	80.43	0.84	0.44	16.88
AraSum	SFT	–	93.40	10.76	22.91	34.32	4.29	3.14	9.04
	SFT + RL	ROUGE	93.20	10.69	23.34	34.58	4.08	3.02	9.56
		METEOR	93.33	10.74	23.26	34.52	3.83	2.74	9.23
		BERTScore	92.79	10.33	23.99	35.38	**2.81**	**1.91**	10.26
		NLI	92.54	10.36	**24.50**	**35.88**	3.88	2.79	9.35
		AVG	93.27	10.39	23.48	35.09	3.88	2.78	9.74
		Adaptive	93.23	10.35	23.66	35.13	3.91	2.78	9.35

those of XL-Sum. These results align with the inherent characteristics of the datasets, demonstrating that our models, both supervised and reinforced, have effectively learned to generate summaries that mimic the style of the training data.

These findings demonstrate the effectiveness of our RL approach in producing more abstractive, concise summaries, while also reducing redundancy. Importantly, this is achieved without compromising performance, as shown in Table 5.2.

5.3 Test-Only Transfer

In order to assess the generalizability of ours models, we take our SFT and SFT + RL models finetuned on the AraSum dataset and test them directly on a new, unseen dataset, in a test-only transfer setup. We use a random 500-example subset from the Arabic News Articles dataset sourced from Aljazeera.net website. We employ the models trained on AraSum since it is the closest in characteristics to the new test set. We report the results in Table 5.4.

From the table, it is evident that the RL-optimized models attain superior scores relative to the SFT baseline, in all metrics. The model trained using BERTScore as reward achieves the highest scores in terms of ROUGE, BERTScore, and factual consistency. Furthermore, all rewards boost faithfulness, with the BERTScore-based reward showing a 5.19% increase over SFT. The NLI reward-based model also demonstrates strong generalization ability, especially in terms of METEOR. Surprisingly, in this setup, multiple rewards lead to improvements in this metric, unlike the case of our main experiments (Section 5.2).

Table 5.4: Test-only transfer results for SFT and SFT + RL models on a random subset from Aljazeera News Articles dataset. Highest scores are marked in bold while the second-best values are underlined.

Method	Reward	ROUGE				METEOR	BERTScore	Fact. cons. (NLI)
		R-1	R-2	R-L	R-Avg			
SFT	–	52.40	40.53	49.56	47.50	53.99	90.37	26.03
SFT + RL	ROUGE	53.52	41.70	50.82	48.68	<u>54.93</u>	<u>90.58</u>	27.93
	METEOR	52.89	41.02	50.10	48.00	53.92	90.49	29.18
	BERTScore	**53.76**	**41.79**	**50.96**	**48.84**	53.36	**90.76**	**31.22**
	NLI	<u>53.60</u>	<u>41.72</u>	<u>50.91</u>	<u>48.74</u>	**54.96**	90.57	28.68
	AVG	53.24	41.28	50.52	48.35	54.38	90.53	27.80
	Adaptive	53.12	40.95	50.22	48.10	54.00	90.54	<u>29.82</u>

These results indicate that RL-finetuning produces more generalizable summarization models, capable of learning better transferable skills on unseen datasets.

5.4 Ablative Analysis

In this section, we present and analyze the results of our ablation studies, which were conducted to evaluate the impact of two key components in our approach: (1) supervised finetuning and (2) the KL regularization coefficient. For these experiments, we chose the AraSum dataset since it is relatively more diverse, providing a robust testing environment. We focused on the reward average (AVG) as a representative signal for all the rewards utilized in our main experiments.

5.4.1 Impact of Supervised Finetuning

To investigate the role and necessity of the supervised finetuning phase, we conducted an ablation experiment where we directly initialized our RL policy with the pretrained language model, without prior finetuning. The training dynamics resulting from this ablation are illustrated in Figure 5.3.

Upon analyzing the plots, we observe a sharp decrease in the reward for the model initialized without supervised finetuning compared to the policy after SFT (with values not exceeding 0.03 vs. 0.43 for the policy initialized with the SFT model). This decline is particularly evident in the return plot, where values drop below zero, indicating a significant performance degradation. The KL divergence plot shows severe instability throughout the training period. Values range between very low and high, suggesting that the policy has deviated considerably from the reference model. Furthermore, we notice an increasing trend regarding the loss, which indicates the model's failure to converge or learn effectively.

In addition to the observations from the training dynamics, we found that the policy

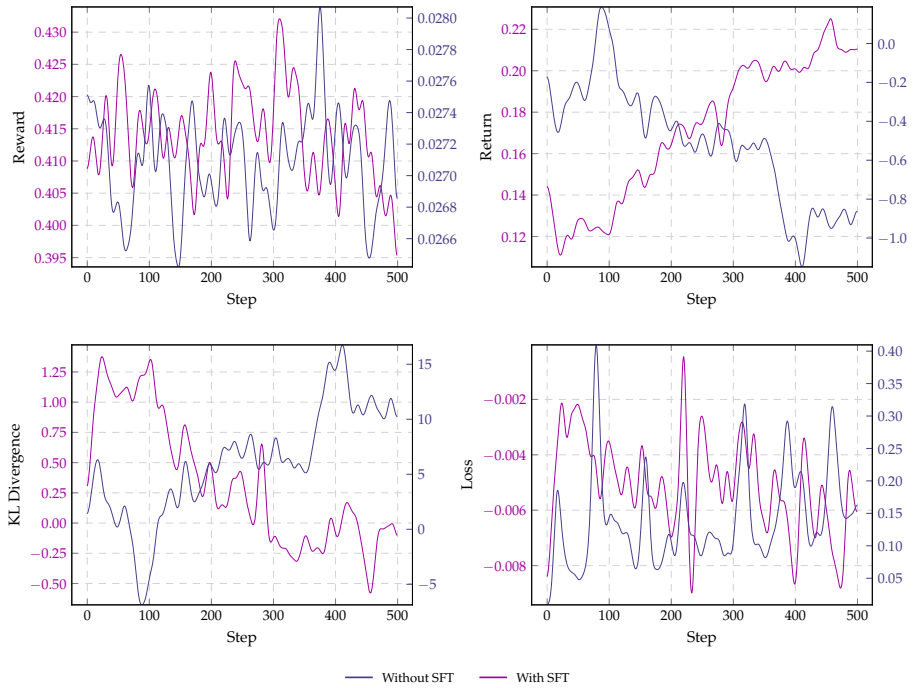


Figure 5.3: Training dynamics for the RL policy initialized with and without the SFT phase.

without SFT produces totally unintelligible generations. This means that the RL model completely loses its language modeling skills if it is not warm-started with a capable model a priori.

These results suggest the necessity of supervised finetuning and its complementary role in preserving the policy's language modeling abilities, as well as establishing foundational summarization capabilities. It serves as an indispensable part, allowing for successful task-specific optimization through reinforcement learning.

5.4.2 Impact of KL Regularization Coefficient

Another important component of our approach is the KL penalty coefficient β. An interesting question is how different values of this regularization hyperparameter affect the performance and training stability of the RL-optimized models. To investigate this, we conducted a series of experiments where we systematically scaled up the KL coefficient and monitored both the evaluation results, in terms of automatic metrics, and the training dynamics.

Table 5.5 details the results for different KL penalty values, while Figure 5.4 shows the policy behavior during the training process. Generally, we see that higher scores correlate with less regularization but increased instability as shown in the KL divergence plot, where the policy drifts farther away from the reference model. In contrast to other metrics, METEOR seems to benefit more from a stronger penalty. According to these results, the value 0.1 strikes a good balance between both overall performance and training stability. This emphasizes

Table 5.5: Ablation study results for SFT + RL models with varying KL strengths (β) on the AraSum test set, using AVG reward. Highest scores are marked in bold.

KL coef. β	ROUGE				METEOR	BERTScore	Fact. cons. (NLI)
	R-1	R-2	R-L	R-Avg			
0.05	31.05	**14.80**	**25.59**	23.81	24.68	**86.43**	**29.00**
0.1	**31.14**	14.75	25.57	**23.82**	25.71	86.35	27.09
0.2	30.95	14.50	25.33	23.59	25.86	86.25	26.49
0.5	30.93	14.45	25.26	23.55	26.28	86.22	26.59
0.8	30.92	14.49	25.26	23.56	**26.32**	86.22	26.02

the importance of carefully choosing the KL regularization coefficient through empirical testing.

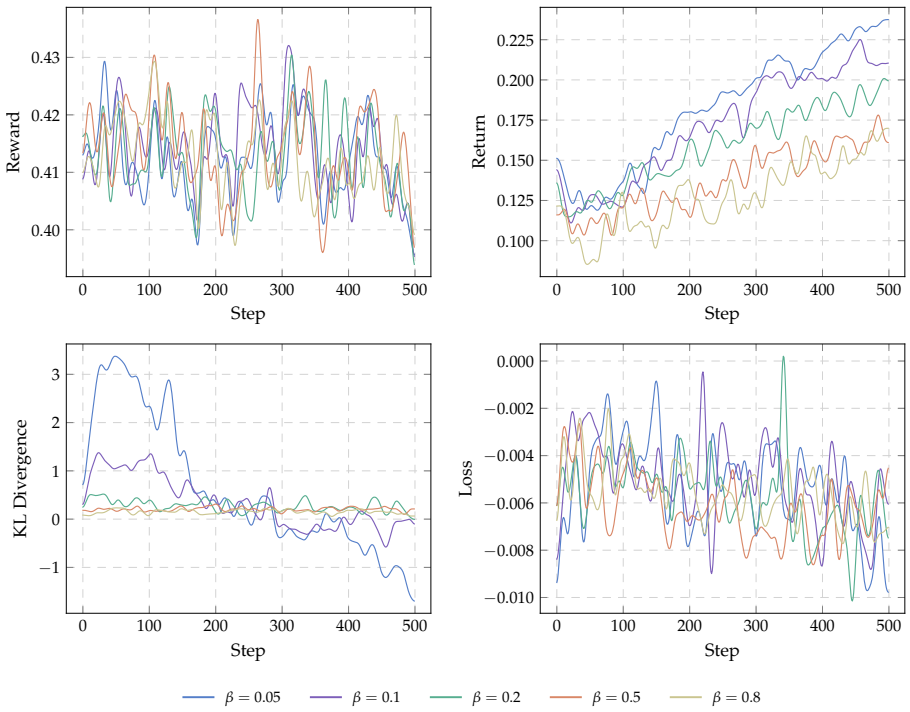


Figure 5.4: Training dynamics for different KL penalty coefficients (β). We use the AVG reward on the AraSum training set.

5.5 Comparison with Prior Work

This section compares our proposed approach against all previous works on the XL-Sum and AraSum benchmark datasets. We summarize the results in Table 5.6.

In terms ROUGE, the supervised finetuned model shows highly comparable performance

Table 5.6: Comparison of our results with SOTA works on the XL-Sum and AraSum datasets. For each test set, the highest scores are given in bold while the second-best values are underlined. *This score was taken from the paper of Kamal Eddine et al. (2022), and was not reported in the original paper.

Work	Method	ROUGE				METEOR	BERTScore	Fact. cons. (NLI)
		R-1	R-2	R-L	R-Avg			
		XL-Sum test set						
Hasan et al. (2021)	SFT (mT5)	33.23	13.74	27.84	24.94	-	65.80*	-
Kamal Eddine et al. (2022)	SFT (AraBART)	**34.50**	14.60	**30.50**	26.53	-	67.00	-
Kahla et al. (2023)	SFT (mT5)	29.12	11.04	24.07	21.41	-	-	-
Alqahtani & Al-Yahya (2023)	SFT (AraBART)	31.77	**18.81**	29.63	26.74	-	78.84	-
Ours	SFT (AraT5)	33.97	16.71	29.77	26.82	**28.85**	87.68	25.22
	SFT + RL (ROUGE)	34.22	17.00	30.11	**27.11**	28.39	87.83	27.00
	SFT + RL (METEOR)	34.07	16.80	29.89	26.92	28.50	87.73	25.81
	SFT + RL (BERTScore)	33.80	16.91	29.93	26.88	27.42	**87.85**	28.77
	SFT + RL (NLI)	33.93	16.87	29.96	26.92	27.86	87.80	**29.79**
	SFT + RL (AVG)	33.97	16.86	29.88	26.90	28.17	87.77	27.76
	SFT + RL (Adaptive)	33.85	16.81	29.83	26.83	27.67	87.80	28.00
		AraSum test set						
Kahla et al. (2023)	SFT (mT5)	**32.85**	13.84	24.57	23.76	-	-	-
Ours	SFT (AraT5)	30.89	14.43	25.19	23.50	**26.43**	86.20	26.16
	SFT + RL (ROUGE)	31.23	**14.76**	**25.57**	**23.85**	26.13	86.30	26.87
	SFT + RL (METEOR)	31.13	14.56	25.43	23.71	26.37	86.25	26.08
	SFT + RL (BERTScore)	30.74	14.60	25.32	23.55	24.61	**86.35**	26.47
	SFT + RL (NLI)	31.04	14.64	25.55	23.74	26.10	86.27	**28.59**
	SFT + RL (AVG)	31.14	14.75	**25.57**	23.82	25.71	**86.35**	27.09
	SFT + RL (Adaptive)	31.05	14.56	25.41	23.67	26.08	86.26	27.27

to existing works on both datasets, making it indeed a strong baseline. On XL-Sum, all of our models achieve a superior ROUGE-Avg, with the ROUGE-based RL model attaining the highest value (27.11% vs. SOTA 26.74%). This indicates a more balanced performance while maintaining competitiveness across other ROUGE variants.

On AraSum, the SFT model surpasses the work by Kahla et al. (2023) in terms of ROUGE-2 and ROUGE-L, but falls behind in ROUGE-1 (which is relies on simpler 1-gram overlap). The RL models using ROUGE and AVG reward signals improve upon the baseline and achieve a more balanced general ROUGE score, outperforming existing work on ROUGE-Avg.

Regarding BERTScore, ours models surpass all previous works by a large margin, establishing a new state-of-the-art on the XL-Sum dataset. The RL-optimized models, particularly the one leveraging BERTScore as reward, score the highest on this metric (87.85% vs. the previous SOTA score of 78.84%). It is worth mentioning that BERTScore has better correlation with human judgements than lexical metrics such as ROUGE (T. Zhang et al., 2020).

For better clarity, a comparison of our best ROUGE-Avg and BERTScore values with SOTA scores on XL-Sum are depicted in the bar plot of Figure 5.5.

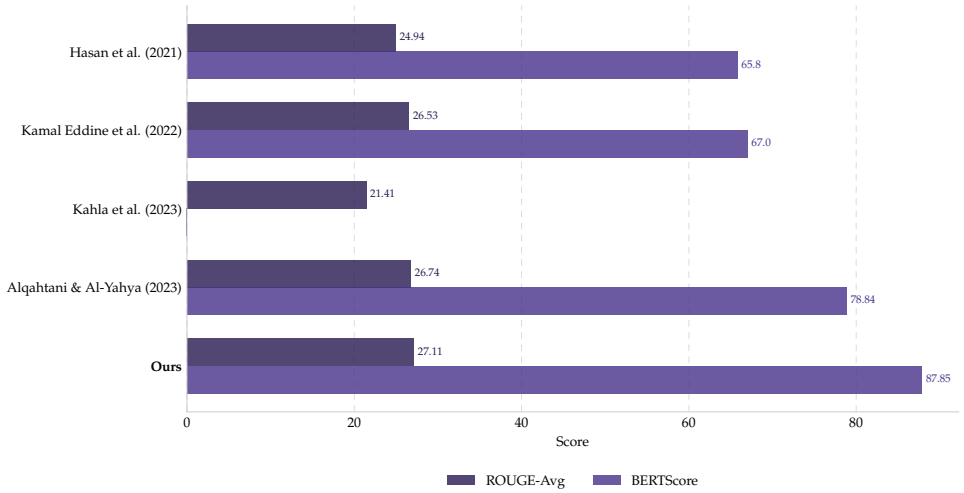


Figure 5.5: Comparison of our best scores in terms of ROUGE-Avg and BERTScore on XL-Sum with those of SOTA works.

Furthermore, our introduction of factual consistency assessment using natural language inference (NLI) addresses some limitations of traditional metrics. The NLI-optimized model achieves the highest factual consistency score among our variants, complementing the strong BERTScore performance. Within our knowledge, this is the first work employing textual entailment (NLI) to evaluate the faithfulness of generated summaries in Arabic ATS research.

5.6 Conclusion

This chapter has presented the empirical findings of our research. We have compared various models trained using our proposed RL-based approach and analyzed the obtained results. Our findings demonstrate that our RL-finetuned models generally outperform the purely supervised baseline across multiple evaluation metrics, thus validating the effectiveness of our approach in enhancing Arabic abstractive summarization models. Furthermore, evaluation on a new dataset in a test-only transfer setup revealed the improved generalizability of our RL-models. An extensive ablative analysis regarding the necessity of the SFT phase and the KL regularization coefficient were also covered. Moreover, when compared with state-of-the-art methods, our approach shows superior performance, particularly in BERTScore, while maintaining a competitive and more balanced overall performance in terms of ROUGE.

6

Conclusion

In this thesis, our primary objective was to explore reinforcement learning as a finetuning paradigm alongside supervised transfer learning for the task of Arabic abstractive text summarization. To this end, we proposed a two-phase approach combining supervised finetuning of pretrained transformer-based language models with deep reinforcement learning.

In the first phase, our framework involved finetuning a pretrained LM to adapt it to the abstractive summarization task. We experimented with two pretrained encoder-decoder LMs: AraT5 and AraBART. We finetuned these models on the XL-Sum and AraSum benchmark Arabic ATS datasets. This phase aimed at establishing a strong baseline that serves both as a warm-starting checkpoint for our RL policies, and as a baseline for comparison with the RL-finetuned models. The results of this phase show that AraT5 outperforms AraBART on both datasets, and thus was chosen to be further optimized in the subsequent RL stage.

In the second phase, we further finetuned our SFT model via reinforcement learning using the proximal policy optimization algorithm (PPO). We leveraged different rewards derived from automatic evaluation metrics, capturing lexical overlap, semantic similarity, as well as a textual entailment. Specifically, we leveraged ROUGE, METEOR, BERTScore, and an NLI classifier as reward signals. This latter was utilized to encourage our models to generate more faithful summaries. In addition to that, we experimented with reward combinations, including a reward average and an adaptive reward using a linear combination with dynamically adjusted weights.

Our findings demonstrate the effectiveness of the proposed RL-based method. In most evaluation scenarios, our RL+SFT models outperform the supervised baseline across multiple evaluation metrics, while also demonstrating improved factual consistency, as measured by our NLI classification model. For instance, we observed a +4% improvement in factual accuracy compared to the baseline model on XL-Sum. The summaries generated by our

approach also proved to be of higher abstractiveness, conciseness, and have lower repetition rates, enhancing overall quality and readability. Moreover, our RL-finetuned models are more generalizable to new, unseen datasets. This shows that our approach is more robust than purely supervised methods and leads to models with enhanced transfer abilities.

Comparison with state-of-the-art methods demonstrates the superiority of our approach over all previous works in terms of BERTScore, a metric that has higher correlation with human judgments than traditional metrics. Results also show that our RL-finetuned models attain competitive and more balanced ROUGE scores.

To investigate critical components of our approach, we conducted two key ablation studies. The first examined the role of the supervised finetuning (SFT) phase. Our findings revealed that SFT plays an indispensable role, serving as a important foundation for effective RL-based optimization. The second ablation study explored the effect of the KL regularization coefficient, which we employed to constrain our RL policies from diverging too far from the initial SFT model. We observed that, generally, lower penalty strengths led to higher metric scores. However, this improvement came with a trade-off: decreased training stability and an increased risk of reward hacking. Specifically, as the penalty weakened, the policy showed a tendency to drift further from the initial SFT (reference) model, potentially leading to less coherent or factual summaries despite higher automated metric scores.

Even though our results are quite promising, there is always room for improvement and exploration. In light of this, we propose the following directions for future work:

- Experimenting with causal (decoder-only) large language models (LLMs) by utilizing them as a base model in our proposed methods. This approach could benefit from their general capabilities across various generative language tasks, potentially improving performance and generalizability in Arabic summarization.
- Extending the application of our approach to other Arabic natural language generation tasks, such as Arabic abstractive question answering, and even multimodal tasks such as Arabic image captioning.
- As the evaluation of summarization systems remains an open area of research, we plan to develop improved evaluation metrics that exhibit higher correlation with human judgments. These new metrics could serve dual purposes: providing more accurate assessment of summarization quality and potentially serving as improved reward functions within our RL framework.

This page was intentionally left blank.

References

Aksenov, D., Moreno-Schneider, J., Bourgonje, P., Schwarzenberg, R., Hennig, L., & Rehm, G. (2020). *Abstractive Text Summarization based on Language Model Conditioning and Locality Modeling*. arXiv: 2003.13027 [cs].

Alahmadi, D., Wali, A., & Alzahrani, S. (2022). TAAM: Topic-aware abstractive arabic text summarisation using deep recurrent neural networks. *Journal of King Saud University - Computer and Information Sciences*, *34*, 2651–2665. https://doi.org/10.1016/j.jksuci.2022.03.026

Al-Huri, I. (2016). *Arabic Language: Historic and Sociolinguistic Characteristics*. https://doi.org/10.13140/RG.2.2.16163.66089/1

AlMahamid, F., & Grolinger, K. (2021). Reinforcement Learning Algorithms: An Overview and Classification. *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–7. https://doi.org/10.1109/CCECE53047.2021.9569056

Al-Maleh, M., & Desouki, S. (2020). Arabic text summarization using deep learning approach. *Journal of Big Data*, *7*(1), 109. https://doi.org/10.1186/s40537-020-00386-7

Alomari, A., Idris, N., Sabri, A. Q. M., & Alsmadi, I. (2022). Deep reinforcement and transfer learning for abstractive text summarization: A review. *Computer Speech & Language*, *71*, 101276. https://doi.org/10.1016/j.csl.2021.101276

Alqahtani, D., & Al-Yahya, M. (2023). Exploring Arabic Pre-Trained Language Models for Arabic Abstractive Text Summarization. *2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 1–7. https://doi.org/10.1109/SNAMS60348.2023.10375464

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, *1142*, 012012. https://doi.org/10.1088/1742-6596/1142/1/012012

Antoun, W., Baly, F., & Hajj, H. (2020). AraBERT: Transformer-based Model for Arabic Language Understanding. *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, 9–15.

Antoun, W., Baly, F., & Hajj, H. (2021). AraGPT2: Pre-Trained Transformer for Arabic Language Generation. *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, 196–207.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv: 1409.0473 [cs, stat]. https://doi.org/10.48550/arXiv.1409.0473

Banerjee, S., & Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72.

Bani-Almarjeh, M., & Kurdy, M.-B. (2023). Arabic abstractive text summarization using RNN-based and transformer-based architectures. *Information Processing & Management*, *60*(2), 103227. https://doi.org/10.1016/j.ipm.2022.103227

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Cao, Y., Sheng, Q. Z., McAuley, J., & Yao, L. (2023). *Reinforcement Learning for Generative AI: A Survey*. arXiv: 2308.14328 [cs]. https://doi.org/10.48550/arXiv.2308.14328

Chouikhi, H., & Alsuhaibani, M. (2022). Deep Transformer Language Models for Arabic Text Summarization: A Comparison Study. *Applied Sciences*, *12*(23), 11944. https://doi.org/10.3390/app122311944

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv: 1412.3555 [cs]. https://doi.org/10.48550/arXiv.1412.3555

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Dong, H., Ding, Z., & Zhang, S. (2020). *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Singapore. https://doi.org/10.1007/978-981-15-4095-0

El-Haj, M., & Koulali, R. (2013). KALIMAT a multipurpose Arabic corpus.

El-Haj, M., Kruschwitz, U., & Fox, C. (2010). Using Mechanical Turk to Create a Corpus of Arabic Summaries.

Elmadani, K. N., Elgezouli, M., & Showk, A. (2020). *BERT Fine-tuning For Arabic Text Summarization*. arXiv: 2004.14135 [cs]. https://doi.org/10.48550/arXiv.2004.14135

Elsaid, A., Mohammed, A., Ibrahim, L. F., & Sakre, M. M. (2022). A Comprehensive Review of Arabic Text Summarization. *IEEE Access*, *10*, 38012–38030. https://doi.org/10.1109/ACCESS.2022.3163292

Etaiwi, W., & Awajan, A. (2022). SemG-TS: Abstractive Arabic Text Summarization Using Semantic Graph Embedding. *Mathematics*, *10*(18), 3225. https://doi.org/10.3390/math10183225

Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., & Radev, D. (2021). SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, *9*, 391–409. https://doi.org/10.1162/tacl_a_00373

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Grusky, M., Naaman, M., & Artzi, Y. (2018). Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 708–719. https://doi.org/10.18653/v1/N18-1065

Gunasekara, C., Feigenblat, G., Sznajder, B., Aharonov, R., & Joshi, S. (2021). Using Question Answering Rewards to Improve Abstractive Summarization. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 518–526. https://doi.org/10.18653/v1/2021.findings-emnlp.47

Hasan, T., Bhattacharjee, A., Islam, M. S., Mubasshir, K., Li, Y.-F., Kang, Y.-B., Rahman, M. S., & Shahriyar, R. (2021). XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 4693–4703. https://doi.org/10.18653/v1/2021.findings-acl.413

He, P., Gao, J., & Chen, W. (2023). *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. arXiv: 2111.09543 [cs].

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., & Wang, W. (2022). The 37 implementation details of proximal policy optimization [https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/]. *ICLR Blog Track*.

Inoue, G., Alhafni, B., Baimukan, N., Bouamor, H., & Habash, N. (2021). The Interplay of Variant, Size, and Task Type in Arabic Pre-trained Language Models. *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, 92–104.

Kahla, M., Novák, A., & Yang, Z. G. (2023). Fine-tuning and multilingual pre-training for abstractive summarization task for the Arabic language. *Annales Mathematicae et Informaticae*, 24–35.

Kahla, M., Yang, Z. G. H., & Novák, A. (2021). Cross-lingual Fine-tuning for Abstractive Arabic Text Summarization. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 655–663.

Kamal Eddine, M., Tomeh, N., Habash, N., Le Roux, J., & Vazirgiannis, M. (2022). AraBART: A Pretrained Arabic Sequence-to-Sequence Model for Abstractive Summarization. *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, 31–42. https://doi.org/10.18653/v1/2022.wanlp-1.4

Khan, B., Shah, Z. A., Usman, M., Khan, I., & Niazi, B. (2023). Exploring the Landscape of Automatic Text Summarization: A Comprehensive Survey. *IEEE Access*, *11*, 109819–109840. https://doi.org/10.1109/ACCESS.2023.3322188

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

Ladhak, F., Durmus, E., Cardie, C., & McKeown, K. (2020). *WikiLingua: A New Benchmark Dataset for Cross-Lingual Abstractive Summarization* (1). arXiv: 2010.03093 [cs].

Laurer, M., Atteveldt, W. van, Casas, A., & Welbers, K. (2024). Less Annotating, More Classifying: Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT-NLI. *Political Analysis*, *32*(1), 84–100. https://doi.org/10.1017/pan.2023.20

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out*, 74–81.

Liu, Y. [Yang]. (2019). *Fine-tune BERT for Extractive Summarization*. arXiv: 1903.10318 [cs].

Liu, Y. [Yinhan], Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., & Zettlemoyer, L. (2020). Multilingual Denoising Pre-training for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*, *8*, 726–742. https://doi.org/10.1162/tacl_a_00343

Loshchilov, I., & Hutter, F. (2019). DECOUPLED WEIGHT DECAY REGULARIZATION.

Lu, X., Van Roy, B., Dwaracherla, V., Ibrahimi, M., Osband, I., & Wen, Z. (2023). *Reinforcement Learning, Bit by Bit*. arXiv: 2103.04047 [cs]. https://doi.org/10.48550/arXiv.2103.04047

Nagoudi, E. M. B., Elmadany, A., & Abdul-Mageed, M. (2022). AraT5: Text-to-Text Transformers for Arabic Language Generation. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 628–647. https://doi.org/10.18653/v1/2022.acl-long.47

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2024). *A Comprehensive Overview of Large Language Models*. arXiv: 2307.06435 [cs]. https://doi.org/10.48550/arXiv.2307.06435

Nenkova, A. (2011). Automatic Summarization. *Foundations and Trends® in Information Retrieval*, *5*(2), 103–233. https://doi.org/10.1561/1500000015

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs]. https://doi.org/10.48550/arXiv.2203.02155

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. https://doi.org/10.3115/1073083.1073135

Parker, Robert, Graff, David, Chen, Ke, Kong, Junbo, & Maeda, Kazuaki. (2011). *Arabic Gigaword Fifth Edition*. https://doi.org/10.35111/P02G-RW14

Paulus, R., Xiong, C., & Socher, R. (2018). A Deep Reinforced Model for Abstractive Summarization. *6th International Conference on Learning Representations*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2016). *Sequence Level Training with Recurrent Neural Networks*. arXiv: 1511.06732 [cs]. https://doi.org/10.48550/arXiv.1511.06732

Reda, A., Salah, N., Adel, J., Ehab, M., Ahmed, I., Magdy, M., Khoriba, G., & Mohamed, E. H. (2022). A Hybrid Arabic Text Summarization Approach based on Transformers. *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, 56–62. https://doi.org/10.1109/MIUCC55081.2022.9781694

Rhouati, A. (2019). *Arabic News Articles from Aljazeera.net*.

Roit, P., Ferret, J., Aharoni, R., Cideron, G., Dadashi, R., Geist, M., Girgin, S., Hussenot, L., Keller, O., Momchev, N., Ramos Garea, S., Stanczyk, P., Vieillard, N., Bachem, O., Elidan, G., Hassidim, A., Pietquin, O., & Szpektor, I. (2023). Factually Consistent Summarization via Reinforcement Learning with Textual Entailment Feedback. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6252–6272. https://doi.org/10.18653/v1/2023.acl-long.344

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*.

Schmidt, R. M. (2019). *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. arXiv: 1912.05911 [cs, stat]. https://doi.org/10.48550/arXiv.1912.05911

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs]. https://doi.org/10.48550/arXiv.1707.06347

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., & Christiano, P. F. (2020). Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, *33*, 3008–3021.

Suleiman, D., & Awajan, A. (2020). Deep learning based abstractive Arabic text summarization using two layers encoder and one layer decoder. *Journal of Theoretical and Applied Information Technology*, *98*, 3233.

Suleiman, D., & Awajan, A. (2022). Multilayer encoder and single-layer decoder for abstractive Arabic text summarization. *Knowledge-Based Systems*, *237*, 107791. https://doi.org/10.1016/j.knosys.2021.107791

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks*. arXiv: 1409.3215 [cs].

Sutton, R. S., & Barto, A. (2020). *Reinforcement Learning: An Introduction* (Second edition). The MIT Press.

Uc-Cetina, V., Navarro-Guerrero, N., Martin-Gonzalez, A., Weber, C., & Wermter, S. (2023). Survey on reinforcement learning for language processing. *Artificial Intelligence Review*, *56*(2), 1543–1575. https://doi.org/10.1007/s10462-022-10205-5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in Neural Information Processing Systems*, *28*.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., & Huang, S. (2020). Trl: Transformer reinforcement learning.

Wazery, Y., Saleh, M. E., Alharbi, A., & Ali, A. A. (2022). Abstractive Arabic Text Summarization Based on Deep Learning. *Computational Intelligence and Neuroscience*, *2022*, 1–14. https://doi.org/10.1155/2022/1566890

Weng, L. (2018). Attention? attention! *lilianweng.github.io*.

Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., Affandy, A., & Setiadi, D. R. I. M. (2022). Review of automatic text summarization techniques & methods. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1029–1046. https://doi.org/10.1016/j.jksuci.2020.05.006

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., … Rush, A. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 483–498. https://doi.org/10.18653/v1/2021.naacl-main.41

Zaki, A. M., Khalil, M. I., & Abbas, H. M. (2019). Deep Architectures for Abstractive Text Summarization in Multiple Languages. *2019 14th International Conference on Computer Engineering and Systems (ICCES)*, 22–27. https://doi.org/10.1109/ICCES48960.2019.9068171

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *Proceedings of the 37th International Conference on Machine Learning*, 11328–11339.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., Xiong, L., Chen, L., Xi, Z., Xu, N., Lai, W., Zhu, M., Chang, C., Yin, Z., Weng, R., … Huang, X. (2023). *Secrets of RLHF in Large Language Models Part I: PPO*. arXiv: 2307.04964 [cs].

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). *A Comprehensive Survey on Transfer Learning*. arXiv: 1911.02685 [cs, stat]. https://doi.org/10.48550/arXiv.1911.02685

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., & Irving, G. (2020). *Fine-Tuning Language Models from Human Preferences*. arXiv: 1909.08593 [cs, stat].

Zitouni, I. (2014). *Natural Language Processing of Semitic Languages*. Springer. https://doi.org/10.1007/978-3-642-45358-8