2020

Transformer Neural Networks for Automated Story Generation

Kemal Araz
Technological University Dublin

Transformer Neural Networks for Automated Story Generation

Kemal ARAZ

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Analytics)

2020

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Date: **05 January 2020**

ABSTRACT

Towards the last two-decade Artificial Intelligence (AI) proved its use on tasks such as image recognition, natural language processing, automated driving. As discussed in the Moore's law the computational power increased rapidly over the few decades (Moore, 1965) and made it possible to use the techniques which were computationally expensive. These techniques include Deep Learning (DL) changed the field of AI and outperformed other models in a lot of fields some of which mentioned above. However, in natural language generation especially for creative tasks that needs the artificial intelligent models to have not only a precise understanding of the given input, but an ability to be creative, fluent and, coherent within a content. One of these tasks is automated story generation which has been an open research area from the early days of artificial intelligence. This study investigates whether the transformer network can outperform state-of-the-art model for automated story generation. A large dataset gathered from Reddit's WRITING PROMPTS sub forum and processed by the transformer network in order to compare the perplexity and two human evaluation metrics on transformer network and the state-of-the-art model. It was found that the transformer network cannot outperform the state-of-art model and even though it generated viable and novel stories it didn't pay much attention to the prompts of the generated stories. Also, the results implied that there should be a better automated evaluation metric in order to assess the performance of story generation models.

Key words: *text generation, automated story generation, deep learning, transformer networks, convolutional sequence to sequence networks*

ACKNOWLEDGEMENTS

First, most of all I would like to express my sincere thanks to Dr. Pierpaolo Dondio for his assistance, support and patience throughout this MSc dissertation.

I would like to express my sincere gratitude to David NG and Dr. Robert Ross for allowing and helping me to use the servers of the university and Dr. Luca Longo for his advices on several aspects about the dissertation.

Also, I would like to thank all of the Technological University Dublin staff who guided and encouraged me throughout my time in the university.

Last but not least, I would like to thank all my relatives and Esra Aynali who encouraged and supported me during this MSc.

TABLE OF CONTENTS

ABSTRACT II

1. INTRODUCTION 1

1.1 BACKGROUND 1
1.2 RESEARCH PROBLEM 3
1.3 RESEARCH OBJECTIVES 3
1.4 RESEARCH METHODOLOGIES 4
1.5 SCOPE AND LIMITATIONS 6
1.6 DOCUMENT OUTLINE 7

2. LITERATURE REVIEW 8

2.1. NATURAL LANGUAGE PROCESSING 8
2.2. NATURAL LANGUAGE GENERATION 9
2.3. NLG APPROACHES AND ARCHITECTURES 10
2.3.1 MODULAR BASED APPROACHES 12
2.3.2 PLANNING BASED APPROACHES 18
2.3.3 MACHINE LEARNING APPROACHES 20
2.3.4 DEEP LEARNING APPROACHES 22
2.4 TRANSFORMER NETWORK 26
2.5 EVALUATION METHODS IN NLG 30
2.5.1 AUTOMATED (METRIC-BASED) EVALUATION 30
2.5.2 HUMAN EVALUATION 32
2.6. AUTOMATED STORY GENERATION 32
2.6.1 GRAMMAR BASED APPROACHES 34
2.6.2 PLANNING BASED APPROACHES 34
2.6.3 MODULAR BASED APPROACHES 35
2.6.4 CASE BASED REASONING APPROACHES 36
2.6.5 DEEP LEARNING APPROACHES 36
2.7 SUMMARY 38
2.7.1 OVERVIEW 38
2.7.2 GAPS IN THE RESEARCH 38

2.7.3 Research Question 39

3. DESIGN & METHODOLOGY 40

3.1 HYPOTHESIS 40
3.2 DATA 41
3.2.1 Data Collection and Understanding 41
3.2.2 Data Preparation 43
3.3 TRANSFORMER NETWORK DESIGN 44
3.3.1 Hyperparameters of Transformer Network 44
3.3.2 Hyperparameters of the Model 46
3.4 EVALUATION OF RESULTS 47
3.4.1 Perplexity 47
3.4.2 Human Evaluation Methods 48
3.4.3 Hypothesis Testing 49
3.5 SUMMARY 50
3.5.1 Strengths 50
3.5.2 Weaknesses 51

4. RESULTS, EVALUATION & DISCUSSION 52

4.1 EXPERIMENT RESULTS 52
4.1.1 Transformer Network 52
4.1.2 Hierarchical Neural Story Generation Model 54
4.1.3 Reliability of Human Annotators 55
4.2 EVALUATION 56
4.3 DISCUSSION 57
4.3.1 Strengths 57
4.3.2 Weaknesses 58

5. CONCLUSION 60

5.1 RESEARCH OVERVIEW 60
5.2 PROBLEM DEFINITION 60
5.3 DESIGN/EXPERIMENTATION, EVALUATION, RESULTS 60
5.4 CONTRIBUTIONS & IMPACT 61
5.5 FUTURE WORK & RECOMMENDATIONS 61

BIBLIOGRAPHY 63

APPENDIX A **76**

TABLE OF FIGURES

FIGURE 2.3-1 NLG SYSTEM ARCHITECTURE 11
FIGURE 2.3.1-1 MODULAR ARCHITECTURE PIPELINE THAT INCORPORATES SUB PROBLEMS FROM SECTION 2.3 12
FIGURE 2.3.3-1 BASIC STRUCTURE OF PERCEPTRON (LEFT) AND NEURAL NETWORK (RIGHT) 23
FIGURE 2.3.3-2 A HIGH-LEVEL ILLUSTRATION OF ENCODER DECODER ARCHITECTURE.. 25
FIGURE 2.4-1 THE TRANSFORMER MODEL ARCHITECTURE.. 27
FIGURE 2.4-2 ENCODER OF THE TRANSFORMER NETWORK WITH DETAILS 28
FIGURE 2.4-3 DECODER OF THE TRANSFORMER NETWORK WITH DETAILS 29
FIGURE 2.6- 1 FUSION MODEL, TAKEN FROM (FAN ET AL., 2018)..... 37
FIGURE 3.2.1-1 HISTOGRAM FOR SENTENCE LENGTH OF THE PROMPTS..... 42
FIGURE 3.2.1-2 HISTOGRAM FOR SENTENCE LENGTH OF THE STORIES 42
FIGURE 3.2.1- 3 A SAMPLE PROMPT FROM THE DATASET 42
FIGURE 3.2.1- 4 CORRESPONDING STORY OF THE SAMPLE PROMPT..... 42
FIGURE 3.2.2-1 TOKENIZED INPUT AND OUTPUT DATASETS OF MODEL, PROMPTS (LEFT) AND STORIES (RIGHT)..... 43
FIGURE 3.3.1-1 SCALED DOT-PRODUCT ATTENTION (LEFT) AND MULTI-HEAD ATTENTION, TAKEN FROM VASWANI ET AL., (2017)..... 44
FIGURE 3.3.2- 1 CHANGE IN THE LEARNING RATE..... 46
FIGURE 4.1.1-1 A SAMPLE PROMPT (TOP) AND ITS STORY GENERATED BY TRANSFORMER NETWORK (BOTTOM)..... 53
FIGURE 4.1.2-1 A SAMPLE PROMPT (TOP) AND ITS STORY GENERATED BY FUSION MODEL (BOTTOM)..... 55
FIGURE 4.1.3-1 SUMMARY OF PROMPT PAIRING RESULTS (LEFT) AND BLIND COMPARISON (RIGHT) 55
FIGURE 4.2- 1 PERPLEXITY SCORE OF THE MODELS..... 56
FIGURE 4.2- 2 ACCURACY OF PROMPT PAIRING TASK AND KAPPA SCORES 56
FIGURE 4.2- 3 BLIND COMPARISON TASK AND KAPPA SCORE 57

TABLE OF TABLES

TABLE 3.3.2-1 HYPERPARAMETERS OF THE MODEL 47

TABLE 4.1.1-1 SUMMARY OF PAIRING RESULTS FOR TRANSFORMER NETWORK 52

TABLE 4.1.1- 2 SUMMARY OF BLIND TEST RESULTS THAT WERE IN FAVOUR OF TRANSFORMER NETWORK..... 53

1. INTRODUCTION

Towards the last two-decade Artificial Intelligence (AI) proved its use on tasks such as image recognition, natural language processing, automated driving. As discussed in the Moore's law the computational power increased rapidly over the few decades (Moore, 1965) and made it possible to use the techniques which were computationally expensive. These techniques include Deep Learning (DL) changed the field of AI and outperformed other models in a lot of fields some of which mentioned above. However, in natural language generation especially for creative tasks that needs the artificial intelligent models to have not only a precise understanding of the given input, but an ability to be creative, fluent and, coherent within a content. One of these tasks is automated story generation which has been an open research area from the early days of artificial intelligence. According to Martin et al. (2018) "automated story generation is the problem of automatically selecting a sequence of events, actions, or words that can be told as a story" (p.2250). Because of its nature it is not like many other text generation tasks, since the generated text should have events, actions and characters which should be consistent within the generated story and in addition to that it should fulfil the primary concerns of automated story generation which are to create a meaningful, fluent and consistent story. The latest developments in Natural Language Processing (NLP), especially transformer networks (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez & Polosukhin, 2017), made it possible to capture important information and allow parallelization when generating the output. For automated story generation topic that means, capturing events, actions or important words for the story that is going to be generated.

1.1 BACKGROUND

A story conveys a sequence of events over time (Labov and Waletzky, 1997) and those events contains characters, locations, actions and sometimes subevents that occur within an event. The result of the events can lead to another event or conclude a story. Even for humans, story generation is a hard task which requires a set of skills and expertise. Automated story generation is a sub field of Natural Language Processing (NLP) which tackles the problem of giving a machine to generate stories like a human.

It is considered one of the key areas for creating a general artificial intelligence (Shapiro, 1992). By hearing stories every day, we figure out how the world works. We order its events to search for meaning in them. This ability to craft, tell and understand stories has long been a hallmark of human intelligence and a long-term goal of artificial intelligence (Mehta, Gala & Kurup, 2016).

NLP tasks such as machine translation, summarization or segmentation does not need to track events or actions that occurs within a story neither characters or locations nor the causal relationship that appears between events. However, for automated story generation it is crucial for a model to keep track of the concepts stated above and while doing so, be coherent and creative. Researchers have taken many different approaches to solve this problem.

Before the latest developments in the field of Deep Learning most of the research conducted relied on human-based rules and planning which restricts the story generation to domains that are specified by the designer (Meehan, 1977;Lebowitz, 1984). Some achieved good results on restricted domains and with human interaction involved in every step. Recent progresses in Deep Learning made it possible to generate stories in an open domain (Clark, Ji & Smith, 2018). Fan, Lewis and Dauphin's attempt was reached outstanding results however "narratives still lacked coherence and human like outcomes and because of the convolutional sequence to sequence and fusion architectures focused on some aspects more than necessary concluded in repetition of words" (Fan et al., 2018). Although latest researches created fluent sentences, capturing the causal relationship between events, creating a coherent story which is consistent from beginning till the end still needs improvement. Essentially, it still remains an open question for the field of NLP. The aim of this paper is to enhance the latest successes on automated story generation field by applying a deep learning approach, the Transformer, which proved its success on several other NLP tasks. The success behind transformer networks is coming from their attention mechanism. It has been proved that when predicting the next word in a sequence transformer networks are capable of attending right words and capture the long-term dependencies which can solve the problem of seizing the causal relationships and by attending the right parts of the story at right moment it can provide a coherent narrative.

1.2 RESEARCH PROBLEM

The model which can be considered as state-of-the-art, Fan, Lewis and Dauphin (2018) presented a novel approach with convolutional sequence to sequence (Gehring, Auli, Grangier, Yarats & Dauphin 2017) architecture and decomposed the decoder with attention mechanism (Bahdanau, Cho & Bengio, 2015) and created a fusion model with two of these architectures combined. The work also compared various state-of-the-art seq2seq models to compare their model on a 300K dataset gathered from an online forum and improved the perplexity score and human evaluation scores on automated story generation dramatically.

In this paper, those two evaluation metrics used by the Fan, Lewis and Dauphin (2018) is adopted for comparison. The first one is an automated evaluation technique called perplexity. Formally it is defined as "the geometric mean of the inverse of the per-word likelihood" (Kobayashi, 2014, p. 797). This score shows how fluent the generated text is given the preceding words and the basic formula is two to the power of cross-entropy. The lower the perplexity score is the better the model. The second evaluation method is human evaluation which has two parts. The first part tests whether the story generated is relevant to the prompt that is given to the model or not. Second human evaluation is about comparing the quality of the stories generated by the models.

"Can a transformer network used on Reddit's WRITINGPROMTS forum dataset achieve statistically significant improvement on the perplexity score and human evaluation metrics prompt matching and blind model comparison test, presented in the paper of Fan et al., (2018), for automated story generation when a prompt is given, over the state-of-the-art Hierarchical Neural Story Generation model (Fan et al., 2018)?"

1.3 RESEARCH OBJECTIVES

One objective is to do a comprehensive literature review on NLG techniques and automated story generation in order to understand the text generation techniques and their relationships with the story generation approaches proposed over time. Also, the evaluation methods proposed for text generation examined in order to comprehend the

methods that should be used for the comparison of the state-of-the-art model and the model that this paper proposes. Recent attempt by Fan, Lewis and Dauphin (2018) about the automated generation field, showed that attention mechanism (Bahdanau et al., 2015) combined with Seq2seq CNN architecture does a decent job tracking long-term dependencies for a consistent story . However, stories still diverge from original course on some aspects and repeat some words or phrases occasionally because of the local aspects of convolutional part of the architecture.

Another objective is to design an experiment which investigates instead of using the convolutional networks for tracking the long-term dependencies and for attending to the essential parts and having a decoder with attention mechanism to enhance its power, the transformer model solely without a fusion model architecture, which had been proven its use for tracking long-term dependencies, can or cannot yield to better perplexity and human evaluation results which leads to a more coherent and consistent story.

Third objective is to evaluate the design with the perplexity automated metric and the human evaluation metrics defined in the paper (Fan et al., 2018).

Last objective is to validate the design with Free-Marginal Fleiss's Kappa (Rudolph, 2005) for human evaluation results, report and discuss the results on two metrics perplexity and human evaluations.

1.4 RESEARCH METHODOLOGIES

For the purpose of having reliability and validity throughout the research and the results gained when conducting the experiment, the methodologies are clearly defined.

The overall methodology used for this thesis is Knowledge Discovery in Databases (KDD) which is in broad terms, making sense of the data with the development of related methods and techniques (Fayyad & Uthurusamy, 1996). The investigation started with the selection of the appropriate data for automated story generation task. Then a pre-processing and transformation step was in place in order to prepare the data for the prerequisites of the model and according to the time and resource constraints allocated for this thesis. Those steps followed by constructing the model and

performing the experiment. Lastly, in order to gain knowledge, results gained by the aid of several evaluation methods examined and discussed.

This research used existing data in order to compare the model built with another model that used the same data, thus it is a secondary (desk) research in addition to that, it is concerned about measuring and comparing the performance of the models built, with numerical evaluation approaches as a result quantitative methods are used. A systematic empirical investigation of the quantitative properties observed in order to build up intelligence about the field of research. This is accomplished by deductive reasoning which starts by constructing a theory, formulating a hypothesis based on the theory, gathering the data, in this case the data is collected and ready for the research, conducting an experiment then examining the results of the experiment for approving or rejecting the constructed theory.

The data that used in this investigation was Reddit's WRITINGPROMPTS[1] dataset. Reddit is a social sharing website where users share a content and based on its popularity those contents rise to the top, making others become less visible. "WRITINGPROMPTS is a community where online users inspire each other to write by submitting story premises, or prompts, and other users freely respond. Each prompt can have multiple story responses. The prompts have a large diversity of topic, length, and detail" (Fan et al., 2018, pp. 2251). Respected writers of the community share prompts and others write short stories about those prompts. Main constraints about writings are, the proposed story should have more than 30 words, plagiarism results in a ban so all stories are unique and off-topic writing is not allowed all stories checked regularly by the administrators thus, consistency among prompts and stories are ensured. The dataset is available online and gathered from here [2].

Once collected, stories trimmed to 1000 words and vocabulary size for prompts and stories restricted to 19,025 and 104,960 respectively. The average length for the prompts and stories 28.4 and 734.5 respectively and validation and test set were set to 5% (Fan et al., 2018). Tokenization of the dataset is done with Byte-Pair Encoding tokenization method proposed by Sennrich, Haddow, and Birch (2016). The maximum

[1] https://www.reddit.com/r/WritingPrompts/

[2] https://github.com/pytorch/fairseq/tree/master/examples/stories

length of tokenized prompts and tokenized stories are 71 and 1157 respectively therefore the input of the transformer model is a vector of the size 71 and the output vector has size of 1157. Once the model is trained the perplexity scores of a various test sets calculated, and its distribution was compared to the state-of-the-art model's perplexity scores, therewithal the human evaluation. To ease human evaluation the generated stories are limited to 150 words and scientific measures were in place for human evaluation methods in order to have accurate comparison of those quantitative methods.

1.5 SCOPE AND LIMITATIONS

The scope of this thesis is the automated story generation (Li, Lee-Urban, Jonston & Riedl, 2013) which is a subfield of Natural Language Generation using deep neural networks.

Based on a given topic the model generates stories; the inputs are prompts and outputs are corresponding stories. Since the purpose of this paper is to compare the model designed and the state-of-the-art model, the dataset used for both models should match in order to have significant comparison. At training time, due to the lack of time and resources 50000 instances were used and the length of the prompts were 71 and, generated stories were limited to 500 words. The reason behind that is the complexity per layer for a transformer network is roughly $O(n^2.d)$ for self-attention layer (Vaswani et al., 2017) where n is sequence length and d is the dimension of the layer. With 50000 stories the dimensions of the input and output are 71 columns and 50000 rows and 500 columns 50000 rows respectively, it took two weeks to train the network, thus there was no time for tuning the hyperparameters of the model because of the time allocated for the dissertation. Also because of the length of the stories the node on the server that this model was trained on gave memory error for more than 50000 prompts and corresponding stories with 500 length. Even for 50000 data instances the model was using 400gb ram on a single GPU.

1.6 DOCUMENT OUTLINE

The rest of this thesis is outlined below.

Chapter 2 – Review of Existing Literature

This chapter is dedicated to the literary review of previously examined venues of research that investigated text generation and the subfield of it, automated story generation. The initial methods used in text generation and story generation, early machine learning attempts and recent deep learning methods discussed and compared. Recently proposed NLG models analysed and deeply examined, using that information a state-of-the-art model which solved some problems in this area such as translation, language understanding, introduced in an expectation for outperforming the state-of-the-art model (Fan et al., 2018) for automated story generation.

Chapter 3 –Design and Methodology

This chapter provides insight to the experiment that was conducted, in order to test the hypothesis and eliminate the gaps that have been defined in Chapter 2. It underpins an inclusive clarification to the design process of the experiment and methods to evaluate the performance of the proposed model and compare it to the state-of-the-art model specified in Chapter 2.

Chapter 4 – Results, Evaluation and Discussion

The results of the experiment are presented here, and the performance of the model is evaluated also compared to the state-of-the-art model described in Chapter 2. Design flaws that led to inaccurate results and possible improvements that may guide to build a better automated story generation model is discussed.

Chapter 5 – Conclusion

In this chapter, the results, observations and insights gathered throughout this investigation is summarized, further research that can be carried out as a potential extension to this paper is presented.

2. LITERATURE REVIEW

2.1. NATURAL LANGUAGE PROCESSING

“Computational linguistics, also known as natural language processing (NLP), is the subfield of computer science concerned with using computational techniques to learn, understand, and produce human language content.” (Hirschberg & Manning, 2015). Essentially, NLP is an area that starts with modifying the spoken or written language into structured data that computers can understand then creating a system that responds back according to the task given to them by humans. Some NLP tasks are machine translation, text summarization, text segmentation, question answering and some subjects that require creativity such as poem or story generation.

One of the first task that NLP paid attention on was machine translation. Hutchins (2000), stated that Warren Weaver was the pioneer in machine translation which started in 1946 for translating enemy language. However, those were systems similar to dictionary look up, so they lacked capturing the semantics or the arrangement between words. Therefore, in early years of NLP scientists tried to create a set of rules describing the human language and vocabulary that machines can understand and interpret. Chomsky (1965), created a syntax that is considered one of the first grammars or a set of rules for the field. Grammars such as the one Chomsky created were indispensable for language understanding processes such as semantic interpretation and made it possible to create applications such as ELIZA (Weizenbaum, 1966) which is an early dialogue system that replicates the dialogue between a psychologist and a patient.

However, until 1990s NLP systems were based heavily on hand crafted rules, the data was limited which restricted the vocabulary used to build those systems and more importantly computational power was limited. Efforts done by linguistic data consortium (LDC) made it possible to have digital copies of written documents and with the arrival of internet made many statistical models feasible for NLP. By the time of 2000s the increase on the computational power shifted this area towards the Deep Learning as some researchers proved its use on NLP (Bengio, Ducharme, Vincent, & Janvin, 2003 ; Schwenk & Gauvain, 2005).

NLP has two major subtasks which are Natural Language Generation and Natural Language Understanding. There must be a clear distinction between NLU and NLG in order to avoid confusions that might be occur later. “Natural language generation is the process of mapping internal computer representations of information into human language, whereas natural language understanding is the process of mapping human language into internal computer representations” (Reiter & Dale, 1997). So, they can be considered as opposite sides of NLP, NLU is concerned about comprehension thus, makes analysis of text however NLG is about construction and planning not analysis (McDonald, 2010). In this manner, the generator can be viewed as “the equivalent of a person with something to say. Its work begins with the initial intention to communicate, and then on to determining the content of what will be said, selecting the wording and rhetorical organization and fitting it to a grammar, through to formatting the words of a written text or establishing the prosody of speech” (McDonald, 2010).

2.2. NATURAL LANGUAGE GENERATION

NLG is considered as one of the subtopics of Natural Language Processing. A broad definition of NLG made by Reiter & Dale (1997, p. 57) “the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information”. Despite the efforts such as Reiter & Dale (1997) to define NLG precisely, still there are contradictions among researchers. The reason behind that is the input of the text generation. Depending on the task to be performed and type of the input it has different applications such as text-to-text, image-to-text etc. It is obvious that the output has to be text, but the input types are diverse (McDonald, 1993). Although input types can be diverse most NLG tasks have text as input. Some of the NLG tasks outlined below.

- Free form Question-Answering
- Image Captioning
- Dialogue (chit-chat or task-based)
- Grammar correction

- Story generation
- Poetry generation
- Joke and pun generation

Conventionally, the problem of generating text from a given input divided into subproblems to simplify the process. Most common sub problems from determining the information needed to generating text are presented below.

- **Content determination:** Determining the information involved in the generated text.
- **Text structuring:** Deciding the order of the information presented in the generated text.
- **Sentence aggregation:** Selecting, structuring and ordering the information to present in the individual sentences.
- **Lexicalisation:** Finding the words and phrases that should be included in the text to articulate information.
- **Referring expression generation:** Selection process of words to determine the way of referring the entities.
- **Linguistic and structure realisation:** Generating the final text by linking the words, terms and phrases taking the requirements under consideration.

Early approaches to solve the text generation problem relied heavily on solving these tasks. After the invention of internet and increase in data, data-driven statistical methods gained power thus, the research on this area shifted to machine learning and deep learning methods which blurred the boundaries between these tasks. Nowadays most of the state-of-the-art NLG models are using deep learning.

2.3. NLG APPROACHES AND ARCHITECTURES

Over the years, many approaches presented to solve the NLG problem and most common NLG sub problems outlined in the previous section. Early approaches relied on symbolic or knowledge-based methods (Reiter & Dale, 1997), however data-driven methods proved their success after the access to the data became easier and some

methods incorporated statistical learning in them to generate outcomes that were less dependent on human interaction and domain restrictions. For the sake of this paper, NLG approaches divided into three broad groups. First one is defined as modular approaches which are grouped together by the common architecture they used which is a reference pipeline architecture is shown in Figure 2.3-1, has roots in early sequential approaches in the field (Dale, Mellish, & Zock, 1990) and it was first demonstrated by Reiter (1994).

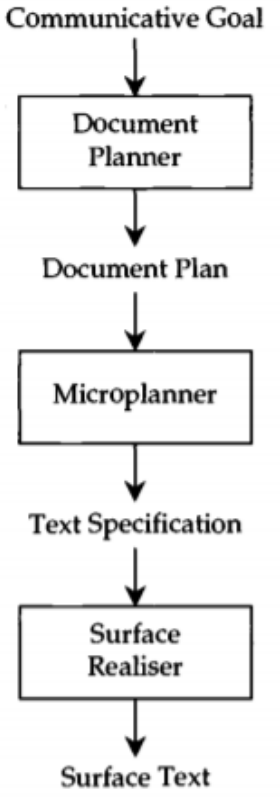


Figure 2.3-1 NLG system architecture Reprinted from "Building applied natural language generation systems," by E. Reiter and R. Dale, 1997, *Natural Language Engineering, 3,* page number. Copyright 1997 by Reiter, et al.

Second group is named as planning-based approaches which are based on planning which is a process of selecting the actions to be performed for reaching a specified goal, and the last part focused on machine learning and deep learning approaches.

2.3.1 Modular Based Approaches

Reiter (1994) taken most early approaches into account when proposing a general architecture for the NLG field which is shown in Figure 2.3.1-1. The modules in the pipelines accommodates some combinations of sub-tasks examined in Section 2.3.

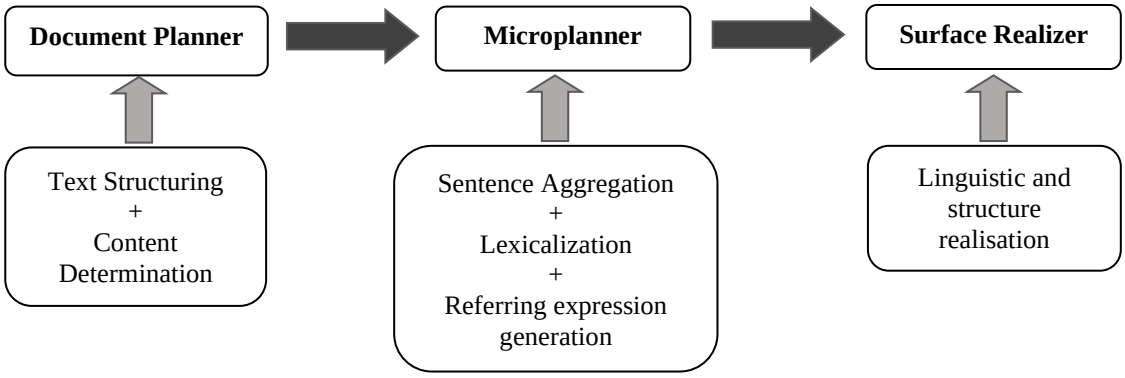


Figure 2.3.1-1 Modular architecture pipeline that incorporates sub problems from Section 2.3

Document Planner pays attention to the decision of "what to say", in this manner it tries to solve the text structuring and content selection problem. As McDonald (1993) pointed out it deals with the strategic generation. The output of Document Planner which is a text plan serves as an input to the Sentence Planner (Microplanner). Sentence aggregation, lexicalization and referring expression generation are the main problems that Sentence Planner is concerned about (Reiter & Dale, 1997). Since Document Planner deals with the problem of "what to say", Sentence Planner takes it one step forward and it focuses on "how to say it". What remains is the generation of text which is the problem of "literally saying it". Surface realiser is concerned about the linguistic and structure realisation sub problem which solves that sub problems by taking grammar of the language and its rules into account which results into the final text. The Sentence Planner and Surface Realiser performs the tasks that identified as the tactical generation.

The first module, the Text Planner (or Document Planner, or Macroplanner), combines content selection and text structuring (or document planning). Thus, it is concerned mainly with strategic generation (McDonald, 1993), the choice of 'what to say'. The resulting text plan, a structured representation of messages, is the input to the Sentence Planner (or Microplanner), which typically combines sentence aggregation, lexicalisation and referring expression generation (Reiter & Dale, 1997). If text

planning amounts to deciding what to say, sentence planning can be understood as deciding how to say it. All that remains then is to actually say it, i.e., generate the final sentences in a grammatically correct way, by applying syntactic and morphological rules. This task is performed by the Linguistic Realiser. Together, sentence planning and realisation encompass the set of tasks traditionally referred to as tactical generation.

The pipeline was widely used because of the clear distinction it made among the assignment of sub problems to modules and describing the proceeding sub problems for information (Smedt, Horachek & Zock, 1996). Systems such as CORAL and SUM-TIME-MOUSAM (Dale, Geldof & Prost, 2003 ; Reiter, Sripada, Hunter, Yu, & Davy, 2005) inherited modular approach which are route finding and weather forecasting generator systems respectively. When the architecture starts working the representations are created at the Document Planner stage then passed to the Sentence Planner and at last those internal representations transferred to the Surface Realiser in order to create the actual text. As discussed above those modules solves specific sub problems of NLG with various approaches which are examined alongside with the details of modules, in the next sub sections.

1.1.1.1 Document Planner

Document Planner involves messages which are grouped together by considering the requirements of the domain that the system is trying to build and represented as trees whose leaf nodes specifies individual messages. The messages are built by changing and outlining input data, labelling the entities, concepts and relationships gathered from the domain (Reiter & Dale, 1997). There are several approaches taken in order to solve the two sub problems, content determination and text structuring involved in the module of Document Planner.

Many researchers taken the reasoning approach for solving the content determination problem .Allen & Perrault (1980) recommended a plan recognition to solve that problem however that approach required sophisticated reasoning and vast among of knowledge about the world and the user because many different plans require similar request of information. Even though there are many investigations conducted about plan recognition, no real-world applications of NLG systems noted about that approach

because of the reasons stated above (Reiter & Dale, 1997). Another approach is to have domain-specific rules for the information to be included when determining content. Most used process for this is examining text with pre-determined operations based on the domain which is very similar to the knowledge acquisition task investigated by Scott, Clayton, & Gibson (1991).

The second sub problem in Document Planner is text structuring which is structuring the content produced by content determination in order to have coherent text. While content determination focusses on what information should be included in the text to be generated, text structuring determines the organisation of structure to create meaningful output. There is no agreement upon a de facto text structuring in the field, in most cases the text structure created is domain specific. To solve the second sub problem text structing in Document Planner, two main approaches proposed. First is planning-based approach which is in broad terms, for reaching a specific target determining a sequence of actions. Here the target is to create a coherent text structure and the actions are plans that should be in place in order to reach that target. The text should be structured by putting the right messages one after another so there must be a plan in place for having a knowledge of which information comes one after another. NOAH (Nets of Action Hierarchies) proposed by Sacerdoti (1977) takes the problem presented to it and applies series of pre-defined actions in an initial state of the world. First it creates a one-step solution then expands it further progressively to have a deeper more detailed solution then plans are represented as partial orderings of actions. Essentially, it is a system of hierarchical planning that used artificial intelligent style planning operators that has rules to what message should be used in where in order to have a coherent text. Many approaches came after based on NOAH (Cawsey, 1993 ; Hovy, 1991 ; Maybury, 1992 ; Moore & Paris, 1993) however the usage of procedural semantics for taking care of specific domain problems and their reliance on customized planning made them unprincipled and difficult to analyse. In addition to that, there are no rules specified for them to generate incorrect plans, they have redundant steps in their planning and, sometimes they fail to find plans in situations where they exist (Young & Moore, 1994). DPOCL system proposed by Young and Moore (1994) in order to overcome these problems by two new properties completeness and soundness. Completeness ensures that there is a plan for all situations that might occur and no incorrect solutions when creating the text structure, soundness create steps in a plan in

a way that steps in a plan doesn't interfere with each other and redundancy is not an issue. Around the time that modular approaches were used, because of the computational power and the lack of knowledge about the text structuring space, made it difficult to create real world applications that uses planning for text structure. The approach that was widely used in modular systems is schema-based approach. Since the approach is domain specific, it can be easily adopted to real world applications. A relatively small number of patterns can be extracted in order to satisfy the needs of text structuring and these patterns discovered by a thorough analysis of the corpus determined by content selection and, with the aid of domain experts. The patterns involve the order, relations and words to connect messages. Most widely accepted approach is SCHEMA, is "a computational model of discourse strategies encoding text organization" which is developed to guide the generation process in order to decide what to say next (McKeown, 1985).

1.1.1.2 Sentence Planner

Sentence Planner consists of the NLG subproblems sentence aggregation, lexicalization and referring expression generation which are inter-related however separate tasks which broadly concerns about syntactic structure of the text and the decides how to merge them together into one or more sentences (Walker, Rambow, & Rogati, 2001). There are various concepts proposed in order to solve the three sub problems related to NLG which eventually means constructing the Sentence Planner.

Sentence aggregation tries to solve the problem merging two or more messages in order to create a sentence. As mentioned in the beginning of Section 2.3.1.1 the output of the Document Planner are trees which includes leaf nodes that consists of specific messages. Sentence aggregation takes those messages in the leaf nodes as input and combines them to form sentences. Which message should the sentence contain and how should those messages represented in order to be syntactically correct, must be determined by the sentence aggregation sub problem. Sole approach to solve this problem is to create a set of rules which is decided by examining a corpus to have a grasp about the aggregations appeared in it. Those rules the applied to the NLG system to have similar aggregations that are extracted from the corpus. After this process lexicalization and referring expression generation takes place which concerns about

selecting the right words for referring to an entity or expressing a concept related to the domain respectively. Like sentence aggregation they are both essential for having a coherent and fluent text, the better those concepts get the better the reader comprehend the generated text however bad lexicalization and referring expression generation can still produce understandable text.

Lexicalization covers the problem of expressing a concept related to the domain. In the following sentence “TUDublin city campus will be relocated to Grangegorman in 2020” the word relocated probably referred as a relation and defined as “RELOCATE” in the message. In order to express the concept lexicalization mapped the relation relocate into “relocated” it can also map it into “moved”. An early approach to solve this problem is using decision trees for choosing the right expression of concepts depending on the rules of the domain and syntax (Goldman, 1975). This approach however is monolingual and needs to be reconstructed for another languages and cannot deal with more than one language which means it is not possible to use that approach for machine translation. A frequent approach to solve the lexicalization on both mono and bilingual NLG tasks is to treat the message as a graph which consists of concepts in the domain and relations and map that graph into another graph which can be identified as an output graph that includes the words and the syntax of relations. To do that researchers developed dictionaries that consists of the concepts in the domain and relations which is mapped into a structure that has various words and representations, when executing the matching of the input and output occurs to get results (Nogier & Zock, 1992 ; Stede, 1996).

Referring expression generation is the problem of deciding how to refer or describe the entity for the reader to understand which entity this sentence is referring to or describing. In the following sentences “TUDublin is moving to Grangegorman, it will be one of the best campus in Ireland. The demand for this university will definitely increase.” the entity TUDublin referred as third times, first with its original name, second with “it” and third since the entity had been stated with its original name before “this university” is used. Those decisions are the responsibility of referring expression generation. Most of the work done to identify these relationships between the first introduction of entity and pronouns are based on conditional clauses. If the entity should be used which is decided based on the information of other words around it,

hasn’t introduced then use its name, if it is introduced select a noun that describes it and if necessary add adjectives or similar distinguishers to separate it from the other objects (Mani & Dale, 1995).

1.1.1.3 Surface Realizer

Surface realizer involves only linguistic realization which is the last part in the modular approach. Its task is to take the relevant words and phrases and link them in a way to generate grammatically correct sentences. The problem that needs to be solved is the right ordering of the words and generating right morphology. Usually, it needs to modify its input by adding auxiliary words, prepositions and punctuation marks. One of the first approaches taken to solve this problem is to create templates to specify outputs. It is domain specific, can be used for relatively small domains and when low deviation in the sentences is expected. The reason for those limitations is, this approach creates templates that are fixed other than the words which comes from the output of the lexicalization and referring expression generation (Mcroy, Channarukul, & Ali, 2003). An example sentence in a template can be “Weather will be $weather_type in $city on $weekdays”. Marked parts are the outputs of the Sentence Planner which can be “rainy”, “Dublin” and “Sunday” and other parts are fixed. For having a general-purpose realization module hand coded grammar system proposed, which in broad terms make the decisions based on the grammar of a language. Early grammar-based realizers the grammar and its rules are written by hand, some examples are KPML (Bateman, 1997) and RealPro (Lavoie & Rambow, 1997). Since they require the whole grammatical and semantic information of a language it was a hard task to prepare them for a real-world scenario. Because of this, researchers proposed a hybrid method which combine templates with hand coded grammar systems. In the approach the realisation supplies the morphology and syntax however the decision making is left to the developer (Vaudry & Lapalme, 2013). Another widely used approach is statistical which obtains a large corpus and extracts the probabilistic information of grammars which decrease the amount of manually hand coding while extending the amount of knowledge gathered. One approach proposed by Langkilde (2000) creates a hand-crafted grammar for all possible representations which is demonstrated as a forest then relied on statistical models to build a stochastic re-ranker to perform ranking (Stent & Srinivas Bangalore, 2014). Despite of being effective that

approach is computationally expensive, so another method is proposed which makes statistical decisions at the very end generation level. The approach defined by Belz (2008) as “a language generation framework that was developed with the aim of providing the formal underpinnings for creating NLG systems that are driven by comprehensive probabilistic models of the entire generation space (including deep generation)”. It creates most probable sentence using a context-free grammar, which handles the decision-making process statistically in order to find the optimal solution.

2.3.2 Planning Based Approaches

Modular approaches are easy to comprehend because of the distinction it made among the sub problems of NLG however there were some difficulties observed when using modular approaches. One of the first problem encountered was the generation gap (Meteer, 1991) which are the confusions between the strategic and tactical generation parts. For instance, at the strategic stage an ordering can be determined but when linguistic realizer starts to produce the actual text that constructed ordering become obscure (Inui, Tokunaga & Tanaka, 1992). Another problem is the constraints problem. There can be some constraints regarding to the text to be generated which can be defined in the early stages in development such as content determination or text structuring however it was discovered that controlling if those constraints met cannot assessed until the generation of text. This is discussed in a study conducted by Reiter (2000) and grounded with an example in the beginning of the study which is “Some types of documents need to meet size constraints, such as fitting into a limited number of pages. This can be a difficult constraint to enforce in a pipelined natural language generation (NLG) system, because size is mostly determined by content decisions, which usually are made at the beginning of the pipeline, but size cannot be accurately measured until the document has been completely processed by the NLG system”. In addition to that, when trying to create a general-purpose system since all the modules must have sub systems to conduct it was computationally expensive. Those problems stated above led researchers to seek different solutions to text generation problem. The approaches discussed on this and the next section blurred the boundaries between the modules and involves methods that intended to solve all sub problems with one architecture. With an unlimited space of actions planning approaches can pass through the boundaries represented in the modular approaches (Gatt & Krahmer, 2018). It can

solely decide on “what to say” and “how to say it” and actually saying it in the same system. “In the subject of AI, planning refers to determining a sequence of actions that are known to achieve a particular objective when performed” (Inder, 1996, pp. 23). It is the method for determining a sequence of one or more actions in order to reach a particular goal. The main goal can be divided into sub goals, accomplished by actions which has pre-determined conditions and effects (Fikes & Nilsson, 1971). The methods used in the planning process may differ however the fundamentals which involves the initial state, state transition system and goal expressed in every method (Garoufi, 2014) and defined below.

- **Initial state:** In a formal logic-based language the initial state impose which propositions are true during the stage of planning.
- **State transition system:** When transitioning from one state to another, state transition system defines the changes occur in the world according to the consequences of actions. The preconditions in the actions determine whether the actions can be carried out according to the state of propositions, then specify the changes made to the propositions after the execution.
- **Goal:** Goal is the state or states which the system tries to reach eventually after carrying out several actions.

The aim of the planner is to arise with specific actions which are actions when performed, guides the planner from initial state to the goal and that solution is called a plan (Garoufi, 2014).

Planning methods were first used for building a plan for physical actions one of the first study that investigated whether the planning can be used for text generation is a paper by Cohen and Perrault (1979). They sought to find a definitive formulation of plans containing preconditions for speech acts with the aid of Searle’s formulation. One other early study which cut across the borders of modular approaches was KAMP (Appelt, 1985) which was in broad terms modelling the user's beliefs and goals. It maps the mutual beliefs and intentions of the speaker and the hearer and uses a planning system for producing the utterances (Paris, 2015, pp. 158). The system created sub-goals according to the mapped beliefs and intentions and, used those from the initial state of the system till reaching specified goals. If a comparison made between KAMP and modular approaches, the initial state begins from the Document

Planner and ends with Surface Realiser, and KAMP solves those tasks in one system. Solely using planning means giving the system the ability to do decision-making at every step which requires deciding on the rules of grammar. Since the possibilities for deciding about the syntax, morphology and semantics require a huge action and state space the computational expense becomes infeasible.

Rather than unrestricted reasoning, using a linguistic framework for the planning process was considered a solution for speeding up the operations. Some early investigations in modular approaches such as KPML (Bateman, 1997) which uses planning to solve linguistic realization, is based on Systematic-Functional Grammar (Halliday & Matthiessen, 2004) that is considered as a baseline for creating a planning approach through grammar. More recent approach is adapting planning into the grammar and treating linguistic structures as planning operators. This requires grammar formalisms which integrate multiple levels of linguistic analysis, from pragmatics to morpho-syntax (Gatt & Krahmer, 2018). Lexicalised Tree Adjoining Grammar - LTAG (Joshi & Schabes, 1997) is considered as a common method for planning based approaches. For LTAG to generate smooth text based on the semantic preconditions and the goals that it will satisfy when the specified text is generated semantic information was combined with the linguistic structure (Garoufi & Koller, 2013).

2.3.3 Machine Learning Approaches

In spite of having a various of planning approaches available for practical use, as indicated by Koller and Petrick (2011) planners spend considerable amount of time on pre-processing only after pre-processing they tend to produce efficient results, in addition to that there are still aspects that needs human interaction which made the systems dependent to human interference and the skills of its developer. A vast increase in available data in almost every field of AI made it possible for researchers to fully utilise data-driven solutions one of which and probably the most common is machine learning. Despite the popularity it has today, it is an old field. The inventor of the term “Machine Learning” Arthur Samuel described it as “the field of study that gives computers the ability to learn without being explicitly programmed” (Samuel, 1959). It is an interdisciplinary field that combines computer science with statistics. The broad purpose of machine learning is learning from past experience in order to

improve future performance (Kumbhar & Kunjir, 2017). Learning term in machine learning models stands for, extracting and enhancing the knowledge hidden in the data and adjusting themselves to accomplish the task that is presented to them. In other words, “a computer program is said to learn from experience E, with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E” (Mitchell, 2017). A high-level pipeline of machine learning models starts with data preparation then training of the model for adjusting its parameters for representing a general function for the data and finally testing the model with new instances in order to assess its performance. There are several learning paradigms proposed for machine learning models and most common of those discussed below.

- **Supervised Learning:** This learning method assumes that the category structure in the data is intact and the models that use supervised learning demands labels and the instances present on the data that correspond to them in order to determine a function that maps the instances to the class labels.
- **Unsupervised Learning:** There is no knowledge of pre-determined labels to figure out which instance in data belong to which class. Instead of creating a function that maps the instances to class labels , unsupervised learning models concerns about extracting the existing clusters or patterns in the dataset and its aim is, when a new instance is present, mapping it to the correct cluster that was defined at training time.
- **Reinforcement Learning:** It is learning what to do in order to maximize the reward signal by mapping situations to actions. There are no instructions given to the model about which actions it should take in order to reach the ultimate reward instead it discovers actions that yield the most reward by trial and error (Sutton & Barto, 2018, pp. 1). This is a learning technique not only designed for machine learning purposes, it can be used in machine learning, planning, operations research etc. and has different names related to its use such as dynamic programming.

Machine learning models or statistical models which was what some researchers called them in early days in the field, tend to improve in performance with more data

presented to them. There are various machine learning approaches presented in order to solve the NLG problem in various tasks.

Since decision-making in text generation is choosing the right choice when dealing with the sub problems of NLG such as choosing the right structure, pronoun or word etc. it can be viewed as a classification model. In order to do that a cascade of classifiers can be built in order to construct the output incrementally where the next classifier uses the output of the previous classifier as an input (Daelemans & Van den Bosch, 1998). Marciniak & Strube (2005) used LTAG (Joshi & Schabes, 1997) to represent text in grammatical form and used decision tree classifiers (C4.5) that at any stage of generation produced most probable output which then passed onto the next classifier. That approach used classical pipeline system described in Section 2.3.1 and compared with integer linear programming to integrate decisions involved in every step, which uses Naïve Bayes algorithm with reinforcement learning.

Most of the machine learning models are built for classification or prediction of continuous values such as decision trees, support vector machines and linear regression. However, text is a discrete type of data and text generation requires delicate planning and may involve classification at some steps as discussed above however when generating there are not many suitable machine learning architectures proposed for NLG especially long text generations such as story generation. However recent focus on deep learning created models that can handle discrete data and has the power of capturing the semantics, grammar and linguistic structures which required delicate planning and human interaction before.

2.3.4 Deep Learning Approaches

The expression “Deep Learning” first proposed by Aizenberg, Aizenberg and Vandewalle (2000), is a subfield of machine learning which “allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction” (LeCun, Bengio, & Hinton, 2015, pp. 436). Most of the models built in deep learning based on neural networks which was first proposed by McCulloch and Pitts (1943). They consist of simple processing units which are called perceptrons that calculate a function of their input. A perceptron or neurons may

not achieve much but when integrated into a complex architecture they become quite powerful. Both perceptrons and neural network shown in Figure 2.3.3-1.

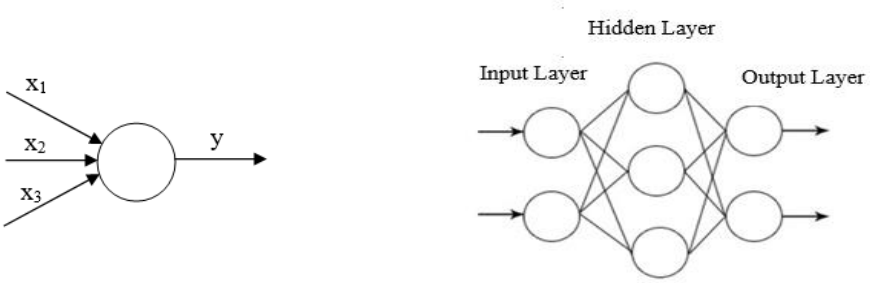


Figure 2.3.3-1 Basic structure of Perceptron (left) and Neural Network (right)

Over the years many functions put to work inside the neuron of a network but originally first the linear regression of the input values and weights are calculated then passed to an activation function which is a sigmoid function. The activation function computed in order to get the hypothesis (output) of the neuron. The activation function pushes the result of linear regression into 1 or 0 which means it decides if the calculations made inside the neuron should or shouldn't passed to the following layers. The power of the neural networks comes from ability to build non-linear models without the manual encoding of higher order featuring from its base features. The network itself looks at building these high order features as a part of its training process. The advantage of this there is no need for the developer of the neural networks to construct high order features and decide which ones are useful, the network will only create high order feature that are shown to have effect. The training process of the network has two parts, first one is forward propagation which means feeding an input into the system and from left to right doing the necessary calculations till the prediction is produced, the second one is backpropagation which is calculating the error produced when the system makes a prediction and back propagating it through the network for adjusting the trainable parameters of the network in order to make better predictions next time. The backbone of the training process is backpropagation because without it network only does predictions and cannot learn to build better ones. The perceptron, the neural network architecture and propagations form the baseline for deep neural networks which used in deep learning. In order to call it "deep" it must have more than one hidden layer.

Despite of the recent impact made by deep learning into the NLG field, neural networks had been investigated in various studies in terms of generation such as the paper by Kukich (1987), however due to the lack of computational power and data till 2010s the research on neural networks was limited. Due to the increase in hardware that can support the intensive computations required for deep neural networks such as GPUs and the creation of frameworks that comes with the backpropagation and recent improvements in deep learning (i.e. Geoffrey Hinton and his teams implementation of backpropagation to Tensorflow framework in 2009) made it possible to use deep neural networks in many fields of AI one of which is NLG. The advantage of deep neural networks on NLG is, they are great at capturing grammar and semantics which made several early hand crafted rule based approaches for identifying grammatical structure and semantics in languages obsolete (Mikolov, Chen, Corrado & Dean, 2013 ; Pennington, Socher, & Manning, 2014).

In the last two decade many approaches, and architectures related to deep learning proposed in order to solve the NLG problem including sequential models that use several architectures including feed-forward networks (Bengio et al., 2003 ; Schwenk & Gauvain, 2005), recurrent neural networks (Mikolov, Karafiat, Burget, Cernocky & Khudanpur, 2010) and its variations long short-term memory (Hochreiter & Schmidhuber, 1997) and gated recurrent networks (Cho, Merriënboer, Gulcehre, Bougares, Schwenk & Bengio, 2014), transformer neural networks (Vaswani et al., 2017) and convolutional networks combined with some other features such as attention (Gehring et al., 2017). Main benefit of these architectures is they are capable of processing text with multiple lengths avoiding both data sparseness and explosion in the number of parameters through the projection of histories into a low-dimensional space, so that similar histories share representations (Gatt & Krahmer, 2018).

One other architecture followed the footsteps of sequential models is the encoder-decoder architecture which enabled end-to-end approach to map sequences to sequences (Seq2Seq) (Sutskever, Vinyals & Le, 2014). The intuition behind this approach is the encoder maps the sequence to a vector of a fixed dimension which is then passed to the decoder in order to decode into another sequence and Sutskever et al., (2014) used long short-term memory (LSTM) to do those operations and also there are other studies that used encoder decoder architecture that are combined with other

networks (Gehring et al., 2017 ; Vaswani et al., 2017). Despite valuable studies in tasks such as machine translation (Kalchbrenner & Blunsom, 2013) if the input text is more than one sentence or a long text such as articles, an encoder decoder model may need some information from previous sentences at encoding in order to decode properly for the given task however there was no significant results reported that uses vanilla encoder decoder architectures for long text that include more than one sentence. The reason for that is when the network is encoding it forces all the information captured from the input text to be stored in the encoded vector and as shown in Figure 2.3.3-2 it is the only connection between the encoder and decoder thus, the decoder generates text according to the information it gets from the encoded vector. However, for long texts the network puts too much pressure into the encoded vector which results in an information bottleneck (Bahdanau et al., 2015). Essentially, by the time the encoding process gets to the end for long input text, since the network forces all the information gathered from the long process of encoding the input, the encoded vector might lose some valuable information presented at the early stages which results in poor decoding.

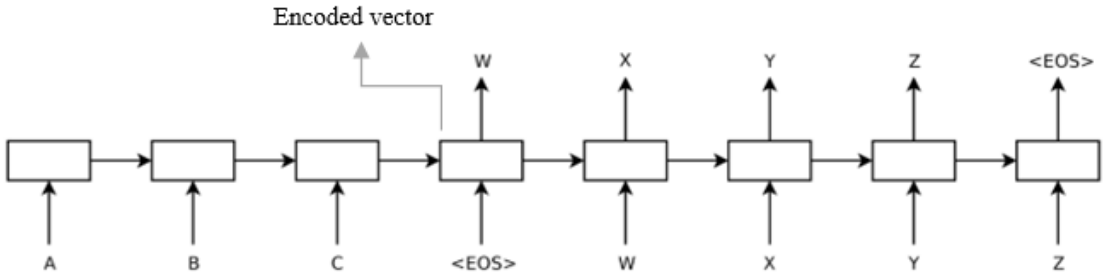


Figure 2.3.3-2 A high-level illustration of encoder decoder architecture

In order to solve this problem Bahdanau et al., (2015) presented a structure called attention which implemented inside the encoder decoder architecture. The difference of attention based encoder decoder model is when the model is doing generation at every step of decoder, in addition to gathering information from the encoded vector, using a direct connection to every vector in encoder should result in attending the important parts of previous sentences in order to generate next words in the sequence properly. This proposal adopted by several text generation approaches one of which is a study proposed by Serban, Sordoni, Bengio, Courville, & Pineau, J. (2016). They used encoder to capture the dialogue context and utterances and decoder to predict the

next word in the sequence which reached results close to the state-of-the-art models. Despite their success dealing with long texts there were some other problems. In order to track long term dependencies, they used attention mechanism at every sequence in the encoder layer and dependencies on the computations made at the previous steps of the network which doesn't allow parallelization thus, for example at training time the backpropagation needs to be done through time which slows down the training process substantially (Gehring et al., 2017). Also, because of that reason those networks need serious amounts of computational power for storing and reusing that information. This resulted in inefficient use of hardware thus it brought many difficulties to implement them into real world scenarios.

One approach to solve this problem was presented by Gehring et al., (2017) which uses convolutional neural networks to implement parallelization, gated linear units to ease the gradient propagation and attention modules in the decoder. That convolutional seq2seq model outperformed LSTM models in both performance and efficiency in various tasks. Because of their efficient structure they used in various long text dependent tasks such as story generation (Fan et al., 2018) thus, this architecture is discussed in Section 2.4.3 thoroughly. Another approach to resolve this matter is an architecture called transformer network (Vaswani et al., 2017). Since one of the models adopted in this paper is transformer network it is described in the next section with more details.

2.4 TRANSFORMER NETWORK

In order to ensure parallelization transformer network gets rid of the recurrent structure and focuses solely on the attention. It adopts the encoder decoder architecture and the architecture described in the paper has 6 stacked encoder and decoder. There are two main processes take place in the encoder, attention and feed forward network, which are also present in the decoder and in addition to that decoder has another attention layer which has inputs from both encoder and previous decoder attention layer. The high-level architecture of the model is shown in Figure 2.4-1.

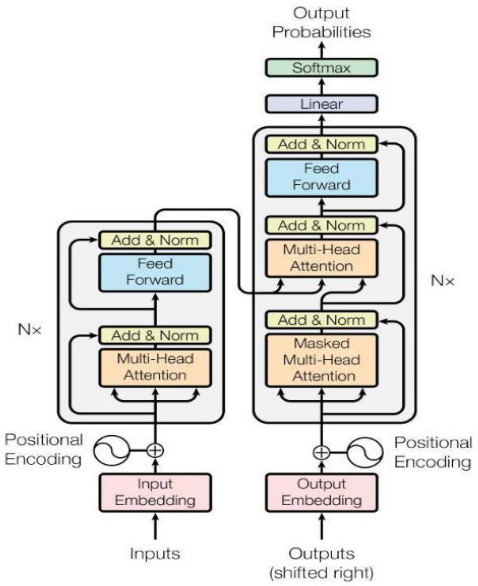


Figure 2.4-1 The Transformer model architecture. Reprinted from "Attention is all you need" by Vaswani et al., 2017, Proceedings of the 2017 Neural Information Processing Systems (NIPS 2017), 6001. Copyright 2017 by Vaswani et al.

To increase the power of attentions Vaswani et al., (2017) presented a new attention mechanism called multi-headed attention which has 8 attention head concatenated to form attention layer thus, it increases the models ability to focus on different parts of the text presented to it. The study used byte-pair encodings (Sennrich, Haddow & Birch, 2016) for creating input embeddings and as explained when discussing the study made by Gehring et al., (2017) transformer networks also used positional encodings to determine the positions of the words in the input text with sine and cosine functions of different frequencies which shown in below where i is the current dimension, pos is the current position and dmodel is the dimension of the model.

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

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

After positional encodings implemented inside the input embeddings, they are linearly transformed into three vectors which are queries, keys and values vectors. The vectors that are used in the multiplying process were first initialized randomly then trained during the training process. All attention matrices inside the multi-headed attention consist of those three vectors. The dot product of queries and keys followed by a

division (the square root of the dimension of the key vectors used in the paper -64) to have more stable gradients and a softmax function determines how much focus should the network needs to pay attention to the other parts of the text when encoding a particular word. Then those outputs of the softmax multiplied by the value vector to create the weighted embeddings. After that, those embeddings summed up to create the output of one attention layer. This is done 8 times in parallel then those results concatenated and multiplied with a weight matrix which is also randomly initialized and trained during training process, in order to get the output of the multi-headed attention. In the following process those results passed into a layer normalization which has the output of multi-headed attention and the input the encoder with the help of a residual connection as input then the output of the layer normalization passed into a feed-forward network. The output of the feed-forward network then passed into another layer normalization, with another residual connection it has the input and output of feed-forward network as input and the output of the layer normalization is the output of the encoder. An illustration of an encoder is shown in Figure 2.4-2.

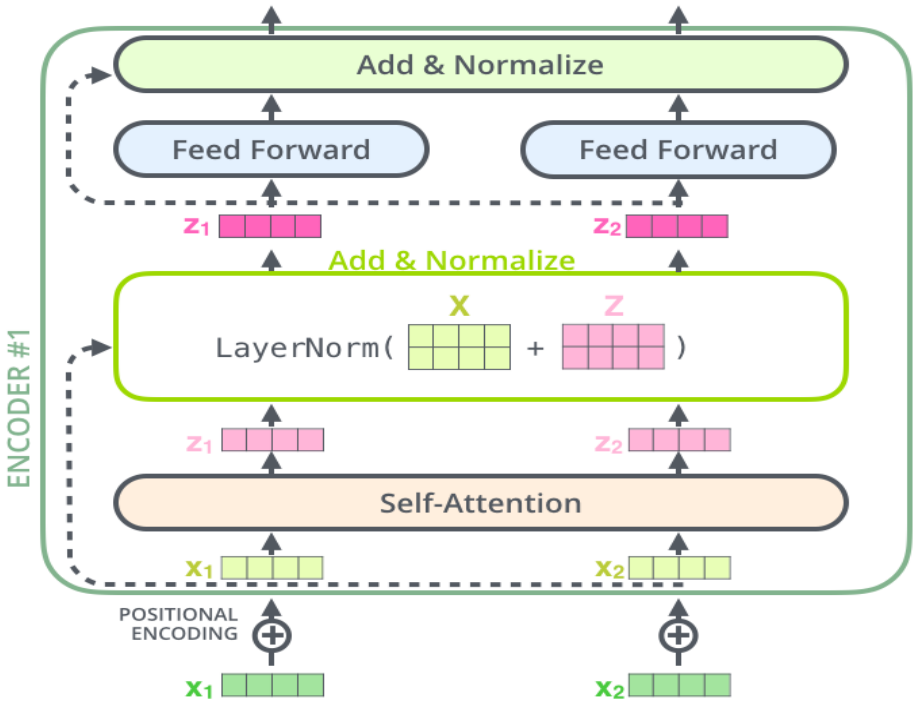


Figure 2.4-2 Encoder of the transformer network with details[3]

[3] http://jalammar.github.io/illustrated-transformer/

The decoder of the network has the same architecture with the encoder except it has one more attention layer between the multi-headed attention and feed forward network which is another multi-headed attention which takes the outputs of the encoder and the output of the first multi-headed attention layer as an input in order to focus appropriate places in the input when decoding. In addition to that the first multi-headed attention layer is a masked attention layer thus, it ensures the decoder to attend to the earlier positions in the output in order to prevent bias. After the decoding step the outputs of the network passed into a fully connected neural network that is a linear network that outputs the logits vector which then passed into the softmax layer in order to convert them into probabilities. The details of the decoder shown in Figure 2.3.3-5. The difference between decoder and the encoder is, encoder takes all the text as input however decoder generates words one by one and uses teacher forcing at training time when generating which means after predicting a word if it is different from the ground truth it changes that word back to the ground truth word in order to force the decoder to predict next word according to the prior ground truth.

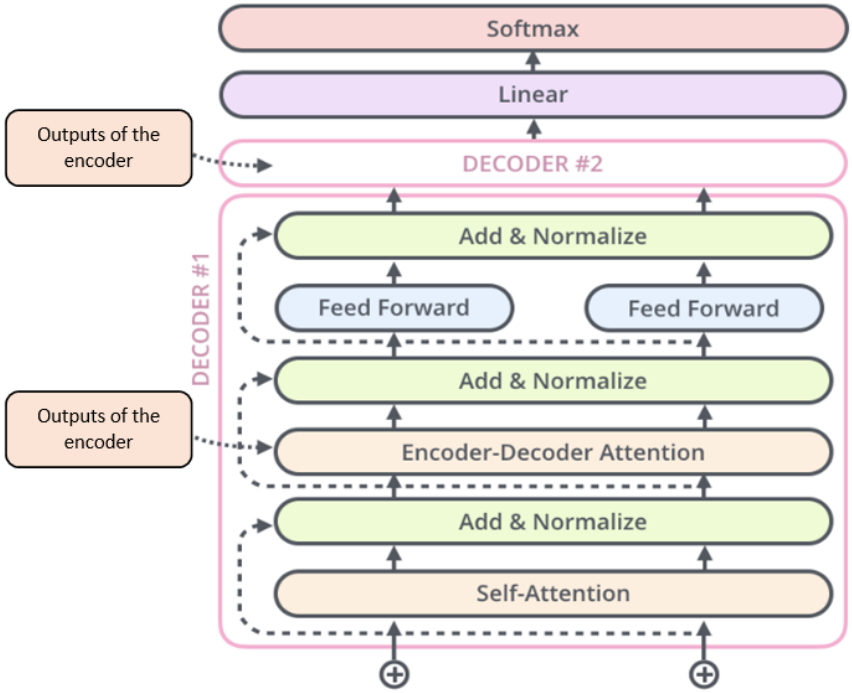


Figure 2.4-3 Decoder of the transformer network with details

When training a sequence to sequence neural models the standard loss function is maximum likelihood estimation (MLE), which maximizes the log-likelihood of

observing each word in the text given the ground-truth proceeding context (Poon, Yap, Tee, Lee, & Goi, 2019). This model uses a multi-class cross-entropy as a loss function which is a different representation of log-likelihood thus, they are actually the same formula (Bishop, 2016, pp. 168).

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Multi-class cross-entropy loss can be calculated with both categorical cross-entropy or sparse categorical cross-entropy. Sparse categorical cross-entropy calculates the loss with integers unlike categorical cross-entropy which is dependent on one-hot encoded vector. The formula is same for both, but categorical cross-entropy needs a vector however sparse categorical entropy needs integers, so it is faster. In the formula presented above $\hat{y}_i$ and y_i represents the predicted value and the ground truth respectively since transformer networks use byte pair encoding the values of the outputs are integer values between 0 and the token number that the byte pair encoding encoded, the predicted and ground truth values must be integers in that range thus, transformer network uses sparse categorical entropy.

There are two main important aspects of this model. First it does attention naturally like humans do it has the ability to attend all the word simultaneously which made this and several other variations of this model very powerful and become state-of-the-art in some tasks such as machine translation (Vaswani et al., 2017). The other important aspect is it because of its attention mechanism it allows parallelization.

2.5 EVALUATION METHODS IN NLG

2.5.1 Automated (metric-based) Evaluation

Natural Language Generation models inherited automated metrics from several related fields such as summarization, image captioning and machine translation. Essentially, these metrics compute a score which shows the similarity between the text generated by the model and the ground truth. There are several automated metrics adopted to

NLG such as BLEU, ROUGE, METEOR and perplexity which are cheap and easy to implement however those metrics are metrics for comparison thus, they are not concerned with the quality of the text generated and in many occasions even the generated text is diverted from the ground truth it can have significant quality. However, those metrics are not totally useless Reiter and Belz (2009) proved that there is a weak correlation between automated metrics and human evaluation, which is far more useful than automated metrics for text generation, so they can be considered as an acceptable starting point.

- **BLEU:** Found by Papineni, Roukos, Ward and Zhu (2002) essentially it is a modified precision for comparing the result of a model and the ground truth thus, it tries to find how many of the predicted text is actually correct. Since build for machine learning tasks it generates more than one output so the precision it does can be called multi-class precision. For translation it has high correlations with human evaluations.

- **ROUGE:** Rouge consists of several metrics that proved their use for text summarization. ROUGE has more than one metric. One of them is ROUGE-n which tries to assess the performance of the model by counting how many n-grams of generated summaries, matches the n-grams of ground truth. In addition to that, the other ROUGE metrics ROUGE-l, ROUGE-w and ROUGE-s are based on longest common subsequence , weighted longest common subsequence, and skip-bigram co-occurrence statistics, respectively. They have shown high correlations with human evaluations (Lin & Hovy, 2003).

- **METEOR:** Lavie and Agarwal (2007) the founders of this metric claim that the metric has higher correlation with human annotators then BLEU score for machine translation. They used weighted F-score that penalizes wrong ordered translations which was not considered by BLEU.

- **Perplexity:** Perplexity can be considered most adequate metric for text generation since it is a metric for assessing the fluence of the generated text. It shows how good a model is when predicting the next word in the sequence. It is a widely used metric for speech recognition systems. Perplexity PP can be

defined as the geometric of the inverse of the per-word likelihood on the held-out test corpus so lower perplexity means better generalization.

$$\text{Perplexity} = 2^{\textit{cross-entropy}}$$

Another evaluation method cross-entropy can be calculated as $\log_2 PP$ (Kobayashi, 2014) which means perplexity can be calculated as shown in the formula above. Since perplexity can assess how fluent the generated text is, it can be a starting point when evaluating text generation tasks such as story generation.

2.5.2 Human Evaluation

When evaluating the readability or quality of the generated text automatic methods becomes obsolete which are tasks suitable for humans. Human can be intrinsic or extrinsic evaluations. Intrinsic evaluations usually done by showing the annotators both the generated text and the ground truth and the performance of the system is the comparison between the ratings given to them (Sparck Jones & Galliers, 1996). Another implemented intrinsic method is comparing taking one NLG model as baseline and making a comparison with others (Lester & Porter, 1997). Intrinsic methods tend to ask people how fluent, coherent or readable the generated text is by questionnaire or by a ranking system. For doing an extrinsic evaluation the proposed model should be embedded to a real world environment according to its use since its time consuming and some model's generation times require time that don't make them suitable for such an environment intrinsic evaluation methods are chosen much more then extrinsic ones. Human evaluation for creative text generation tasks such as poem, story or joke generation is much more essential then others since they require the assessment of measures such as enjoyment, expectation, coherence and creativity.

2.6. AUTOMATED STORY GENERATION

The study of narrative described as narrative theory or narratology by Fludernik (2009), is the theory of the structures of narrative (Phelan & Rabinowitz, 2008) which studies mostly the formal features of a story. The history of narratology can be viewed

as two phases, classical and post-classical. Classical narratology focused more on identifying the genres, structure and the systematics of telling a story, on the other hand, post-classical era defined narratology as an inter-disciplinary field which aims more on practical applications and studies with other fields such as cognitive narratology (Amerian & Jofi, 2015). Classic narratology divides the story generation into two parts, fabula and suzjet (Lemon, Reis, Ėĭkhenbaum, Shklovskiĭ, & Tomashevskiĭ, 1965). Fabula corresponds to story in English and it is not outlined by the writer, it is the story interpreted by the reader. More precisely it is a series of events that occur in a narrative which has no perspective. On the other hand, suzjet bears the perspective of the author, it is about designing a coherent story by ordering the events and, planning the structure of the story. These two terms served as a guideline for many of the work done in the field, one of which is Genette (Prince, Genette, & Lewin, 1980), fabula put into action as the events of the story and had precise computer representations which were ordered by taking their chronology and causal relationships into account, suzjet is the generated text. Essentially, fabula is the document and text planner in NLG and suzjet is the surface realiser. In addition to humanities narratology, linguistics was also an influential field for automated story generation. Constructing grammars for story generation which is called story grammars is an area in linguistics and used for story generation applications such as BRUTUS (Bringsjord & Ferrucci, 2000). To generate a long coherent narrative, automated story generation field have been made use of computational models. Many early models tried to decode the structure of a story by investigating the classical narratology which "is a humanities discipline dedicated to the study of the logic, principles, and practices of narrative representation" (Gradmann, Hühn, & Schönert, 2006). Since computer programs need a structure and precise knowledge of what to do and when to do it, early models focussed on examining the structure of the story to be generated by finding the causal relationship between events, the roles of the characters, the actions they might take and the results of those actions. Latter models however tried to give the models the ability to capture that information without explicitly programmed, especially deep learning approaches.

2.6.1 Grammar Based Approaches

One of the oldest approaches for automated story generation is generations based on story grammars. These grammars involve a set of transformation rules and those rules expand the story grammars in order to transform them into final stories. First story grammars were created by Lakoff (1972) rewrote the rules and structure created by Propp (1968) in order to create a grammar for Russian folk tales. Thorndyke (1977) creates his own set of rules by extracting rules that are common to a set of narratives. It was argued that those set of grammars and rules are focused intensively on grammar that they fail to create coherent stories (Andersen & Slator, 1990). Regardless of those investigations several successful story systems created one of which is TELLTALE (Correira, 1980). It generates a story by a set of grammars and rules held within a database and expanded in order to generate a story. The database was manually built to generate short stories itself. An example of a rule is, that each fairy tale must contain a setting and at least one episode, terminating with the main characters living happily ever after (McIntyre, 2011).

2.6.2 Planning Based Approaches

Stories generated with this method is based on planning in NLG. The planning systems created for storytelling usually took initial state of the world and the goal as an input then produced sequence of actions which might comprise sub actions that lead from one to the other in order to reach a specific goal. The goals for those systems can be character or authorial goals. The first automated storytelling model was created by Klein et al., (1973) generated stories in a weekend party setting. They created actions that follow previous events but instead of a goal they used a system similar to story grammars which tried to satisfy the grammatical structure of a story. Not only there was one domain the only thing that changed in the stories generated are the role of the characters. Another planning approach was TALESPIN (Meehan, 1977) created stories about the lives of woodland creatures. In order to create the story, the character was given a goal and then a plan was in place in order to reach that goal. TALESPIN combined events and related consequences with goals that occurred from previous events which are bound to the events that come before them (Kybartas & Bidarra, 2017).

Some other models concerned more on plot planning which aims to plan the plot which will lead to a story that satisfies one or more predetermined goals. They created stories chunk by chunk rather than creating them as a whole. UNIVERSE (Lebowitz, 1984) generated plots for soap opera episodes. For generating events which are melodramatic conflicts between characters planning was used (Lebowitz, 1984). The system paid more attention to the characters in the story which helped to create diverse roles that story grammars are in lack of. UNIVERSE kept track of the goals that aims to expand the plots to reach a complete story. BRUTUS (Bringsjord & Ferrucci, 2000) also used plot-planning for building stories for its betrayal narratives which is combined by story grammars. It used a knowledge base of story plot templates and planning for the association between causes and effects.

2.6.3 Modular Based Approaches

Several story generation systems followed the NLG pipeline examined in Section 2.3.1. McIntyre (2011) stated that the production of high-quality narratives clearly involves an integration of the story generation process with the natural language generation pipeline. GESTER (Pemberton, 1989) is a storyteller that generate medieval French epic story summaries which used a modular approach. Its grammar is hand coded based on nine French epic poems. From those grammars a tree is constructed in order to determine the document plan. In order to plan and generate the text Pemberton used a set of rules which makes the model dependent on human interaction. Callaway and Lester (2001) built one architecture and its implementation, in order to integrate the story generation process with the NLG pipeline which are AUTHOR and STORYBOOK respectively. The input, narrative plan examined, and the linguistic structures extracted in order to generate the story however since their focus was on more with the sentence planner and surface realiser. In order to fully integrate NLG pipeline with story generation Lönneker, Meister, Gervás, Peinado and Mateas (2005) stated that the systems must also take document planner into account. To fulfil the requirements of many stories such as flashbacks, ellipses or stories within stories story generation systems has to consider rhetorical structure of the document being developed which can be done by document planner (Lönneker et al., 2005).

2.6.4 Case Based Reasoning Approaches

Case-based reasoning was one of the widely used problem-solving task for generating stories. It solves problems by "recalling past scenarios where a solution was gained and apply those solutions to the current problem" (Slade, 1991). MEXICA (Pérez ý Pérez, 2001) is a short story generator based on early inhabitants of Mexico and generates the story by combining plot chunks together piece by piece. The system takes previous stories as input for assessing the performance of the next story generated. Story plot generation based on CBR (Gervás et al., 2005), VBPOCL (Riedl and Sugandh, 2008) are also used case-based reasoning to create stories for a closed domain. However, since case-based reasoning depends on the previous tasks when a new task appears, when those models are applied to new domains, they have no way of dealing with the new domains, thus problem-solving fails. Because of that those models couldn't generate stories on an open domain they were bound to the domains that the model has seen before. FABULIST (Riedl and Young, 2010) used same technique with analogical mapping and expanded the search space of planners to account for coherence and character believability to create more creative stories, although they managed to come close creating open domain stories, the model wasn't coherent.

2.6.5 Deep Learning Approaches

Some of the models mentioned above gained noteworthy results however almost all approaches were restricted to limited domains till deep learning approaches. One of the most important reason behind that is the deep learning models generalization power. They are very capable of capturing the semantics and grammar of the text in some domains and can adopt themselves into other domains. Peng, Ghazvininejad, May and Knight (2018) extracted keywords from the ground truth stories via RAKE algorithm (Rose, Engel, Cramer, & Cowley, 2010) then provided those keywords to a bidirectional LSTM in order to generate stories based on those keywords. Those keywords provided to the model in order to increase the controllability of the model and reached more coherent results than non-controlled models. Another approach by Jain et al. (2017) used GRU specialization of RNN for generating stories. The model's input was descriptions of a story and the output was the story generated. Martin et al. (2018) focused on the events and used reinforcement learning in order to determine

what action to take next when generating a story. There were two seq2seq networks built, first one is a LSTM network with attention (Sutskever et al., 2014) which is called event2event. Purpose of that network is trying to determine which event can lead to the best story and after that event2sentence translates those successful events back to sentences which is also LSTM network with attention. However, the results of the model were not coherent, and it lost the course of the story very easily. In order to solve those problems Xu et al. (2018) proposed a generative adversarial network. The discriminator of the network is a seq2seq network which consists of hierarchical encoder and an attention-based decoder, and the generator is a seq2seq LSTM network. Since text is discrete data the backpropagation of the network with traditional methods failed so reinforce algorithm was implemented. They used policy gradient as optimizer and monte-carlo search in order to solve the derivation problem. Model overcame the problem of losing course but still wasn't coherent and failed in human evaluation. In another approach Fan et al., (2018) used gated convolutional neural networks (Dauphin, Fan, Auli & Grangier, 2017) and for generating prompts convolutional sequence to sequence model (Gehring et al., 2017) is used with both the pre-trained gated convolutional neural networks in a fusion model architecture (Sriram, Jun, Satheesh, & Coates, 2018) that enables the generator to access the hidden states of the pre-trained model.

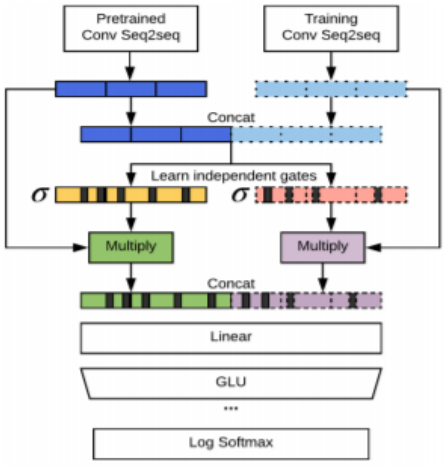


Figure 2.6- 1 Fusion model, taken from (Fan et al., 2018)

The first model is trained with the prompts only and the researchers tried to give the model ability to generate a sketch of the original prompt. The gated convolutional model builds a hierarchical representation of the input and that made capturing the

long-term dependencies easier, similar to the tree-structured analysis of linguistic grammar formalisms (Dauphin et al., 2017). Then for training the fusion model, they combined the pre-trained gated convolutional model with convolutional sequence to sequence model (Gehring et al., 2017) for generating the actual story because they are well suited for long text generation because they allow parallelism of computations within the sequence (Fan et al., 2018). Since the fusion model allows the convolutional seq2seq model to access the hidden states of the first pre-trained model the story is grounded that means it doesn't lose its original course which is the prompt of the story. The model reached significant results on terms of both perplexity and human evaluation. However, the prompts were given to the model by another model, so the model is controlled and because of the convolutional networks ability to focus model tends to focus more than necessary in some parts which resulted in repetition of words. In addition to that, the results of the model were coherent however lacked creativity they focussed on one aspect in the prompt and kept generating text according to it.

2.7 SUMMARY

2.7.1 Overview

As it can be inferenced from this section there have been many approaches proposed for NLG and almost every one of them used in automated story generation in order to create narratives which are human-like. Approaches before deep learning created significant results however they were bound to one domain or several but neither of them could propose a method for open story generation. With the rise in deep learning approaches several approaches reached tremendous results in open domain generation however there are still restrictions about what they can achieve. As a result, a model which can be considered new, the transformer networks applied to story generation in order to resolve those restrictions.

2.7.2 Gaps in the Research

More researches began to investigate this area with recent advances in deep learning and improved the quality of generated stories according to the human evaluations and dramatic increases have been seen on perplexity scores as seen in the model of Fan, Lewis and Dauphin (2018). However, because of unnecessary attention provided by

convolutional networks the model attended some parts of the prompts more than others. This resulted in repetitions of words and stories were stuck on some aspects of the prompt didn't have a clear sense of progression which was a problem for a coherent story. Transformer networks (Vaswani et al., 2017) provided optimum amount of attention to the right parts of the text and reached state-of-the-art results in many tasks in NLG and to my knowledge they haven't been used for story generation field yet. They have the potential of providing the attention the story generation needs, in addition to that the model proposed by Fan et al., (2018) uses a pre-trained network in order to create stories which are not independent from the prompt. Transformer networks can also provide stories which are linked to their prompt with their attention mechanism.

2.7.3 Research Question

Those evidences lead to a theory that, convolutional seq2seq models combined with attention can reach to significant results for automated story generation so transformer networks which are solely based on attention can improve those results in that field.

"Can a transformer network used on Reddit's WRITINGPROMTS forum dataset achieve statistically significant improvement on the perplexity score and human evaluation metrics prompt matching and blind model comparison test, presented in the paper of Fan et al., (2018), for automated story generation when a prompt is given, over the state-of-the-art Hierarchical Neural Story Generation model (Fan et al., 2018)?"

3. DESIGN & METHODOLOGY

This chapter describes the experiment undertaken in order to determine whether the null hypothesis can be rejected or not. For the experiment transformer network is designed for decreasing the perplexity and having more human annotators chosen the transformer network in terms of story quality and, more correct pairing of prompts and stories made by human annotators then the compared Hierarchical Neural Story Generation model (Fan et al., 2018).

The data collection and understanding step followed by its preparation for transformer network is described. Before the experiment, the details of the model are described in detail and the experiment process is examined.

The evaluation metrics, both perplexity and human evaluations described in detail including how they are used to compare the two models and how the human evaluation results can be specified statistically in order to accept or reject the null hypothesis is defined.

3.1 HYPOTHESIS

$H_{1:}$ If a transformer network (Vaswani et al., 2017) is designed and applied to the Reddit's WRITINGPROMTS forum dataset for automated story generation, it can statistically significantly outperform the Hierarchical Neural Story Generation model (Fan et al., 2018) on perplexity and two human evaluation metrics prompt pairing and blind model comparison.

H_0: If a transformer network (Vaswani et al., 2017) is designed and applied to the Reddit's WRITINGPROMTS forum dataset for automated story generation, it cannot statistically significantly outperform the Hierarchical Neural Story Generation model (Fan et al., 2018) on perplexity and two human evaluation metrics prompt pairing and blind model comparison.

3.2 DATA

3.2.1 Data Collection and Understanding

The dataset used for this investigation is the forum Reddit's WRITING PROMPTS dataset which is available online and gathered from the link provided by Ott et al., (2019). Reddit is an online forum where people create subreddits in order to discuss some matters, presenting the latest news, socializing and so on. "WRITING PROMPTS" is also a subreddit where administrators of the forum create prompts such as topics which can be considered as descriptions of a story or poems and users write stories or poems corresponding to that prompt. Then respected administrators rank those stories and the best topic is presented at the top of the forum which can be looked at as the best story for its corresponding prompt. The dataset contains the prompts and only best stories written in order to provide a certain quality. Main constraints about writings are, the proposed story should have more than 30 words, plagiarism results in a ban so all stories are unique and off-topic writing is not allowed. All stories checked regularly by the administrators thus, consistency among prompts and stories ensured. Also, prompts have a large diversity of topic, length, and detail (Fan et al., 2018). The average length for the prompts and stories 28.4 and 734.5 and the maximum lengths measured as 71 and 3784 respectively. As seen in histogram charts of the Figure 3.2.1-1 and Figure 3.2.1-2 the length of the sentences is quite diverse for both stories and prompts. The minimum length of a sentence in a story is 1 because there are questions such as "How?" or answers such as "yes" present in the corpus and the maximum length is measured as 38. The minimum and maximum length for a sentence in prompts is 3 and 26 respectively. Also, there is a prompt restriction which is six words referenced from the famous six-word stories concept so the minimum prompt can be six words which then should be expanded to a story by the writers. An example of a prompt and corresponding story is shown in Figure 3.2.1-3 and Figure 3.2.1-4 respectively.

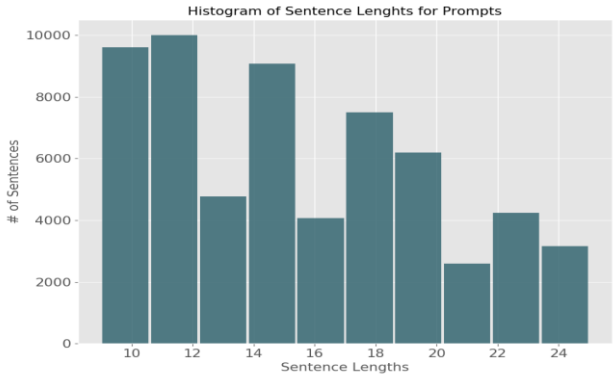


Figure 3.2.1-1 Histogram for sentence length of the prompts

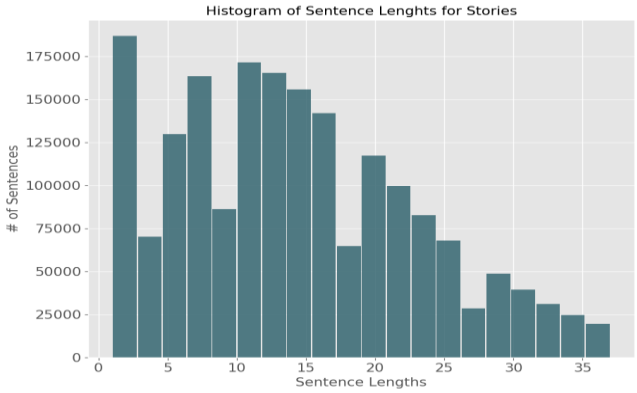


Figure 3.2.1-2 Histogram for sentence length of the stories

```
You are at the park with your kids , when you see the telltale signs of a
lightning strike . You divert your kids from danger , but are hit by lightning .
Soon after , you discover that your Dad Senses have increased 100 fold .
```

Figure 3.2.1- 3 A sample prompt from the dataset

```
“ Sadie ! I told you not to stand under the tree in the middle of a thunderstorm ! ”
<newline> <newline> My mom frantically dialed 911 , as I laid there , shocked .
Literally . <newline> <newline> “ Hello ? ! My daughter was just struck- “ <newline>
<newline> All of a sudden , something was wrong . I sat up , my fingertips crackling
with this new power . Or was it the aftermath of the lighting ? I couldn ’ t tell . Nor
did I care . No . Something was definitely wrong , and it was nearby . <newline>
<newline> “ If I take 295 south right now , I can get to Home Depot in ten minutes flat
. And then I can finally build that deck you ’ ve wanted , they have the planks on sale
. I can feel it in my blood . Now , I know , they ’ re plum colored and you did
specifically say you wanted pine , but hear me out here . Plum is the pinest color out
there , and you ’ d be a birch to say otherwise. ” <newline> <newline> My mom looked
back at me , a hint of recognition in her eyes , but the rest of her face read as one
emotion : terror . <newline> <newline> “ You know what else ? I ’ m gon na get like ,
50 windows from craigslist , they have the best deals , and make a greenhouse ! ”
<newline> <newline> Somehow , the words kept spilling out of my mouth . But something
else was off here . Something with my house… <newline> <newline> Suddenly , I knew . I
felt my blood boiling as I continued to be filled with rage . I jumped up the rest of
the way , and dashed the two blocks to my house , where my brother ’ s laptop was .
<newline> <newline> “ Not . Home . And his computer is plugged in . ENERGY WASTER !
Doesn ’ t he care about this house ? ! He could ’ ve burned the whole place down ! ”
```

Figure 3.2.1- 4 Corresponding story of the sample prompt

3.2.2 Data Preparation

The acquired dataset prepared for satisfying the limitations and pre-processed for the model. Initial dataset was already randomly divided into training, validation and test sets which are %90, %5 and %5 respectively. Training set contains 272.600, validation set 15.620 and test set 15.138 instances for both prompts and stories. The model presented by Fan et al., (2018) limited the story length to 1000 words and did their experiments on 8 GPUs however, because of the resource limitations the length of the stories for this investigation was limited to 500 words per story and the training instances were limited to 50.000. Since the training set is limited to 50.000, and there was no hyperparameter tuning applied with the validation, the percentage of the test set increased to %23. Prompts contain information about which writing mode is dedicated for that blog, such as "[WP]" which corresponds to writing prompts, removed from the prompts. All the transformer models in the NLP field used byte pair encoding over character encodings. There are two reasons for that, first one is it was empirically proven that byte-pair encoding increases the performance of the models and second since models need to capture every character rather than words and then predict them one by one, models that used the character encodings are computationally expensive than other models. For byte-pair encoding the target size chosen to be 32.000 and the words in the dataset tokenized according to it. The dimension of the tokenized training dataset is 50.000 rows and 71 columns for prompts (inputs) and 50.000 rows and 624 columns for the stories (outputs) which is shown in Figure 3.2.2-1. After that in order to give the model the ability to learn the start and end of the prompts and stories, start and end tokens added to the beginning and endings of the sequences.

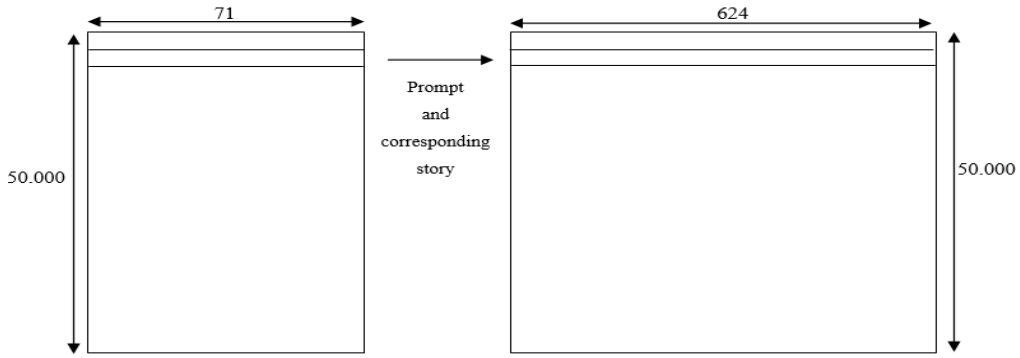


Figure 3.2.2-1 Tokenized input and output datasets of model, prompts (left) and stories (right)

3.3 TRANSFORMER NETWORK DESIGN

In order to design the transformer network, there are various hyperparameters that needs to be determined. However, one model for hyperparameter tuning took 15 days, thus it was not feasible to do hyperparameter selection because of the time interval allocated for this thesis. Instead of randomly choosing the hyperparameters, they are taken from the original paper of transformers (Vaswani et al., 2017) which proved its use for many tasks with those hyperparameters which are defined in the next section.

3.3.1 Hyperparameters of Transformer Network

- **Number of layers:** The layers of the stacked encoders and decoders. Encoders function is to map the input and the attention information into vector representations. On the other hand, decoders take that information to turn that representation into output text. There is one more attention mechanism that the decoder has which makes the decoder attend the previously decoded words in addition to the attention for input words.

- **Number of heads:** This hyperparameter is designed in order to determine the number of scaled dot-product attention heads used inside multi-headed attention. The h represented in the Figure 3.3-1 below shows number of heads.

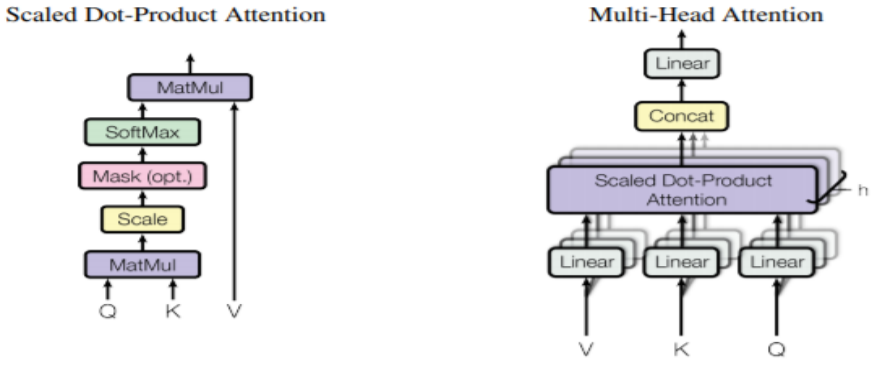


Figure 3.3.1-1 Scaled dot-product attention (left) and multi-head attention, taken from Vaswani et al., (2017)

- **Dimension of the model:** Determines the input and output dimensions of the feed-forward network inside the encoders and decoders and eventually the output of encoders and decoders since the last layer inside them is a feed

forward network. In addition to that the input and output tokens are converted to vectors and its length is determined by this hyperparameter.

- **Dimension of the feed forward network (FFN):** The dimension of the hidden layer inside the fully connected feed forward network which can also be understood as two convolutions with kernel size 1. More dimensions mean more functions will be computed inside the hidden layer which can increase the performance however that might result in overfitting.

- **Dropout rate:** The dropout rate eliminates some neurons from the network when training in order to prevent overfitting. In transformers it is implemented for the feed forward networks inside the encoder and decoders.

- **Activation function:** A function that decides if the information of a particular neuron or a function should be passed to the next layer in the network or not. It is used inside the feed-forward networks. There are many activation functions proposed for many different tasks in deep learning models, some of which are linear function, sigmoid function, hyperbolic tangent function, rectified linear unit (RELU) and leaky RELU.

- **Loss function:** Also called as cost function, in training time it determines whether the outputs of the model are correct or not in order to calculate the error rate and pass it back to the model by backpropagation. There are various loss functions for different purposes. Some of them are mean squared error(MSE) and cross-entropy.

- **Optimizer:** An algorithm, which is responsible for making the loss function converge to its optimum. Some examples are stochastic gradient descent, Adagrad, Adadelta and Adam optimizer.

- **Number of epochs**: The number of times allowed for the network to see the dataset.

- **Batch size:** Determines the number of instances passed through the model before the backpropagation for each epoch. Small batch sizes lead to more backpropagation, so the loss function can converge faster however it gets computationally expensive and big batch size may lead to bad performance.

- **Top k sampling:** This is a method for the generation step. When generating text word by word it allows the model to randomly choose between most k probable words for that position.

3.3.2 Hyperparameters of the Model

The hyperparameters defined for the original transformer model (Vaswani et al., 2017) is selected for this model. The number of layers for both encoder and decoder defined as 6 which are identical to one another. The number of scaled-dot attention inside the multi-headed attention is 8 which means there are 48 scaled dot-attentions present in the encoder and 96 in the decoder. The dimension of the network is 512 thus, when the input embeddings converted to vectors their length will be 512 and the input and output dimension is also 512 for the feed forward networks which has 2048 neurons inside its one hidden layer. Also, the dropout rate and activation function chosen for those feed forward networks is 0,1 and RELU respectively. Since the byte-pair encoding assigns integers to words with 32.000 chosen as the target vocabulary there are around 32.000 different tokens in the dataset because of that reason sparse categorical cross-entropy is chosen to be the loss function of the model. In addition to that Adam optimizer is used with β1 = 0.9, β2 = 0.98 and Ɛ = 10−9. Adam optimizer chosen because it aids to a faster convergence of the loss function and computationally cheap. The learning rate of the optimizer is chosen to be a custom learning rate as described in the paper. The formula for the learning rate shown in the following formula and learning rates in the Figure 3.3.1-1 where warmup steps are the number of training steps from the beginning of the training till a specified training step which chosen to be 4000.

$$lrate = d_{model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

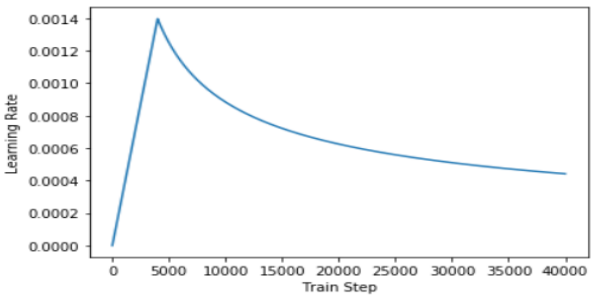


Figure 3.3.2- 1 Change in the learning rate

The number of epochs and batch size were unfortunately limited because of the time and resource limitations. The number of epochs chosen to be 10 and batch size 64 because with bigger batch sizes the model got memory error even with 400 GB RAM. A table of the hyperparameters implemented to the model is shown below.

Number of layers	6
Number of heads	8
Dimension of the model	512
Dimension of the FFN	2048
Dropout rate	0.1
Activation function	RELU
Loss function	Sparse categorical loss function
Optimizer	Adam
Epochs	10
Batch size	64
Top k sampling	3

Table 3.3.2-1 Hyperparameters of the model

3.4 EVALUATION OF RESULTS

3.4.1 Perplexity

"Perplexity is commonly used to evaluate the quality of language models, and it reflects how fluently the model can produce the correct next word given the preceding words" (Fan et al., 2018). The perplexity can be formulized as 2 to the power of cross-entropy which is a measure for calculating the differences of two distribution functions (Chen, Kar, & Ralescu, 2012).

$$\text{Perplexity} = 2^{\textit{cross-entropy}}$$

The perplexity for the proposed model calculated by measuring the cross-entropy of test instances one by one then taking the average of them and calculating the power of two of the cross entropy. Lower perplexity means better model.

3.4.2 Human Evaluation Methods

There are two human evaluations designed for this model. First one is prompt pairing where stories and their corresponding prompts are presented to 15 human annotators and asked to match them. This is done in order to assess if the story generated according to the prompt is related to it or not. After that process is done the accuracy of the annotators are calculated for both models. In the original state-of-the-art model (Fan et al., 2018) this is done with more than two models however for this thesis only two models are compared so an option given to the annotators which is "not related to the prompt". Thus, when two prompts and stories are given to the annotator if a story found to be unrelated, he or she can choose "unrelated" option which result as negative to the model of that story. This evaluation is done by taking 100 prompts and corresponding stories from both models and then they are shuffled and presented to the annotators simultaneously.

Second evaluation is a blind test where the prompts of the stories are hidden from the annotators. The number of annotators used for this task is 5 and they are asked to assess the qualities of the stories. If the story found to be good, it is considered as positive and negative otherwise.

Both human evaluations then gone through a Free-Marginal Fleiss's Kappa test (Randolph, 2005) which is a variation of original Fleiss's Kappa test for assessing the annotators reliability statistically (Fleiss, 1971). This test is chosen because of two reasons. First reason, this is a test for multi-rater reliability, other tests such as Cohen's Kappa (Cohen, 1960) measures the reliability of just two annotators. Second reason, original Fleiss's Kappa test assumes that there is a predetermined amount of cases in each category however the human evaluations in this thesis don't give the annotators such limits for categories. As shown in the formula below, K_{free} value is the overall agreement between annotators and is the expected agreement.

$$K_{free} = \frac{\sum_{i=1}^{M}\left(\frac{1}{N \times A}\sum_{c=1}^{N} a_{ic}\right)^2 - \frac{1}{k}}{\frac{1}{k}}$$

In order to calculate P_o, the proportion of overall observed agreement calculated according to the formula below where M corresponds to the number of classes which are positive and negative in this case, N is the number of annotators, A is the total number of tests presented to one annotator and a_{ic} is i^{th} annotators total annotations made for c^{th} class.

$$P_o = \sum_{i=1}^{M} \left(\frac{1}{N \times A} \sum_{c=1}^{N} a_{ic} \right)^2$$

Since there is no lower limit for the number of times of choosing a category which can also be expressed as free marginals, the expected agreement is assumed as all categories in the dimension have same probability of being chosen by the annotators therefore the expected aggrement formulized as follows:

$$P_e = \frac{1}{k}$$

3.4.3 Hypothesis Testing

For determining whether the transformer networks outperformed the state-of-the-art model presented by Fan et al., (2018) series of tests conducted. To compare the transformer model's results with the state-of-the-art model the prompts and stories generated by that model gathered from the website[4] provided by Ott et al., (2019).

The dataset encoded by the byte pair encoding in order to get the input embeddings which then passed into the designed transformer model. The model trained with the hyperparameters stated in the Section 3.3.1 and the results for both perplexity and two human evaluation methods are gathered.

For comparing the perplexities of two models the simplest of cross-validations the hold-out method is used. Since to train the transformer with 50.000 instances took 15 days for 10 epochs with batch size 64 and because of the time restrictions for allocated for the thesis hold-out method is chosen. As specified in the Section 3.2.2 the dataset is divided in to %77 (50.000) training, %23 (15.138) test set respectively.

[4] https://github.com/pytorch/fairseq/tree/master/examples/stories

There were 15 annotators chosen for pairing the prompts with the stories. 100 prompts and their corresponding stories were chosen randomly and presented to the annotators. They matched the prompts with the stories and if they think the prompt is not related with the story, they chose “not related” option. For both models the matched and unmatched answers are gathered and Free-Marginal Fleiss’s Kappa used twice, one for each model to assess the annotators results reliability.

For the second human evaluation, annotators are asked to choose the better story from two stories that are presented to them. There were 5 annotators and 100 stories used for this method and one Free-Marginal Fleiss’s Kappa score calculated to measure the reliability of the annotators.

After gathering the perplexity and human evaluation results for both models they are compared in order to accept or reject the null hypothesis.

3.5 SUMMARY

For conducting the experiment, first the dataset was gathered from the website described in Ott el al. (2019) and it was pre-processed in order to convert it to the embeddings that the model required. After that the transformer model was designed with the hyperparameters described in the original paper of transformer networks (Vaswani et al., 2017). The state-of-the-art model (Fan et al., 2018) and the designed transformer network trained and evaluated with the same training and testing dataset in order to compare them through perplexity and human evaluation metrics described in Section 3.4 to test the hypothesis.

3.5.1 Strengths

- **Easy to comprehend:** Transformer networks focus solely on the attention mechanism and exclude any recurrence and convolutional structures. They neither have any time step nor confusing calculations such as LSTM networks has or convolutional features that gets hard to understand when networks get deeper thus, they are much easier to comprehend.

- **Faster training times compared to previous approaches:** Since there is no time step, the calculations in the attention layers doesn’t depend on the

previous time steps, all can be done simultaneously in parallel which makes the network faster than many previous approaches done on this field.

- **Diversity of topics:** The topics (prompts) in the dataset is examined and found to be quite diverse as stated in the paper by Fan et al., (2018)

3.5.2 Weaknesses

- **Reproducibility:** In order to reproduce the same design, one must have the tokenized dictionary. Since there is a stochastic aspect in tokenization every tokenization can result in different set of tokens.

- **Bias from tokenization:** Since the tokenization is done with whole dataset in order to avoid <unknown> tokens, tokenization process includes bias into the process.

- **Long training times for long text:** The speed of the transformer networks is proven for tasks such as machine translation and text summarization however for text generation it takes longer times than expected. The reason can be understood especially when compared to text summarization. Text summarization can also contain long texts however the input is long not the output. Thus, because of the parallelization in the encoder architecture of transformers the input text can be processed fast however for the decoder the network generates texts one by one so there is no parallelization there but since the summaries of the texts rather short this can be done fast. However, for story generation it is the opposite the prompts are short, and the outputs are long which takes transformer train times much higher than expected.

4. RESULTS, EVALUATION & DISCUSSION

The experiment process is examined in this chapter as well as the results of the evaluation metrics described in the Section 3.4.2, then they are compared in order to test the null hypothesis. The strengths and weaknesses according to the findings when conducting the design, experiment and evaluation is discussed along with the changes and enhancements that can be implemented into the design examined.

4.1 EXPERIMENT RESULTS

In this section, the perplexity and human evaluation results measured from the experiment, is demonstrated in three sections. First section is dedicated to the transformer network, the second section is for the state-of-the-art model from Fan et al., (2018) and the last section dedicated to Free-Marginal Fleiss's Kappa scores from all human evaluations in order determine the reliability of the annotators.

4.1.1 Transformer Network

When the network is generating text on unseen data which is the test data in this case, the cross-entropy scores are tracked, and the perplexity of the transformer network calculated with the cross-entropy scores as discussed in the Section 3.4.1. The perplexity of the transformer network is measured 67,48.

The pairing task involves the prompts and their corresponding stories. The test is done with randomly chosen 100 instances from the test set and given to 15 annotators.

Annotators	Positive	Negative
Annotator 1	4	96
Annotator 2	7	93
Annotator 3	3	97
Annotator 4	3	97
Annotator 5	5	95
Annotator 6	4	96
Annotator 7	6	94
Annotator 8	4	96
Annotator 9	2	98
Annotator 10	4	96
Annotator 11	6	94
Annotator 12	4	96
Annotator 13	4	96
Annotator 14	5	95
Annotator 15	4	96

Table 4.1.1-1 Summary of pairing results for transformer network

As shown in the Table 4.1.1-1 above positive means annotators were able to match the correct prompt with the story, negative means they weren't able to match or chosen the option of "not related". Annotators were able to choose the correct prompt with the story with 4.3% accuracy.

Second human evaluation is a blind test which is, giving the annotators two stories, one created by transformers and the other by the state-of-the-art model (Fan et al., 2018). 100 randomly chosen stories were given to 5 annotators without their prompts in order for them to choose which one is better. The results for the blind test are shown in the Table 4.1.1-2 below. Annotators preferred the stories generated by the transformer network with 23.8%. The table only shows the numbers that chosen this model, a table for both models is shown in the Section 4.2.

Annotators	TRNET (from 100 stories)
Annotator 1	25
Annotator 2	24
Annotator 3	23
Annotator 4	25
Annotator 5	22

Table 4.1.1- 2 Summary of blind test results that were in favour of transformer network

A sample prompt and its corresponding story generated by transformer network shown below in the Figure 4.1.1-1.

```
[ WP ] It is believed that the cure for
cancer has been discovered on Mars ,
astronauts have only one chance to bring
it home where they hope to duplicate its
properties for mass distribution
```

```
I do not know what I do for this place . <> I do know how I got here .
I ' m not like I have to say what it was to me to think about how to
get to the world . I don ' t, in my opinion , but I 've got to go on
to the right side and go on to the side . I have to do this , I dont
have a chance for it . I dont have to be a part of the job , I dont
have to go for the money , but I have to do it for the money . I
have n'T got a job . But when I get the job , I get a job for my life .
<newline>. I ' ll
```

Figure 4.1.1-1 A sample prompt (top) and its story generated by transformer network (bottom)

4.1.2 Hierarchical Neural Story Generation Model

Since both models are tested on the same dataset the perplexity result of this model is taken from the paper of it (Fan et al., 2018) which is 36,56. The generated stories and their corresponding prompts by the model (fusion model) gathered as described in the Section 3.4.2 in order to conduct the human evaluation.

The pairing task again used the same 15 annotators and randomly chosen 100 stories and prompts. As mentioned earlier both the transformer network's and fusion model's generated stories and corresponding prompts were given to the annotators simultaneously in order to do the pairing task however in the Table 4.1.2-1 below only the fusion models results are shown. Positive means correct matches, while negative is the count of wrong or "not related" matches. Annotators were able to choose the correct prompt 70.67% of the time for the fusion model.

Annotators	Positive	Negative
Annotator 1	64	36
Annotator 2	69	31
Annotator 3	74	26
Annotator 4	62	38
Annotator 5	75	25
Annotator 6	76	24
Annotator 7	68	32
Annotator 8	78	22
Annotator 9	66	34
Annotator 10	72	28
Annotator 11	64	36
Annotator 12	68	32
Annotator 13	72	28
Annotator 14	77	23
Annotator 15	75	25

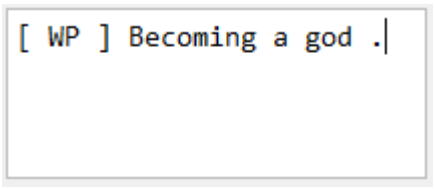
Table 4.1.2-1 Summary of pairing results for the state-of-the-art model

The second human evaluation is a blind test. Same 5 annotators and 100 stories randomly taken from the generated stories of the fusion model. This is a task for deciding which model generates more human like results and only the number of stories generated by the fusion model and preferred by the annotators are displayed in the Table 4.1.2-1 below.

Annotators	Fusion Model (from 100 stories)
Annotator 1	75
Annotator 2	76
Annotator 3	77
Annotator 4	75
Annotator 5	78

Table 4.1.2-2 Summary of pairing results for the fusion model

Annotators preferred the stories generated by the fusion model 76.20% of the time. Also, an example of a prompt and its corresponding story generated by the fusion model is shown in the Figure 4.1.2-1 below.



`` God , it 's been a long time since I 've been a god . '' <newline> <newline> I stood up , and made my way up the stairs . My hands , covered in sweat , were shaking . I could hear the crowd outside as I reached a large , wooden door at the back of the house . It opened , and a small white man stepped through . <newline> <newline> `` Hello , I 'm Lucifer , '' the man said , `` and you are the first God of my kind . '' <newline> <newline> `` I am , '' I said , my head still spinning . This was my chance . `` I am , '' I repeated myself . <newline> <newline> `` I 'm God . I am God . '' He paused for a moment . `` You are ,

Figure 4.1.2-1 A sample prompt (top) and its story generated by fusion model (bottom)

4.1.3 Reliability of Human Annotators

Free-Marginal Fleiss's Kappa score calculated three times. Firstly, for the pairing task of the transformer network, secondly again for the pairing task but this time for the fusion model and thirdly for the blind comparison. Figure 4.1.3-1 shows summary of all human evaluations. Original results of the human evaluations can be found in Appendix A.

Annotators	TRNET		Fusion Model	
	Positive	Negative	Positive	Negative
Annotator 1	4	96	64	36
Annotator 2	7	93	69	31
Annotator 3	3	97	74	26
Annotator 4	3	97	62	38
Annotator 5	5	95	75	25
Annotator 6	4	96	76	24
Annotator 7	6	94	68	32
Annotator 8	4	96	78	22
Annotator 9	2	98	66	34
Annotator 10	4	96	72	28
Annotator 11	6	94	64	36
Annotator 12	4	96	68	32
Annotator 13	4	96	72	28
Annotator 14	5	95	77	23
Annotator 15	4	96	75	25

Annotators	TRNET	Fusion Model
Annotator 1	25	75
Annotator 2	24	76
Annotator 3	23	77
Annotator 4	25	75
Annotator 5	22	78

Figure 4.1.3-1 Summary of Prompt pairing results (left) and blind comparison (right)

4.2 EVALUATION

In order to evaluate the model, three evaluation metrics were taken into consideration and the details are explained in this section.

First evaluation metric examined is perplexity. Because of the time limitations the transformer network was trained on 50.000 data instances which took 15 days thus, only one perplexity score gathered from the whole instances of the test set which is 15.138 prompts and corresponding stories. The perplexity of the transformer network and fusion model (Fan et al., 2018) is 67,48 and 36,56 respectively. Since no metrics such as k-fold cross validation for gathering more than one perplexity score, the statistical significance of the perplexity score is an issue and in terms of perplexity there is no evidence found to reject the null hypothesis.

Models	# of Parameters (mil)	Perplexity score
Transformer network	93.3	67,48
Fusion model (Fan et al., 2018)	255.4	36,56

Figure 4.2- 1 Perplexity score of the models

The second and third metrics are human evaluation metrics which are prompt pairing and blind comparison test of generated stories. In order to interpret the Free-Marginal Fleiss's Kappa measure for both human evaluations, the guideline determined by Richard and Koch (1977) was used. For the prompt pairing task, the percentage of pairing accuracy for the transformer network is measured as 4.33%. The Free-Marginal Fleiss's Kappa score for the prompt pairing task related to the transformer networks calculated as 0.86±0.04 with %95 confidence interval. There was a significant perfect reliability found between annotators about the prompt pairing task for the transformer network. The prompt pairing accuracy for the fusion model found as 70.67% and the Free-Marginal Fleiss's Kappa score for fusion networks measured as 0.45±0.05 with %95 confidence interval. There was a significant moderate reliability found among annotators for the prompt pairing task related to the fusion model.

Models	Accuracy of Prompt pairing task	Free-Marginal Fleiss's Kappa
Transformer network	4.33%	0.86±0.04
Fusion model (Fan et al., 2018)	70.67%	0.62±0.04

Figure 4.2- 2 Accuracy of prompt pairing task and kappa scores

The third metric is also a human evaluation task which is a blind comparison task given to the annotators for them to prefer a story between two stories. For this task the annotators chosen the fusion model over the transformer network with 76.20% and the Free-Marginal Fleiss's Kappa score measured as 0.62±0.04 with %95 confidence interval. There was a significant substantial reliability found among annotators for the blind comparison test. Results of the both metrics shown that there is no statistically significant evidence found to reject the null hypothesis.

Models	Blind Comparison Task	Free-Marginal Fleiss's Kappa
Transformer network	23.80%	0.62±0.04
Fusion model (Fan et al., 2018)	76.20%	

Figure 4.2- 3 Blind comparison task and kappa score

4.3 DISCUSSION

This section is dedicated to the strengths and limitations of the proposed model.

4.3.1 Strengths

- **Competitive results with simpler network:** The fusion mechanism has one pre-trained convseq2seq network and another convseq2seq with attention and gating mechanism implemented in it, however even with 22% training data that fusion network trained with transformer networks generated results that preferred by annotators in every 1 out of 4 stories.

- **No bias in reliability of the annotators:** The popular technique Fixed-Marginal Fleiss's Kappa (Fleiss, 1971) is shown to be influenced by prevalence and bias which can lead to high agreement but low kappa among annotators (Randolph, 2005). On the other hand, the Free-Marginal Fleiss's Kappa used in this thesis resolve that problem so there is no bias in reliability of agreements among annotators.

- **Generated novel stories:** The transformer network generated novel stories without copying the stories from the original dataset.

4.3.2 Weaknesses

- **High memory usage:** The network has 93.3 million trainable parameters which is almost %30 of the fusion model. However even with 93.3 million parameters the attention that needed to be kept in RAM during training is substantially high. In addition to that, there is no parallelization in the decoder layer which is the part that generates the whole story, which increases the training time and memory usage enormous amounts. The RAM needed to train the dataset was 400GB.

- **Repetition:** The seq2seq network architecture of the transformer network resulted in repetition of words which also examined in some previous research for LSTM seq2seq on story generation (Jain et al., 2017). Even the transformer architecture which based on solely powerful attention mechanism which has no recurrence in it endured repetition, so the problem seems to be the seq2seq architecture, not recurrence.

- **Grammatical errors:** When generating stories in several occasions the model generated wrong grammar especially for abbreviations. An example can be given from the Figure 4.1.1-1 “have n’T”. There is an unnecessary space between have and the abbreviation also capital “t” is used.

- **Better automated evaluation metric needed for story generation:** Even the perplexity score of the transformer model is almost two times high or in other words the difference between the perplexity of two models is 30.92, the 1 out of 4 stories generated by transformer networks are chosen by the annotators as better story.

- **Stories are not related to prompts:** The seq2seq network architecture of the model resulted in having stories that lose its course which brought the problem of generating stories that are unrelated to their corresponding prompts as examined in the first human evaluation task.

- **No sense of progression in the stories:** Stories tend to keep telling the same events no sense of progression found in the generated stories.

- **Inefficient generation time:** The time needed for generating a story that consists of 150 words with a prompt that has 23 word in it measured as 43 seconds on an average CPU which makes the network inefficient for real time generation.

- **No statistical comparison for perplexity:** Because of the training time took to train one model there was no time for validation methods such as cross-validation to make a statistical comparison of perplexity results.

5. CONCLUSION

5.1 RESEARCH OVERVIEW

The main objective of this research is to investigate the automated story generation field and propose a new model that can increase the automated generation capabilities of machines. In order to reach that goal, a transformer network (Vaswani et al., 2017) was designed and the performance of it compared with the state-of-the-art automated story generator (Fan et al., 2018) on the same evaluation metrics.

5.2 PROBLEM DEFINITION

Many approaches taken in order to solve the automated story generation over the years as reviewed comprehensively in Section 2.6. Early approaches tried to solve the problem on restricted domains and some of the investigations got good results. However, those methods were relied heavily on human interaction and since they were bound to the domains that specified by their developers and lacked creativity. Deep learning approaches brought open domain solutions to the field, but they also introduced several other problems some of which is the coherency, repetitive generations and keep generating on the same event with no progression so it is still an open field. In order to solve those issues a transformer network is designed, evaluated and tested against the hypothesis in Section 3 and Section 4 respectively.

5.3 DESIGN/EXPERIMENTATION, EVALUATION, RESULTS

The data was gathered and in order to have a clear understanding some features of it examined. Because of the time limitations allocated to this thesis the data was cropped into a feasible dataset. The network was designed according to the hyperparameters specified by Vaswani et al., (2017) and the experiment conducted, and the strengths and weaknesses of the design was examined.

After the training step the transformer network, the quality and performance of the generations done by the proposed transformer network on the test dataset was measured with perplexity and human evaluations as explained in Section 3 then the

reliability of the human evaluation results examined in order to assess their statistical significance.

The comparison of the perplexity and human evaluation results between the transformer network and the state-of-the-art model proposed by Fan et al., (2018) reported and the hypothesis presented in Section 3.1 is tested, in addition to that, the strengths and weaknesses of the experiment is investigated. There was no statistically significant evidence found to reject the null hypothesis.

5.4 CONTRIBUTIONS & IMPACT

This investigation shows that even one of the most powerful models in the field of NLG which is solely attention-based transformer network cannot generate stories dependent on a prompt and solve the repeatability problem because of its seq2seq architecture. Still the most viable approach is to take the prompts written by a human, turn them into sketches and use them as a prompt (Fan et al., 2018).

Usually among the NLP field an increase by 10 on perplexity considered as a dramatical increase, however even there was substantial differences between the two models perplexity scores, 1 out of 4 stories chosen by the annotators are generated by transformer networks which indicates there should be a better automated metric in place in order to assess the performance of text generations especially tasks that require a certain level of creativity such as story or poem generation.

Although, transformer networks allow parallelization in the encoder, for long text generation transformer networks take considerable amount of time to generate stories because of the one by one generation mechanism at the decoder.

5.5 FUTURE WORK & RECOMMENDATIONS

- In order to pay more attention to the prompts the encoder architecture of the network can be changed to bidirectional encoder (Devlin, Chang, Lee, & Toutanova, 2019) in order to pay more attention to the prompt.

- An implementation of generative adversarial networks (Goodfellow et al., 2014) with bidirectional transformer encoder as discriminator and the decoder

of the transformer network as generator with REINFORCE algorithm has the potential of reaching more creative, coherent and human like results when paying attention to the prompt.

BIBLIOGRAPHY

Kumbhar, O., & Kunjir, A. (2017). A Survey on Optical Handwriting Recognition System Using Machine Learning Algorithms. International Journal of Computer Applications, 175(5), 28–31. doi: 10.5120/ijca2017915539

Aizenberg, I. N., Aizenberg, N. N., & Vandewalle, J. P. L. (2000). *Multi-valued and Universal Binary Neurons: theory, learning and applications*. London, NY: Springer.

Allen, J. F., & Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, *15*(3), 143–178. doi: 10.1016/0004-3702(80)90042-9

Amerian, M., Jofi, L. (2015). Key concepts and basic notes on narratology and narrative. *Scientific Journal of Review*, *4*(10), 182-192. doi: 10.14196/sjr.v4i10.1927

Andersen, S., & Slator, B. M. (1990). Requiem for a theory: the 'story grammar' story. *Journal of Experimental & Theoretical Artificial Intelligence*, *2*(3), 253–275. doi: 10.1080/09528139008953726

Appelt, D. E. (1985). Planning English referring expressions. *Artificial Intelligence*, *26*(1), 1–33. doi: 10.1016/0004-3702(85)90011-6

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *Neural machine translation by jointly learning to align and translate*. Paper presented at 3rd International Conference on Learning Representations, ICLR 2015, San Diego, United States. Abstract retrieved from https://nyuscholars.nyu.edu/en/publications/neural-machine-translation-by-jointly-learning-to-align-and-trans-2

Bateman, J. A. (1997). Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, *3*(1), 15–55. https://doi.org/10.1017/s1351324997001514

Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, *14*(4), 431–455. doi: 10.1017/s1351324907004664

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*, 1137–1155. Retrieved from http://jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

Bishop, C. M. (2016). *Pattern Recognition and Machine Learning*. (1st ed.). New York, NY: Springer.

Bringsjord S., & Ferrucci D. (2000). *Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine*. Hillsdale, NJ: Erlbaum Associates Inc.

Callaway, C. B., & Lester, J. C. (2002). Pronominalization in generated discourse and dialogue. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 88–95. doi: 10.3115/1073083.1073100

Cawsey, A. (1993). *Explanation and Interaction: the computer generation of explanatory dialogues*. Cambridge, MA: MIT Press.

Chen, X., Kar, S., & Ralescu, D. A. (2012). Cross-entropy measure of uncertain variables. *Information Sciences*, *201*, 53–60. doi: 10.1016/j.ins.2012.02.049

Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP 2014)*. 1724 – 1734. Retrieved from https://www.aclweb.org/anthology/D14-1179.pdf

Clark, E., Ji, Y., & Smith, N. A. (2018). Neural Text Generation in Stories Using Entity Representations as Context. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1(Long Papers), 2250–2260. Retrieved from https://www.aclweb.org/anthology/N18-1204/

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37–46. doi: 10.1177/001316446002000104

Cohen, P. R., & Perrault, C. R. (1979). Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, *3*(3), 177–212. doi: 10.1207/s15516709cog0303_1

Correira, A. (1980). Computing story trees. *American Journal of Computational Linguistics*, *6*(4), 135–149. Retrieved from https://www.aclweb.org/anthology/J80-3001/

Daelemans, W. & Van den Bosch, A. (1998). Rapid development of NLP modules with memory-based learning. Proceedings of ELSNET in Wonderland (pp. 105-113). Utrect, Netherlands: ELSNET. Retrieved from

https://pdfs.semanticscholar.org/3e2e/8d6fe25b9c1ba9a0556b8c231ff02a7a8a32.pdf

Dale, R., Geldof, S. & Prost, J. (2003). CORAL: using natural language generation for navigational assistance. *Proceedings of the 26th Australasian computer science conference*, *16*(1), 35-44. Retrieved from https://www.lirmm.fr/~prost/publis/daleGeldofProst2003-ACSC.pdf

Dale, R., Mellish, C. S., & Zock, M. (1990). *Current research in natural language generation*. London: Academic.

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language Modeling with Gated Convolutional Networks. *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, *70*, 933-941. Retrieved from https://dl.acm.org/doi/pdf/10.5555/3305381.3305478?download=true

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North*, 4171–4186. doi: 10.18653/v1/n19-1423

Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical Neural Story Generation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (*Volume1: Long Papers)*, 2250–2260. Retrieved from https://www.aclweb.org/anthology/P18-1082/

Fayyad, U., & Uthurusamy, R. (1996). Data mining and knowledge discovery in databases. *Communications of the ACM*, *39*(11), 24–26. doi: 10.1145/240455.240463

Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3–4), 189–208. doi: 10.1016/0004-3702(71)90010-5

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378–382. doi: 10.1037/h0031619

Fludernik, M. (2009). *Introduction To Narratology* (1st ed.). London, New York: Routledge.

Garoufi, K. (2014). Planning-Based Models of Natural Language Generation. *Language and Linguistics Compass*, *8*(1), 1–10. doi: 10.1111/lnc3.12053

Garoufi, K., & Koller, A. (2013). Generation of effective referring expressions in situated context. *Language, Cognition and Neuroscience*, *29*(8), 986–1001. https://doi.org/10.1080/01690965.2013.847190

Gatt, A., & Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, *61*(3), 65–170. doi: 10.1613/jair.5477

Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y.N. (2017). Convolutional Sequence to Sequence Learning. *Proceedings of the 34th International Conference on Machine Learning (ICML 2017), 70*, 1243-1252. Retrieved from http://proceedings.mlr.press/v70/gehring17a.html

Goldman, N. (1975). Conceptual Generation. In R. Schank (Ed.), *Conceptual Information Processing* (pp. 289-371). New York, NY: Elsevier.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Nets. *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2672–2680. Retrieved from http://papers.nips.cc/paper/5423-generative-adversarial-nets

Gradmann, S., Hühn, P., & Schönert, J. (2006). Ein netzgestütztes Living Handbook of Narratology. *Jahrbuch Für Internationale Germanistik*, *2006*(1), 109–114. doi: 10.3726/82024_109

Halliday, M. A. K., & Matthiessen, C. (2004). *An introduction to functional grammar*. (3rd ed.). London, NY: Hodder Education Publishers.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266. doi: 10.1126/science.aaa8685

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

Hovy, E. H. (1991). Approaches to the Planning of Coherent Text. *The Kluwer International Series in Engineering and Computer Science*, *6*(2), 83–102. doi: 10.1007/978-1-4757-5945-7_3

Hutchins, J. (2000). *Early Years in Machine Translation* (97th ed., p. 400). Amsterdam: John Benjamins.

Inder, R. (1996). Planning and Problem Solving. *Artificial Intelligence*, 23–53. doi: 10.1016/b978-012161964-0/50004-2

Inui, K., Tokunaga, T., & Tanaka, H. (1992). Text revision: A model and its implementation. In Dale, R., Hovy, E. H., Rosner, D., & Stock, O. (Eds.), Lecture Notes in Computer Science*: Vol. 587, International Workshop on Natural Language Generation – IWNLG 1992: Aspects of automated natural language generation* (pp. 215-230). Berlin, Germany: Springer. doi: 10.1007/3-540-55399-1_15

Jain, P., Agrawal, P., Mishira, A., Sukhwani, M., Laha, A., & Sankaranarayanan, K. (2017). Story Generation from Sequence of Independent Short Descriptions. In S. Matvin, S. Yu, F. Farooq (Eds.), *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD2017*. Halifax, NS: Association for Computing Machinery.

Joshi, A. K., & Schabes, Y. (1997). Tree-Adjoining Grammars. *Handbook of Formal Languages*, *3*(1), 69–123. doi: 10.1007/978-3-642-59126-6_2

Kalchbrenner, N., & Blunsom, P. (2013). Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, *2*, 1700–1709. Retrieved from https://www.aclweb.org/anthology/D13-1176.pdf

Klein , S., Aeschlimann, J. F., Balsiger, D. F., Converse, S. L., Court, C., Foster, M., … Smith, J. (1973). *Automatic novel writing: A status report* (Report No. 168). Madison, WI: University of Wisconsin-Madison Department of Computer Sciences.

Kobayashi, H. (2014). Perplexity on Reduced Corpora. *Proceedings of the 52nd Annual Meeting of the Association for Computational* Linguistic, 1(Long Papers), 797-806. Retrieved from https://www.aclweb.org/anthology/P14-1075/

Koller, A., & Petrick, R. P. A. (2011). Experiences with planning for natural language generation. Computational Intelligence, 27(1), 23–40. doi: 10.1111/j.1467-8640.2010.00370.x

Kukich, K. (1987). Where do Phrases Come from: Some Preliminary Experiments in Connectionist Phrase Generation. *Natural Language Generation*, *135*(1), 405–421. https://doi.org/10.1007/978-94-009-3645-4_26

Kybartas, B., & Bidarra, R. (2017). A Survey on Story Generation Techniques for Authoring Computational Narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, *9*(3), 239–253. doi: 10.1109/tciaig.2016.2546063

Labov, W., & Waletzky, J. (1997). Narrative Analysis: Oral Versions of Personal Experience. *Oral Versions of Personal Experience Journal of Narrative and Life History*, *7*(1-4), 3–38. doi: 10.1075/jnlh.7.02nar

Lakoff, G. (1972). Structural complexity in fairy tales. *The Story of Man*, *1*(2), 128–190. Retrieved from https://georgelakoff.files.word-press.com/2014/06/structural-complexity-in-fairy-tales-lakoff-1972.pdf

Langkilde, I. (2000) . Forest-based statistical sentence generation. Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (pp. 170-177). Stroudsburg, PA: ACL Press. Retrieved from https://www.aclweb.org/anthology/A00-2023/

Lavie, A., & Agarwal, A. (2007). METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. *Proceedings of the Second Workshop on Statistical Machine Translation - StatMT '07*, 65–72. doi: 10.3115/1626355.1626389

Lavoie, B., & Rambow, O. (1997). A fast and portable realizer for text generation systems. *Proceedings of the Fifth Conference on Applied Natural Language Processing* (pp. 255–268). Groningen, Netherlands: ACL Press. doi: 10.3115/974557.974596

Lebowitz, M. (1984). Creating characters in a story-telling universe. *Poetics*, *13*(3), 171–194. Retrieved from https://doi.org/10.1016/0304-422x(84)90001-9

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

Lemon, L. T., Reis, M. J., Ėĭkhenbaum, B., Shklovskiĭ, V., & Tomashevskiĭ, B. V. (1965). *Russian Formalist Criticism: Four Essays (Regents Critics Ser)* (3rd Printing). Lincoln: University of Nebraska Press.

Lester, J. C., & Porter, B. W. (1997). Developing and empirically evaluating robust explanation generators: the KNIGHT experiments. *Computational Linguistics*, *23*(1), 65–101. Retrieved from https://core.ac.uk/display/24692874

Li B., Lee-Urban S., Johnston G., & Riedl M. (2013). Story generation with crowdsourced plot graphs. *Proceedings of the Twenty-Seventh AAAI*

Conference on Artificial Intelligence, 598-604. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6399

Lin, C.-Y., & Hovy, E. (2003). Automatic evaluation of summaries using N-gram co-occurrence statistics. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, 177–198. doi: 10.3115/1073445.1073465

Liu, L., & Özsu, M. T. (2009). *Encyclopedia of Database Systems*. New York: Springer

Lönneker, B., Meister, J. C., Gervás, P., Peinado, F., & Mateas, M. (2005). Story Generators: Models and Approaches for the Generation of Literary Artefacts. *Proceedings of the 17th Joint International Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*, 126–133. Retrieved from https://core.ac.uk/display/20932257

Mani, I., & Dale, R. (1995). Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes. *Language*, *71*(2), 381-405. doi: 10.2307/416185

Marciniak, T., & Strube, M. (2005). Beyond the Pipeline: Discrete Optimization in NLP. Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL). Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/W05-0618.pdf

Martin, L. J., Ammanabrolu, P., Wang, X., Hancock W., Singh, S., Harrison, B. & Riedl, M. O. (2018). Event representations for automated story generation with deep neural nets. *Proceedings of the 2018 Conference of the North American Chapter of the ACL: Human Language Technologies, Vol. 1 (Long Papers)*, 2250-2260. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/download/17046/15769

Maybury, M. T. (1992). Communicative acts for explanation generation. *International Journal of Man-Machine Studies*, *37*(2), 135–172. doi: 10.1016/0020-7373(92)90083-w

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. doi: 10.1007/bf02478259

McDonald, D. D. (1993). Issues in the Choice of a Source for Natural Language Generation. *Computational Linguistics*, 19 (1), 191–197. Retrieved from https://www.aclweb.org/anthology/J93-1009/

McDonald, D. D. (2010). Natural Language Generation . In R. Herbrich, & T. Graepel. (Eds.), *Handbook of Natural Language Processing* (2nd ed., pp. 121-141). London, New York: CRC Press.

McIntyre, N. (2011). *Learning to Tell Tales: Automatic Story Generation from Corpora* (Doctoral thesis, Institute for Communicating and Collaborative Systems School of Informatics University of Edinburgh, Edinburgh, UK). Retrieved from https://era.ed.ac.uk/handle/1842/5039

McKeown, K. R. (1985). Discourse strategies for generating natural-language text. *Artificial Intelligence*, *27*(1), 1–41. doi: 10.1016/0004-3702(85)90082-7

Mcroy, S. W., Channarukul, S., & Ali, S. S. (2003). An augmented template-based approach to text realization. *Natural Language Engineering*, *9*(4), 381–420. doi: 10.1017/s1351324903003188

Meehan, J. R. (1977). TALE-SPIN, an interactive program that writes stories. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, *1*, 91–98. Retrieved from https://dl.acm.org/citation.cfm?id=1624452

Mehta, A., Gala, R., & Kurup, L. (2016). A roadmap to auto story generation. *2016 3rd International Conference on Computing for Sustainable Global Development* (pp. 2306-2310). New Delhi: IEEE. Retrieved from https://ieeexplore.ieee.org/document/7724674

Meteer, M. W. (1991). Bridging the Generation Gap Between Text Planning and Linguistic Realization. Computational Intelligence, 7(4), 296–304. doi: 10.1111/j.1467-8640.1991.tb00402.x

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, *2*(1),

3111–3119. Retrieved from https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent Neural Network based Language Model. *Proceedings of the 2010 International Speech Communication Association (INTERSPEECH 2010)*, 1045–1048. Retrieved from https://www.isca-speech.org/archive/interspeech_2010/i10_1045.html

Mitchell, T. M. (2017). Machine learning. New York, NY: Mcgraw Hill.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8). New York: Mcgraw-Hill.

Moore, J. D., & Paris, C. (1993). Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information. *Computational Linguistics*, *19*(4), 651–694. Retrieved from https://www.aclweb.org/anthology/J93-4004/

N Chomsky. (1965). *Aspects of the Theory of Syntax*. Cambridge: The M.I.T. Press

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Auli, M. (2019). fairseq: A Fast, Extensible Toolkit for Sequence Modeling. *Proceedings of the 2019 Conference of the North*, 48–53. doi: 10.18653/v1/N19-4009

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 311–318. doi: 10.3115/1073083.1073135

Paris, C. L. (2015). *User modelling in text generation*. London: Bloomsbury.

Pemberton, L. (1989). A modular approach to story generation. *Proceedings of the Fourth Conference on European Chapter of the Association for Computational Linguistics* , 217–224. doi: 10.3115/976815.976845

Peng, N., Ghazvininejad, M., May, J., & Knight, K. (2018). Towards Controllable Story Generation. *Proceedings of the First Workshop on Storytelling*, 43–49. doi: 10.18653/v1/w18-1505

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. doi: 10.3115/v1/d14-1162

Pérez ý Pérez, R. (2001). MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, *13*(2), 119–139. doi: 10.1080/09528130118867

Phelan, J., & Rabinowitz, P. J. (2008). *A companion to narrative theory*. Malden, Ma: Blackwell Pub.

Poon, H. K., Yap, W. S., Tee, Y. K., Lee, W. K., & Goi, B. M. (2019). Hierarchical gated recurrent neural network with adversarial and virtual adversarial training on text classification. *Neural Networks*, *119*, 299–312. doi: 10.1016/j.neunet.2019.08.017

Prince, G., Genette, G., & Lewin, J. E. (1980). Narrative Discourse: An Essay in Method. *Comparative Literature*, *32*(4), 413-427. doi: 10.2307/1770890

Propp, V. I. (1968). *Morphology of the folktale* (1st ed.). Austin, TX: University of Texas Press.

Randolph, J. (2005, January). *Free-Marginal Multirater Kappa (multirater _free): An alternative to Fleiss' Fixed-Marginal Multirater Kappa*. Paper presented at the Joensuu Learning and Instruction Symposium, Joensuu, Finland. Abstract retrieved from

Reiter, E. (1994, June). *Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible?*. Paper presented at the INLG '94 Proceedings of the Seventh International Workshop on Natural Language Generation, Kennebunkport, Maine. Abstract retrieved from https://www.aclweb.org/anthology/W94-0319.pdf

Reiter, E., & Belz, A. (2009). An Investigation into the Validity of Some Metrics for Automatically Evaluating Natural Language Generation Systems. *Computational Linguistics*, *35*(4), 529–558. doi: 10.1162/coli.2009.35.4.35405

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 57–87. doi: 10.1017/s1351324997001502

Reiter, E., Sripada, S., Hunter, J., Yu, J., & Davy, I. (2005). Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, *167*(1–2), 137–169. https://doi.org/10.1016/j.artint.2005.06.006

Richard, L. J., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *International Biometric Society*, *33*(1), 159–174. doi: 10.2307/2529310

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic Keyword Extraction from Individual Documents. *Text Mining Applications and Theory*, 1–20. doi: 10.1002/9780470689646.ch1

Sacerdoti, E. D. (1977). *A Structure for Plans and Behaviour* (Doctoral thesis, Stanford University Department of Computer Science, Stanford, CA). Retrieved from https://dl.acm.org/citation.cfm?id=907010

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210–229. doi: 10.1147/rd.33.0210

Schwenk, H., & Gauvain, J. L. (2005). Training neural network language models on very large corpora. *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*. doi: 10.3115/1220575.1220601

Scott A. C., Clayton, J. E., & Gibson, E. L. (1991). *A practical guide to knowledge acquisition*. Reading, MA: Addison-Wesley Pub Co.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. doi: 10.18653/v1/p16-1162

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. (2016). Building EndTo-End Dialogue Systems Using Generative Hierarchical Neural Network Models. *Proceedings of the 30th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence and the 28th Innovative Applications of Artificial Intelligence Conference*, *4*, 3776–3784. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/11957/12160

Shapiro, S. C. (1992). *Encyclopedia of artificial intelligence* (2nd ed.). New York: Wiley.

Slade, S. (1991). Case-based reasoning: A research paradigm. *AI Magazine*, *12*(1), 42-55. doi: 10.1609/aimag.v12i1.883

Smedt, K., Horachek, H. & Zock M. (1996). Architectures for Natural Language Generation: Problems and Perspectives. *Proceedings of the Fourth European Workshop on Natural Language Generation* (pp. 17-46). Pisa, Italy: Springer. doi: 10.1007/3-540- 60800-1 22

Sparck Jones, K., & Galliers, J. R. (1996). *Evaluating natural language processing systems: an analysis and review*. Berlin: Springer.

Sriram, A., Jun, H., Satheesh, S., & Coates, A. (2018). Cold Fusion: Training Seq2Seq Models Together with Language Models. *Interspeech 2018*, 387–391. doi: 10.21437/interspeech.2018-1392

Stede, M. (1996). Lexical options in multilingual generation from a knowledge base. In G. Adorni and M. Zock (Eds.) *Lecture Notes in Artificial Intelligence: Trends in Natural Language Generation* (pp. 222-237). Springer.

Stent, A., & Srinivas Bangalore. (2014). *Natural language generation in interactive systems*. Cambridge: Cambridge University Press.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Proceedings of the 2014 Neural Information Processing Systems (NIPS 2014), 3104–3112. Retrieved from https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Sutton, R. S., & Barto, A. G. (2018). An Introduction to Reinforcement Learning. Cambridge, MA: MIT Press.

Vaswani, A., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. *Proceedings of the 2017 Neural Information Processing Systems (NIPS 2017)*, 5998–6008. Retrieved from https://papers.nips.cc/paper/7181-attention-is-all-you-need

Vaudry, P. L., & Lapalme, G. (2013). Adapting SimpleNLG for bilingual French-English realisation. *Proceedings of the 14th European Workshop on Natural Language Generation* (pp. 183-187). Sofia, Bulgaria: ACL Press. Retrieved from https://www.aclweb.org/anthology/W13-2125/

Walker, M. A., Rambow, O., & Rogati, M. (2001). SPoT: A Trainable Sentence Planner. *Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies 2001 - NAACL '01*. doi: 10.3115/1073336.1073339

Weizenbaum, J. (1966). ELIZA-A--a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45. doi: 10.1145/365153.365168

Xu, J., Ren, X., Zhang, Y., Zeng, Q., Chi, X., & Sun, X. (2018). A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4306–4315. doi: 10.18653/v1/D18-1462

Young, R. M., & Moore, J. D. (1994, June). *DPOCL: A Principled Approach to Discourse Planning.* Paper presented at the INLG '94 Proceedings of the Seventh International Workshop on Natural Language Generation, Kennebunkport, Maine. Abstract retrieved from https://www.aclweb.org/anthology/W94-0302.pdf

APPENDIX A

Stories	TRNET		Fusion Model	
	Match	No Match	Match	No Match
story #1	0	15	2	13
story #2	0	15	12	3
story #3	2	13	11	4
story #4	0	15	14	1
story #5	0	15	13	2
story #6	0	15	12	3
story #7	2	13	14	1
story #8	4	11	10	5
story #9	0	15	11	4
story #10	4	11	12	3
story #11	3	12	12	3
story #12	1	14	12	3
story #13	2	13	0	15
story #14	0	15	3	12
story #15	0	15	13	2
story #16	0	15	10	5
story #17	0	15	14	1
story #18	1	14	3	12
story #19	0	15	10	5
story #20	1	14	0	15
story #21	1	14	15	0
story #22	0	15	13	2
story #23	0	15	12	3
story #24	0	15	1	14
story #25	0	15	11	4
story #26	0	15	12	3
story #27	1	14	11	4
story #28	0	15	13	2
story #29	1	14	10	5
story #30	0	15	11	4
story #31	0	15	10	5
story #32	0	15	11	4
story #33	0	15	12	3
story #34	0	15	10	5
story #35	0	15	2	13
story #36	0	15	11	4
story #37	0	15	13	2
story #38	0	15	15	0
story #39	0	15	13	2
story #40	1	14	13	2
story #41	1	14	11	4
story #42	0	15	14	1
story #43	0	15	13	2
story #44	0	15	12	3
story #45	0	15	13	2
story #46	5	10	10	5
story #47	1	14	14	1
story #48	0	15	1	14
story #49	0	15	13	2
story #50	0	15	12	3
story #51	0	15	14	1
story #52	1	14	13	2
story #53	1	14	14	1
story #54	0	15	0	15
story #55	2	13	2	13
story #56	0	15	13	2
story #57	0	15	14	1
story #58	0	15	13	2
story #59	0	15	2	13
story #60	0	15	13	2
story #61	5	10	10	5
story #62	0	15	0	15
story #63	0	15	14	1
story #64	1	14	11	4
story #65	0	15	14	1
story #66	1	14	10	5
story #67	0	15	12	3
story #68	2	13	1	14
story #69	4	11	11	4
story #70	0	15	13	2
story #71	0	15	12	3
story #72	0	15	15	0
story #73	2	13	12	3
story #74	1	14	11	4
story #75	0	15	14	1
story #76	0	15	13	2
story #77	1	14	11	4
story #78	0	15	14	1
story #79	0	15	13	2
story #80	1	14	2	13
story #81	3	12	14	1
story #82	1	14	14	1
story #83	0	15	12	3
story #84	1	14	11	4
story #85	1	14	13	2
story #86	0	15	11	4
story #87	0	15	12	3
story #88	1	14	13	2
story #89	0	15	14	1
story #90	1	14	11	4
story #91	0	15	2	13
story #92	1	14	11	4
story #93	0	15	10	5
story #94	2	13	13	2
story #95	0	15	12	3
story #96	0	15	14	1
story #97	1	14	12	3
story #98	0	15	2	13
story #99	0	15	13	2
story #100	0	15	14	1

Results for prompt pairing task

Stories	TRNET	Fusion
story #1	1	4
story #2	0	5
story #3	0	5
story #4	1	4
story #5	0	5
story #6	1	4
story #7	0	5
story #8	1	4
story #9	0	5
story #10	1	4
story #11	0	5
story #12	0	5
story #13	1	4
story #14	5	0
story #15	1	4
story #16	1	4
story #17	0	5
story #18	1	4
story #19	1	4
story #20	5	0
story #21	0	5
story #22	1	4
story #23	1	4
story #24	0	5
story #25	1	4
story #26	0	5
story #27	0	5
story #28	1	4
story #29	0	5
story #30	1	4
story #31	1	4
story #32	1	4
story #33	1	4
story #34	0	5
story #35	5	0
story #36	0	5
story #37	0	5
story #38	0	5
story #39	1	4
story #40	0	5
story #41	0	5
story #42	1	4
story #43	5	0
story #44	0	5
story #45	0	5
story #46	5	0
story #47	1	4
story #48	0	5
story #49	5	0
story #50	0	5
story #51	1	4
story #52	0	5
story #53	1	4
story #54	5	0
story #55	0	5
story #56	5	0
story #57	1	4
story #58	5	0
story #59	0	5
story #60	0	5
story #61	1	4
story #62	5	0
story #63	1	4
story #64	0	5
story #65	1	4
story #66	1	4
story #67	1	4
story #68	4	1
story #69	4	1
story #70	1	4
story #71	1	4
story #72	0	5
story #73	1	4
story #74	4	1
story #75	1	4
story #76	0	5
story #77	4	1
story #78	1	4
story #79	0	5
story #80	0	5
story #81	1	4
story #82	1	4
story #83	1	4
story #84	1	4
story #85	0	5
story #86	5	0
story #87	0	5
story #88	1	4
story #89	1	4
story #90	0	5
story #91	1	4
story #92	0	5
story #93	0	5
story #94	0	5
story #95	5	0
story #96	0	5
story #97	1	4
story #98	0	5
story #99	0	5
story #100	1	4

Results for blind comparison task