A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF ÇANKIRI KARATEKİN UNIVERSITY

TEXT GENERATION WITH RECURRENT NEURAL NETWORKS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRONICS AND COMPUTER ENGINEERING

BY

MUSTAFA ABBAS HUSSEIN HUSSEIN

ÇANKIRI

2024

TEXT GENERATION WITH RECURRENT NEURAL NETWORKS

By Mustafa Abbas Hussein HUSSEIN

January 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science

Advisor **:** Assoc. Prof. Dr. Serkan SAVAŞ

Examining Committee Members:

Chairman **:** Assoc. Prof. Dr. Serkan SAVAŞ
Computer Engineering
Kırıkkale University

Member **:** Asst. Prof. Dr. Mustafa TEKE
Electrical and Electronics Engineering
Çankırı Karatekin University

Member **:** Asst. Prof. Dr. Taha ETEM
Electronics and Computer Engineering
Çankırı Karatekin University

Approved for the Graduate School of Natural and Applied Sciences

Prof. Dr. Hamit ALYAR
Director of Graduate School

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mustafa Abbas Hussein HUSSEIN

ABSTRACT

TEXT GENERATION WITH RECURRENT NEURAL NETWORKS

Mustafa Abbas Hussein HUSSEIN

Master of Science in Electronics and Computer Engineering

Advisor: Assoc. Prof. Dr. Serkan SAVAŞ

January 2024

This study delves into a comprehensive study aimed at advancing the field of natural language processing (NLP) through the development of a refined language model. This model leverages the power of recurrent neural networks (RNN) with Long Short Term Memory (LSTM) layers, focusing on contextual understanding for enhanced text generation. The study meticulously compares the performance of this model against historical datasets attributed to Nietzsche and Shakespeare, showcasing its effectiveness and efficiency in handling complex linguistic structures. At the core of this research is the objective to surpass traditional text generation methodologies by integrating advanced contextual analysis into the training of language models. This approach is rooted in the hypothesis that a deeper understanding of context and the semantic relationships between words can significantly improve the accuracy and coherence of generated text. To test this hypothesis, the study employs a state–of–the–art RNN architecture equipped with LSTM layers, known for their capacity to retain long–term dependencies in data sequences, a crucial feature for understanding context in language. The methodology adopted in this study is multi–faceted. Firstly, the research team applies a clustering technique to analyze the context vectors derived from the training data. This technique involves calculating the distances between words within these vectors and assessing their semantic proximity. This method is instrumental in identifying and emphasizing words that are pivotal in conveying meaning within sentences. Subsequently, the model is trained using these context vectors, alongside the traditional input data, to create a more linguistically adept generation model. The results of the study are indicative of the model's superiority in text generation. When applied to Nietzsche’s dataset, the model achieved an impressive accuracy of 95.21% with a loss of just 0.25. In contrast, the

model's performance on Shakespeare's data, though slightly lower, was still remarkable, achieving an accuracy of 91.25% and a loss of 0.3876. These results not only affirm the efficacy of using context vectors in language model training but also underscore the model's versatility across different literary styles and epochs. Moreover, the study introduces an innovative feedback mechanism within the training process. This mechanism allows for continuous evaluation and adjustment of the model, ensuring that training is halted only when the model reaches a satisfactory level of accuracy and reliability. This iterative approach to training and testing guarantees the development of a robust model that is well–tuned to the nuances of human language. In conclusion, this study makes a significant contribution to the field of NLP by demonstrating the effectiveness of context–based training in language models. The use of LSTM layers, in conjunction with clustering methods for context vector analysis, provides a novel approach to understanding and generating complex text structures. The success of this model on diverse datasets highlights its potential applicability in various NLP tasks, paving the way for more sophisticated and accurate text generation technologies..

2024, 39 pages

Keywords: Text generatıon, Recurrent neural network, Training platform, Bidirectional recurrent neural networks, Gated recurrent units

ÖZET

YENİLENEN SİNİR AĞLARI İLE METİN ÜRETİMİ

Mustafa Abbas Hussein HUSSEIN
Elektronik and Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Doç. Dr. Serkan SAVAŞ
Ocak 2024

Bu çalışma, doğal dil işleme (NLP) alanını geliştirmeyi amaçlayan kapsamlı bir çalışmaya derinlemesine dalıyor. Bu model, tekrarlayan sinir ağları (RNN) ile Uzun Kısa Süreli Bellek (LSTM) katmanlarının gücünden yararlanarak, metin üretimini geliştirmek için bağlamsal anlamayı odak noktası haline getiriyor. Çalışma, bu modelin performansını Nietzsche ve Shakespeare'e atfedilen tarihsel veri setleriyle titizlikle karşılaştırarak, karmaşık dil yapılarını ele almadaki etkinliğini ve verimliliğini sergiliyor. Bu araştırmanın temelinde, gelişmiş bağlamsal analizi dil modeli eğitimine entegre ederek geleneksel metin üretim metodolojilerini aşma hedefi bulunmaktadır. Bu yaklaşım, bağlamın ve kelimeler arasındaki anlamsal ilişkilerin daha derin bir anlayışının, üretilen metnin doğruluğunu ve tutarlılığını önemli ölçüde artırabileceği hipotezine dayanmaktadır. Bu hipotezi test etmek için çalışma, dildeki bağlamı anlamak için hayati bir özellik olan veri dizilerindeki uzun süreli bağımlılıkları koruma kapasitesiyle bilinen LSTM katmanlarına sahip en son RNN mimarisini kullanmaktadır. Bu çalışmada benimsenen metodoloji çok yönlüdür. Öncelikle, araştırma ekibi, eğitim verilerinden türetilen bağlam vektörlerini analiz etmek için bir kümeleme tekniği uygulamaktadır. Bu teknik, bu vektörler içindeki kelimeler arasındaki mesafeleri hesaplamayı ve anlamsal yakınlıklarını değerlendirmeyi içerir. Bu yöntem, cümleler içindeki anlamı iletmekte kilit rol oynayan kelimeleri belirlemekte ve vurgulamakta önemlidir. Ardından, model, bu bağlam vektörleri kullanılarak, geleneksel girdi verileriyle birlikte eğitilerek, daha dil bilimsel açıdan yetkin bir üretim modeli oluşturmak için eğitilir. Çalışmanın sonuçları, modelin metin üretimindeki üstünlüğünü göstermektedir. Nietzsche'nin veri setine uygulandığında, model %95.21 etkileyici bir doğruluk ve sadece 0.25'lik bir kayıp ile başarılı oldu. Buna karşılık, modelin Shakespeare verilerindeki performansı, biraz daha

düşük olmasına rağmen yine de dikkate değerdi ve %91.25 doğruluk ve 0.3876 kayıp elde etti. Bu sonuçlar, dil modeli eğitiminde bağlam vektörlerinin kullanımının etkinliğini doğrulamanın yanı sıra, modelin farklı edebi stiller ve dönemler arasındaki çok yönlülüğünü de vurgulamaktadır. Ayrıca, çalışma, eğitim süreci içinde yenilikçi bir geri bildirim mekanizması tanıtıyor. Bu mekanizma, modelin sürekli değerlendirilmesini ve ayarlanmasını sağlayarak, model yeterli doğruluk ve güvenilirlik seviyesine ulaştığında eğitimin sadece o zaman durdurulmasını sağlar. Bu yinelemeli eğitim ve test yaklaşımı, insan dilinin inceliklerine iyi ayarlanmış sağlam bir modelin geliştirilmesini garanti eder. Sonuç olarak, bu çalışma, dil modellerinde bağlama dayalı eğitimin etkinliğini göstererek NLP alanına önemli bir katkıda bulunmaktadır. LSTM katmanlarının kullanımı, bağlam vektör analizi için kümeleme yöntemleriyle birlikte, karmaşık metin yapılarını anlama ve üretme konusunda yenilikçi bir yaklaşım sağlar. Bu modelin çeşitli veri setlerindeki başarısı, çeşitli NLP görevlerinde uygulanabilirliğinin potansiyelini vurgulamakta ve daha sofistike ve doğru metin üretim teknolojileri için yol açmaktadır.

2024, 39 sayfa

Anahtar Kelimeler: Metin üretimi, Tekrarlayan sinir ağı, Eğitim platformu, Çift yönlü tekrarlayan sinir ağları, Geçitli tekrarlayan birimler

PREFACE AND ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Assoc. Prof. Dr. Serkan SAVAŞ, for his patience, guidance and understanding.

Mustafa Abbas Hussein HUSSEIN

Çankırı-2024

CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial intelligence
LAS	Labeled attachment score
LSTM	Long short term memory
NLP	Natural language processing
NLG	Nature language generation
RNN	Recurrent neural network
TG	Text generation

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

A software method known as Natural Language Generation (NLG) produces outputs in natural language. One of its applications is Text Generation (TG). One of the most widely cited surveys of TG techniques defines TG as a subfield of artificial intelligence and computational linguistics. It is concerned with the development of computer systems capable of producing understandable texts in English or other human languages from some underlying non–linguistic representation of information (Yu *et al.* 2022). There is some disagreement over whether the inputs to a TG system must be non–linguistic. However, it is generally agreed that text is the outcome of any TG process. TG approaches are frequently used in the generation of various reports, including weather and medical reports, image captioning, and chatbots. Automated TG is similar to how humans express their ideas verbally or in writing. Psycholinguists refer to this process as language production, which can also be theoretically or computationally characterized for the purpose of psychological study (Rebuffel *et al.* 2022).

The output of some nodes in a Recurrent Neural Network (RNN), a class of artificial neural networks where connections between nodes can form a loop, may influence subsequent input to those same nodes. This allows them to exhibit temporal dynamic behavior. RNNs, which are built from feedforward neural networks, can process input sequences of various lengths using their internal state (memory). As a result, they are applicable to tasks like speech recognition or connected, unsegmented handwriting recognition. Theoretically, RNNs can execute any program to process any input sequence (Alhumoud and AlWazrah 2022). RNNs are the class of networks with an infinite impulse response, in contrast to Convolutional Neural Networks (CNNs), which have a finite impulse response. Both types of networks exhibit temporally dynamic activity. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and substituted with a strictly feedforward neural network. In contrast, an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled (Wang *et al.* 2022).

1.1 Historical Background of Recurrent Neural Networks

The history of RNNs dates back to the 1980s, with the initial idea being to create networks that could remember previous inputs using internal state representations. The early 1990s saw the introduction of simpler RNNs, primarily used for tasks like speech recognition and language modeling. Key milestones included the introduction of Long Short–Term Memory (LSTM) in 1997 and Gated Recurrent Units (GRUs) in 2014, which simplified the LSTM architecture while maintaining its ability to capture long–term dependencies (Yu *et al.* 2019)

Advanced optimization techniques like RMSprop and Adam contributed to mitigating these challenges, leading to more stable and efficient training of RNNs. Overcoming these challenges has enabled RNNs to be effectively used in more complex applications, such as speech recognition, machine translation, and text generation.

1.2 Text Generation Technology

TG is a fascinating and rapidly evolving field within artificial intelligence (AI) that focuses on creating written content through automated means. This technology is primarily driven by advanced algorithms and machine learning models that have the capability to generate coherent, contextually relevant, and often highly creative text. The applications of text generation are vast and varied, ranging from creating fictional stories and poetry to generating news articles, reports, and even code (Fatima *et al.* 2022). At the heart of modern text generation systems are neural network models, such as the Transformer architecture, which have been trained on vast datasets of text. These models learn patterns and structures within the language, enabling them to predict and generate subsequent text that is syntactically and semantically coherent. The most advanced of these models are capable of producing text that is often indistinguishable from that written by humans. Text generation technology has numerous practical applications. In the creative industries, it can assist in drafting narratives and scripts. In business, it can automate content creation for reports, emails, and marketing materials. In technology, it aids in coding and documentation. However, this technology also poses ethical and

societal challenges, such as the potential for misuse in creating misleading information or deepfakes (Klahold and Fathi 2019).

As the field continues to grow, it raises important questions about the role of AI in content creation, the ethics of automated text generation, and the future of human–AI collaboration in writing. The ongoing developments in this area represent a significant step forward in the capabilities of AI and offer a glimpse into the future of automated content creation.

1.3 RNN in Text Genration

RNN are a crucial tool in text generation due to their unique structure and capabilities. RNNs are designed to handle sequential input, making them ideal for text where the order of words is crucial for meaning. They have a memory element, retaining information from previous inputs through internal states, which influences the processing of current and future inputs. The hidden layers in RNNs are loops, allowing for the remembering of information across time steps (Baktha and Tripathy 2017, Marhon *et al.* 2013). RNNs have applications in language modeling, machine translation, text summarization, and speech recognition. They are adept at predicting the next word or character in a sequence, generating coherent and contextually relevant text. They can also analyze and condense long pieces of text while maintaining key information and context. Advantages of RNNs include context–awareness and flexibility, as they can handle variable–length input sequences. However, they face limitations such as the vanishing gradient problem and computational intensity. Despite these challenges, RNNs have made significant advancements in text generation, with newer architectures like Long Short Term Memory LSTM and gated recurrent units being developed to address these limitations. These advances have led to more robust and effective models for text generation, pushing the boundaries of natural language processing and AI–driven content creation (DiPietro and Hager 2020).

1.4 Development of Language Models

Before the advent of neural networks, language generation was primarily based on statistical models like n–gram models. These models generate text based on the probability of a word sequence occurring in a given corpus. However, they were limited in their ability to understand context or generate coherent long–form content. The development of language models for text generation has been a significant advancement in artificial intelligence and natural language processing. Previously, language generation relied on statistical models like n–gram models, which were limited in their ability to understand context or generate coherent long–form content. The introduction of neural networks, particularly RNNs and LSTMs, improved the quality of text generation. Attention mechanisms, particularly in models like the Transformer, allowed models to focus on different parts of the input sequence, leading to more coherent and contextually relevant text generation (Milanova *et al.* 2019). Fine–tuning and task–specific models have also been a trend, with a focus on addressing ethical considerations such as mitigating biases in generated text and ensuring responsible use. Future trends include continued improvements in model architectures, training methods, and the integration of multimodal inputs, as well as an ongoing effort to make these models more efficient and accessible. So, the development of language models for text generation is a rapidly advancing field, pushing the boundaries of natural language understanding and generation.

LSTM network is a powerful methods for text generation. LSTM networks are RNN that are effective for processing sequences like text, as they remember information over extended periods. They can learn patterns in word sequences and generate new sequences based on these learned patterns. Dropout is a regularization technique used to prevent overfitting in neural networks, preventing overfitting when a model performs well on training data but poorly on new data. Dropout involves randomly "dropping out" a fraction of the network units during training, allowing the model to generalize better to new data. When combined, these techniques can generate diverse and contextually coherent text. This combination is widely used in language modeling, machine translation, and content creation (Souri *et al.* 2018).

1.5 Advancements and Improvements in RNN Technology

Recent advancements in RNN technology have led to improvements in LSTM architectures, making them more efficient in handling longer sequences and complex datasets. Bidirectional RNNs have significantly enhanced context understanding in tasks like sentiment analysis and language translation. Attention mechanisms have been integrated into RNNs, allowing the network to focus on specific parts of the input sequence, improving performance in tasks like machine translation and text summarization. Advanced regularization methods, such as dropout variants, have been introduced to prevent overfitting. Optimization algorithms, such as evolving gradient descent methods, have improved the training speed and stability of RNNs, enabling them to learn more effectively from complex datasets. Hybrid models, which combine RNNs with CNNs or Transformers, are also emerging to leverage the strengths of each architecture for specific applications like text generation. RNNs are good at processing sequence data and capturing time–series information, but struggle with very long sequences due to memory limitations and are slower to train and infer compared to Transformers. Transformers are highly efficient in parallel processing, superior in handling long–range dependencies, and can be prone to overfitting on smaller datasets due to their large number of parameters (Milanova *et al.* 2019).

1.6 Problem Statement

One major challenge in early RNN development was the vanishing and exploding gradient problem. RNNs often encountered issues where gradients would either become too small or too large , making the training process unstable and inefficient. This problem affected the network's ability to learn long–range dependencies, crucial for tasks like language modeling and text generation. To address this problem, the development of LSTM and GRU architectures introduced gates that controlled the flow of information, allowing the network to retain important information over long sequences without gradients vanishing or exploding. Advantages of RNNs include context–awareness and flexibility, as they can handle variable–length input sequences. However, they face limitations such as the vanishing gradient problem and computational intensity. Despite

these challenges, RNNs have made significant advancements in text generation, with newer architectures like LSTM being developed to address these limitations. These advances have led to more robust and effective models for text generation, pushing the boundaries of natural language processing and AI–driven content creation.

1.7 Objective of Study

This study presents a systematic survey on recent development of neural text generation model. This study propose a generative model, called RNN, by combining with text generative adversarial net to better learn the data distribution and generate various realistic text. This study aims to advance Natural Language Processing (NLP) by developing a sophisticated language model that excels in understanding and generating text. The model employs RNN with LSTM layers to improve text generation by focusing on context. The model's effectiveness is tested against historical datasets attributed to notable figures like Nietzsche and Shakespeare, showcasing its ability to handle complex linguistic structures across different literary styles and epochs.The study is based on the hypothesis that a deeper understanding of context and semantic relationships between words can significantly improve the accuracy and coherence of generated text. The model's performance is quantified using a state–of–the–art RNN architecture equipped with LSTM layers, which are adept at retaining long–term dependencies in language. Advanced training methodologies, such as clustering techniques, are employed to analyze context vectors derived from training data. The model's performance is quantified, revealing impressive accuracy and low loss rates when applied to Nietzsche's and Shakespeare's datasets. An innovative feedback mechanism is introduced in the training process, allowing for continuous evaluation and adjustment of the model.The study makes a significant contribution to NLP by demonstrating the effectiveness of context–based training in language models and the use of LSTM layers in conjunction with clustering methods for context vector analysis. This paves the way for more sophisticated and accurate text generation technologies.

1.8 Contribution of Study

This study presents an approach to natural language processing (NLP) by focusing on contextual understanding in language models. It uses RNN with LSTM layers to enhance the accuracy and coherence of generated text. The study also integrates context vectors in model training, using clustering techniques to identify and emphasize contextually important words. This methodology improves the capability of language models to understand and replicate human–like language patterns more effectively.

The thesis is empirically validated using historical literary datasets attributed to Nietzsche and Shakespeare, demonstrating the model's effectiveness across different literary styles and setting a benchmark for comparing modern NLP models against complex historical texts. The study introduces an innovative feedback mechanism in the training process, allowing for ongoing evaluation and refinement to ensure high accuracy and reliability. The quantitative analysis of the model performance demonstrates high accuracy and low loss rates when applied to challenging datasets, supporting the thesis's claims about the effectiveness of the proposed methodologies. The thesis suggests a wide range of potential applications for this advanced language model in various NLP tasks, opening avenues for more sophisticated text generation and analysis tools. Lastly, the thesis contributes significantly to the evolution of NLP techniques by presenting a blend of advanced machine learning techniques, such as LSTM and context vector analysis, setting a new precedent for future research and development.

1.9 Organization of Study

As shows in Figure 1.1, This study is structured into five primary sections. First, the introduction delves into the concept of text generation, framing the research problem and defining its objectives. The subsequent section provides a comprehensive background and literature review, focusing on recent advancements aimed at enhancing existing text generation models. In the third section, detailed insights into the project's methodology are presented, showcasing our approach to evaluating and contrasting current student success prediction models. The fourth segment presents the conclusions drawn from our

experimental findings, further supplemented by discussions. Finally, the concluding section encapsulates the principal discoveries, delves into broader discussions, and hints at prospective directions for future research.

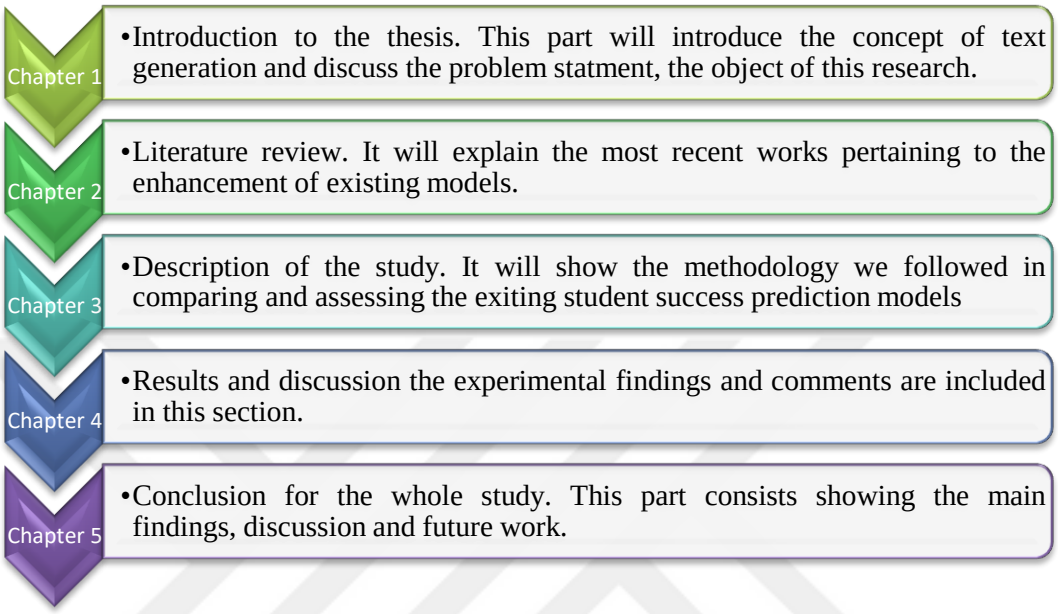

Figure 1.1 Thesis outlines

2. LITERATURE REVIEW

Pawade *et al.* (2018) aimed to compile previously submitted stories into new narratives. They explored two methods for generating these stories based on the types of submissions received. Firstly, they considered narratives with a range of themes and styles. Secondly, they worked with various iterations of interconnected narratives and similar characters. The system's output was evaluated based on grammar accuracy, event causality, level of interest, and originality.

Khatri *et al.* (2015) developed a deep learning and natural language processing framework to generate context for products on e–commerce websites. Their approach, consisting of five unsupervised steps, involved abstraction and extraction. These steps included: a) gathering item–specific keywords and descriptions, b) creating a blacklist dictionary to avoid duplication, c) selecting sentences and paragraphs aligned with the user's intent, d) sentence creation using RNN and LSTM, followed by extraction with Skip Grams and Negative Sampling, and e) organizing the output contextually with TextRank.

Thomaidou *et al.* (2013) proposed a method to create persuasive text excerpts from advertisements using the website's landing page. Their process involved information extraction and sentiment analysis, followed by natural language development.

Jain *et al.* (2017) described a classical approach for creating stories from brief narrations. They utilized phrase–based statistical machine translation to create phrase–maps in the target language, concatenating each phrase–map to produce the final text.

Li *et al.* (2013) introduced 'SCHEHERAZADE', a story creation method utilizing crowdsourced plot graphs. Each graph represents an action and includes nodes such as standard events, conditional events, and optional events, connected by mutual exclusion and precedence rules. The plot graph establishes the order of events, although improvements are needed in story quality and reliability.

Nallapati *et al.* (2016) developed an encoder–decoder RNN with a large vocabulary and attention mechanism for abstract text summarization. They also used a switching decoder/pointer architecture for handling uncommon words and employed hierarchical attention to identify key sentences. The method was tested on two distinct databases, yielding promising results.

Jaya *et al.* (2010) presented an ontology–based system for story creation. Users input people, objects, and settings, and the system uses ontology to assign attributes and construct stories within the domain's context.

(GV 2011) improved upon Proppian's System with enhanced semantic capabilities. Each sentence undergoes a reasoning process to maintain semantic significance, with ontology providing a clear declaration of common concepts.

Le *et al.* (2013) showcased an abstract text synthesis method using discourse rules and syntactic constraints for keyword–based sentence creation, followed by a Word graph for generating coherent and concise statements.

Sakhare *et al.* (2014) suggested a hybrid text summarization approach combining sentence structure and attribute–based algorithms. After preprocessing the document, a neural network is trained on sequence structures and frequency tags using backpropagation. Features are scored in a feature–based neural network, and the final sentence score is calculated by averaging weighted sentence scores. The top–ranked sentences are then selected for summarization. (Kavila *et al.* 2015) focused on an extractive text summarization technique based on modified sentence symmetry and weighting methods, specifically targeting IEEE papers.

Hatipoglu and Omurca (2016) developed a smartphone app for summarizing Turkish Wikipedia articles, improving effectiveness by considering structural and semantic features, including Wikipedia–based and LAS–based semantic features.

Colon *et al.* (2014) mentioned a professional writing prompts application, where three random WordNet words are chosen, and ConceptNet is used to explore and correlate concepts to establish a theme and characters, iteratively searching ConceptNet to develop a complete plot.

Mehta *et al.* (2016) created a system using a 'generate and rank' approach combined with NLG techniques. Inputs include the theme and desired duration of the story. The content is identified using centric theory, action graph, local coherence, and a lexical database. Planning involves a dependency tree, and the story is generated using a deep syntactic structure tree and rule–based approach.. A list of all the strategies covered is provided in Table 2.1.

Table 2.1 Literature review according to several studies

REFERRENCE	AIM OF STUDY	DESCRIPTION	RESULTS
Pawade *et al.* (2018)	In terms of the nature of the inputted stories, generation ne story via explored two alternatives.	Different storylines, characters, and volumes of the same stories in which the characters are similarly similar and the storylines are interconnected.	63%
Khatri *et al.* (2015)	Create the product context from the online store	Framework for algorithms focused on Deep Learning and Natural Language Processing	88%
Thomaidou *et al.*(2013)	Create an advertisement from website content	Sentiment analysis and the creation of natural language	54%
Jain *et al.* (2017)	Create a tale using several narration sequences	Boyang uses phrase centered statistical translation of machine with architecture recurrent neural network sequencing	65%
Li *et al.* (2013)	Generating stories established on events	Graphs of Crowdsourced of Plots Ramesh	64.5%
Nallapati *et al.* (2016)	Text Summarization in the Abstraction	RNN encoder–decoder with large vocabulary and attention	55.6%
Jaya *et al.* (2010)	Story creation based on the user's chosen characters, items, and setting	Ontology	64.3%
GV (2011)	Story generation that is effectively semantic	The Proppian's System and Ontology have been updated	76.2%
Le *et al.* (2013)	Text Summarization in the Abstraction	Discourse conventions, syntactic restrictions, and word clouds	56.9%
Sakhare *et al.* (2014)	Summarizing a text	Using average weighted of the sentence scores from syntactic structure constructed neural networks and feature centered neural networks, we may rank the sentences.	80%
Kavila *et al.* (2015)	Extracting and summarizing text	modified sentence symmetry and the modified weighting technique	80%
Hatipoglu and Omurca (2016)	Turkish Wikipedia Version Summarization using Mobile App	approach hybrid using organizational characteristics, LSA, and semantic features based on Wikipedia	78.5%
Colon *et al.* (2014)	A top–notch writing exercise program	WordNet, ConceptNet, Natural Language Processing, Concept Correlation, Plot creation, Knowledge Base, If Then rules–based Inference engine	67%
Mehta *et al.* (2016)	Create a tale depending on the user–provided theme and length.	Process and approach for NLG	65%

3. MATERIALS AND METHODS

In this methodology, we use two historical novel datasets (Shakespeare and Nietzsche). These datasets have undergone preprocessing to remove numbering and citations, and then they were split according to a 70% training and 30% testing ratio. In this study, an RNN–LSTM model has been developed to evaluate the performance of our proposed approach. Figure 3.1 depicts the process in a flowchart.

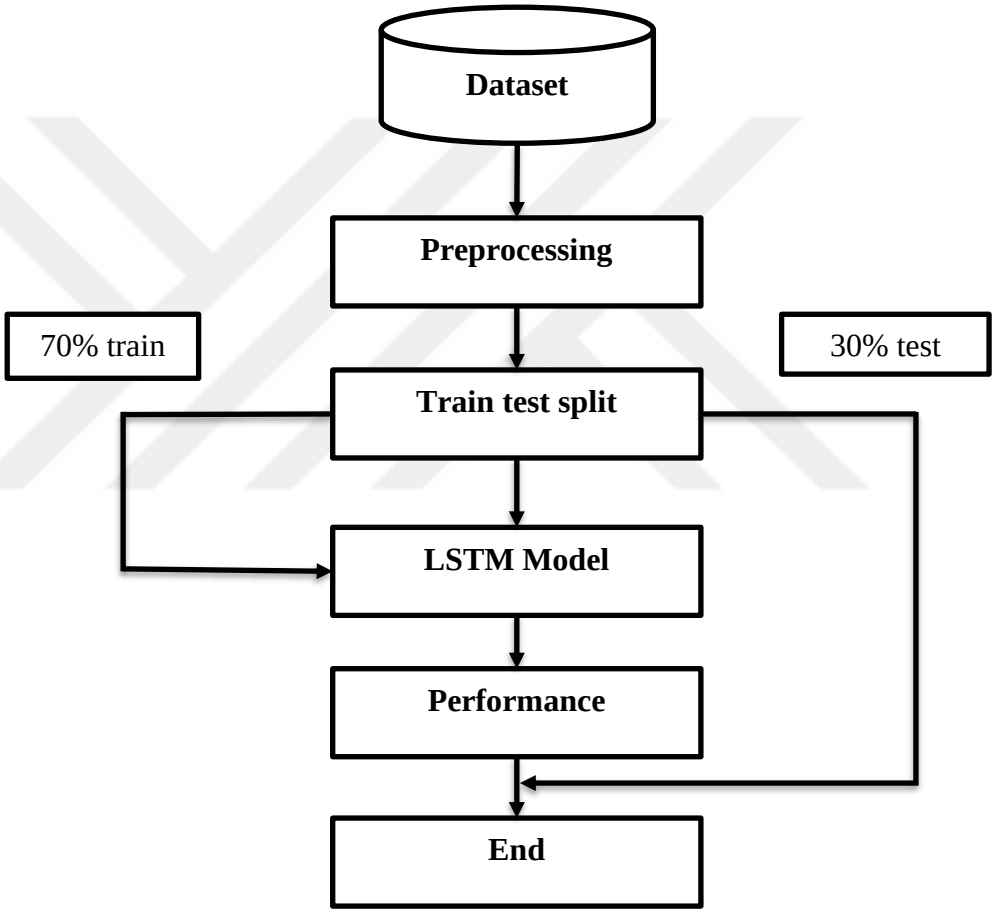

Figure 3.1 Flowchart of the proposed study

For the data preprocessing, Figure 3.2 shows the process used in this methodology

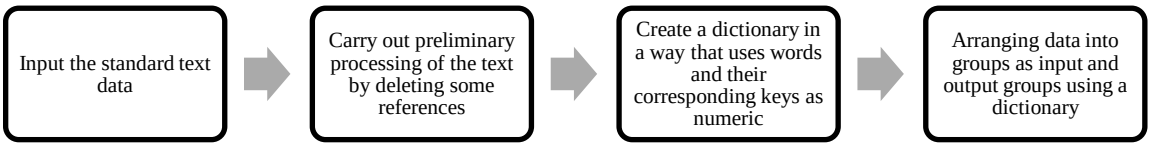

Figure 3.2 Flow chart of dataset collection and preprocessing

To process text data, which a computer cannot handle directly, it's essential to load the necessary libraries for RNNs, data pre–processing, and NLP. This is because we must convert the text into a form that is readable and interpretable by a computer (Abbas 2022). This conversion can be achieved using one of two operations: one–hot encoding or word embedding. One–hot encoding transforms text into vectors of 1s and 0s, generating a dictionary of words that recur in various forms within the text. These models are simple to train and retain a considerable amount of useful information for accurate predictions, making them highly versatile. In these models, text words are represented within a numerical beam, with values greater than 0 and 1. The input is appropriately configured to match the output. The RNNs are implemented with LSTM layers through the Keras library, a crucial interface for TensorFlow applications. To avoid overfitting, we use a 3–layer model. The first layer of the LSTM network comprises 250 memory nodes, responsible for storing information and sequentially returning it. The last layer, the output layer, provides accurate character sequence predictions. The model is trained using Adam's algorithm. Post–training, we compare the model's performance by analyzing the proportionality between the generated inputs and outputs. Finally, a dictionary is required to convert the model's outputs back into numbers, as they were initially letters.

3.1 Text Generation

Text generation is one of the basics in natural language processing by predicting the letter A in the initial word of the text to be generated, which we encounter in many daily applications such as writing texts in Gmail and documents in Google, and keyboards in smart phones, computers and some modern chat bots. Text generation of NLP tasks in advanced neural machine translation in this research, the text was generated to predict the next letter in the sentence from the previous letters, and the appropriate sentence was predicted by recognizing the appropriate context to complete the text, and RNN that

depends on LSTM layers was used to achieve the goal. We will be using a deep learning process using TensorFlow's, Keras, which is a programming interface in python in high–level applications.

3.2 Recurrent Neural Networks

RNNs are among the most important networks in deep learning based on time–dependent data modeling due to their ability to hold input information and time steps in memory. Since texts contain contextual information and sentences that are only defined by word order, natural language processing relies heavily on sequence structures such as RNN (Figure 3.3).

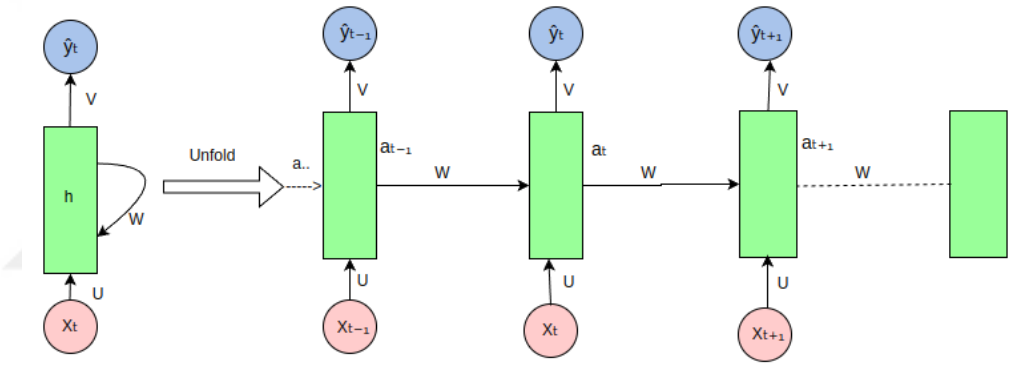

Figure 3.3 RNN Architecture (Poudel 2023)

A RNN based on LSTM layers has been developed for order prediction. While executing LSTM layers, we set the state type as True to preserve the time step memory of previous states while learning with subsequent batches of input in aperiod. It helps in recognizing the context between successive sequences (Bo Chang 2019).

3.3 Long Short Term Memory

Because the problem of RNNs is that they cannot save information for a long time, they forget and delete the entries. To solve this problem, LSTM networks (Mir–Levy 2018) are used as a solution to learn RNNs that feature short–term memory. Also, when adding new entries, RNN completely modifies and arranges the existing information. Because RNN

is unable to differentiate between important and unimportant information. Whereas in LSTM you make a small adjustment in the information entered whenever a new batch is added because the LSTM gate determines the flow of information to the RNN as in the Figure 3.4.

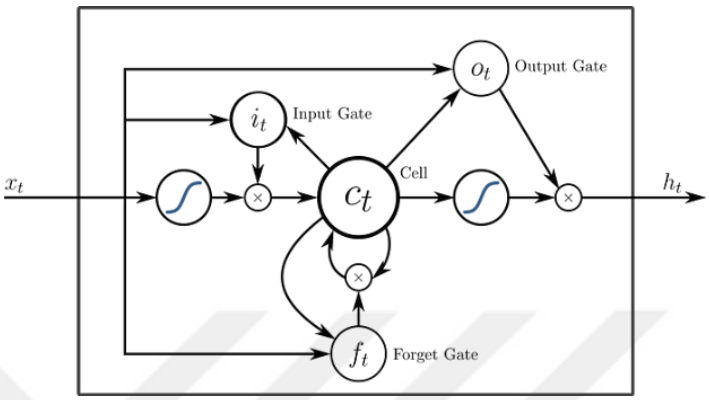

Figure 3.4 LSTM gate define the flow of information into the iterative network (Abdurakipov *et al.* 2018)

As shown in Figure 3.5, the gates in the LSTM identify which data are useful and unhelpful in prediction in the appropriate sentence and which data to omit as unnecessary. The gates are (input – output – forget).

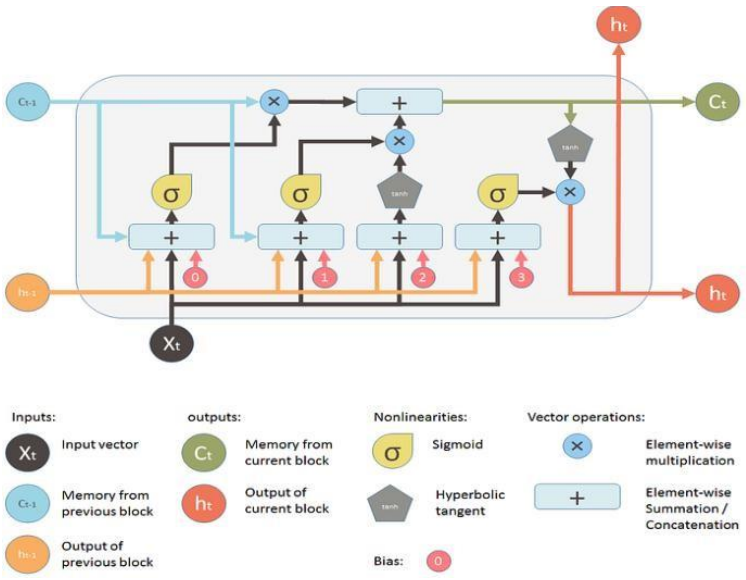

Figure 3.5 LSTM gates (input, output, forget) (Rahman *et al.* 2019)

- Input gate: Through this gate, data is entered, consisting of neurons whose number is according to the number of data, and it is the cell responsible for the values that must be added and their values modified using the sigmoid function. There is also an activation function called tanh that makes the input values within the range [–1,1] that modifies and coordinates the information before entering it to the next layer (Figure 3.6) (Dautel *et al.* 2020).

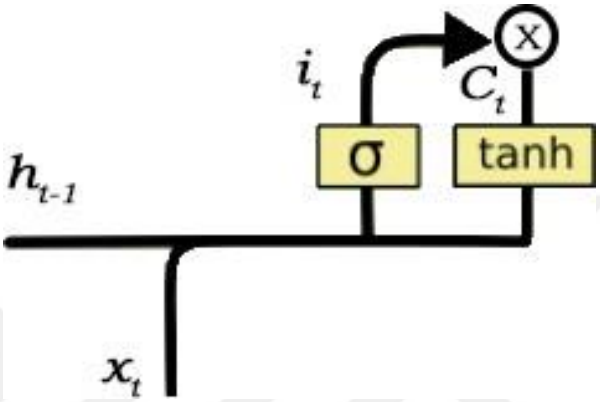

Figure 3.6 Input gate into the RNN (Dautel *et al.* 2020)

- Output Gate: Through this cell we get the data that interests us from the previous cell after creating a matrix of values within the range [–1,1] using the tanh function,and the sigmoid function determines the final values that will be displayed as output (Figure 3.7).

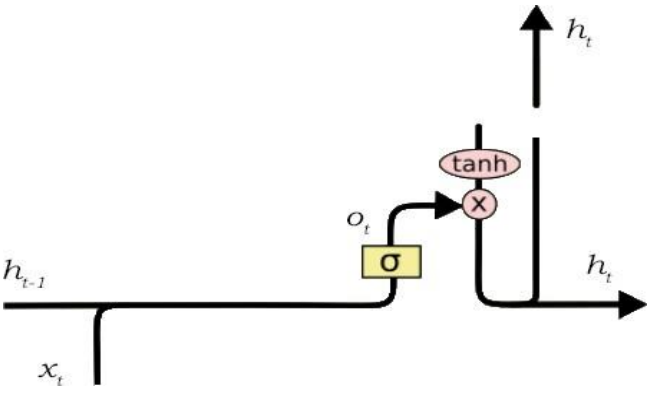

Figure 3.7 Output gate into the RNN (Dautel *et al.* 2020)

- Forgetting Gate: When entering data, this gate identifies useful information to store and deletes and forgets non–important information to improve the performance of the

recurrent neural network and train it quickly. This gate consists of two inputs, one of which is the result of the previous node, and the other is the input of the next node. After determining the biases and matrix weights and applying the sigmoid function for each neuron The information is preserved and remembered according to thevalues. If the value is 0, the oblivion gate deletes the information. If the value is 1, then the data is important (Figure 3.8).

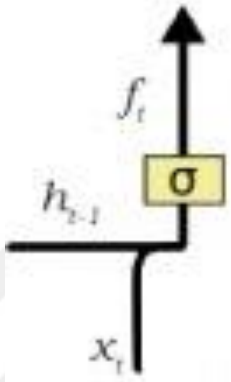

Figure 3.8 Forget gate into the iterative network (Dautel *et al*. 2020)

An RNN receives a sequence of characters or words as input, such as a string of characters for predicting the next character. It has internal memory, which captures information about previous inputs, crucial for understanding textual context. RNNs process sequences considering both the current input and what it has learned from previous inputs, unlike other neural networks that process each input independently.

3.4 Integrating LSTM Based RNNs

By studying RNNs that are characterized by short–term memory and applied in complex tasks and language programming, we propose to build the proposed model that depends on LSTM layers that is characterized by long–term memory and is ableto recognize the appropriate context of the text to be generated instead of predicting the character after determining the necessary abnormalities and outputs.

RNNs are taught after sequencing sequences of sentences and successive wordswithout knowing the context. The structure of the network is similar when moving from machine translation to a question–answer system. To capture the important features of the input strings using the seq2seq encoding and decoding networks that give grammatically understandable sentences, but they are illogical. With the continuation of training the model and creating words, the sentences will not have coherence, but by training the proposed model with all the units together, the appropriate sentences are recognized and the ambiguity in the lexicon is avoided.

Based on this, a model was proposed using RNNs to generate text using the appropriate context for the input information, and the proposed model is taught on input, output, and the relationship of words to each other through the appropriate context, by applying the language model, where the network generates sentences on a specific set of contextual words. When building the model, it is evaluated by not repeating the same sentences, predicting new and grammatically understandable sentences, and the degree of convergence between the input context and the resulting text in terms of linguistic significance, regardless of the quality in constructing sentences. A clustering algorithm was proposed to detect important information in sentences and evaluate thisalgorithm using the cosine similarity measure.

3.5 RNNs with Sequential Data

The recurrent neural network makes a prediction based on the previous characters, which requires a string of input characters and a string of characters in the training data. The model taught by the similarity and probability value of the match (Zhang *et al.* 2019) Figure 3.9 shows the predict the next word in a string.

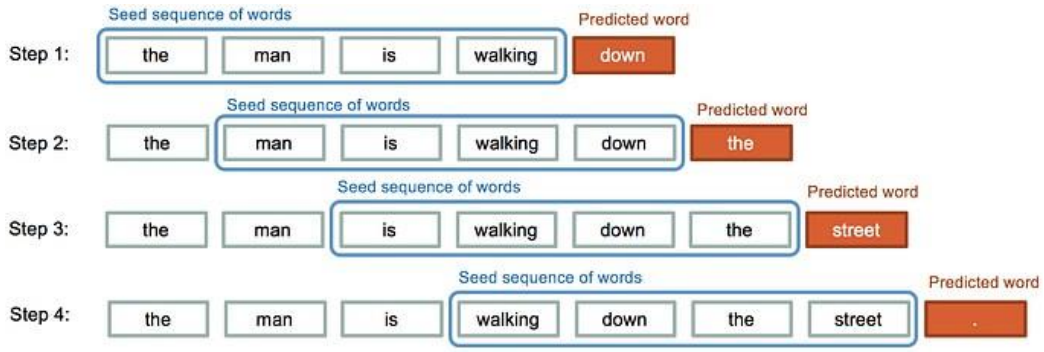

Figure 3.9 Steps to predict the next word in a string

3.6 Context Based Text Generation

The linguistic model is trained on a set of selected text input data, predicts the next word, then the resulting word is appended to the existing input words which is a new input, and this process is repeated for several iterations until an understandable and grammatically strong text is generated (Puduppully *et al.* 2019).

The proposed model is trained at the word or letter level. Here we trained it at the word level. Since it is a good method for directed learning, there are no restrictions in choosing the data set. To train the model, a series of sentences are given as input and the next word in the beam is set as the predictive word (the word suitable for prediction).

Sets of input sequences consisting of predictive words are formed as data. As the neural network is trained on several instances, it approximates detection with the next word of a group of significant words in the sequence beam.

For input words i, . . . , i + n, the predictor word is i + n + 1, where i represents the iteration to generate training examples in the dataset and n represents the length of the input beam in terms of words. Regarding the number of hidden layers in the network and the size of the input data, the training process requires a long time and a large computational process for the device. The model is trained on textual data written in the form of a story with a clear and appropriate sequence to create a text with a clear meaning. The proposed context–dependent model is based on the same word prediction strategy with construct contrast with input and output vectors. A method was adopted in the research to extract

the context from the text through context scans and inputs to learn the proposed model efficiently and with high accuracy.

In this study, a different method for extracting context vectors using sentence grouping, having the same dimension of input and output, is proposed. The proposed model is based on the hypothesis that the context vector in the sequence maintains the important information of the sequence. For example, the important context vector in the sentence "The fast deer jumps over the lazy bunny" contains words like "pets, woods, motions". Context rays are computed for all input and output vectors before training the model to generate context–based text. The context can be just an important word, a group of words, a comprehensible topic, or any sentence that represents the semantic content of the sentence and is meaningful.

3.7 Dataset and Preprocessing

In this study, two dataset used to evaluate the model performance. The first dataset, which includes all of Shakespeare's plays (Liamlarsen 2018), is organized for text analysis or natural language processing tasks. Each column in the dataset has a unique identifier or counter, facilitating easy reference and management of the data. The Play column identifies the specific Shakespearean play from which the lines are taken, categorizing them according to the play they belong to. The Act–Scene–Line column provides detailed location information for each line within the play's structure, indicating the specific act, scene, and line number. The Player column names the character who is speaking the line, as each character has a distinct voice and role in Shakespeare's plays. The Line Being Spoken column contains the actual words written by Shakespeare and spoken by the characters in his plays, essential for textual analysis, including studying Shakespeare's language, themes, literary devices, and character development. The dataset is a comprehensive collection of Shakespeare's theatrical works, organized for detailed literary analysis, and can be used for various purposes, such as studying Shakespeare's writing style, performing textual analysis to find patterns or themes across different plays, training machine learning models for tasks like text generation or sentiment analysis, and more.

The second dataset is the Nietzsche Texts dataset (Kris 2019) is a comprehensive collection of English text, specifically designed for natural language processing and text generation tasks. The dataset is a valuable resource for experimenting with NLP and machine learning techniques, as it consists of texts by philosopher Friedrich Nietzsche. Nietzsche's writing is known for its rich vocabulary, complex sentence structures, and philosophical content, making it an ideal resource for challenging NLP models. The language is likely to be philosophical and abstract, offering interesting challenges for text generation models. The dataset is used in an example in François Chollet's book "Deep Learning with Python," specifically in a notebook demonstrating text generation with an LSTM model. It is recommended for educational and experimental use, as LSTM networks are well–suited for learning sequences and patterns in text. The dataset is also useful for character–level text generation, natural language understanding, style transfer in text, and philosophical text analysis.

The data was processed by removing punctuation marks and usingsentences consisting of 7 words at the minimum level of training. The training sentences consist of 100,000 sentences and are arranged within rays of the input process to preserve the computational memory space used in the device for ease of training. from the set of standard data, the input and output training vectors are taken and all sentences are directed using one hot symbol. Thus, each word is represented using a single vector of dimension V, where V is the magnitude of the vector and thelength of each input sequence ray except for the context extracted over 8 words. The context vector generated by the clustering process of the training sample is also a vector of dimension V. Context vector encoding is based on the extraction method, where context vectors andinput vectors are formed and the output vector contains the next word in the sequence.

A sequential model is implemented for RNNs with configuration: bi–directional for LSTM, 250 hidden units, activation method uses 'relu' for inner layers and activation method 'softmax' for outer layer. Keras library (Chollet 2015) is used to train and test the model. Since the language model is a prediction problem, and because the model must predict the one–coded word to be predicted, the "softmax" function is suitablefor determining the maximum probability value of the word.

3.8 Method of Grouping Words

All methods used in prediction depend on selecting the most appropriate word for the given context. In this method, we aim to obtain the context vector by using all words in the sentence rather than a single word. This is based on the hypothesis that the context vector exists within a matrix of high–dimensional vectors, which capture the semantic and grammatical meaning of the sentence. The inclusion of an appropriate word provides an oriented representation, encoded with specific semantic information. Models such as Word2Vec (Tomas 2013), GloVe (Pennington *et al.* 2014), and FastText (Mikolov *et al.* 2018) are among the highly regarded and commonly used methods. Word2Vec, in particular, was chosen for its efficiency in storing and embedding the word grouping process. Text embeddings manage to preserve two key properties: linguistic similarity of meaning and semantic relations. By exploiting these advantages, context vectors extracted from text using word grouping can be particularly effective.

This proposed method hypothesizes that the context of a sentence is determined by a context vector contained within the word vector matrix. The aim is for the semantic and syntactic similarity between the context vector and the compound sentence vector to be high. The sentence beam, consisting of word vectors in the sentence, is calculated by summing them up, taking into account the semantic relations. This summation retains the semantic and grammatical content of all included words. Through semantic similarity, we can obtain the closest beam of compound word vectors from the word vector matrix. Consequently, we create a word vector space composed of a high–dimensional space containing all words and vocabularies from the dataset, as represented in the word embedding. In this space, sentences and words with useful and important meanings are positioned close to each other. The contextual word vector matrix is computed by training a self–supervised Word2Vec model on the training data.

3.9 Training an RNN

Traditional RNNs face challenges in training due to the vanishing gradient problem, which causes information decay over time. To overcome this, advanced variants like LSTM have

been developed. These architectures are capable of learning long–term dependencies and are widely used in tasks like language modeling, machine translation, and speech recognition. For example, in text prediction, an RNN model is trained on a large dataset of text, learning the probability of each character occurring in a given context. This makes RNNs ideal for tasks involving sequential data like text, speech, and time series analysis. In this study, The RNN is trained on a proposed two datasets, which are large corpus of text, for text prediction. It learns the probability of a character following a sequence of characters using backpropagation through time to reduce prediction errors. The RNN is trained to produce outputs that match the training data in probability and structure, such as predicting 'h' with higher probability after seeing 't'.

4. RESULTS AND DISCUSSION

In this study, contextual information was applied to train language models for text generation, and a recurrent neural network based on LSTM layers with long–term memory was chosen. The focus of the work is to discover a way to extract the context from the sentences and from the training data and the important and useful words in the meaning and train the model through the context vectors extracted and the vectors formed from the input data aimed at finding a better linguistic model. The clustering method was used by calculating the distances between words in context vectors and proximity in meaning. This method provided good results and obtained context vectors with important meaning. The cosine scale evaluation method was used to calculate the semantic closeness between the generated sentences and the context provided. The proposed method also provided feedback to the training process until stopping when the built model is appropriate in order to stop training and testing. To evaluate the accuracy and efficiency of the proposed methodology, the model will be trained and tested on two data sets (Shakespeare dataset and Nietzsche's dataset) related to the topic of text generation, and the results will be compared with the results reached in previous studies that worked on the same two data sets. Figure 4.1 shows the type of layers used in the model and the output shape and the number of parameters used to train this model. The network was tracked for 100 epochs, and in each epoch the loss and accuracy were calculated.

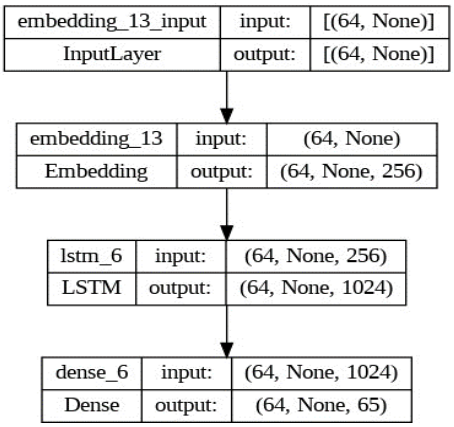

Figure 4.1 Model parameters with layer types and output shape

4.1 Nietzsche's Data for Text Generation

The dataset Nietzsche provides researchers and readers complete online access to the editions, interpretations and reference works on one of the most important philosophers. Users thus obtain access to a comprehensive database containing the research results of the last forty years.

This data set is relied upon in many researches dealing with the topic of text generation (Gao 2016). The accuracy curve may vary depending on factors like the model architecture, dataset size, and the learning rate, So, the experimental approach was used to reach the proposed model, where many training operations were conducted, monitoring the accuracy and loss curves, and modifying the model structure and parameters experimentally until the proposed model that achieved optimal accuracy was reached. The following Figure 4.2 shows how the model's accuracy improves as it learns from the training data over multiple epochs. It starts with a relatively low accuracy and gradually increases as the model refines its parameters.

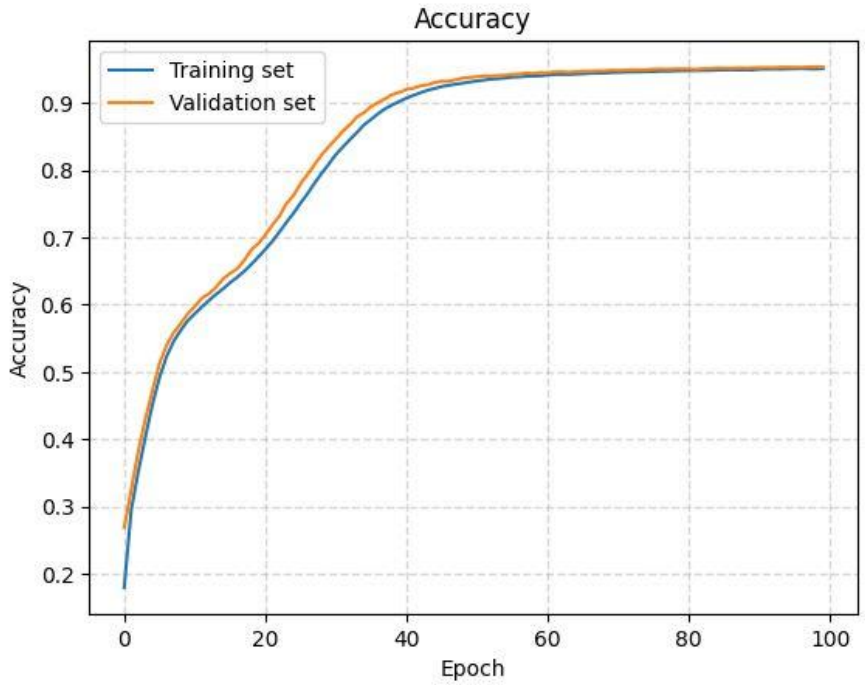

Figure 4.2 The model's accuracy curve based on Nietzsche's data

The loss curve showed in Figure 4.3 illustrates how the model's loss decreases over epochs. Initially, the model makes significant errors in its predictions, and these errors gradually reduce as training continues. The goal is to minimize the loss as much as possible.

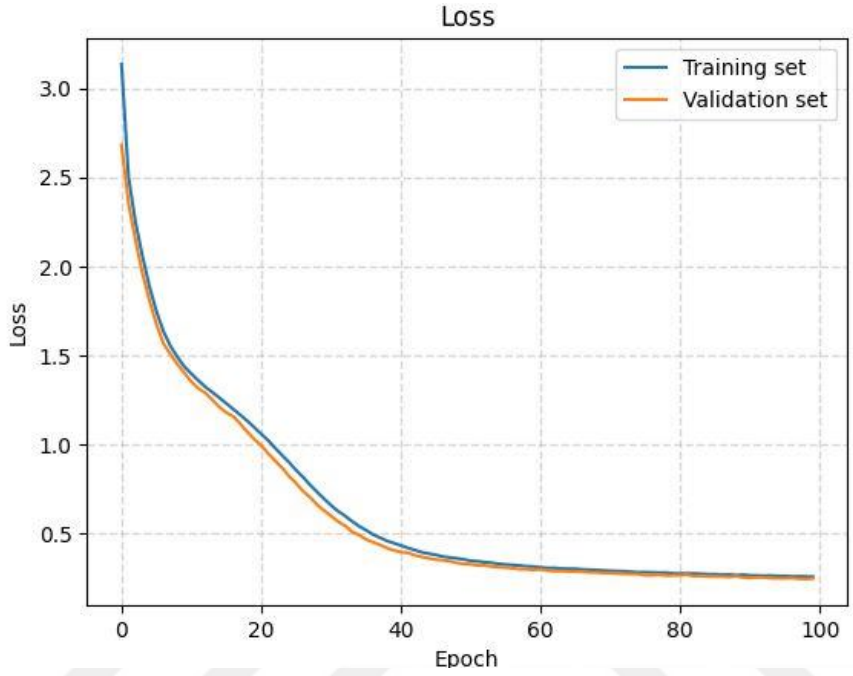

Figure 4.3 The model's loss curve based on Nietzsche's data

Balancing the loss and accuracy curves is important for ensuring that the model is learning effectively and generalizing well to unseen data.

Table 4.1 shows a comparison between the accuracy of the our model with the accuracy achieved in the study (Gao 2016). During our search for scientific articles, we only found this study that dealt with the topic of text generation using Nietzsche data. In order for the comparison to be fair, it must be made with articles that dealt with the same topic using the same data set.

Table 4.1 A comparison between the accuracy of the our model with the accuracy achieved in the study according to Nietzsche's data

MODELS	ACCURACY	LOSS	NUMBER ITERATIONS
Gao (2016)	0.758	–	160
Our model	0.9521	0.2518	100

We note that our model significantly outperforms (Gao 2016) in terms of accuracy, achieving 95.21% accuracy compared to 75.8%. Additionally, "Our model" reached this level of accuracy with a lower loss of 0.2518, which indicates better model performance in terms of minimizing the error, compared to (Gao 2016), for which the loss is not provided. Furthermore, our model required fewer iterations (100) to achieve these results compared to the 160 iterations of (Gao 2016). This suggests that "Our model" may converge faster and is more efficient in terms of training time. These results suggest that "Our model" is a more effective and efficient model compared to the reference model (Gao 2016) for the given task Shakespeare data for text generation.

4.2 Shakespeare Data for Text Generation

The dataset you're describing, which includes 40,000 lines from a variety of Shakespeare's plays, is an extensive and rich source for training a machine learning model. When such a model is trained on this dataset, it initially exhibits relatively low accuracy. This is expected, as the model is just beginning to learn the complex language, styles, and nuances found in Shakespeare's works. As the training progresses over multiple epochs, the model gradually improves its accuracy. This improvement is a result of the model refining its parameters and learning to better understand and predict the patterns in Shakespearean language. The increase in accuracy over time is a common phenomenon in machine learning, especially with large and complex datasets like the complete works of Shakespeare.

This progression, from low to high accuracy, indicates that the model is effectively learning from the training data. It's adapting to the intricacies of the language, the rhythm of the iambic pentameter, the use of metaphors, and the rich vocabulary that characterizes

Shakespeare's plays. As a result, the model becomes more proficient at tasks such as text generation, sentiment analysis, or even thematic classification of the plays, depending on the specific objectives of the machine learning project.The following Figure 4.4 shows how the model's accuracy improves as it learns from the training data over multiple epochs. It starts with a relatively low accuracy and gradually increases as the model refines its parameters.

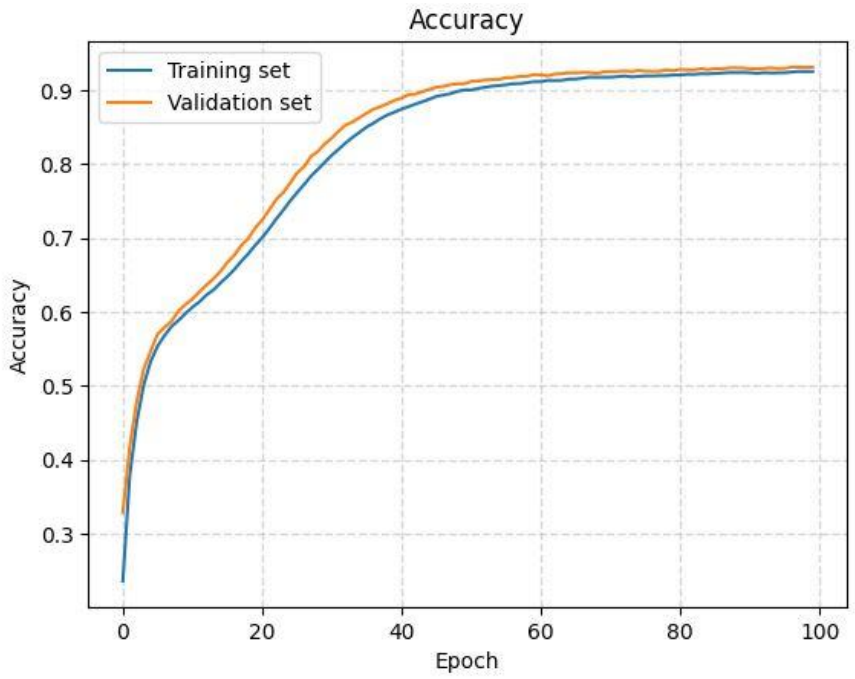

Figure 4.4 The model's accuracy curve based on Shakespeare data

The loss curve depicted in Figure 4.5 is a critical aspect of understanding the model's learning process. This curve visualizes the decrease in the model's loss over successive training epochs. At the outset, when the model is first exposed to the training data, its predictions are often significantly off–target, resulting in a high loss value. These errors are manifestations of the model's initial unfamiliarity with the complex patterns and nuances in the dataset. As training progresses, the model continuously adjusts its internal parameters in an attempt to better align its predictions with the actual outcomes. This adjustment process is guided by the loss function, a mathematical formulation that

quantifies the difference between the model's predictions and the true data. By minimizing this loss, the model is effectively learning to make more accurate predictions.

The declining trend of the loss curve is a positive indicator of learning and improvement. Each epoch represents a complete pass through the dataset, and with each pass, the model refines its understanding, leading to a reduction in prediction errors. The goal in training machine learning models like this is to minimize the loss as much as possible, which corresponds to maximizing the accuracy of the model's predictions. However, it's important to monitor the loss curve for signs of overfitting, where the model performs well on the training data but poorly on unseen data. This is typically indicated by a continued decrease in training loss but no corresponding improvement or even a worsening in validation loss. Therefore, while a decreasing loss curve is desirable, it is essential to balance it with the model's ability to generalize to new, unseen data. This balance ensures that the model is robust and performs well in practical applications, not just in the controlled environment of training.

The loss curve showed in Figure 4.5 illustrates how the model's loss decreases over epochs. Initially, the model makes significant errors in its predictions, and these errors gradually reduce as training continues. The goal is to minimize the loss as much as possible.

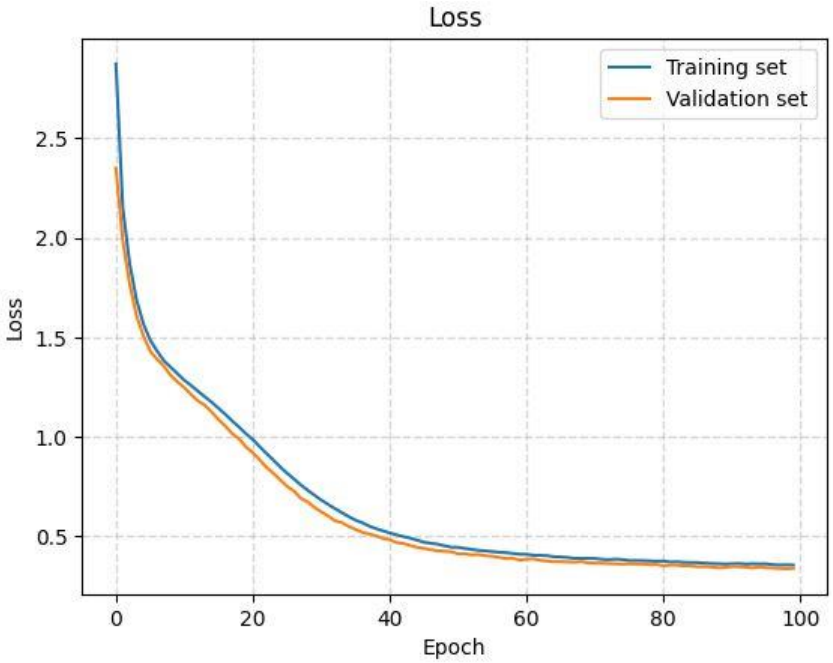

Figure 4.5 The model's loss curve based on Shakespeare data

The results obtained were compared with the results of other research on Shakespeare's data. According to (Hussain *et al.* 2021) words were generated from a given set of input words. The LSTM model was trained using Shakespeare's writings. The results obtained from using this model show that the final accuracy is 74.6%. The batch size for training was 500 epochs. 300 words were generated using the LSTM search model. Table 4.2 compares our results with those of other research, highlighting the preference for the model we proposed over other models.

Table 4.2 Comparing results that represents the accuracy, Loss and iterations number on text for Shakespeare data

Models	**Accuracy**	**Loss**	**Number Iterations**
(Hussain 2021)	0.746	1.020	500
Our model	0.9125	0.3876	100

Comparing the models, it's clear that our model has the highest accuracy (91.25%), indicating that it's performing better in terms of making correct predictions. Additionally, it has the lowest loss (0.3876), which suggests that it's better at minimizing errors during training. On the other hand, (Hussain 2021) has the lowest accuracy (74.6%) and the

highest loss (1.020), indicating it may not be as effective as the other models. The number of iterations can also provide insights into the efficiency of model training. Our model achieved its high accuracy and low loss in just 100 iterations, whereas (Hussain 2021) needed 500 iterations for its results. This suggests that our model may be more efficient in terms of training time. So, our model outperforms the other two models in terms of accuracy, loss, and training efficiency.

In this study, contextual information was applied to train language models for text generation, and a recurrent neural network based on LSTM layers with long–term memory was chosen. The focus of the work is to discover a way to extract the context from the sentences and from the training data and the important and useful words in the meaning and train the model through the context vectors extracted and the vectors formed from the input data aimed at finding a better linguistic model . The clustering method was used by calculating the distances between words in context vectors and proximity in meaning. This method provided good results and obtained context vectors with important meaning. The proposed method also provided feedback to the training process until stopping when the built model is appropriate in order to stop training and testing.

5. CONCLUSIONS AND RECOMMENDATION

This comprehensive study marks a significant advancement in the field of natural language processing (NLP) by developing a refined language model that leverages Recurrent Neural Networks (RNN) with Long Short Term Memory (LSTM) layers. Focused on contextual understanding, this innovative approach has demonstrated remarkable effectiveness in handling complex linguistic structures across various literary styles and epochs. The study's methodology, which integrates advanced contextual analysis through clustering techniques and context vectors, has proven to be highly effective. The impressive accuracy and low loss rates achieved with datasets from Nietzsche and Shakespeare underscore the model's efficacy and versatility. Moreover, the introduction of an innovative feedback mechanism in the training process ensures continuous improvement and adaptation, leading to a robust model attuned to the nuances of human language. This research not only showcases the potential of context–based training in language models but also opens new avenues for more sophisticated and accurate text generation technologies in various NLP tasks. The analysis conducted in this study has demonstrated the superior performance of our model in generating text from Shakespearean data, with a remarkable accuracy of 91.25% and a minimal loss value of 0.3876. Our model has shown not only an ability to make correct predictions with greater reliability but also to minimize errors effectively during training. Its efficiency is further underscored by the fact that such high levels of accuracy were achieved in just 100 iterations, indicating a significant reduction in training time compared to other models. The contextual training approach adopted by our model, leveraging LSTM layers to capture long–term dependencies, has evidently paid dividends. It enabled the extraction of meaningful context vectors and facilitated the clustering of words based on semantic proximity. This methodology contrasts favorably with the results obtained by (Hussain *et al.* 2021), which reported lower accuracies and required more iterations for training, reflecting a less efficient process. Moreover, when compared with (Gao 2016), our model not only surpassed in terms of accuracy by a substantial margin but also demonstrated a lower loss, suggesting a more robust performance in error minimization. The reduced number of iterations required to attain optimal performance further cements our study as a highly efficient and effective tool for text generation tasks. The findings of this study

provide compelling evidence that the innovative training techniques and contextual understanding employed in our model are pivotal in achieving high–quality text generation. These results highlight the potential of our model for application in areas requiring advanced language models and set a benchmark for future research in the field. The promising performance of this study may thus inspire continued advancements and explorations into the optimization of LSTM networks and context–aware learning for natural language processing tasks.

For future works in LSTM–based text generation, several areas hold significant promise. Enhancing contextual understanding is paramount, focusing on a more nuanced grasp of language intricacies, including idiomatic expressions. The development of multilingual and cross–lingual models could revolutionize text generation, enabling seamless translation and maintenance of stylistic nuances across languages. Integrating LSTM with other neural network architectures, like Transformers, could harness the unique strengths of each, potentially improving long–distance dependency recognition and efficiency. Optimizing training processes is also crucial, aiming to reduce computational demands without compromising output quality. Tailoring models for specific applications, such as creative writing, automated journalism, or conversational interfaces, will make LSTM text generation more relevant and effective in various fields. Concurrently, addressing ethical concerns and potential biases in generated text is essential to ensure fairness and prevent harmful stereotypes. The development of interactive models that generate text in response to human input, adapting narratives based on real–time feedback, represents an exciting frontier. Personalizing text generation according to individual preferences or demographic specifics can vastly improve user engagement and relevance. Enhancing scalability and efficiency of LSTMs will allow their deployment across a wider range of platforms, including mobile and embedded systems. Conducting longitudinal studies to understand the long–term impacts of LSTM–generated text in various domains is vital to gauge its societal effects. Additionally, incorporating emotional intelligence into LSTMs could lead to more empathetic and context–aware conversational agents. Finally, exploring novel applications, such as educational content generation, interactive storytelling, or scriptwriting for virtual reality, could open new avenues for the use of LSTM text generation, further expanding its utility and impact in our digital world.

REFERENCES

Abbas, F. 2022. An improved NLP for syntactic and semantic matching using bidirectional LSTM and attention mechanism. Artificial Intelligence and Applications, 69–75.

Abdurakipov, S. and Butakov, E. 2018. Combustion anomalies detection for a thermal furnace based on recurrent neural networks. In Journal of Physics: Conference Series, 1105(1): 1–5.

Ahmad, F. and Faisal, D. M. 2022. A novel hybrid methodology for computing semantic similarity between sentences through various word senses, International Journal of Cognitive Computing in Engineering 3: 58–77.

Baktha, K. and Tripathy, B. K. 2017. Investigation of recurrent neural networks in the field of sentiment analysis, In: 2017 International Conference on Communication and Signal Processing (ICCSP), pp. 2047–2050, India.

Blevins, T., Levy, O. and Zettlemoyer, L. 2018. Deep RNNs encode soft hierarchical syntax, In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (2): 14–19.

Bo Chang, M. C. 2019. Antisymmetric RNN: A dynamical system view on recurrent, In: 2019 ICLR Conference, pp. 1–15.

Colon, R. S., Patra, P. K. and Elleithy, K. M. 2014. Random word retrieval for automatic story generation. In: Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education, pp. 1–6, USA.

Dautel, A. J., Härdle, W. K., Lessmann, S. and Seow, H. V. 2020. Forex exchange rate forecasting using deep recurrent neural networks. Digital Finance, 2: 69–96.

DiPietro, R. and Hager, G. D. 2020. Deep learning: RNNs and LSTM. Handbook of Medical Image Computing and Computer Assisted Intervention, 503–519.

Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M. and Soomro, A. 2022. A Systematic Literature Review on Text Generation Using Deep Neural Network Models. IEEE Access, 10: 53490–53503.

CHOLLET, F., 2015. Website: https://github.com/fchollet/keras. Date of access: 05.02.2015.

Gao, Y. and Glowacka, D. 2016. Deep gate recurrent neural network, In: Asian Conference On Machine Learning, pp. 350–365. PMLR.

Gv, U., 2011. An intelligent automatic story generation system by revising Proppian's system. In Advances in Computer Science and Information Technology: First International Conference on Computer Science and Information Technology, CCSIT 2011, Bangalore, India, January 2–4, 2011. Proceedings, Part I 1 (pp. 594–603). Springer Berlin Heidelberg.

Hatipoglu, A. and Omurca, S. I., 2016. A Turkish wikipedia text summarization system for mobile devices, IJ Information Technology and Computer Science, 1: 1–10.

Hussain, H. 2021 Shakespeare like text generation using lstm, International Journal of Emerging Technologies and Innovative Research, 8(5): 991–993.

Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A. and Sankaranarayanan, K., 2017. Story generation from sequence of independent short descriptions. SIGKDD Workshop on Machine Learning for Creativity, 1–7.

Jaya, A. and Uma, G.V. 2010. An intelligent system for semi– automatic story generation for kids using ontology. In: Proceedings of the Third Annual ACM Bangalore Conference, 8: 1–6.

Jeffrey, P., Richard, S. and Christopher, M. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 1532– 1543.

Kang, C., Zhang, Z. 2020. Application of lstm in short–term traffic flow prediction, In: 5th International Conference on Intelligent Transportation Engineering (ICITE), pp. 98–101, China.

Kavila, S.D. and Radhika, Y. 2015. Extractive text summarization using modified weighing and sentence symmetric feature methods. International Journal of Modern Education and Computer Science, 7(10): 33.

Khatri, C., Voleti, S., Veeraraghavan, S., Parikh, N., Islam, A., Mahmood, S., Garg, N. and Singh, V. 2015. Algorithmic content generation for products. In 2015 IEEE International Conference on Big Data (Big Data), pp. 2945–2947, USA.

Klahold, A., & Fathi, M. (2019). Automatic text generation. *Springer EBooks*, 131–154.

KRIS, 2019. Website: https://www.kaggle.com/datasets/pankrzysiu/nietzsche–texts. Date of access: 28.10.2019.

Le, H.T. and Le, T. M. 2013. An approach to abstractive text summarization. In 2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR), pp. 371–376, Vietnam.

LIAMLARSEN, 2018. Website: https://www.kaggle.com/datasets/kingburrito666/shakespeare–plays/data. Date of access: 01.02.2018.

Li, B., Lee–Urban, S., Johnston, G. and Riedl, M. 2013, June. Story generation with crowdsourced plot graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, 27(1): 598–604.

Lindström, A., Bensch, S., Björklund, J., Drewes F. 2021. Probing multimodal embeddings for linguistic properties: The visual–semantic case, COLING 2020 Conference, 1–15.

MAGNUS, B. 2023. Website: https://www.britannica.com/biography/Friedrich Nietzsche. Date of access: 01.11.2023.

Mansy, A., Rady, S., Gharib, T. 2022. An ensemble deep learning approach for emotion detection in arabic tweets. International Journal of Advanced Computer Science and Applications 13(4).

Marhon, S. A., Cameron, C. J. F. and Kremer, S. C. 2013. Recurrent neural networks. Intelligent Systems Reference Library, 29–65.

Mehta, A., Gala, R. and Kurup, L., 2016, March. A roadmap to auto story generation, In: 2016 3rd International Conference on Computing for Sustainable Global Development, pp. 2306–2310, India.

Mer Levy, K. L. 2018. Long short-term memory as a dynamically computed element-wise weighted sum. Artificial Intelligence, 1–8.

Milanova, I., Sarvanoska, K., Srbinoski, V. and Gjoreski, H. 2019. Automatic text generation in Macedonian using recurrent neural networks. Communications in Computer and Information Science, 1–12.

Nallapati, R., Zhou, B., Gulcehre, C. and Xiang, B. 2016. Abstractive text summarization using sequence–to–sequence rnns and beyond, In: SIGNLL Conference on Computational Natural Language Learning, 1–11.

Pawade, D., Sakhapara, A., Jain, M., Jain, N. and Gada, K. 2018. Story scrambler–automatic text generation using word level RNN–LSTM. International Journal of Information Technology and Computer Science, 10(6): 44– 53.

Pennington J., Socher R., Manning C.D. 2014. Glove: Global vectors for word representation, In: conference on empirical methods in natural language processing (EMNLP), 1– 12.

POUDEL, S. 2023. Website: https://medium.com/@poudelsushmita878/recurrent–neural–network–rnn–architecture–explained–1d69560541ef. Date of access: 28.08.2023.

Puduppully, R., Dong, L. and Lapata, M. 2019. Data–to–text generation with content selection and planning. In Proceedings of the AAAI conference on artificial intelligence 33(01): 6908–6915.

Rahman, M. M. and Siddiqui, F. H. 2019. An optimized abstractive text summarization model using peephole convolutional LSTM. Symmetry, 11(10): 1290.

Rijkhoff, J. 2002. Qualifying modifiers in the noun phrase. Oxford University Press.

100-145, England.

Sakhare, D.Y. and Kumar, R. 2014. Syntactic and sentence feature based hybrid approach for text summarization. Internation Information Technology and Computer Science, 2014(3): 38–46.

Souri, A., El Maazouzi, Z., Al Achhab, M., & Badr. 2018. Arabic Text Generation Using Recurrent Neural Networks. Communications in Computer and Information Science, 523–533.

Thomaidou, S., Lourentzou, I., Katsivelis–Perakis, P. and Vazirgiannis, M. 2013. Automated snippet generation for online advertising. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, 1841–1844.

Tomas, M., Ilya, S., Kai, C., Greg, S. C. and Jeff D. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 3111–3119.

Yu, Y., Si, X., Hu, C. and Zhang, J. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7): 1235–1270.

Zhang, R., Wang Z., Yin K. and Huang Z. 2019 Emotional text generation based on cross–domain sentiment transfer, IEEE Access, 7: 100081–100089.

CURRICULUM VITAE

Personal Information

Name and Surname : Mustafa Abbas Hussein HUSSEIN

Education

MSc	Çankırı Karatekin University Graduate School of Natural and Applied Sciences Department of Electronics and Computer Engineering	2021-2024
Undergraduate	Kirkuk University Faculty of Engineering Department of Electronics and Computer Engineering	2015–2019