

**Автономная некоммерческая организация высшего образования  
«Университет Иннополис»**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
(БАКАЛАВРСКАЯ РАБОТА)  
по направлению подготовки  
09.03.01 - «Информатика и вычислительная техника»**

**GRADUATION THESIS  
(BACHELOR'S GRADUATION THESIS)  
Field of Study  
09.03.01 – «Computer Science»**

**Направленность (профиль) образовательной программы  
«Информатика и вычислительная техника»  
Area of Specialization / Academic Program Title:  
«Computer Science»**

**Тема /  
Topic**

**Определение текстового плагиата в области больших  
языковых моделей с использованием обучения с  
подкреплением /  
Text plagiarism detection in the field of large language models  
using the reinforcement learning**

**Работу выполнил /  
Thesis is executed by**


**Береснев Дмитрий  
Владимирович /  
Dmitry Beresnev**

  
подпись / signature

**Руководитель  
выпускной  
квалификационной  
работы /  
Supervisor of  
Graduation Thesis**

**Бекларян Армен Левонович  
/  
Armen Beklaryan**

подпись / signature



# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Considered text plagiarism types . . . . .	9
1.2	The relevance of applying DRL to text plagiarism detection . . .	10
1.3	Deep reinforcement learning approaches . . . . .	11
1.3.1	Q-learning . . . . .	11
1.3.2	Policy Gradients . . . . .	12
1.4	Limitations and implications . . . . .	12
<b>2</b>	<b>Literature Review</b>	<b>13</b>
2.1	Search process . . . . .	13
2.2	Existing approaches . . . . .	14
2.2.1	Policy Gradients approach . . . . .	14
2.2.2	Deep Q-network approach . . . . .	15
2.2.3	Ensemble approach . . . . .	15
2.3	Conclusions and insights . . . . .	16
<b>3</b>	<b>Methodology</b>	<b>17</b>
3.1	Research design . . . . .	17
3.2	Data collection . . . . .	18

---

3.2.1	Base dataset selection . . . . .	19
3.2.2	Plagiarism generation . . . . .	19
3.2.3	Dataset compilation . . . . .	25
3.3	Solution architecture . . . . .	30
3.3.1	Decision Network (DNet) . . . . .	31
3.3.1.1	REINFORCE . . . . .	33
3.3.1.2	Advantage Actor-Critic (A2C) . . . . .	36
3.3.1.3	Deep Q-Network (DQN) . . . . .	39
3.3.2	Structured representation model (SRM) . . . . .	40
3.3.3	Regression Network (RNet) . . . . .	42
<b>4</b>	<b>Evaluation and Discussion</b>	<b>44</b>
4.1	Training details . . . . .	45
4.2	Experiment settings . . . . .	45
4.3	Baselines . . . . .	48
4.4	Regression results . . . . .	50
4.5	Representations analysis . . . . .	51
4.5.1	Qualitative analysis . . . . .	53
4.5.2	Quantitative analysis . . . . .	53
4.6	Limitations and Future Work . . . . .	56
<b>5</b>	<b>Conclusion</b>	<b>58</b>
	<b>Bibliography cited</b>	<b>59</b>
<b>A</b>	<b>Additional experiments</b>	<b>65</b>

# List of Tables

I	LLMs prompts for text plagiarism generation calls . . . . .	21
II	Example of Type 1 plagiarism generation results . . . . .	24
III	Example of Type 2 plagiarism generation results . . . . .	25
IV	Train and test dataset sizes . . . . .	29
V	Sizes of different sub-datasets . . . . .	29
VI	Experiment parameters shapes . . . . .	47
VII	Number of trainable parameters of baseline and solution architectures . . . . .	50
VIII	Regression MSE loss on synthetic dataset . . . . .	51
IX	The initial average length and the obtained average length by SRM-R, SRM-A and SRM-D in test dataset . . . . .	54
X	Examples of the most and least deleted words in the test dataset	55
XI	The initial average length, mean of removed ratio and the standard deviation of obtained length by SRM-R, SRM-A and SRM-D in AG News test dataset . . . . .	65
XII	Tendency to delete words denoting the days of the week in the AG News test dataset . . . . .	66

---

XIII	Tendency to delete words denoting the brands in the AG News test dataset . . . . .	67
XIV	Tendency to delete words connected to sport in the AG News test dataset . . . . .	68
XV	Tendency to delete words denoting jobs in the AG News test dataset	69
XVI	Tendency to delete words denoting the high technologies in the AG News test dataset . . . . .	70

# List of Figures

1	Processes of text plagiarism generation . . . . .	22
2	Example of maintaining LLM session history . . . . .	23
3	Solution architecture overview . . . . .	31
4	Advantage Actor-Critic (A2C) architecture with shared body . .	37
5	Epsilon strategy for Deep Q-Network algorithm . . . . .	46
6	CNN-based baselines outline . . . . .	49
7	Examples of text representations discovered by SRM-R, SRM-A and SRM-D . . . . .	52

## **Abstract**

The active development of large language models (LLMs) and the ease of access to them are the reasons for the emergence of an enormous amount of plagiarized texts. Modern research provides several techniques to detect text plagiarism. Most of such techniques use machine learning, especially deep neural networks (DNNs). One of the possible approaches working with DNNs is to build text representation for further processing. However, constructing the effective text representation for plagiarism detection task can be challenging.

This study proposes a model for text plagiarism detection in the field of LLMs, with uses deep reinforcement learning (DRL) method to automatically learn effective text representation.

This study examines two types of text plagiarism: text paraphrased by LLM and text generated by LLM given the idea of the original text. To build text representation, the representation learning task was formulated as a sequential decision problem and the structured representation model (SRM) was designed, which extracts only task-relevant words from the text. For models training and performance evaluation, synthetic dataset was built using SOTA LLMs.

Experiments demonstrate that proposed model architectures can learn task-specific text representation by removing irrelevant words, and as achieve competitive performance against SOTA representation building approaches.

This research illustrates a new perspective approach to the text plagiarism detection problem. The proposed model architectures can increase the quality of text plagiarism detection in general, and in the field of LLMs in particular.

# Chapter 1

## Introduction

The active development of large language models (LLMs) and the ease of access to them are the reasons for the emergence of an enormous number of plagiarized texts. Modern research suggests several approaches to detecting text plagiarism, the main of which remain approaches based on mathematical statistics and approaches based on machine learning. Despite the fact that both approaches demonstrate decent performance, in recent years preference has increasingly been given to solutions based on machine learning, in particular on deep neural networks (DNNs).

One of the popular approaches to handle text using DNN is to build task-specific text representation, which is easier to work with for neural networks ([1], [2] and [3]). However, constructing the effective and task-relevant text representation can be challenging.

In case of this research, the task is text plagiarism detection, so the one of the possible ways for improvement of text representation building is to extract only relevant information from the text. This can be implemented as retaining of significant, task-relevant words and deletion of redundant words with no infor-



mation. This task can be formulated as a sequential decision problem with two actions, which can be naturally addressed to reinforcement learning (RL) and specifically to deep reinforcement learning (DRL).

The DRL techniques are designed precisely in such a way as to demonstrate adequate performance and respond appropriately in an environment with a long-term effect actions. For example, Silver et al. [4] used the reinforcement learning in AlphaGo model implementation, which became the first computer Go program to beat a human professional Go player. However, the DRL has not yet been applied in the field of text plagiarism detection.

Therefore, this study is an attempt to build and evaluate a structured representation model (SRM) to detect text plagiarism in the field of LLMs. With this purpose, DRL algorithms are used to learn efficient structured text representations, which are then passed to the subsequent neural network. By doing so, this research aims to expand the scope of DRL algorithms and develop a promising approach to text plagiarism detection.

## **1.1 Considered text plagiarism types**

The definition of text plagiarism is still a matter of dispute and discussion, especially after the active development of artificial intelligence systems that allow you to create a huge number of new texts based on existing ones. The starting point for plagiarism definition in this research is the original human-written text which is taken from open sources and public datasets and considered to have no plagiarism. Then this study examines the following types of text plagiarism, which cover the basic needs for analyzing texts generated by LLMs:

1. **Type 1 plagiarism.** The original text paraphrased by LLM is considered to have Type 1 plagiarism in relation to original text.
2. **Type 2 plagiarism.** The completely new text, fully generated by LLM given only the short extracted idea of original text is considered to have Type 2 plagiarism in relation to original text.

Also, to obtain more relevant results, several state-of-the-art (SOTA) LLMs are used to generate plagiarism of both types. The details of collecting original human-written texts, selecting LLMs, and generating plagiarized texts are described in details in the Section 3.2 of Methodology chapter.

## 1.2 The relevance of applying DRL to text plagiarism detection

In a Section 2.2 of Literature review chapter, it can be seen that application of DRL algorithms to text processing is actively studied topic. However, the number of studies in the field of applying DRL methods specifically to the text plagiarism detection is small. This, as well as the growing popularity of LLMs, highlights the prospects for research and development of DRL-based solution for text plagiarism detection in the field of LLMs.

Currently, methods based on effective text representation, embeddings, are used for text analysis and processing. However, in most of these methods text representations either have pre-defined structures ([5], [6]) or are provided as input ([7]). The use of DRL methods can be used to automatically identify task-relevant text features and build effective representations, without any pre-specified banks.

Moreover, relatively simple model used in DRL algorithm can significantly reduce size of the text by filtering out the unnecessary parts. Obtained text is then passed to the main task-specific neural network, which is usually quite complex. In this sense, usage of DRL methods potentially reduces the running time of the overall model.

Thus, this study can be considered relevant, and the chosen topic is promising for expanding the scope of DRL and further study.

## **1.3 Deep reinforcement learning approaches**

There are two most popular DRL method classes: Q-learning, which approximates the action-value function  $Q$ , and Policy Gradients, which directly learns policy of an agent. As it is demonstrated in the subsequent Literature review chapter, both method classes can be used for text processing. This research aims to explore the application of algorithms from both classes to the text plagiarism detection in the field of LLMs.

### **1.3.1 Q-learning**

Q-learning focuses on estimating the action-value function. Usually in practice, the deep neural network is used as  $Q$  function approximation. Such variant is called Deep Q-network (DQN). DQN optimizes policy indirectly by learning the value first and then using this value to choose the optimal behavior. The main advantage of the DQN algorithms is ability to benefit from data, obtained from the old policy, and from other sources, for example, human behavior. Also, DQN approach is considered to be more stable than Policy Gradients, and requires fewer

interactions with the environment.

### **1.3.2 Policy Gradients**

Policy Gradients is an alternative to Q-learning which directly optimizes the behavior of an agent. The most famous Policy Gradient algorithm is REINFORCE, which uses deep neural network for estimation the actions distribution based on the current state. Policy Gradients algorithms can learn stochastic policy, while DQN can learn only deterministic one, and handle continues action space. The main problem of Policy Gradients is high gradient variance, which can lead to slow and unstable convergence. However, there are different variance reduction methods, such as Actor-Critic and Advantage Actor-Critic (A2C) algorithms.

## **1.4 Limitations and implications**

In this study, the simplest cases of the text plagiarism are considered. The obtained results for considered cases may not directly translate to more complicated forms of text plagiarism. Further experimentation is necessary to evaluate the applicability of proposed methods in addressing complex plagiarism types. Moreover, despite employing several SOTA LLMs for dataset generation, the quality of produced dataset remains a topic open to discussion.

Nevertheless, the proposed methods have significant potential to enhance the detection of specific cases of text plagiarism by possible enhancing detection accuracy and reducing computational overhead.

# Chapter 2

## Literature Review

This chapter is structured as follows: Section 2.1 describes the method of literature search. Section 2.2 provides an overview and discussion of the existing approaches of DRL-based applications for text processing. Finally, Section 2.3 contains conclusions and insights related to this study.

### 2.1 Search process

According to the survey of DRL approaches [8], the policy-based methods and Deep Q-Network (DQN) algorithms are the most suitable for text processing tasks. Policy-based methods include Actor-Critic (AC) and Policy Gradients algorithms. Therefore, the set of search keywords was the following: Text Processing, Deep reinforcement learning, Actor-Critic, Deep Q-network, Policy Gradients, Natural Language Processing. The inclusion criterion is that the publication should describe an architecture or an algorithm for solving text processing tasks, for example text classification. Studies with low credibility level and those that provide solutions that do not compete in efficiency with state-of-the-art

(SOTA) approaches were excluded. As a result, most of the papers were taken from SpringerLink and Google Scholar.

## **2.2 Existing approaches**

The found DRL-based text processing approaches can be categorized as follows: Policy Gradients approaches, Deep Q-network approaches and ensemble approaches.

### **2.2.1 Policy Gradients approach**

Policy Gradients approach optimize parameterized policy with respect to the expected reward. The most known variant of the policy gradients method is the REINFORCE algorithm. Zhang et al. in [1] proposed two models: Information Distilled LSTM (ID-LSTM) and Hierarchical Structured LSTM (HS-LSTM), which use the REINFORCE algorithm to optimize parameters of the policy network. Both models build efficient structured text representations, which are then passed to the classification network. Authors compared the performance of both models with such baseline architectures as LSTM, biLSTM and CNN. As a result, proposed models achieve the classifications accuracy of the baselines on all used datasets. Even in cases where the proposed models did not demonstrate the best performance, their results were inferior to the best by less than 1%. Teng et al. in [9] used the ID-LSTM and HS-LSTM from Zhang's study [1] in the extreme multi-label classification task (XMC). Authors demonstrated that proposed architectures consistently produced best or the second best result against the most representative XMC methods, such as FastText [10], FastXML [11],

TextCNN [12] and SLEEC [13].

### 2.2.2 Deep Q-network approach

The DQN algorithm was presented in [14], where it achieved professional video games player scores across a classic Atari 2600 games. The DQN uses the neuron network to compactly represent high-dimensional observations and to estimate the Q-function. Therefore, the main strength of this algorithm is the ability to generalize. Lin in [15] introduces the model, which uses the DQN algorithm to solve the problem of low performance of conventional classification algorithms in the case of imbalanced data distribution. Authors tested the proposed model on image datasets, but in general this architecture can be used with different types of data, including text. Results show that the performance of the suggested model in imbalanced data sets is better than other SOTA imbalanced classification methods, such as DNN, ROS [16], RUS [16], MFE [17], CSM [18], and DTA [19].

### 2.2.3 Ensemble approach

Ensemble approach is combining multiple models into one unit called Mixture of Experts models (MoE) and aggregating their results. Li et al. in [20] combine SOTA solutions for text classification tasks, such as Naive Bayes, SVM-SGD, FastText [10], BiLSTM and TextCNN [12] into a MoE called RL-ERT. The DRL policy gradient algorithm is used in the form of weight-tuning policy network (WTPN) to generate suitable weights for results of each expert model for each sample. The results show that on the PHEME and RumorEval datasets RL-ERT demonstrates the best or second best result against all the models included

in its composition, individually and in combinations.

## 2.3 Conclusions and insights

This chapter attempted to give a summary of the most relevant DRL-based text processing approaches. The most universal technique for text processing among mentioned is building effective text representation with the use of Policy Gradients DRL methods. So, this research is an attempt to transfer idea of Zhang et al. in [1] to a more complex task and implement other algorithms, such as Advantage Actor-Critic (A2C) and DQN, to solve the problem of text plagiarism detection in the field of LLMs.

As a subject for further research, combinations of several mentioned approaches can be considered: for example, an ensemble model consisting of Policy Gradients based model and with other SOTA solutions, or improving the quality of classification on an imbalanced data set through integration of DQN into final architecture.



# Chapter 3

## Methodology

This chapter outlines the objectives of the present research and the employed methodology. Specifically, Section 3.2 examines the process of data collection and utilization of SOTA LLMs in it. Finally, Section 3.3 describes the architecture of text plagiarism detection models and provides overview of different used DRL algorithms along with the theoretical foundations.

### 3.1 Research design

The goal of this research is to build and evaluate DRL-based solution for text plagiarism detection. Therefore, the research question of this study can be formulated as following: how effective can a solution, which uses DRL methods for learning effective text representation, be for detecting text plagiarism in the field of LLMs?

To answer the stated research question, it was necessary to perform the following steps:

1. **Hypotheses formulation.** Null hypothesis is that the DRL-based solution

has no significant difference in the performance compared to SOTA architectures for text plagiarism detection. Conversely, the alternative hypothesis states that the DRL-based model can outperforms SOTA architectures in considered settings.

2. **Data collection.** Gather a diverse dataset containing original human-written texts and the corresponding plagiarized versions. It is crucial to ensure that dataset include both type of the text plagiarism examined in this research: Type 1 plagiarism and Type 2 plagiarism.
3. **Solution development.** Formulate the task in terms of reinforcement learning problem and design a solution architectures for text plagiarism detection. Specifically, each part of final architecture, Decision Network (DNet), structured representation model (SRM) and Regression Network (RNet), are described in details. Also, the different DRL algorithms for learning effective text representation within the DNet, such as REINFORCE, Advantage Actor-Critic (A2C) and Deep Q-Network (DQN), are explained.

The subsequent sections of the current chapter examine each research step in details.

## 3.2 Data collection

Data collection and preparation plays crucial role in the performance of machine learning models and reinforcement learning algorithms. This study considers two types of text plagiarism, which were specifically defined for analyzing texts generated by LLMs, namely Type 1 plagiarism and Type 2 plagiarism.

Therefore, the dataset should include original human-written texts together with corresponding examples of both considered plagiarism types. Preferably, the dataset should consist of quite diverse topics of original texts to exclude the situation when the plagiarism version of one text can be easily incorrectly considered as plagiarism version of another. The last requirement for the dataset is having several examples of plagiarism for one original text in order to create a task close to the real world scenarios.

Finding fully prepared dataset that would satisfy all the above criteria is not an easy task, so it was decided to generate custom dataset from existing (base) one.

### 3.2.1 Base dataset selection

As a human-written texts the subset of AG’s corpus of news articles, used by Zhang et al. in [21], was chosen. This dataset is called AG News and contains titles and descriptions of articles on the topics ‘world’, ‘sports’, ‘business’ and ‘science’. Originally it consists of 120 000 train samples and 7 600 test samples. All these samples are considered to be human-written and unique, what means no plagiarized texts in the initial dataset are presented. Hereafter, texts of these samples are referred to as source text.

### 3.2.2 Plagiarism generation

The next step was to generate corresponding examples of two plagiarism types for each sample. Two fundamental approaches to do this exists:

1. **Not using LLMs.** This approach includes methods, which do not utilize

LLMs to produce plagiarism versions of the source text. For example, for generating Type 1 plagiarism version (paraphrasing), the simple synonym replacement within the source text can be applied. Also, the procedure of translation the source text into another language and then back into the original language can generate Type 2 plagiarism versions of text.

2. **Using LLMs.** With this approach, special prompts to LLMs can be used to generate both types of plagiarism based on the source text.

The approach without LLMs requires at least two different methods to generate plagiarism versions of two types. Also, this approach almost certainly requires careful design: for example, it is not always obvious how to choose correct synonym without taking into account the context of the whole text. Moreover, translation, as an example of Type 2 plagiarism generation procedure, not only requires additional tools, such as machine translation software, but also can not guarantee notable modifications of the source text and its coherence. Although these considerations are applicable only to two specific methods (synonym replacement and translation), there is no reason to believe that the more complex methods of this approach eliminate the mentioned disadvantages. Therefore, the approach without LLMs was not the best choice, especially, as it is shown below, compared to approach with LLMs.

To generate plagiarized versions of the source texts the following SOTA LLMs were chosen:

- **OpenChat.** Model learned from mixed-quality data and fine-tuned with C-RLFT — conditional reinforcement learning fine-tuning, proposed by Wang et al. in [22]. Specifically, model version *openchat 3.5* was used.

- **Pi.** AI-based personal assistant powered by Inflection AI.
- **Mixtral.** Mixture of experts (MoE) decoder-only transformer powered by Mistral AI. Specifically, model version *mixtral-8x7b* was used.

The selection criteria for the mentioned LLMs were API availability and competitive performance comparable to Llama 2 or GPT-based models.

TABLE I

LLMs prompts for text plagiarism generation calls. *SOURCE TEXT* is replaced with text the plagiarism is generated for. *MAX SYMBOLS* is a parameter and set to 150. *IDEA* in the context of this paper refers to the result of the ‘Main idea extraction’ call

Call	Prompt
Type 1 plagiarism generation	Paraphrase and reformulate the following text as much as you can keeping the initial idea. Do not make it longer than initial text. Let it be only the paraphrased text in your answer, without any annotations: <i>SOURCE TEXT</i>
Main idea extraction	Write directly in several words the main topic of the following text, without details: <i>SOURCE TEXT</i>
Type 2 plagiarism generation	Write a short sentence (no more than <i>MAX SYMBOLS</i> symbols) on the following topic: <i>IDEA</i>

To generate different plagiarized texts of different types (Type 1 and Type 2), different prompts to LLMs were used (Table I). Note that generation of Type 1 plagiarism requires only one LLM API call (Fig. 1a), while for Type 2 plagiarism generations it is necessary to call the LLM API twice: first API call is for extraction the main topic of the text, and the second call is for generation new text based on the extracted idea (Fig. 1b). Also, it is important to mention that the idea extraction call for the concrete source text is performed only once, and

then the result is reused by other LLMs. The LLM for the idea extraction call is chosen in round-robin manner when iterating over the source texts.

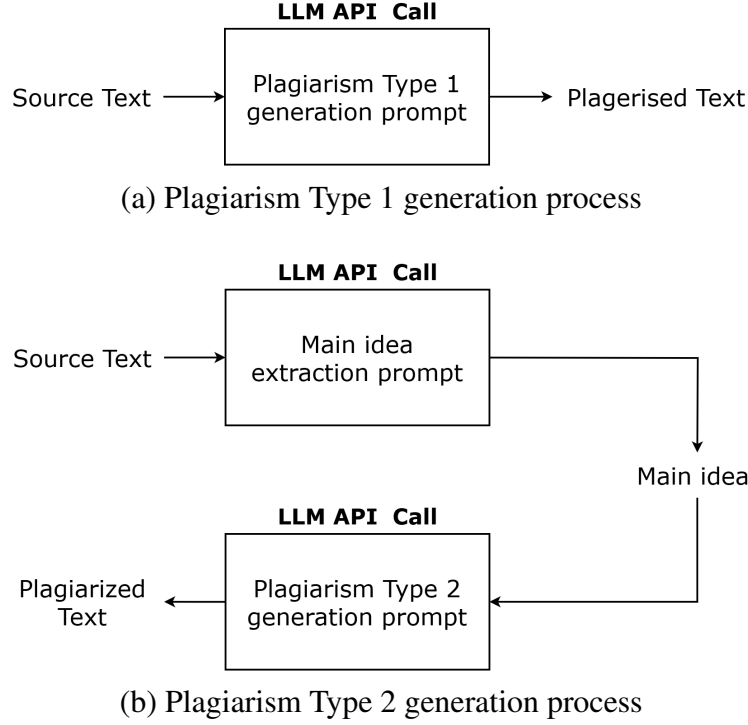


Fig. 1. Processes of text plagiarism generation. Type 1 plagiarism generation requires only one API call, while Type 2 plagiarism generation requires two.

To generate more diverse versions of text plagiarism, the conversation context (history) with LLM was maintained, so it was possible within the single session call the same generation prompt multiple times. To ensure that LLM would not generate the same results within a session, after the first generation call special history prefix ‘*One more time but in different way*’ was added to the prompt. In this paper, the number of the same generation prompt calls within a single session with LLM was set to 2 (Fig. 2).

Due to the restrictions of the LLM API calls number and lack of computational resources, the initial AG News dataset with 120 000 train and 7 600 test samples were truncated to only 120 train and 7 test samples. It will be demon-

strated in the subsequent subsection that such number of samples is enough to build satisfactory dataset.

User:	Write a short sentence (no more than 150 symbols) on the following topic: <i>Elephant escape at Moscow Zoo</i>
LLM:	Yesterday, an elephant escaped from Moscow Zoo, causing a brief panic before being safely recaptured. Zoo officials are investigating how the elephant was able to escape.
User:	<b>One more time but in different way</b> write a short sentence (no more than 150 symbols) on the following topic: <i>Elephant escape at Moscow Zoo</i>
LLM:	An elephant made a surprise exit from Moscow Zoo, leading to a temporary uproar until it was secured. Authorities are examining the cause of the elephant's escape.

Fig. 2. Example of maintaining LLM session history. It allows getting two different Type 2 plagiarism variants from the once extracted idea.

Hence, the overall plagiarism generation algorithm for each initial dataset (train and test) can be described as follows:

1. Shuffle and truncate the initial dataset to be the desired size: in case of this paper it means 120 samples for train dataset and 7 samples for test dataset. Note that shuffling is performed independently for each dataset, what means, in particular, that different samples for truncated train and test datasets before plagiarism generation can present.
2. For every source text from the dataset generate  $K$  (different) plagiarized versions of Type 1 plagiarism for every available LLM. In case of this paper,  $K = 2$  and available LLMs are *openchat*, *pi* and *mixtral*.
3. Shuffle and truncate the initial dataset to be the desired size, using different random seed.

4. For every source text from the dataset generate  $K$  (different) plagiarized versions of Type 2 plagiarism for every available LLM.

The results of the plagiarism generation algorithm are presented in the form of 4 tables: for each dataset, training and testing, as well as for each of the two types of plagiarism (Tables II and III).

TABLE II

Example of Type 1 plagiarism generation results. The initial sentence was '*A teenager from Brooklyn won a million dollars in the lottery and is now going to spend all the money on ice cream*'

LLM	Call Result
first generation call	
<b>openchat</b>	A young person hailing from Brooklyn has secured a million-dollar prize through the lottery, intending to invest every cent into indulging their sweet tooth with delectable frozen treats
<b>pi</b>	A young person from New York City has become a millionaire through a lottery win and plans to use the entire amount on frozen treats
<b>mixtral</b>	A young person hailing from Brooklyn has emerged victorious in the lottery, securing a grand prize of one million dollars, and is now determined to splurge it all on gelato
second generation call	
<b>openchat</b>	15-year-old resident of Brooklyn has recently emerged victorious in a lottery, pocketing a substantial sum of one million dollars, which they have resolved to allocate entirely towards the purchase of their favorite chilled dessert variety
<b>pi</b>	A Brooklyn teen has won a million-dollar lottery prize and intends to allocate the entire sum to purchasing ice cream.
<b>mixtral</b>	A youth from Brooklyn has struck gold in the lottery, winning a substantial sum of one million dollars, and is now decided to indulge entirely in frozen desserts



TABLE III

Example of Type 2 plagiarism generation results. The initial sentence was '*A teenager from Brooklyn won a million dollars in the lottery and is now going to spend all the money on ice cream*'. The main idea was extracted by openchat and was '*Teenager wins lottery, spends money on ice cream*'

LLM	Call Result
first generation call	
<b>openchat</b>	17-year-old lottery winner indulged in a massive ice cream spree, delighting friends and neighbors alike
<b>pi</b>	A teenager won the lottery and decided to indulge in their sweet tooth, spending the money on a lifetime supply of ice cream
<b>mixtral</b>	Thrilled teenage lottery winner splurges on a mountain of ice cream, turning their sweet tooth dream into a chilly reality
second generation call	
<b>openchat</b>	Lottery-winning teen splurges fortune on sweet treats, creating an ice cream frenzy among peers
<b>pi</b>	A lucky teenager hit the jackpot in the lottery and chose to satisfy their cravings by investing the winnings in a never-ending supply of ice cream
<b>mixtral</b>	Luck strikes a teen, as lottery winnings turn into an avalanche of ice cream, fulfilling their frozen treat fantasy

### 3.2.3 Dataset compilation

The next step after plagiarism generation was dataset compilation. This means that obtained source texts together with corresponding plagiarized versions had to be organized into desired datasets for further training and evaluation of models.

The following dataset structure was chosen for each data point:

- **Target text**, or just **target**. This is the text that would be selected as a starting point for detecting plagiarism.

- **Candidate text**, or just **candidate**. This is the text that would be checked for any plagiarism, or absence of plagiarism, of the target.
- **Plagiarism score**, or just **score**. The non-negative float number between 0 and 1, which indicates how much does the candidate text plagiarize the target text. Score 1 means the total plagiarism, and score 0 means no plagiarism between target and candidate.

Based on definition of the Type 1 and Type 2 plagiarism types and design of plagiarism generation, the scores 1.0 and 0.5 were chosen for Type 1 and Type 2 plagiarism types obtained in previous subsection respectively.

Besides the above-mentioned structure of data points, the desired dataset had to meet the following criteria:

- **Representativeness**. Both Type 1 and Type 2 plagiarisms as well as absence of plagiarism should present in the resulting dataset. Ideally, all the data obtained in the previous subsection should be utilized.
- **Balance**. Dataset should include data points with both plagiarism types and the absence of plagiarism in equal proportions. In this case, the number of examples with plagiarism of both types and examples with no plagiarism should be  $\frac{1}{3}$  of the dataset size each.
- **Order invariance**. The model is expected to predict plagiarism score equally well, regardless of the order in which the texts appear. In other words, if text A and text B are sent to the model in statuses target and candidate respectively, the predicted score should not differ much from the one that would have been obtained if text A had been assigned candidate status,

and text B had been assigned target status. Therefore, it is necessary to sometimes set the different statuses to the same text within a single dataset.

- **Flexibility.** The size of the dataset should be easy to change, which means changing the number of data points within the dataset, if possible without losing others desired dataset criteria.

After the plagiarism generation, 127 (120 for training and 7 for testing datasets) source texts in total were present, and for each source text 6 plagiarized versions of Type 1 plagiarism and 6 plagiarized versions of Type 2 plagiarism were generated. All the data can be represented as two tables, for both plagiarism types, where single row contains source text together with 6 corresponding plagiarized versions. Hereafter, these tables are referred to as Type 1 plagiarism table and Type 2 plagiarism table respectively.

Taking into account the data point structure chosen above and the outlined criteria of the desired dataset, the compilation process of the dataset was the following:

1. **Type 1 positive sampling.** For each row of Type 1 plagiarism table, generate all possible 2-length permutations, where the first element in each result pair is treated as target text, and the second element is treated as candidate text. Assign score 1.0 to each pair.
2. **Type 1 negative sampling.** Let the size of Type 1 plagiarism table be  $N_1$ , so the number of resulting pairs from previous step is  $\frac{7!}{(7-5)!}N_1 = 42N_1$ . For each source text from Type 1 plagiarism table sample  $42N_1/N_1/2 = 21$  other different source texts from the same table at random. Assign score 0.0 to each pair.

3. **Type 2 positive sampling.** For each row of Type 2 plagiarism table, generate all possible 2-length permutations, where the first element in each result pair is treated as target text, and the second element is treated as candidate text. Assign score 0.5 to each pair.
4. **Type 2 negative sampling.** Let the size of Type 2 plagiarism table be  $N_2$ , so the number of resulting pairs from previous step is  $\frac{7!}{(7-5)!}N_2 = 42N_2$ . For each source text from Type 2 plagiarism table sample  $42N_2/N_2/2 = 21$  other different source texts from the same table at random. Assign score 0.0 to each pair.
5. Combine the results of steps 1–4 and produce a single dataset with the size  $42N_1 + 21N_1 + 42N_2 + 21N_2 = 63N_1 + 63N_2$ . By design of plagiarism generation given in this research, the size of Type 1 plagiarism table always equals the size of Type 2 plagiarism table, which means  $N_1 = N_2 = N$ . So the size of resulting dataset would be  $126N$ .

Remember, that sizes of plagiarism tables for testing and training datasets were 120 and 7 respectively. Therefore, the procedure above produced the resulting training and testing datasets of corresponding sizes 15 120 and 882 (Table IV).

By compilation algorithm design, the resulting datasets always meet the Representativeness and Order invariance criteria. Produced dataset would be always balance: the number of examples with plagiarism of both types and examples with no plagiarism is exactly  $42N = \frac{1}{3} \cdot 126N$  each. The produced dataset can also be easily shrunk: for each part of rows, which corresponds to the data portion obtained during one of 1–4 compilation algorithm step, just every  $k$ -th row can be retained. In such case, the shrink ration would be  $\frac{1}{k}$  and the only criterion

that can be affected is Order invariance criterion. For the purpose of this study, several sub-datasets with different sizes were generated in addition to the entire training and testing datasets (Table V).

TABLE IV  
Train and test dataset sizes after each step of compilation procedure

Step	Train dataset	Test dataset
Plagiarism tables size	120	7
Type 1 positive sampling	5 040	294
Type 1 negative sampling	2 520	147
Type 2 positive sampling	5 040	294
Type 2 negative sampling	2 520	147
<b>Total</b>	15 120	882

TABLE V  
Sizes of different sub-datasets. Label *Initial* corresponds to the entire dataset obtained using dataset compilation procedure. For other dataset labels, every  $k$ -th row from the initial dataset was retained

Label	$k$	Train dataset size	Test dataset size
Initial	—	15 120	882
Medium, or <i>md</i>	3	5 040	294
Small, or <i>sm</i>	6	2 520	147

### 3.3 Solution architecture

The final solution architecture has conceptual similarities to the one proposed by Zhang et al. in [1]. The model has three main components: **Decision Network (DNet)**, **structured representation model (SRM)** and **Regression Network (RNet)** (Fig. 3). DNet is responsible for sampling agent actions on each state. Three different DNetS were designed, each for one of the following DRL algorithms: Advantage Actor-Critic (A2C), REINFORCE and Deep Q-Network (DQN). DNet samples actions for the target and candidate texts, so the two action sequences for both texts are produced. Action decision process can be based on explicit stochastic policy, as in case of A2C and REINFORCE algorithms, or can rely on implicitly derived policy, as in case of DQN. Then SRM converts the action sequences to the text representation of both target and candidate. RNet produces the plagiarism score based on target and candidate representations. After all, depend on the RNet loss the reward is computed and then passed to the DNet for update. Note that the reward can be calculated after full pass of both target and candidate texts through DNet and RNet.

All three solution parts are interchanged and the results of each part is then passed to the next one. The SRM text representations are derived from DNet actions. RNet utilizes the SRM representations to predict the plagiarism score. Finally, DNet uses the delayed reward obtained from RNet to explicitly or implicitly update the action sampling policy (Fig. 3).

In this section the word ‘word’ is widely used in the context of solution architecture pipeline, and it should be understood in the meaning of ‘token’.

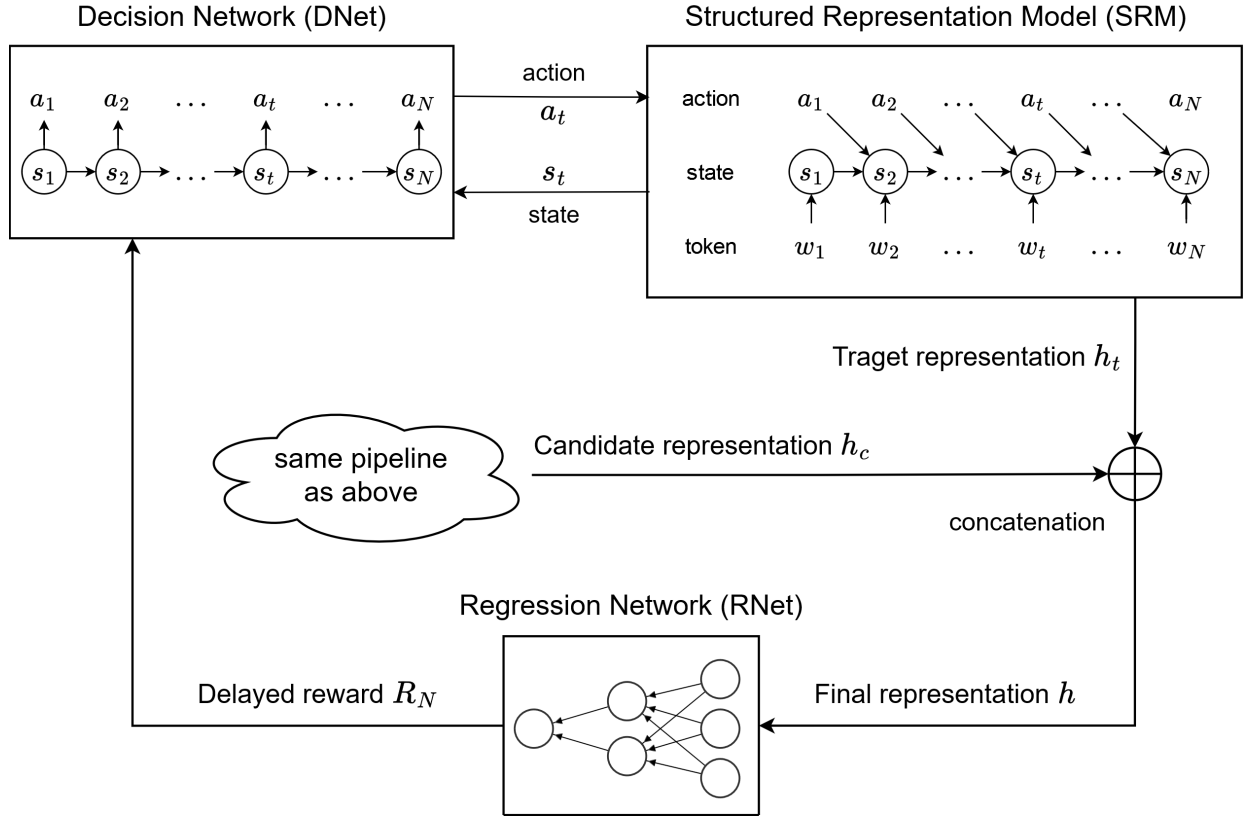


Fig. 3. Solution architecture overview. The decision network (DNet) samples action for each state. The structured representation model (SRM) provides state representation to DNet, and results in final text representation based on DNet actions. This pipeline is repeated for the target and candidate texts, the final representations are then concatenated and passed to regression network (RNet). RNet calculates the plagiarism score and offers the reward to DNet.

### 3.3.1 Decision Network (DNet)

The decision network conducts the policy for sampling actions and uses the delayed reward for policy updates. The policy is either explicit and stochastic (if A2C or REINFORCE are used) or implicit and deterministic (in case of DQN). The details about different policies are stated in consequent Sections 3.3.1.1 to 3.3.1.3. DNet samples actions based on the current state, which representation is got from SRM. The delayed reward obtained from RNet is available after

full pass of entire target and candidate texts through DNet and SRM. When text representations of both target and candidate texts are received, they are concatenated and further passed to RNet to get plagiarism score  $Y$  (Equation (3.23)).

The policies (explicit and implicit) and the objective functions of DNet are specific to the different DRL algorithms, so they are explained separately in corresponding Sections 3.3.1.1 to 3.3.1.3 below.

**State** State stores information about previous history and current input. The state  $s_t$  construction procedure is described in detail in Section 3.3.2.

**Action** This research considers two actions: *Retain* and *Delete*. The examined word  $w_t$  can be retained in the final text representation or deleted from it. Each action directly influence on state construction in SRM, and this influence is described in details in Section 3.3.2.

**Reward** To compute delayed reward  $R_N$ , the mean squared loss (MSE) between  $Y$  and  $Y^*$  is used, where  $Y^*$  is true plagiarism score for given target and candidate texts. Also, like in [1], the ratio of deleted words to the total length of target and candidate texts is included into the reward. Such term is added to encourage the model to delete more words. Therefore, the final formula of delayed reward looks like the following:

$$R_N = \log(1 - (Y - Y^*)^2) + \gamma \frac{N'}{N}, \quad (3.1)$$

where  $N$  — total length of target and candidate texts,  $N'$  — number of deleted words (where the respective action  $a_t$  is *Delete*) and  $\gamma$  is hyperparameter used to



balance the RNet prediction accuracy and deletions ratio.

Since the reward is computed at the end of the episode, after full pass of both target and candidate texts, the reward in all states except the final,  $s_N$  is equal to 0, and the reward at final state equals  $R_N$ . At the same time, the episode return can be computed as

$$R = \sum_{t=1}^N \gamma_{\text{decay}}^{N-t} r_t,$$

where  $\gamma_{\text{decay}}$  — decay factor, which is between 0 and 1, and  $r_t$  is the reward at state  $s_t$ . Therefore, in this research the total episode return  $R$  always equals  $R_N$ :

$$R = R_N,$$

which is usual estimation for action-value function  $q(s, a)$ , which shows the value of action  $a_t$  in state  $s_t$ , in the scope of this research the following is true:

$$q(s, a) \approx R_N. \quad (3.2)$$

The obtained Equation (3.2) is useful for the rest of this section.

### 3.3.1.1 REINFORCE

REINFORCE is a version of Monte-Carlo Policy Gradient algorithm proposed by Williams in [23]. It adopts an explicit stochastic policy, and uses the episode samples to update the policy parameters. To construct the sample space, the full episode should be played, which correlates with the proposed solution architecture.

**Policy** In the case of using REINFORCE, the stochastic policy of selection action  $a_t$  given state  $s_t$  obtained from DNet is defined as the following:

$$\pi_{\theta}(a_t | s_t) = \text{softmax}(\text{relu}(s_t W_1^{\text{rnf}} + b_1^{\text{rnf}}) W_2^{\text{rnf}} + b_2^{\text{rnf}}), \quad (3.3)$$

where  $\theta = \{W_1^{\text{rnf}}, b_1^{\text{rnf}}, W_2^{\text{rnf}}, b_2^{\text{rnf}}\}$  denotes the parameters of DNet. Hereafter, *relu* means ReLU activation function and *softmax* — default softmax function.

During training stage, the corresponding probabilities from Equation (3.3) are used for action sampling. During evaluation and testing stages, the optimal action  $a_t^*$  is chosen in the following way:

$$a_t^* = \text{argmax}_a \pi_{\theta}(a | s_t) \quad (3.4)$$

**Objective function** The expected reward definition is mainly based on one given in [1].

The main goal of policy optimization is maximizing the expected reward, which is defined as

$$\begin{aligned} J(\theta) &= \mathbb{E}_{(s_t, a_t) \sim P_{\theta}(s_t, a_t)} r(s_1 a_1 \dots s_N a_N) = \sum_{s_1 a_1 \dots s_N a_N} P_{\theta}(s_1 a_1 \dots s_N a_N) R_N \\ &= \sum_{s_1 a_1 \dots s_N a_N} p(s_1) \prod_t \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t) R_N. \end{aligned} \quad (3.5)$$

Here for the simplicity it is assumed, that  $N$  — total length of both target and candidate texts. By the solution architecture design, the state  $s_{t+1}$  is completely determined by state  $s_t$  and action  $a_t$ , the probabilities  $p(s_1)$  and  $p(s_{t+1} | s_t, a_t)$  are

both equal to 1. Therefore, the Equation (3.5) can be simplified to

$$J(\theta) = \sum_{s_1 a_1 \dots s_N a_N} \prod_t \pi_\theta(a_t | s_t) R_N. \quad (3.6)$$

The likelihood ratios are then used to compute the policy gradient:

$$\nabla_\theta J(\theta) = R_N \sum_{t=1}^N \nabla_\theta \log \pi_\theta(a_t | s_t). \quad (3.7)$$

In addition to just policy gradient, the entropy  $H$  of the policy is included to the objective function. Entropy is defined as

$$H(\theta) = - \sum_a \pi_\theta(a | s) \log \pi_\theta(a | s). \quad (3.8)$$

The value of  $H$  is always non-negative, and has a single maximum when all the actions have the same probability to be taken, i.e. when policy is uniform. Entropy reaches minimum, when  $\pi_\theta(a_i | s) = 1$  for some action  $a_i$  and  $\pi_\theta(a_j | s) = 0$  for all  $j \neq i$ . Therefore, subtracting the entropy from the objective function pushes the policy to be uniform, what means punishing the agent to be too sure about what action to take.

Finally, the objective function for the policy update combines Equations (3.7) and (3.8), and is defined as

$$Z(\theta) = \nabla_\theta J(\theta) - \beta \nabla_\theta H(\theta), \quad (3.9)$$

where  $\beta$  is hyperparameter to balance two terms, which is usually called *entropy beta*.

### 3.3.1.2 Advantage Actor-Critic (A2C)

The Actor-Critic method combines the policy-based and value-based approaches. In this method, two function approximations are learned: policy function  $\pi_\theta(a | s)$ , which controls what action the agent selects at state  $s$ , and value function  $v_\theta(s)$ , which estimates how profitable the state  $s$  is.

The modification of Actor-Critic method, which uses advantage, is called Advantage Actor-Critic (A2C) and is used for more stable convergence in comparison with only policy-based methods, like REINFORCE. The advantage is calculated in the following way:

$$A(s_t, a_t) = q(s_t, a_t) - v(s_t),$$

where  $q(s_t, a_t)$  — action value function.

Advantage  $A(s_t, a_t)$  indicates the extra reward the agent can get if it selects the action  $a_t$  at state  $s_t$  compared to the mean reward,  $v(s_t)$ , it gets at state  $s_t$ . Using the Equation (3.2), the formula for advantage becomes the following:

$$A(s_t, a_t) = R_N - v(s_t). \quad (3.10)$$

In the case of using A2C, the parameters of DNet are the following:

$$\theta = \{W^{a2c}, b^{a2c}, W_{p_1}^{a2c}, b_{p_1}^{a2c}, W_{p_2}^{a2c}, b_{p_2}^{a2c}, W_v^{a2c}, b_v^{a2c}\}, \quad (3.11)$$

where  $\{W^{a2c}, b^{a2c}\}$  are used in the body of A2C,  $\{W_{p_1}^{a2c}, b_{p_1}^{a2c}, W_{p_2}^{a2c}, b_{p_2}^{a2c}\}$  are used for policy approximation, and  $\{W_v^{a2c}, b_v^{a2c}\}$  are used for value approximation (Fig. 4).

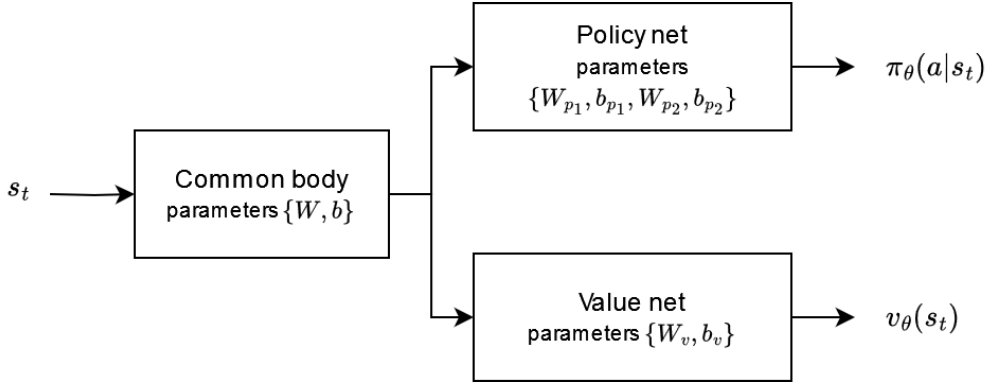


Fig. 4. Advantage Actor-Critic (A2C) architecture with shared body. For each part the used parameters from set  $\theta = \{W, b, W_{p1}, b_{p1}, W_{p2}, b_{p2}, W_v, b_v\}$  are listed. Upper indices of parameters are omitted for clarity.

**Policy** The policy calculation is similar to REINFORCE case (Equation (3.3)). The stochastic policy of selection action  $a_t$  given state  $s_t$  obtained from DNet is defined as the following:

$$\pi_{\theta}(a_t | s_t) = \text{softmax}(\text{relu}(\text{relu}(s_t W^{a2c} + b^{a2c}) W_{p1}^{a2c} + b_{p1}^{a2c}) W_{p2}^{a2c} + b_{p2}^{a2c}), \quad (3.12)$$

where  $\theta$  denotes the parameters of DNet.

During training stage, the corresponding probabilities from Equation (3.12) are used for action sampling. During evaluation and testing stages, the optimal action  $a_t^*$  is chosen in the same way as in Equation (3.4).

**Objective function** At the beginning, it is necessary to define  $v_{\theta}(s_t)$ . The value function of the given state  $s_t$  obtained from DNet is defined as following:

$$v_{\theta}(s_t) = \text{relu}(s_t W^{a2c} + b^{a2c}) W_v^{a2c} + b_v^{a2c}, \quad (3.13)$$

where  $\theta$  denotes the parameters of DNet.

The objective function itself in case of A2C consists of three parts: policy

gradient part, entropy part and value loss part.

First part is policy gradient, which is quite similar to the one give in Equation (3.7):

$$\nabla_{\theta} J(\theta) = \sum_{t=1}^N A(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t).$$

The only difference is that, instead of  $q_{\theta}(s_t, a_t)$  used Equation (3.7), here  $A(s_t, a_t)$  is used. Using the Equation (3.10), the policy gradient formula becomes as following:

$$\nabla_{\theta} J(\theta) = \sum_{t=1}^N (R_N - v_{\theta}(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t). \quad (3.14)$$

Second part of objective function is entropy part,  $H(\theta)$ , which is identical to one proposed for REINFORCE (Equation (3.8)).

Third and final part of objective function is value loss part, which estimates how accurate DNet approximates value. The value loss is defined as

$$L_v(\theta) = \sum_{t=1}^N (q_{\theta}(s_t, a_t) - v_{\theta}(s_t))^2 = \sum_{t=1}^N (R_N - v_{\theta}(s_t))^2. \quad (3.15)$$

The objective function is defined as the following combination of Equations (3.8), (3.14) and (3.15):

$$Z(\theta) = \nabla_{\theta} J(\theta) + \nabla_{\theta} L_v(\theta) - \beta \nabla_{\theta} H(\theta), \quad (3.16)$$

where  $\beta$  has the same meaning as in Equation (3.9).

### 3.3.1.3 Deep Q-Network (DQN)

The Deep Q-Network (DQN) approximates a state-value function  $Q(s, a)$  for each possible action and state, using the experience replay buffer. Therefore, the DQN policy, which is build based on the state-value function, is implicit and deterministic.

Q-Network is usually optimized towards target network with frozen weights, which is denoted as  $\hat{Q}(s, a)$ . In its turn, the target network is periodically updated with the latest Q-Network weights every  $k_{\text{sync}}$  episodes, where  $k_{\text{sync}}$  is a hyperparameter. The target network is introduced to stabilize training.

In this DNet setting, the state-value function of the given state  $s_t$  is defined as following:

$$Q_{\theta}(s_t, a_t) = \text{relu}(s_t W_1^{\text{dqn}} + b_1^{\text{dqn}}) W_2^{\text{dqn}} + b_2^{\text{dqn}}, \quad (3.17)$$

where  $\theta = \{W_1^{\text{dqn}}, b_1^{\text{dqn}}, W_2^{\text{dqn}}, b_2^{\text{dqn}}\}$  denotes the parameters of DNet. Frozen parameters of  $\hat{Q}(s, a)$  are denoted  $\hat{\theta}$ .

**Policy** To perform agent exploration, the  $\varepsilon$ -greedy strategy is used.  $\varepsilon(i)$  is a function  $\mathbb{R}^d \rightarrow (0, 1)$ , which maps the index of current epoch to float value between 0 and 1. On each step, the agent with the probability of  $\varepsilon$  selects random action. Otherwise, the agent selects action which maximizes  $Q(s, a)$ . This can be formalized as follows:

$$\pi_{\theta}(s_t) = \begin{cases} a \sim \{a_m\} & \text{with probability } \varepsilon, \\ \operatorname{argmax}_a Q_{\theta}(s_t, a) & \text{with probability } 1 - \varepsilon. \end{cases} \quad (3.18)$$

where  $\theta$  denotes the parameters of DNet and  $\{a_m\}$  — actions space, which is in case of this research is  $\{Retain, Delete\}$ .

During training stage, the actions are sampled according to the Equation (3.18). During evaluation and testing stages, the optimal action  $a_t^*$  is chosen as  $a_t^* = \operatorname{argmax}_a Q_\theta(s_t, a)$ .

**Objective function** In case of DQN, the objective function for the episode consists only from one part — state-value loss  $L_Q(\theta) = \sum_{t=1}^N L_{Q_t}(\theta)$ , and  $L_{Q_t}(\theta)$  is defined as following:

$$L_{Q_t}(\theta) = \begin{cases} (Q_\theta(s_t, a) - R_N)^2, & \text{if the episode has ended} \\ (Q_\theta(s_t, a) - \max_{a'} \hat{Q}_{\hat{\theta}}(s_{t+1}, a'))^2, & \text{otherwise.} \end{cases}$$

Note that for non-terminal states the target network  $\hat{Q}(s, a)$  is used to predict value of the next state  $s_{t+1}$ . Also recall that every  $k_{\text{sync}}$  episodes  $\hat{\theta}$  is set to  $\theta$ .

Therefore, the objective function looks like the following:

$$Z(\theta) = \nabla_\theta L_Q(\theta). \quad (3.19)$$

### 3.3.2 Structured representation model (SRM)

The purpose of structured representation model (SRM) is to build effective text representation by retaining the relevant words from the initial text and removing the irrelevant, redundant ones. By doing this, the SRM is expected to learn more concise and informative text representation for text plagiarism detection. For example, prepositions, articles and pronouns may contain little information useful for plagiarism detection, and rather add some unwanted noise to text rep-



resentation. Recall that ‘word’ means ‘token’ in the context of text representation within this study. So, most punctuation can be attributed to irrelevant words. In contrast, the numbers, for example, may be highly informative for further plagiarism detection. The idea of SRM presented below is largely based on the definition of ID-LSTM given by Zhang et al. in [1].

SRM translates the actions  $\{Retain, Delete\}$  obtained from DNet to the states and text representation. Namely, for word sequence  $w_1 w_2 \dots w_{N'}$  the action sequence  $a_1 a_2 \dots a_{N'}$  is constructed, where each  $a_i \in \{Retain, Delete\}$ . The action *Retain* means that corresponding word is relevant and should be used for text representation building, while action *Delete* means that word should be ‘deleted’ from the initial text, or skipped, and should have no impact for final text representation. This can be formalized as the following:

$$c_t, h_t = \begin{cases} c_{t-1}, h_{t-1}, & a_t = Delete \\ \Phi(c_{t-1}, h_{t-1}, w_t), & a_t = Retain \end{cases} \quad (3.20)$$

where  $\Phi$  denotes the sequence LSTM block (proposed by Hochreiter and Schmidhuber in [24]) with hidden state  $h_t$  and context state  $c_t$  at step  $t$ . If the  $a_t = Delete$ , all the LSTM cells data is copied from the previous step.

**State** The state provided by SRM to DNet is defined as following:

$$s_t = h_{t-1} \oplus c_{t-1} \oplus w_t, \quad (3.21)$$

where  $c_{t-1}$  and  $h_{t-1}$  are obtained from Equation (3.20), and  $\oplus$  is vector concatenation.

**Text representation** As a final representation for individual text  $w_1 w_2 \dots w_{N'}$ , the last hidden state of SRM  $h_{N'}$  is used.

The final representation of both target and candidate texts, which is then passed to RNet for plagiarism score prediction, is defined as following:

$$h_N = h_{N_t} \oplus h_{N_c}, \quad (3.22)$$

where  $h_{N_t}$  is the last hidden state of SRM after processing target text,  $h_{N_c}$  is the last hidden state of SRM after processing candidate text, and  $\oplus$  is vector concatenation.

### 3.3.3 Regression Network (RNet)

The Regression network (RNet) predicts the plagiarism score based on the both target and candidate texts representation obtained from SRM. The plagiarism score  $Y$  calculation within the RNet looks like the following:

$$Y = \sigma(\text{relu}(h_N W_1^r + b_1^r) W_2^r + b_2^r), \quad (3.23)$$

where  $\{W_1^r, b_1^r, W_2^r, b_2^r\}$  — parameters of RNet,  $\sigma$  — Sigmoid function,  $\text{relu}$  — ReLU function, and  $h_N$  is defined from Equation (3.22).

In order to train RNet, the mean squared error (MSE) is used as loss function. For the individual episode the loss function is defined as the following

$$\mathcal{L} = (Y - Y^*)^2, \quad (3.24)$$

where  $Y^*$  — true plagiarism score, and  $Y$  is defined from Equation (3.23).

Note that SRM also has set of trainable parameters (parameters of  $\Phi$ , Equation (3.20)), which can be directly trained using loss function defined in Equation (3.24), because the predicted plagiarism score  $Y$  depends on  $h_N$  (Equation (3.23)) obtained from SRM.

## Chapter 4

# Evaluation and Discussion

This chapter describes the training and evaluation workflow together with presented limitations and future work suggestions. Specifically, Section 4.1 outlines the process of solution training. Section 4.2 provides training details, such as optimization functions and hyperparameter choices. Section 4.3 lists the baseline models used for comparative analysis. The comparison of the solution architectures with the baselines is made in Section 4.4. Section 4.5 provides analysis of text representations obtained by the solution architectures. Finally, the limitations of this study and the further possible development of the considered topic are described in Section 4.6.

Hereafter, for simplicity, solution architectures with DRL algorithms REINFORCE, A2C and DQN are referred to as SRM-R (Structured Representation Model with REINFORCE), SRM-A (Structured Representation Model with A2C), and SRM-D (Structured Representation Model with DQN) respectively.

## 4.1 Training details

Since DNet, SRM and RNet are connected with each other and are rotated during workflow pipeline, they should be trained and updated jointly. However, as joint training of all components from the entire scratch can be difficult, the pre-training stages are included, like proposed by Zhang et al. in [1].

Then the overall training process consists of the following steps:

1. Pre-train SRM and RNet on warm-state representations by minimizing Equation (3.24). Warm-state representations are just the initial target and candidate texts without any deletions. Warm-state representations are used to reduce the high variance, which could occur in case of training DRL parts from scratch.
2. Fix the parameters of SRM and RNet and pre-train DNet using the corresponding objective function: Equation (3.9) in case of SRM-R, Equation (3.16) in case of SRM-A and Equation (3.19) if SRM-D is used;
3. Unfreeze the parameters of SRM and RNet and train all the tree components (DNet, SRM and RNet) jointly until convergence.

## 4.2 Experiment settings

The medium size *md* train dataset is used for training, and the small size *sm* test dataset is used for testing and evaluation (Table V).

The PyTorch ‘*basic\_english*’ tokenizer is used for text tokenization. The word vectors are initialized using the GloVe vectors proposed by Pennington et

al. in [25]. The dimension of GloVe vectors as well as dimension of hidden states of SRM are set to 300.

During the training stage, the Adam optimizer proposed by Kingma and Ba in [26] is used. The learning rate is similar for the all components and DRL algorithms and is set to 0.005. Mini-batch size is 10, so processing 10 episodes is one epoch. Additionally, the Dropout layer with probability 0.5 is placed before the final layer in RNet.

$\gamma$  in the reward computation (Equation (3.1)) is set to 0.1. The entropy beta  $\beta$  in the objective functions of REINFORCE and A2C from Equations (3.9) and (3.16) is set to 0.001. For DQN algorithm,  $k_{\text{sync}}$  is set to 20.

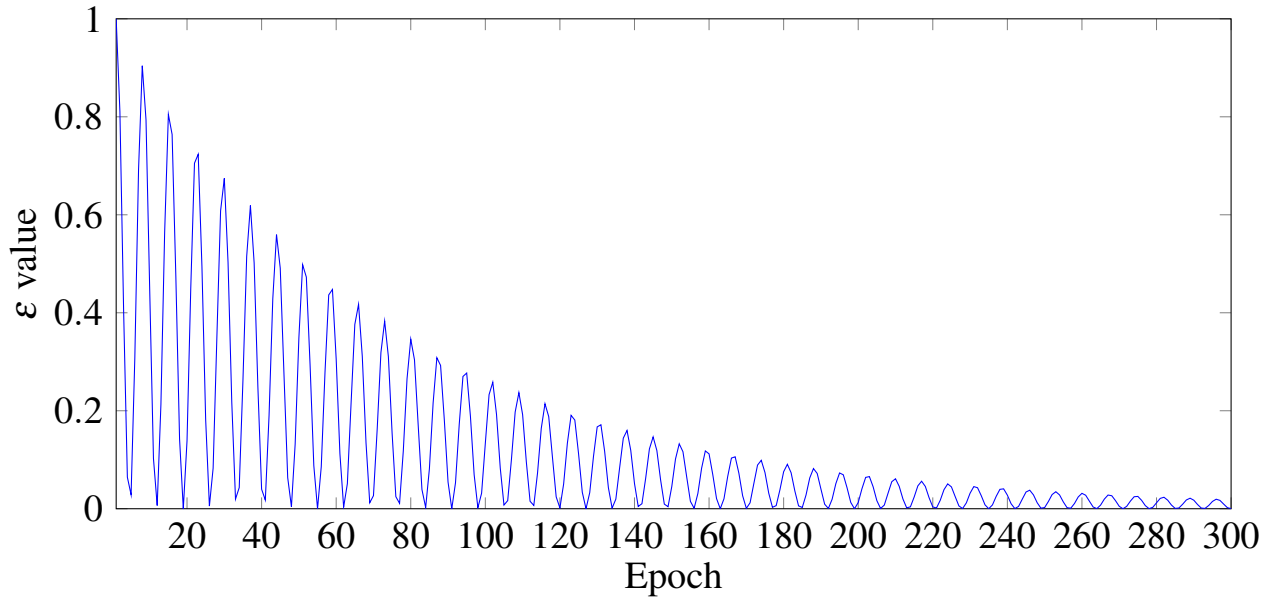


Fig. 5. Epsilon strategy for Deep Q-Network algorithm. The general function is defined in Equation (4.1). In the illustrated case,  $M = 300$  — total number of epochs,  $\epsilon_0 = 1.0$  — starting  $\epsilon$  value, and  $C = 50$ .

The general formula of used epsilon strategy for the DQN algorithm is de-

defined as following:

$$\varepsilon(i) = \frac{\varepsilon_0}{2}(\cos(i \cdot C - C) + 1) \exp(-4(i - 1)/M), \quad (4.1)$$

where  $i$  — the epoch index,  $M$  — total number of training epochs,  $\varepsilon_0$  — starting value for  $\varepsilon$ , and  $C$  — positive constant, which defines how frequently the function oscillates. Equation (4.1) produces oscillated decreasing function (Fig. 5), what alternates the exploration and exploitation of DQN agent. These alternations help the algorithm not to stuck in the local minima or plateau. In this study,  $C = 50$  and  $\varepsilon_0 = 1.0$  are set.

TABLE VI

Experiment parameters shapes. Only shapes for matrix parameters  $W$  are given: the shapes of corresponding bias parameters  $b$  are uniquely determined by matrix parameters from Equations (3.3), (3.12), (3.13), (3.17) and (3.23)

Parameter	Shape	Parameter	Shape
REINFORCE DNet		DQN DNet	
$W_1^{\text{rnf}}$	(900, 16)	$W_1^{\text{dqn}}$	(900, 16)
$W_2^{\text{rnf}}$	(16, 2)	$W_2^{\text{dqn}}$	(16, 2)
A2C DNet		RNet	
$W^{\text{a2c}}$	(900, 16)	$W_1^{\text{r}}$	(600, 128)
$W_{p_1}^{\text{a2c}}$	(16, 16)	$W_2^{\text{r}}$	(128, 1)
$W_{p_2}^{\text{a2c}}$	(16, 2)		
$W_v^{\text{a2c}}$	(16, 1)		

The parameters of RNet and DNet for each DRL are chosen with respect to the required input and output shapes (Table VI). Specifically, input to the RNet has size 600, which is determined by the chosen dimension of hidden states of SRM and Equation (3.22). The input size of DNet, regardless of used DRL algo-

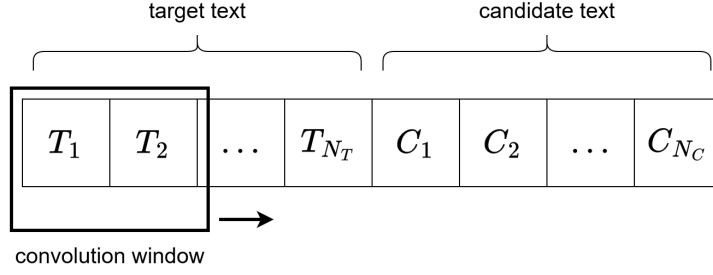
rithm, is also determined by the chosen dimension of hidden states of SRM and Equation (3.21). The outputs are either scalars, as in RNet and in the value part of A2C DNet, or have size of action space. The action space in case of this research is  $\{Retain, Delete\}$ , so its size is 2.

### 4.3 Baselines

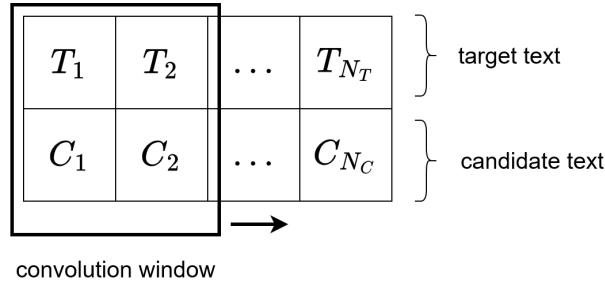
Several widely used and relatively simple architectures were chosen as baselines. Relative simplicity means that architecture has not numerous trainable parameters, which is comparable with the number of trainable parameters used in the proposed solutions. Therefore, the following architectures were selected as baselines:

- **LSTM.** A one-directional sequence LSTM.
- **biLSTM.** A bidirectional sequence LSTM, which is commonly used for text processing text.
- **A-LSTM.** A one-directional sequence LSTM with the preceding self-attention mechanism [27].
- **A-biLSTM.** A bidirectional sequence LSTM with the preceding self-attention mechanism.
- **CNN-1D.** A one-dimensional convolution neural network with one convolution layer. Convolution is applied to the concatenation of target and candidate texts (Fig. 6a).





(a) CNN-1D workflow overview



(b) CNN-2D workflow overview

Fig. 6. CNN-based baselines outline.

- **CNN-2D.** A two-dimensional convolution neural network with one convolution layer. Convolution is applied to the target and candidate texts simultaneously (Fig. 6b).

The baseline models are combined with the proposed RNet for the evaluation. So the set of RNet parameters are the same for the all baselines and solutions, except the input shape, which is architecture-dependent. For all LSTM-based baselines, the same dimension of hidden state as for SRM are used. A-LSTM and A-biLSTM use the multi-head attention with 15 heads. CNN-1D and CNN-2D have the 3 output channels for convolution layers and use the max pooling layers with  $kernel = 2$  and  $stride = 2$  after the convolution layers. The kernel size for convolution layer is set to 4 for CNN-1D, and to  $(2, 4)$  for CNN-2D. Other evaluation settings, such as optimizer and used embeddings, are chosen the same as described in Section 4.2.

Although numbers of trainable parameters of CNN-1D and CNN-2D are approximately 18 times less than of the other architectures (Table VII), they increase linearly of the maximum process text length. In contrast, numbers of trainable parameters of other, LSTM-based, architectures are fixed and do not depend on the length of processing text.

TABLE VII

Number of trainable parameters of baseline and solution architectures. Maximum input length for CNN-1D and CNN-2D are set to 217, which corresponds to the maximum total length of target and candidate texts pairs across train and test datasets, used in the experiments

Architecture	Parameters
baselines	
LSTM	761 057
biLSTM	1 521 857
A-LSTM	1 122 257
A-biLSTM	1 883 057
CNN-1D	44 948
CNN-2D	48 548
solutions	
SRM-R	813 907
SRM-A	814 196
SRM-D	813 907

## 4.4 Regression results

Regression results (Table VIII) show that the proposed solutions, namely SRM-R, SRM-A and SRM-D, demonstrate competitive performance on synthetic

dataset. Proposed architectures completely outperform the LSTM-based (LSTM, biLSTM, A-LSTM and A-biLSTM) and CNN-based (CNN-1D and CNN-2D) baselines. Moreover, even the worst performing solution, SRM-D, has the smaller MSE loss than the best baseline model — CNN-2D. The obtained results prove that building text representation using only relevant task-specific words can increase effectiveness of the model.

TABLE VIII  
Regression MSE loss on synthetic dataset

Architecture	Loss
LSTM	0.1604
biLSTM	0.1601
A-LSTM	0.1658
A-biLSTM	0.1678
CNN-1D	0.1443
CNN-2D	0.1331
SRM-R	<b>0.1082</b>
SRM-A	0.1187
SRM-D	0.1266

## 4.5 Representations analysis

In these sections analyze text representations obtained by SRM-R, SRM-A and SRM-D and examines how these representations influence regression performance. For comprehensive understanding, the analysis is conducted in both qualitative and qualitative points of view.

Initial text	Under a recent legal change, hearse producers are no longer obligated to fit specific car seat requirements.
SRM-R	<del>Under a recent legal change, hearse producers are no longer obligated to fit specific requirements</del>
SRM-A	<del>Under a recent legal change, hearse producers are no longer obligated to fit specific car seat requirements</del>
SRM-D	<del>Under a recent legal change, hearse producers are no longer obligated to fit specific car seat requirements</del>
Initial text	Wanted: skilled programmer for crafting a secure online platform by MyDoom virus innovators
SRM-R	<del>Wanted: skilled programmer for crafting a secure online platform by MyDoom virus innovators</del>
SRM-A	<del>Wanted: skilled programmer for crafting a secure online platform by MyDoom virus innovators</del>
SRM-D	<del>Wanted: skilled programmer for crafting a secure online platform by MyDoom virus innovators</del>
Initial text	During its initial week, Madden NFL exhibited remarkable sales figures, enthralling gaming enthusiasts globally
SRM-R	<del>During its initial week, Madden NFL exhibited remarkable sales figures, enthralling gaming enthusiasts globally</del>
SRM-A	<del>During its initial week, Madden NFL exhibited remarkable sales figures, enthralling gaming enthusiasts globally</del>
SRM-D	<del>During its initial week, Madden NFL exhibited remarkable sales figures, enthralling gaming enthusiasts globally</del>

Fig. 7. Examples of text representations discovered by SRM-R, SRM-A and SRM-D. The cancelled out parts correspond to the parts deleted by models. Punctuation is preserved as in initial text for clarity.

### 4.5.1 Qualitative analysis

All the SRM-R, SRM-A and SRM-D models can efficiently remove irrelevant tokens and still outperform the chosen baselines. Although each solution architecture consider different tokens as uninformative, the general trend is noticeable. Specifically, prepositions and articles, such ‘a’, ‘for’, ‘under’ and ‘by’, are usually removed by all the proposed solutions. Moreover, the entire consecutive subsequence of the text can be deleted, such as ‘are no longer obligated to’ (Fig. 7). Also remember that dataset texts were not somehow cleaned: namely, punctuation were not removed in advance. However, proposed models managed to get rid of most punctuation marks to different extents.

Interestingly, the deletion of irrelevant and noisy words does not worsen the regression results. Conversely, the solutions have notable increase in the performance in comparison with pure LSTM (Table VIII). And since proposed solutions, without taking into account the DNet parts, are essentially LSTMs (Equations (3.20), (3.22) and (3.23)), it can be established that not all tokens are necessary for the successful text plagiarism detection in the field of LLMs.

### 4.5.2 Quantitative analysis

A quantitative analysis is performed to find out what is retained from the initial text and what kinds of tokens are considered relevant or irrelevant for each solution model.

Primarily, the average length of the initial text and the average length of obtained texts for each solution model in the test dataset are compared (Table IX). All the proposed models in average can shrink the initial text by at least 30%.

Interestingly, the number of removed words has no explicit correlation with the performance of the model. So, SRM-A removes about 68% of words in average and has the second-place performance among solutions, while the SRM-D shows the worst performance among the solutions, though it has the smallest average percentage of deleted words (Table VIII). Apparently, the number of deleted tokens is not as important as which tokens are deleted and which ones are retained.

TABLE IX

The initial average length and the obtained average length by SRM-R, SRM-A and SRM-D in test dataset

Architecture	Initial Length	Resulting Length	Removed	Percentage
SRM-R	39.79	26.03	13.76	34.58%
SRM-A		12.59	27.20	68.36%
SRM-D		27.67	12.12	30.46%

Therefore, an analysis of the types of deleted and retained words was carried out (Table X). The most deleted words for all solution models are non-content words, such as prepositions and pronouns. Such words are usually uninformative for plagiarism detection task. In contrast, the least deleted words for all solution models are mainly verbs and modal verbs, which can indeed be considered as relevant for plagiarism detection, as they build a backbone of the text. However, there are some words, the ratio of deletions of which differs significantly for different solution models. For example, SRM-R and SRM-A models always delete word ‘no’, while SRM-D always retains it. Other example is word ‘but’, which is more frequently deleted by SRM-D than by SRM-R. Although both ‘no’ and ‘but’ may radically change the meaning of the text, solution models significantly differ in the assessment of informativeness of these words. The difference in the perception of such kind of words can play a crucial role in the performance

of the model.

TABLE X

Examples of the most and least deleted words in the test dataset. The upper part of table presents words that are frequently deleted by all solution models. The middle part shows words, the ratio of deletions of which differs significantly for different solution models. Finally, the bottom part of the table shows words that are rarely deleted by all solution models

Token	Count	SRM-R		SRM-A		SRM-D	
		Deleted	Percentage	Deleted	Percentage	Deleted	Percentage
now	9	9	100.00%	9	100.00%	9	100.00%
who	13	11	84.62%	13	100.00%	10	76.92%
under	8	5	62.50%	8	100.00%	7	87.50%
but	20	10	50.00%	20	100.00%	18	90.00%
the	581	315	54.22%	563	96.90%	156	26.85%
no	5	5	100.00%	5	100.00%	0	0.00%
understand	14	0	0.00%	0	0.00%	0	0.00%
must	7	0	0.00%	0	0.00%	0	0.00%
new	13	0	0.00%	0	0.00%	0	0.00%

As a result, the qualitative and quantitative analysis show that SRM-R, SRM-A and SRM-D can automatically learn efficient text representation for plagiarism detection in the field of LLMs by removing task-irrelevant words. Such process of building text representations can increase the model performance, and also correlates well with human ideas about informative and uninformative words.

## 4.6 Limitations and Future Work

The process of dataset collection, described in Section 3.2, can be a topic for discussion. The quality of obtained texts highly depends on the used LLMs and prompts for plagiarism generation. Therefore, usage of more innovative versions of LLMs, which are nowadays appearing very quickly, and knowledge of the prompt-engineering can be considered as ways of improving dataset building process for future researches. Moreover, the base dataset can be changed to detect the text plagiarism in the specific fields.

This research considers only the simplest types of text plagiarism, which are referred to as Type 1 plagiarism and Type 2 plagiarism. So, the more complex plagiarism cases were not included into this paper, though detecting sophisticated text plagiarism remains an important task. Therefore, the definition and investigation of complex text plagiarism cases using the method, proposed in this research, stays for the future research.

DRL algorithms implemented in this study, namely REINFORCE, Advantage Actor-Critic and Deep Q-Network, are classical. Integration of other innovative DRL approaches, such as Deep Deterministic Policy Gradients ([28]), Proximal Policy Optimization ([29]) and Trust Region Policy Optimization ([30]), into DNet can be the subject for further research.

Section 3.3.2 defines the Structured representation model (SRM), which is based on LSTM model in this study. However, regression results (Table VIII) show that CNN-based models have a good potential for being used inside SRM. Moreover, not only CNN, but any other context-saving model, such as RAE ([31]), can be integrated to SRM. In such case, the notion of state would be modified too. Therefore, testing different SRM architectures can also be considered



for further research.

Finally, the lack of the time and computational resources can be considered as the main limitation of this study. This limitation affects all stages: dataset building, solution architecture, experiments settings and baseline choice. Therefore, testing the proposed architecture on bigger datasets with more deep models and for longer time is natural next step to further study.

## Chapter 5

# Conclusion

This study presented the method based on deep reinforcement learning for text plagiarism detection in the field of large language models. The synthetic dataset was generated with the use of state-of-the-art large language models to train and evaluate the performance of the solution models. The architectures based on REINFORCE (SRM-R), Advantage Actor-Critic (SRM-A) and Deep Q-Network (SRM-D) deep reinforcement learning algorithms were implemented to learn task-relevant text representations by removing uninformative words from the initial text. Although solution architectures discovered different text representations, conducted experiments show that all the proposed models outperform chosen baselines and are able to build efficient text representations for plagiarism detection task. Therefore, the proposed method has a great potential in further research on the text plagiarism detection in general and in the field of large language models in particular.

# Bibliography cited

- [1] T. Zhang, M. Huang, and L. Zhao, “Learning structured representation for text classification via reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, ISSN: 2159-5399. DOI: 10.1609/aaai.v32i1.12047. [Online]. Available: <http://dx.doi.org/10.1609/aaai.v32i1.12047>.
- [2] Y. Liu and M. Lapata, “Learning structured text representations,” *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 63–75, Dec. 2018, ISSN: 2307-387X. DOI: 10.1162/tacl\_a\_00005. [Online]. Available: [http://dx.doi.org/10.1162/tacl\\_a\\_00005](http://dx.doi.org/10.1162/tacl_a_00005).
- [3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: <https://arxiv.org/abs/1810.04805>.
- [4] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, ISSN: 1476-4687. DOI: 10.1038/nature16961. [Online]. Available: <http://dx.doi.org/10.1038/nature16961>.

- [5] X. Zhu, P. Sobhani, and H. Guo, *Long short-term memory over tree structures*, 2015. DOI: 10.48550/ARXIV.1503.04881. [Online]. Available: <https://arxiv.org/abs/1503.04881>.
- [6] K. S. Tai, R. Socher, and C. D. Manning, *Improved semantic representations from tree-structured long short-term memory networks*, 2015. DOI: 10.48550/ARXIV.1503.00075. [Online]. Available: <https://arxiv.org/abs/1503.00075>.
- [7] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2016. DOI: 10.18653/v1/n16-1174. [Online]. Available: <http://dx.doi.org/10.18653/v1/N16-1174>.
- [8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” 2017. DOI: 10.48550/ARXIV.1708.05866. [Online]. Available: <https://arxiv.org/abs/1708.05866>.
- [9] H. Teng, Y. Li, F. Long, M. Xu, and Q. Ling, “Reinforcement learning for extreme multi-label text classification,” in *Cognitive Systems and Signal Processing*. Springer Singapore, 2021, pp. 243–250, ISBN: 9789811623363. DOI: 10.1007/978-981-16-2336-3\_22. [Online]. Available: [http://dx.doi.org/10.1007/978-981-16-2336-3\\_22](http://dx.doi.org/10.1007/978-981-16-2336-3_22).
- [10] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, *Enriching word vectors with subword information*, 2016. DOI: 10.48550/ARXIV.1607.

04606. [Online]. Available: <https://arxiv.org/abs/1607.04606>.
- [11] Y. Prabhu and M. Varma, “Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD ’14, ACM, Aug. 2014. DOI: 10.1145/2623330.2623651. [Online]. Available: <http://dx.doi.org/10.1145/2623330.2623651>.
- [12] Y. Kim, *Convolutional neural networks for sentence classification*, 2014. DOI: 10.48550/ARXIV.1408.5882. [Online]. Available: <https://arxiv.org/abs/1408.5882>.
- [13] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, “Sparse local embeddings for extreme multi-label classification,” in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: [https://proceedings.neurips.cc/paper\\_files/paper/2015/file/35051070e572e47d2c26c241ab88307f-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2015/file/35051070e572e47d2c26c241ab88307f-Paper.pdf).
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, ISSN: 1476-4687. DOI: 10.1038/nature14236. [Online]. Available: <http://dx.doi.org/10.1038/nature14236>.
- [15] E. Lin, Q. Chen, and X. Qi, “Deep reinforcement learning for imbalanced classification,” *Applied Intelligence*, vol. 50, no. 8, pp. 2488–2502, Mar. 2020, ISSN: 1573-7497. DOI: 10.1007/s10489-020-01637-z.

- [Online]. Available: <http://dx.doi.org/10.1007/s10489-020-01637-z>.
- [16] C. Drummond and R. Holte, "C4.5, class imbalance, and cost sensitivity: Why under-sampling beats oversampling," *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets*, Jan. 2003.
- [17] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2016. DOI: 10.1109/ijcnn.2016.7727770. [Online]. Available: <http://dx.doi.org/10.1109/IJCNN.2016.7727770>.
- [18] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, Jan. 2006, ISSN: 1041-4347. DOI: 10.1109/tkde.2006.17. [Online]. Available: <http://dx.doi.org/10.1109/TKDE.2006.17>.
- [19] J. J. Chen, C.-A. Tsai, H. Moon, H. Ahn, J. J. Young, and C.-H. Chen, "Decision threshold adjustment in class prediction," *SAR and QSAR in Environmental Research*, vol. 17, no. 3, pp. 337–352, Jun. 2006, ISSN: 1029-046X. DOI: 10.1080/10659360600787700. [Online]. Available: <http://dx.doi.org/10.1080/10659360600787700>.
- [20] G. Li, M. Dong, L. Ming, *et al.*, "Deep reinforcement learning based ensemble model for rumor tracking," *Information Systems*, vol. 103, p. 101 772, Jan. 2022, ISSN: 0306-4379. DOI: 10.1016/j.is.2021.101772. [Online]. Available: <http://dx.doi.org/10.1016/j.is.2021.101772>.

- [21] X. Zhang, J. Zhao, and Y. LeCun, *Character-level convolutional networks for text classification*, 2015. DOI: 10.48550/ARXIV.1509.01626. [Online]. Available: <https://arxiv.org/abs/1509.01626>.
- [22] G. Wang, S. Cheng, X. Zhan, X. Li, S. Song, and Y. Liu, *Openchat: Advancing open-source language models with mixed-quality data*, 2024. arXiv: 2309.11235 [cs.CL].
- [23] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, May 1992, ISSN: 1573-0565. DOI: 10.1007/bf00992696. [Online]. Available: <http://dx.doi.org/10.1007/BF00992696>.
- [24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- [25] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1162. [Online]. Available: <http://dx.doi.org/10.3115/v1/D14-1162>.
- [26] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: <https://arxiv.org/abs/1412.6980>.

- [27] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: <https://arxiv.org/abs/1706.03762>.
- [28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, *Continuous control with deep reinforcement learning*, 2015. DOI: 10.48550/ARXIV.1509.02971. [Online]. Available: <https://arxiv.org/abs/1509.02971>.
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. DOI: 10.48550/ARXIV.1707.06347. [Online]. Available: <https://arxiv.org/abs/1707.06347>.
- [30] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust region policy optimization*, 2015. DOI: 10.48550/ARXIV.1502.05477. [Online]. Available: <https://arxiv.org/abs/1502.05477>.
- [31] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, R. Barzilay and M. Johnson, Eds., Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 151–161. [Online]. Available: <https://aclanthology.org/D11-1014>.



# Appendix A

## Additional experiments

The qualitative analysis of obtained text representations (Table X) is not quite representative, mostly due to small test dataset size.

The AG News test dataset consisting of 7600 samples was used. To take into consideration the different mean ratios of deletions of different models (Table IX), the deletion ratio of concrete word was normalized by subtracting the mean deletion ration of this word and dividing by standard deviation.

TABLE XI

The initial average length, mean of removed ratio and the standard deviation of obtained length by SRM-R, SRM-A and SRM-D in AG News test dataset

Architecture	Initial Length	Removed Ratio Mean	Removed Ratio STD
SRM-R	192.4	0.31	0.39
SRM-A		0.74	0.38
SRM-D		0.42	0.41

TABLE XII

Tendency to delete words denoting the days of the week in the AG News test dataset. Although the ratios of deletion vary greatly for different models, each model gives approximately the same degree of relevance to words that are similar in meaning

Token	Count	Model	Deleted	Ratio	Normalized ratio
Monday	497	SRM-R	54	0.11	-0.49
		SRM-A	493	0.99	0.65
		SRM-D	191	0.38	-0.09
Tuesday	487	SRM-R	82	0.17	-0.34
		SRM-A	483	0.99	0.65
		SRM-D	187	0.38	-0.09
Wednesday	443	SRM-R	68	0.15	-0.38
		SRM-A	443	1.00	0.67
		SRM-D	193	0.44	0.03
Thursday	447	SRM-R	45	0.10	-0.51
		SRM-A	445	0.99	0.66
		SRM-D	173	0.39	-0.09
Friday	406	SRM-R	66	0.16	-0.36
		SRM-A	405	0.99	0.67
		SRM-D	93	0.23	-0.47

TABLE XIII

Tendency to delete words denoting the brands in the AG News test dataset. Most of models prefer not to delete the brand names, which is correlate with human understanding of plagiarism generation: brand names can not be paraphrased

Token	Count	Model	Deleted	Ratio	Normalized ratio
AMD	21	SRM-R	6	0.29	-0.05
		SRM-A	0	0.00	-1.91
		SRM-D	2	0.1	-0.80
Dell	30	SRM-R	13	0.43	0.32
		SRM-A	0	0.00	-1.91
		SRM-D	1	0.03	-0.95
Intel	74	SRM-R	26	0.35	0.11
		SRM-A	1	0.01	-1.88
		SRM-D	8	0.11	-0.76
Nvidia	7	SRM-R	0	0.00	-0.77
		SRM-A	0	0.00	-1.91
		SRM-D	2	0.29	-0.33
Xbox	10	SRM-R	2	0.20	-0.26
		SRM-A	0	0.00	-1.91
		SRM-D	1	0.10	-0.78

TABLE XIV

Tendency to delete words connected to sport in the AG News test dataset. Most of models prefer not to delete words relative to sport

Token	Count	Model	Deleted	Ratio	Normalized ratio
Sports	120	SRM-R	0	0.00	-0.77
		SRM-A	2	0.02	-1.87
		SRM-D	15	0.13	-0.72
Chess	3	SRM-R	0	0.00	-0.77
		SRM-A	0	0.00	-1.91
		SRM-D	0	0.00	-1.03
Football	106	SRM-R	1	0.01	-0.74
		SRM-A	20	0.19	-1.43
		SRM-D	37	0.35	-0.18
Hockey	25	SRM-R	3	0.12	-0.47
		SRM-A	6	0.24	-1.30
		SRM-D	9	0.36	-0.15
Tennis	21	SRM-R	1	0.05	-0.65
		SRM-A	8	0.38	-1.93
		SRM-D	8	0.38	-0.10

TABLE XV

Tendency to delete words denoting jobs in the AG News test dataset. Most of models prefer not to delete words relative to jobs

Token	Count	Model	Deleted	Ratio	Normalized ratio
Consumers	42	SRM-R	6	0.14	-0.40
		SRM-A	7	0.01	-1.67
		SRM-D	13	0.31	-0.27
Investors	93	SRM-R	6	0.06	-0.60
		SRM-A	0	0.00	-1.91
		SRM-D	27	0.29	-0.32
Scientists	70	SRM-R	12	0.17	-0.34
		SRM-A	1	0.01	-1.88
		SRM-D	23	0.33	-0.23
Customers	69	SRM-R	5	0.07	-0.58
		SRM-A	4	0.06	-1.76
		SRM-D	20	0.29	-0.32
Shareholders	31	SRM-R	5	0.16	-0.36
		SRM-A	3	0.01	-1.66
		SRM-D	12	0.39	-0.08

TABLE XVI

Tendency to delete words denoting the high technologies in the AG News test dataset. Most of models prefer not to delete words relative to high technologies

Token	Count	Model	Deleted	Ratio	Normalized ratio
Technology	162	SRM-R	9	0.06	-0.63
		SRM-A	0	0.00	-1.91
		SRM-D	10	0.06	-0.88
Semiconductor	28	SRM-R	0	0.00	-0.77
		SRM-A	0	0.00	-1.91
		SRM-D	0	0.00	-1.03
Display	15	SRM-R	1	0.07	-0.60
		SRM-A	0	0.00	-1.91
		SRM-D	2	0.13	-0.70
Wireless	106	SRM-R	4	0.04	-0.67
		SRM-A	1	0.01	-1.89
		SRM-D	12	0.13	-0.75