IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Comparison of Evaluation Metrics for Short Story Generation

PONRUDEE NETISOPAKUL[1], USANISA TAOTO[2]
[1]School of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok (e-mail: ponrudee@it.kmitl.ac.th)
[2]School of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok (e-mail: 61606001@it.kmitl.ac.th)
Corresponding author: Ponrudee Netisopakul (e-mail: ponrudee@it.kmitl.ac.th).

ABSTRACT The aim of this study was to analyze the correlation among different automatic evaluation metrics for text generation. In the study, texts were generated from short stories using different language models: N-gram model, Continuous Bag-of-Word (CBOW) model, Gated recurrent unit (GRU) model, and Generative Pre-trained Transformer 2 (GPT-2) model. All models were trained on short Aesop's fables. The quality of the generated text was measured with various metrics: Perplexity, BLEU score, the number of grammatical errors, Self-BLEU score, ROUGE score, BERTScore, and Word Mover's Distance (WMD). The resulting correlation analysis of the evaluation metrics showed four groups of correlated metrics. Firstly, perplexity and grammatical errors were moderately correlated. Secondly, BLEU, ROUGE and BERTScore were highly correlated. Next, WMD was negatively correlated with BLEU, ROUGE and BERTScore. On the other hand, Self-BLEU, which measures text diversity within the model, did not correlate with the other metrics. In conclusion, to evaluate text generation, a combination of various metrics should be used to measure different aspects of the generated text.

INDEX TERMS Natural language processing, Neural networks, Text processing, Text analysis

I. INTRODUCTION

Natural language generation or text generation is a task of, given an input, generating a sequence of natural language output that appropriately response to, or continuing from the input. There are many applications of text generation, for example, automatic message responses to user requests, such as in a chatbot, text generation from data analysis to describe the finding, text summarizing from long or diverse documents, story generation from a given beginning text, machine translation from another language, and so on. All of these tasks eventually need to be objectively evaluated. The problem of text generation evaluation is crucial and not easy. The standard evaluation metrics used in many machine learning classification tasks, such as precision, recall, and F-1 score cannot be applied to text generation tasks. In addition, each generation task may have different objectives, which require different evaluation metrics. Basically, evaluation methods are either manual evaluation by human evaluators or automatic evaluation [1]. Manual evaluation by humans can be slow and costly, especially when dealing with large datasets or multiple generated texts. Thus, using automatic metrics evaluation are more appropriate in many cases.

There are many types of automatic metrics for evaluating text generation tasks. For example, perplexity [2] and BLEU score [3] are two well-known metrics. However, it is difficult to determine which automatic metric is the best, as each has its own advantages and disadvantages. For example, while perplexity is widely used, it can be difficult to interpret and may not always correlate with human evaluation. BLEU score, on the other hand, is easy to understand and has a clear interpretation, but it has been criticized for being biased towards certain types of text and for not accounting for fluency or coherence [4]. Other categories of automatic metrics are N-gram overlapped and distance-based metrics, which compare the generated text to a gold standard reference, similar to the BLEU score metric.

In this research, we focused on short story generation from a beginning phrase. Stories were generated from different text generation models when given the same beginning text. Automatic text generation metrics are then applied to evaluate the models' performances.

The evaluation metrics covered in this study include perplexity, number of grammatical errors, BLEU score, Self-BLEU [5], ROUGE score [6], BERTScore [7], and Word Mover's Distance [8].

Several language models were used in this study with

the Aesop's Fables corpus, including N-gram model [9], neural network model (specifically, CBOW model) [10], GRU model (gated recurrent unit) [11], Pretrained GPT-2 model [12], and Finetuned GPT-2 model. These models were used for text generation. The study also involved evaluating the performance of each model using several automatic evaluation metrics, including perplexity, BLEU, Self-BLEU, ROUGE, BERTScore, and Word Mover's Distance. Finally, the study analyzed the correlation between these metrics.

The research questions for this study were the following:

1) Were there any correlation between these automatic metrics? and why?
2) Which metric or a set of metrics was the best to use for text generation evaluation?
3) Could automatic text evaluation metrics objectively show the superiority of one text generation model over the other?

In the upcoming section, we review previous research studies related to text generation and their evaluation methodology. In the methodology section, we describe our own experiment involving text generation, which was evaluated with various metrics. The results of the experiments are then analyzed and discussed to answer the research questions above.

II. LITERATURE REVIEW

In the field of natural language intelligence, there are many ways to generate messages that mimic those produced by humans. They can be broadly classified into rule-based models, statistical models, and neural network models. This study did not focus on rule-based models because of their inflexibility [13]. It focused on various statistical and neural network models that have been applied in natural language generation applications.

Text generation is a simple concept that controls the input and output. Just like humans who write message from left to right, most language model works the same way. It takes the context message as input and then predicts the next word. The number of possible next words depends on the model's vocabulary and the provided context. From Equation 1 [14], the input text (also known as context) is a sequence of words or tokens $x_1, ..., x_{i-1}$. The language model generates the next n words or tokens until the text is complete. In general, text generation often ends when it finds an end-of-sentence token (End token). Then, the generated text will be represented as $x_1, ..., x_{m+n}$. According to Equation 1, the probability of generating text from the start to the $(m+n)^{th}$ token of the text is represented as

$$P(x_1, ..., x_{m+n}) = \prod_{i=1}^{m+n} P(x_i | x_1, ..., x_{i-1}) \qquad (1)$$

Equation 1 represents the probability of a word sequence (or token sequence) where the presence of each word is conditional to the previous word. It is used in language models to generate text. The output from the language model is in the form of probabilities indicating the likelihood of each word becoming the next word. Text generation generally stops when it returns an end-of-sentence token.

Markov chains [15] are also a well-known method for describing the process of generating sentences or paragraphs based on existing text. This method computes a given text and builds a probabilistic model that captures the transitions between words or characters

N-gram [16] is a text generation technique that predicts the next word or sequence of words based on the preceding $n-1$ words. N-gram models are based on the Markov assumption [15] that the probability of a word depends only on the preceding $(n-1)$ words.

However, N-gram models have limitations. For example, they are sensitive to the context within the N-gram window, which can result in problems with unseen N-gram. Additionally, they do not capture long-range dependencies in the text. To address the issue of unseen N-gram, N-gram smoothing techniques can be used. They can help handle sparsity problems and improve the model's accuracy. [17] and [18] are two examples of smoothing techniques.

In 2013, word embedding was introduced. Word embedding is a technique to transform a word into a vector in a low-dimensional space using neural network. This technique preserves the meaning of word in its context and is the core reasons for the rise of the neural network language model era. For example, [19] used neural networks to generate biographies.

Recurrent Neural Networks were also widely used for text generation tasks. RNNs could handle flexible lengths of input and output, which were suitable for text generation as text could have varying lengths of words in sentences or passages. RNNs had hidden states, which acted like their inside memory and enabled them to receive and update information based on their input at each step. Even though they were popular, they had some limitations. They had some problems with sentences that were too long. Therefore, there were updated versions like LSTM [20] and GRU [11] to fix those issues. A recurrent model like this can generate text at either the character or word level. For example, the authors of [21] used various RNN language models, trained with a special optimizer called Hessian-Free to generate text at the character level.

The sequence-to-sequence (seq2seq) [22] model was an improvement from the recurrent neural network model. It introduced an encoder-decoder architecture, where the encoder captured the contextual information vector from the input sequence, and the decoder generated the output sequence step by step. The seq2seq allowed the model to handle sequences of different lengths and produce outputs that fit the context. The work in [22] was also the first to apply seq2seq model to a machine translation task.

The Transformer model was a big improvement over the seq2seq model. Not only did it use the encoder and decoder architecture, but it also used attention mechanisms, including self-attention and multi-head attention, as well as positional

embedding. Transformers are used to create dialogue systems, chatbots, and other language generation tasks. GPT [23] and its variants GPT-2 [12] GPT-3, [24] and GPT-3.5 [25] are examples of this technology. The decoder component of the transformer architecture has been widely used for natural language generation tasks.

In our research, it is important to experiment with text generation using both non-transformer and transformer models. Non-transformer models, which include statistical language models, neural network models, and recurrent neural networks, can generate text by predicting the next word in a sequence given the previous words, while transformer models are able to generate text by attending to all the words in a sequence at once.

Both models were important in our research to understand the capability of measurement metrics for assessing the qualities of the generated text.

Automatic evaluation metrics for text generation can be categorized into 5 categories: N-gram overlap metric, distance-based metric, diversity metric, content overlap metric, and grammatical feature-based metric [1]. N-gram overlap metric involves breaking each text into smaller pieces and comparing the overlap between the N-grams of each text. Distance-based metric measures the distance between the reference text and the generated text. Diversity metric measures the diversity of the generated text, while content overlap metric measures the similarity between the generated text and the reference text. Grammatical feature-based metric measures the accuracy of the generated text in terms of grammar.

N-gram overlap metric is used in natural language processing to measure the similarity between two texts. The metric involves breaking each text into smaller pieces of n consecutive words, and then comparing the overlap between the N-gram in each text. Examples of N-gram overlap metrics include BLEU [3], ROUGE [6], and METEOR [26].

Distance-based metric measures the distance between the reference text and the generated text. One example of distance-based metric is Word Mover's Distance (WMD) [8], which measures the distance between the embedding of two texts.

Diversity metric measures the diversity of generated text, which is important for ensuring that the text is both interesting and informative. One example of diversity metric is Self-BLEU [5], which is a variation of the BLEU score. Self-BLEU measures the similarity between different generated texts produced by the same model.

Content overlap metric measures the similarity between the generated text and the reference text in terms of content. Examples of this type of metric include PYRAMID [27] , SPICE [28], and SPIDER [29], which are used in image captioning and dialog systems.

Grammatical feature-based metric gauges the accuracy of the generated text in terms of grammar. Examples of grammatical feature-based metric include Part-Of-Speech accuracy, dependency accuracy, and fluency. They can be challenging to compute and may require additional preprocessing or annotation of the reference and generated texts.

Perplexity [2] is a metric not categorized into any of the categories mentioned above. It is generally considered to be a stand-alone metric used for evaluating the overall performance of a language model. However, it can be seen as a type of distance-based metric that measures the similarity between the probability distributions of the predicted words and the true words in the test set. In this sense, perplexity can be considered a type of evaluation metric related to distance-based metrics.

Each metric has its own limitations. For example, perplexity measures the probability of a sequence of words and is widely used, but it can be difficult to interpret, and it may not always correlate with human evaluation. BLEU score measures the similarity between the generated text and the reference text in terms of N-gram overlap and is easy to understand, but it has been criticized for being biased towards certain types of text and for not accounting for fluency or coherence [30]. Self-BLEU is a variation of BLEU score that measures the similarity between different generated texts produced by the same model. It is used to measure diversity in generated text [5]. ROUGE score measures the similarity between the generated text and the reference text in terms of N-gram overlap, similar to the BLEU score but with a different weighting scheme [6]. BERTScore is similar to BLEU and ROUGE scores, but it uses a transformer-based model instead of N-gram, while Word Mover's Distance measures the distance between the embeddings of two texts and is used to measure similarity between the generated text and the reference text [8].

Various automatic metrics have been used to evaluate text generation tasks. For example, [22] used the BLEU metric to evaluate text generated by sequence-to-sequence models. Similarly, [31] evaluated the quality of generated dialogue responses using metrics like BLEU, distinct N-gram, and human evaluation. In [32], metrics like ROUGE and METEOR were used to evaluate the task of content selection in text summarization.

Although there are some works that use evaluation metrics to assess text generation, there are no related works that use multiple metrics to evaluate text generation from a specific domain, such as short stories generated from various language models.

Based on the information provided, it did not seem appropriate to incorporate content overlap in our research. This was mostly because most of the research on content overlap focused on image captioning, which involved both text and images. However, our work solely focused on text.

Based on the aforementioned reasons, several evaluation metrics were chosen in our research. Those were the number of grammatical errors identified by the Grammatical feature-based metric, the BLEU score, ROUGE which was an N-gram overlap metric, Self-BLEU which was a Diversity metric, Word Mover's Distance which was a Distance-based metric, and Perplexity which was not categorized above.

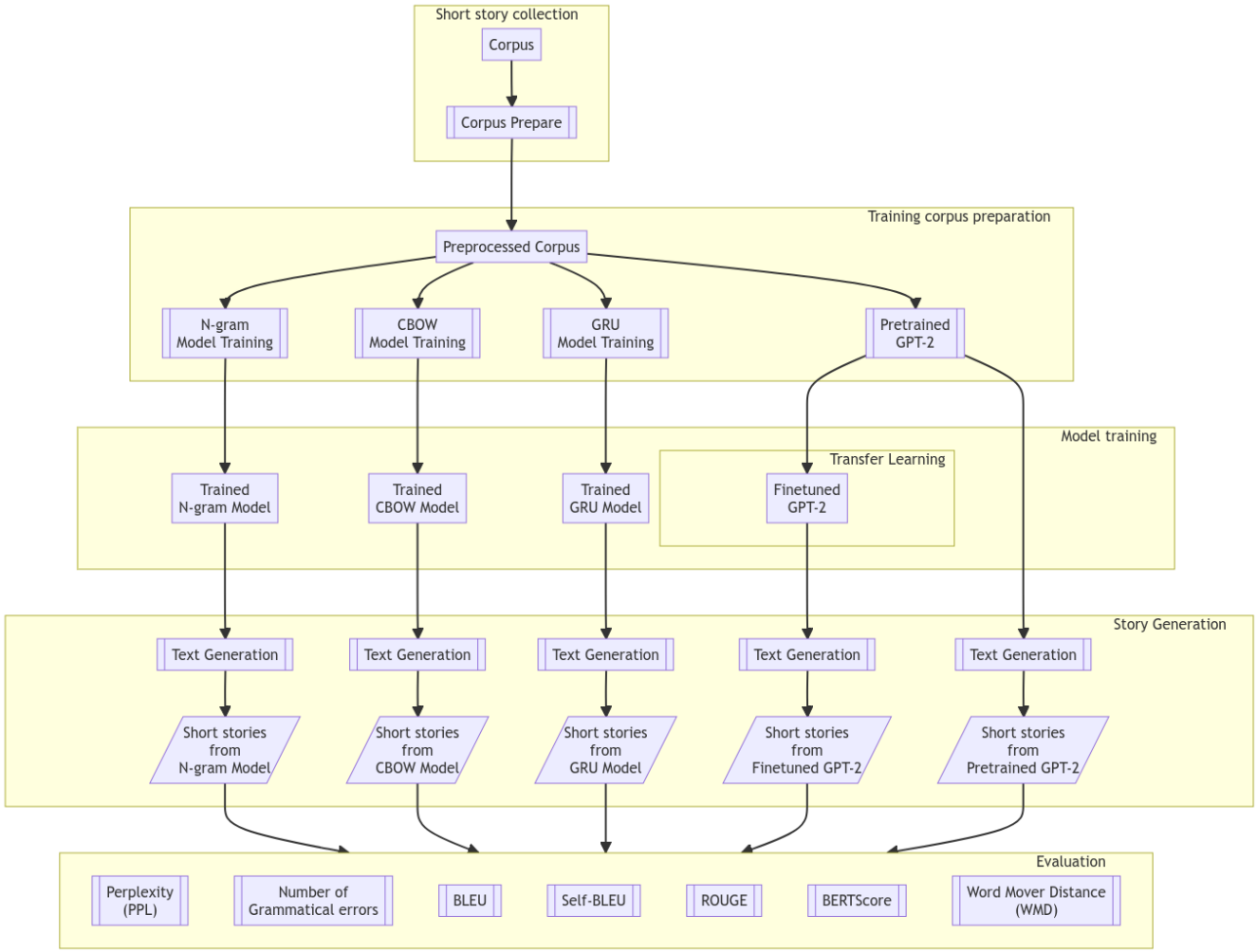

FIGURE 1. Pipeline of the experiments for Text Generation and Evaluation.

III. METHODS

This study generated texts from different language models trained within the same domain. To accomplish this, a short story domain was chosen. The experiments were set up using the same corpus but different language models. Once all the models had been trained, the same starting text was used to generate new text. The following subsection provides greater detail on the corpus and language model used.

The pipeline for this research experiment methodology is shown in Figure 1.

The corpus was first preprocessed into inputs and targets. Then, the language models were trained and used to generate new texts. After that, the generated texts were evaluated with various metrics.

A. SHORT STORY COLLECTION

In the first step of the experimental pipeline, short stories were selected from a corpus named Aesop's Fables from americanliterature.com [33]. These fables are a collection of well-known stories. For this study, we used a total of 160 stories to train our language models. The corpus statistics are shown in Table 1.

TABLE 1. Some statistics from the Aesop's Fables corpus - some cells show Median (Mean ± SD) [Minimum - Maximum]

Corpus property	Value
Number of stories	160
Number of words	32,031
Vocabulary size	3,418
Number of words per story	186 (200 ± 89.82) [74 - 520]
Number of sentences per story	10 (11 ± 5.3) [3 - 31]
Lexical diversity (%)	10.67
Lexical diversity per story (%)	55.56 (55.67 ± 6.92) [37.58 - 75.67]

Table 1 provides the statistics on the corpus used in this study. It consists of 160 short stories from Aesop's Fables. The table displays the number of stories, number of words, vocabulary size, average words per story, and average sentences per story. The Aesop's Fables corpus contains 160

stories, totaling 32,031 words and including 3,418 unique words in its vocabulary. The stories vary in length, with an average of 186 words and a range of 74 to 520 words. The lexical diversity of the corpus is 10.67%, A higher percentage implies a richer vocabulary. The lexical diversity of each story ranges from 37.58% to 75.67%. The lexical diversity per story is higher than the lexical diversity of the entire corpus, which means that each story uses a wider range of words. However, when we consider all the stories together, we see that the corpus as a whole does not have much variation in its vocabulary.

B. TRAINING CORPUS PREPARATION

Before the model training process, the corpus was prepared in a form acceptable to the language models; where input is the sliding window of the context of the stories and output is the next word prediction.

Stories in the corpus were first converted into word tokens. For each story, special tokens were added: a *"begin-of-story"* token before the first word, and an *"end-of-story"* token after the last word in the story.

In training a non-transformer model, the model required an exact number of input nodes, that is, the fixed number of context words for input nodes. For this reason, we would slide windows to capture every K word, when K was the desired context word number. The input was a word sequence $(W_i, W_{i+1}, W_{i+2}, ...W_{i+k})$, the output being the next word in this sequence, word W_{i+k+1}, when $i = 0, 1, 2, 3$, and the process moved one word at a time until the last word was reached.

The table 2 showed an example of window sliding with a context size of 4, and with the target word of the message "a hare was making fun of the tortoise."

TABLE 2. Context and target Word for non-transformer model training

Context (sliding windows of 4)	Target Word
<BEGIN> <BEGIN> <BEGIN> a	hare
<BEGIN> <BEGIN> a hare	was
<BEGIN> a Hare was	making
<BEGIN> a Hare was making	fun
hare was making fun	of
was making fun of	the
making fun of the	tortoise
fun of the tortoise	.
of the tortoise .	<END>
the tortoise . <END>	<END>
tortoise . <END> <END>	<END>
. <END> <END> <END>	<END>

However, preparing texts for training with the transformer model was slightly different. It was not necessary to limit the input to an exact number of words in the context. The input part, where words are the context, starts with the first word in the text. The output would be the next word in the context part. The example output and the context are shown in Table 3.

TABLE 3. Context and Target Word for transformer model training

Context	Target Word
<BEGIN>	a
<BEGIN> a	hare
<BEGIN> a hare	was
<BEGIN> a hare was	making
<BEGIN> a hare was making	fun
<BEGIN> a hare was making fun	of
<BEGIN> a hare was making fun of	the
<BEGIN> a hare was making fun of the tortoise	.
<BEGIN> a hare was making fun of the tortoise .	<END>

C. LANGUAGE MODEL TRAINING

In this study, five types of language models were used. An N-gram model was used for statistical language modelling. A CBOW (Continuous Bag-of-Words) model was used as a neural network model. A GRU model was used as a recurrent neural network model. Two models from Generative Pretrained Transformer 2 (GPT-2), were used as transformer models. Due to the large size of the GPT-2 model and the large number of hyperparameters, pretrained model (Pretrained GPT-2) and transfer learning (Finetuned GPT-2) were chosen instead of training the model from scratch.

We selected the aforementioned models not only because they have different model structures, but also because of the time period when each model was introduced and became well-known in the field of language modeling. The N-gram model has been a fundamental approach to language modeling and has been widely used since the early days. The CBOW model introduced embeddings to make language models more efficient in understanding meaning. The GRU model, with its recurrent neural network that can capture word sequences, is a significant improvement over traditional recurrent neural networks (RNNs). The Transformer architecture is a breakthrough in language modeling. It has impressive text generation performance because it trains on a large language model. During the time of its release, GPT-2 was a significant improvement in the field . It can generate human-like text in a variety of contexts and have an impact across a wide range of industries. That's why GPT-2 stands out as one of the most significant and influential.

The reason is that there are different models available, each with its own historical context for when it was introduced and became prominent in the field of language modeling. That is why we have chosen to use the following models in our experiment. Furthermore, we also want to check the capability of each model in the aspects that we have chosen.

D. SHORT STORY GENERATION

After the language model was trained, the next step was to use it for text generation. The algorithm generated text by predicting from left to right. At each step, the model would select the word with the highest probability and added it to the generated text. However, in order to make the generated text more diverse, the output word is chosen based on the probability weights of next word prediction. That is the higher the probability of the word, the more likely it is

chosen to be the next word output. The algorithm is shown in Algorithm 1.

Algorithm 1 Text Generation Algorithm for Non-Transformer Model

Procedure GenerateText(model, startingText, k, maxLength)
$input \leftarrow$ last k words of $startingText$
$genText \leftarrow startingText$
while length(genText) $\leq$ maxLength **and** END_TOKEN not found **do**
    $nextWordProbs \leftarrow$ model.probs($input$)
    $nextWord \leftarrow$ RandomlySelectWord($nextWordProbs$)
    $genText \leftarrow genText + nextWord$
    $input \leftarrow$ last (k-1) of input $+nextWord$
return $genText$
End Procedure

As the input length limit was fixed in non-transformer models, step *8* of the text generation algorithm slid the input window by removing the first word of the window and concatenating the next word into the window. Meanwhile, for transformers, there was no need to fix the input length because the model embedded padding ability for input length not exceed 1,024 for GPT-2. The algorithm is presented in Algorithm 2.

Algorithm 2 Text Generation Algorithm for Transformer Model

Procedure GenerateText(model, startingText, k, maxLength)
$input \leftarrow$ last k words of $startingText$
$genText \leftarrow startingText$
while length(genText) $\leq$ maxLength **and** END_TOKEN not found **do**
    $nextWordProbs \leftarrow$ model.probs($input$)
    $nextWord \leftarrow$ RandomlySelectWord($nextWordProbs$)
    $genText \leftarrow genText+ nextWord$
    $input \leftarrow input$ + $nextWord$
return $genText$
End Procedure

For each model, we will generate text by using the first sentence from every story in the Aesop's Fables as the initial text. We will generate three stories from each starting text to evaluate and compare generated text from different models for the following aspects: how close to human writing (text from corpus), how good is the grammar (number of grammatical errors), and also, the variation of text generated within the same model.

E. EVALUATION

After the texts have been generated, the results needed to be evaluated to judge the model's performance. Seven different scores were selected for reasons. Perplexity, and BLEU score were chosen because of their popularity, number of grammatical errors to measure rule-based grammar. Self-BLEU to measure variation within the same model, ROUGE-L score to capture common longest word sequence, BERTScore as a measure deploy transformer capability, and finally, Word Mover's Distance to measure the distance from original text.

1) Perplexity

Perplexity is a widely used metric to automatically evaluate text generated by language models.

It is defined as geometric mean of the inverse probabilities of the generated text. Simply put, it is inverse to the likelihood of the generated text appearing in the corpus, that is, how not likely the generated text is written by human. The higher the value, the more perplex (or confuse) for human reading the generated text. The formula for perplexity is shown in Algorithm 3.

Algorithm 3 Compute Perplexity

procedure COMPUTEPPL(gentext, model)
    $tokens \leftarrow$ Tokenize(gentext)
    $N \leftarrow$ length($tokens$)
    log_prob_sum $\leftarrow 0.0$
    $prob\_sum \leftarrow 0.0$
    for $i \leftarrow 1$ to N **do**
        $token \leftarrow$ tokens$[i]$
        $context \leftarrow$ tokens$[1 : i-1]$
        $prob \leftarrow$ model.Probability($token, context$)
        $prob\_sum \leftarrow prob\_sum \times prob$
    $perplexity \leftarrow$ power($prob\_sum, -1/N$))
    return $perplexity$

2) Number of grammatical errors

Grammatical errors were considered an indicator for text quality evaluation. In our experiment, we used an open-source rule-based grammar detection software called LanguageTools [34]. LanguageTools were used to detect grammatical errors both in the generated models and the text corpus. Language Tools had many error detection rules, but we did not use all of them. Table 4 shows the list of rules for what we identified as errors.

TABLE 4. LanguageTools: Error Types as identified errors.

Rule Type	Description
Grammar	Checks for grammatical errors such as subject-verb agreement, tense consistency, and pronoun use
Punctuation	Checks for errors related to the use of punctuation marks such as commas, semicolons, and colons
Misused word	Checks for commonly misused words and suggests alternatives
Contextual	Takes into account the context in which a word or phrase is used to provide more accurate suggestions for correction
Idiom	Checks for incorrect use of idiomatic expressions and suggests appropriate alternatives

3) BLEU

The BLEU score (Bilingual Evaluation Understudy) is a widely used evaluation metric, particularly in machine translation tasks, and can also be applied in other natural language generation applications, including text generation.

BLEU score values range from 0 to 1. If the BLEU score tends to be close to 1, this generated text is quite similar to the reference, i.e., it resembles the corpus story. On the other hand, 0 is the opposite, meaning it differs significantly from the corpus.

The calculation of BLEU scores not weighs the precision of the N-gram in the generated text that match the N-gram in the reference text, but also involves a brevity penalty, which takes into account the length of the generated text compared to the reference text. BLEU score was calculated with Algorithm 4.

Algorithm 4 Compute BLEU

procedure COMPUTEBLEU(refs, gentext, weights))
    ▷ weight is a vector with n-gram size
    ▷ sum of all elements is 1
    $c \leftarrow \text{length}(gentext)$
    $p \leftarrow [0.0] * \text{length}(weights)$ ▷ precision
    for each ref in $refs$ **do**
        $ref\_counts \leftarrow \text{count\_ngrams}(ref, n)$
        $gentext\_counts \leftarrow \text{count\_ngrams}(gentext, n)$
        $overlapped \leftarrow ref\_counts \cap gentext\_counts$
        for $i \leftarrow 0$ to $\text{length}(weights) - 1$ **do**
            $num \leftarrow \text{count}(overlapped)$
            $d \leftarrow max(1, c - i)$
            $p[i] \leftarrow p[i] + weights[i] \times (num/d)$
    $closest\_ref\_len \leftarrow \text{min\_len\_diff}(refs, c)$
    $bp \leftarrow \min(1.0, c/closest\_ref\_len)$
    $bleu \leftarrow bp \times \text{geometric\_mean}(p)$
    return $bleu$

In our experiment, every generate text would be compared to the text from the corpus in each story as a reference text. Because the model was generated by starting with the first sentence of each story in the Aesop's Fables corpus, the original story from the Corpus could be used as the reference text.

4) Self-BLEU

Self-BLEU is a modified version of the BLEU score that was designed to be used for evaluating the diversity of generated text. Self-BLEU compared the generated text to other texts generated which came from the same model instead of comparing the generated text to a set of reference texts.

Similar to the BLEU score, the Self-BLEU score ranges from 0 to 1. A lower score indicates that the model generated more diverse texts, while a higher score indicates that the model generated text that was quite similar. Self-BLEU score was calculated with Algorithm 5.

As mentioned in section III-D, one model generated three stories with the same starting text. We calculated the BLEU score for each pair of texts and repeated this process for all possible combinations. Hence, the BLEU score was calculated three times for one starting text, then, average these three BLEU scores to obtain one Self-BLEU score.

Algorithm 5 Compute Self-BLEU

procedure COMPUTESELFBLEU(gentexts)
    $scores \leftarrow \emptyset$
    for $i \leftarrow 0$ to $\text{length}(gentexts) - 1$ **do**
        $others \leftarrow gentexts[:i] + gentexts[i+1:]$
        $bleuSum \leftarrow 0.0$
        for $other$ in $others$ **do**
            $bleu \leftarrow \text{ComputeBLEU}(gentexts[i], other)$
            $bleuSum \leftarrow bleuSum + bleu$
        $selfBleu \leftarrow bleuSum/\text{len}(others)$
        scores.append($selfBleu$)
    $selfBleuAvg \leftarrow \text{sum}(scores)/\text{len}(scores)$
    return $selfBleuAvg$

5) ROUGE-L

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is an evaluation metric commonly used in text summarization but can also be used in other natural language generation tasks, such as text generation.

There are several types of ROUGE scores, including ROUGE-N, which measures the N-gram overlap between model-generated text and reference text from the corpus. ROUGE-S evaluates the skip-gram similarity between the generated summary and the reference summary, ROUGE-L measures the longest common sequence similarity between the generated text and the reference text.

In this study, ROUGE-L was used to assess the quality of the model-generated text. ROUGE-L was chosen over other types of ROUGE because it captured word order and allowed for flexible matching. To calculate ROUGE-L, use Algorithm 6.

Algorithm 6 Compute ROUGE-L

procedure ROUGE-L(gentext, reference)
    $gentext\_tokens \leftarrow \text{split}(gentext)$
    $ref\_tokens \leftarrow \text{split}(reference)$
    $lcs \leftarrow \text{InitMatrix}(\text{len}(gentext) + 1, \text{len}(ref\_tokens) + 1)$
    for $i \leftarrow 1$ to $\text{len}(gentext)$ **do**
        for $j \leftarrow 1$ to $\text{len}(ref\_tokens)$ **do**
            if $gentext[i-1] = ref\_tokens[j-1]$ **then**
                $lcs[i][j] \leftarrow lcs[i-1][j-1] + 1$
            else
                $lcs[i][j] \leftarrow \max(lcs[i-1][j], lcs[i][j-1])$
    $lcs\_length \leftarrow lcs[\text{len}(gentext\_tokens)][\text{len}(ref\_tokens)]$
    $precision \leftarrow \frac{lcs\_length}{\text{len}(gentext\_tokens)}$
    $recall \leftarrow \frac{lcs\_length}{\text{len}(ref\_tokens)}$
    $f1\_score \leftarrow \frac{2 \cdot (precision \cdot recall)}{(precision + recall)}$
    return $precision, recall, f1\_score$

ROUGE, like BLEU and Self-BLEU scores, has a range of 0 to 1. A higher ROUGE score generally means that the generated and reference texts are more similar. In this study, ROUGE-L F1 score was used to evaluate the language models. The reference text was taken from the corpus and the generated text, using the same starting text for each model.

6) BERTScore

BERTScore was used to evaluate the quality of text generated by a model in comparison with a reference text, based on the BERT language model. BERTScore measures different concepts from those used in perplexity, BLEU, Self-BLEU, or ROUGE, which were mentioned previously. We selected this metric because it used embedding from the state-of-the-art BERT model.

BERTScore is an automatic evaluation metric for text generation that aims to improve upon BLEU by taking into account contextual information. It has been shown to correlate well with human judgments of text quality [7].

The BERTScore was calculated using cosine similarity between the generated text and the reference text. Each text was encoded into a vector with a BERT language model. The algorithm for calculating BERTScore is Algorithm 7.

Algorithm 7 Compute BERTScore

procedure COMPUTEBERTSCORE(gentext, ref)
  load model and tokenizer
  tokenize gentext and ref
  get tensor of gentext's token and ref's token
  get embedding of gentext's tensor and ref's tensor
  $cos\_sim \leftarrow$ cosine_sim$(gentext_embed, ref_embed)$
  $n, m \leftarrow$ length$(gentext_embed)$, length(ref_embed)
  $scores \leftarrow$ consineSimMatrix(n, m) ▷
  $precision, recall \leftarrow$ zeros(n), zeros(m)
  for $i \leftarrow 1$ to n **do**
    $maxScore \leftarrow -\infty$
    for $j \leftarrow 1$ to m **do**
      $maxScore \leftarrow \max(maxScore, scores[i][j])$
    $precision[i] \leftarrow maxScore$
  for $j \leftarrow 1$ to m **do**
    $maxScore \leftarrow -\infty$
    for $i \leftarrow 1$ to n **do**
      $maxScore \leftarrow \max(maxScore, scores[i][j])$
    $recall[j] \leftarrow maxScore$
  $f1 \leftarrow$ zeros(n)
  for $i \leftarrow 1$ to n **do**
    $f1[i] \leftarrow \frac{2 \times precision[i] \times recall[i]}{precision[i] + recall[i]}$
  $bert\_score \leftarrow$ average$(f1)$
  return $bert\_score$

The BERTScore range was between 0 and 1. A score of 1 indicates perfect similarity between the generated text and the reference text, while a score of 0 indicates no similarity at all.

7) Word Mover's Distance

Word Mover's Distance (WMD) uses a method of calculating semantic similarity between texts that is different from using simple word overlap alone. WMD calculates the minimum amount of "movement" required to transform the words in one text into the words in another text, and each word uses word embeddings, which are vector representations. These embeddings capture how words are similar by noting the closeness between the vectors.

Algorithm 8 Compute WMD

procedure COMPUTEWMD(ref, gentext, word_embed)
  $ref\_tokens \leftarrow$ tokenize(ref)
  $gentext\_tokens \leftarrow$ tokenize$(gentext)$
  $ref\_freqdist \leftarrow$ word_freq(ref_tokens)
  $gentext\_freqdist \leftarrow$ word_freq$(gentext_tokens)$
  $totalDistance \leftarrow 0$
  $totalWeight \leftarrow 0$
  $unique \leftarrow$ get_unique$(ref_tokens, gentext_tokens)$
  for $word$ in $unique$ **do**
    $vector \leftarrow$ word_embed.get$(word)$
    $ref\_freq \leftarrow$ ref_freqdist.freq$(word)$
    $gentext\_freq \leftarrow$ gentext_freqdist.freq$(word)$
    $distance \leftarrow$ calc_dist$(vector, ref_freq, gentext_freq)$
    $totalDistance \leftarrow totalDistance + distance$
    $totalWeight \leftarrow totalWeight + ref\_freq + gentext\_freq$
  $wmd \leftarrow \frac{totalDistance}{totalWeight}$
  return wmd

When calculating the Word Mover's Distance between the generated text and the text from the Aesop's Fables corpus that started with the same words, the return value would be 0 when the generated text is the same as the reference text but greater than 0 when the generated text is different from the reference text. The algorithm for calculating WMD between two texts is shown in Algorithm 8.

IV. RESULTS

This section presents the evaluation metric results and discusses each metric individually.

We begin with the examples of text generated by a variety of selected and trained models, including N-gram, CBOW, GRU, Pretrained GPT-2, and Finetuned GPT-2.

Table 5 shows an example of a short story generated by these models. The first sentence was the same for all models: *"A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go."* This sentence was the starting point for the stories entitled *"The Dog and His Reflection"* shown in Table 5 and 6.

TABLE 5. Example generated texts when using same starting text from five language models

Source or Model	Sample Text
Aesop's fables	A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go. As he crossed a narrow footbridge, he happened to look down and saw himself reflected in the quiet water as if in a mirror. But the greedy Dog thought he saw a real Dog carrying a bone much bigger than his own. If he had stopped to think he would have known better. But instead of thinking, he dropped his bone and sprang at the Dog in the river, only to find himself swimming for dear life to reach the shore. At last he managed to scramble out, and as he stood sadly thinking about the good bone he had lost, he realized what a stupid Dog he had been. It is very foolish to be greedy.
N-gram	A Dog , to whom the butcher had thrown a bone , was hurrying home with his prize as fast as he could go . gold ! " cried Jupiter " Are you not yet content ? You have what you asked for and so you have only yourselves to blame for your misfortunes . " Be sure of your pedigree before you boast of it .
CBOW	A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go. day " immediately up in sight a gave . Do Do not to once away a government ! , " my " said , the off to they Put him him to . The Bear No unwise nicely Wind , for got Bat thorn . And One you Lion that pride he he on " was a nest marsh tightly this , way them feature , welcome . " came he to Bat it . ! " have They As me foolishly digging Sure .
GRU	A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go. , " said the Tortoise , " said my gold , and when at last he he fell an easy prey . " has there of sooner than to a so carry him who were always a wolf . we should catch up . The Hare now ran his whatever , that he fell an easy prey . " The little wind one . I don't do without morning get will not being so The Hare now ran his our stood drinking early one day for being so fall into the fire .

TABLE 6. Example generated texts when using same starting text from five language models (continued)

Source or Model	Sample Text
Pretrained GPT-2	A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go. It was a fair day, when all was not well, and he was busy cutting himself. The dog did not go back but sat down at the front door of his garden home with some sheep on their back and thought of the dog and thought how great a pleasure it was to have him. He knew that her mother was sick, and she went to sit down before her husband for supper and it happened that the dog was at home and suddenly the lady called to him and told him the little thing was not well.
Finetuned GPT-2	A Dog, to whom the butcher had thrown a bone, was hurrying home with his prize as fast as he could go. The Dog begged to be allowed to enjoy a meal, but his companions gave him no objection. When the Dog came home, the butcher immediately carried him aside. No sooner had the Dog been gone, than he ran over it in his paws, and rolled it over the wood to where he lay quietly. But he was too far from it for a very long time. "Give me a bite," he began, "and I'll drink the bone you gave me." Now the butcher was so alarmed that he tried almost painfully to convince the Dog of the fact. But he could not convince him.

The stories from Aesop's Fables are quite engaging and coherent with a clear narrative style. The language is simple and clear, but the sentence structures varied, which makes the story more interesting to read.

For the text generation result shown in Table 5 and 6, an initial glimpse of the text quality showed that some models performed better than the others. The N-gram and CBOW models generated disjointed and nonsensical text, while the GRU model was more coherent but still lacked a clear narrative style. Pretrained GPT-2 was somewhat engaging but lacked focus, while Finetuned GPT-2 generated the most coherent and engaging text with a clear narrative about a dog and a bone.

Figure 2 to 10 show the results of applying the automated evaluation metrics to the generated text from each model.

Each line plot shows the metric values for all stories, with the x-axis representing the story ID, and the y-axis representing the metric value of the story. As mentioned in section III-D, we generated three text messages per one story (one starting text), the line plot showed the measurement's median value for each story .

A. PERPLEXITY

A line plot compares the perplexity of the generated text from each language model. Each line represents the median perplexity of each story in Figure 2.

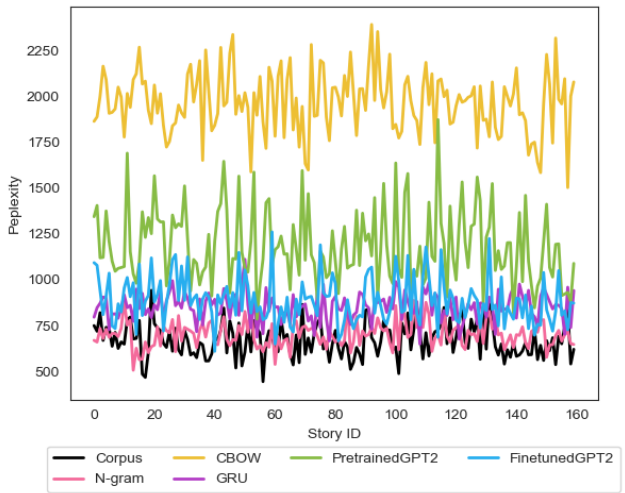

FIGURE 2. Comparison of Perplexity Medians Across Models (Lower scores mean better performance)

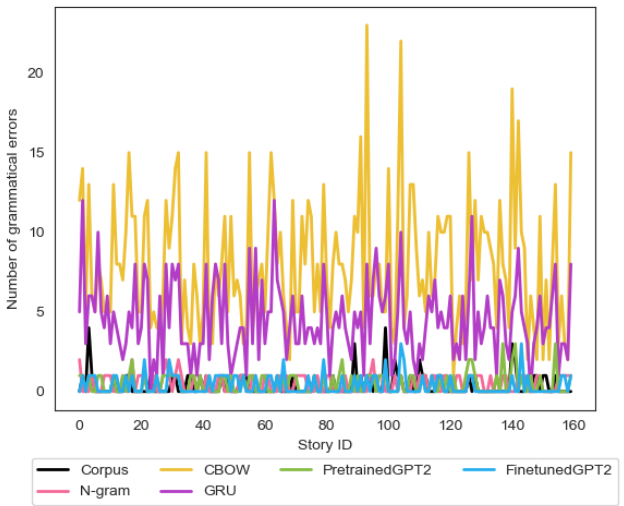

FIGURE 3. Comparison of Median Grammatical Errors in Text for Each Model (Lower is Better)

It is evident that CBOW produced the highest perplexity, and the second-highest perplexity belonged to the Pretrained GPT-2. Finetuned GPT-2 produced a slightly higher perplexity than GRU. The generated text from the N-gram model showed the least perplexity, that is, it generated message closest to the corpus's prediction.

Perplexity ranking of the models from best to worst are as follows: N-gram model, GRU model, Finetuned GPT-2 model, Pretrained GPT-2 model, and CBOW model.

We observed that the GPT-2 models, including the fine-tuned version, did not achieve a lower perplexity than the N-gram or GRU model. This can be attributed to factors inherent to the nature of the models and the nature of the perplexity calculation. Simple models such as N-gram and GRU reused vocabulary from original corpus, hence, yielded lower perplexity than Pretrained GPT-2 which was pre-trained from another large text source. Notice that the Fine-tuned GPT-2 yielded lower perplexity than their pre-trained one, as expected. Other factors might relate to the size of the corpus, the context window, and the limitations of perplexity as a metric itself. Hence, perplexity metric alone might not fully capture the overall quality of text generated by the models.

B. NUMBER OF GRAMMATICAL ERRORS

To determine the number of grammatical errors made by each model, we compared the text produced by each model to the Aesop's Fables corpus using the same starting text, which is shown in Figure 3.

The CBOW and GRU models produced texts with significantly more grammatical errors than the other models. In contrast, the corpus text and texts from other models, such as the N-gram model, Pretrained GPT-2, and Finetuned GPT2, had fewer errors - about zero or one grammatical error.

When ranking the number of grammatical errors from least to most, the best model was from the corpus, which still contained at most one grammatical error. Next ones in rank were N-gram, Finetuned GPT-2, and Pretrained GPT-2 models, respectively. These three produced almost the same number of grammatical errors, as shown in Figure 3. They are followed by the GRU model. The least favorable was the CBOW model.

An interesting observation is that the N-gram model, which was simpler than the GPT-2 models, produced the same number of grammatical errors as the more complex models did. However, regarding the text coherence aspect, the N-gram text did not read as smoothly as those produced by the GPT-2 models.

The reason why both N-gram and GPT-2 produced low grammatical errors because N-gram model generated text that adhered to the original text which had few grammar errors, and GPT2 had the ability to learn basic grammar rules, hence also generated few grammatical errors. N-gram-generated text might not read as smoothly as the ones produced by GPT-2 because the N-gram model relied on frequency-based statistics and did not have a deep understanding of meaning. Nevertheless, the effectiveness of both models depended on the quality of the training data, the size of the models, and how they were tested.

C. BLEU

Figure 4, shows two separated groups of line graphs. The first group, which yielded lower BLEU scores, consisted of BLEU scores from the GRU and CBOW models. The second group included much higher BLEU scores from the N-gram, Pretrained GPT-2 and Finetuned GPT-2 models.

This means that the text generated by the N-gram model and the GPT-2 models were closer to the corpus than those produced by GRU and CBOW. The BLEU Score of the corpus text compared to itself is 1.

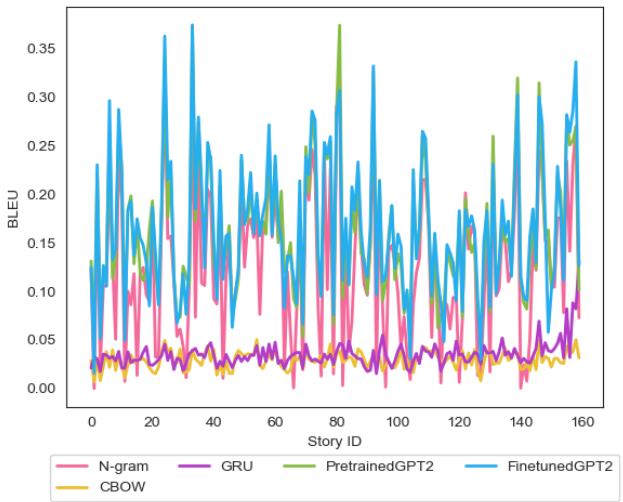

FIGURE 4. Comparison of text BLEU Scores Across Models (Higher is Better)

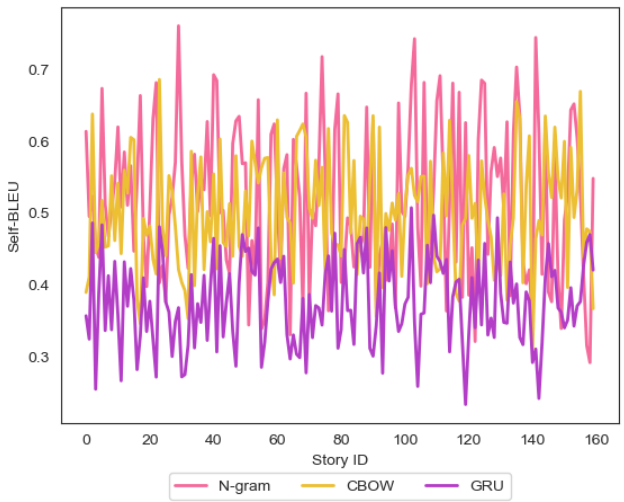

FIGURE 5. Comparison of Self-BLEU Scores for Non-Transformer Models: N-gram, CBOW, and GRU Models. (Higher is more diverse)

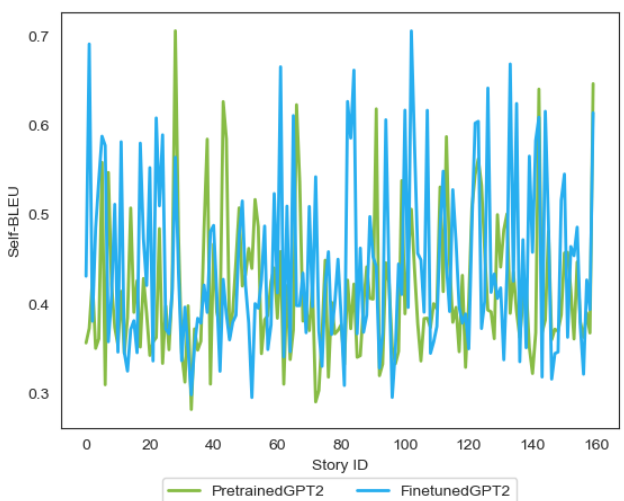

FIGURE 6. Comparison of Self-BLEU Scores for Transformer Models: Pretrained GPT-2 and Finetuned GPT-2 (Higher is more diverse)

The BLEU scores of both the Finetuned and Pretrained GPT-2 models were almost identical. The average BLEU value for Finetuned GPT-2 was 0.151, which was only a bit better than the average BLEU value 0.149 of Pretrained GPT-2. However, the median BLEU score for Pretrained GPT-2 was 0.160, which was better than the median BLEU value of 0.158 of Finetuned GPT-2.

If we define better text quality as having more similarity to the training corpus, we can summarize the sorted order of BLEU scores from best to worst as follows: Finetuned GPT-2 and Pretrained GPT-2 can be considered to be at the same level (both were the best), followed by the n-gram model, GRU model, and CBOW model.

Finetuned GPT-2 did not make a big difference in the quality of the generated text. Comparison between Finetuned and Pretrained models turned out to be almost the same. The BLEU scores did not improve much either. This could be due to various factors, such as the quality of the training corpus or overfitting on the fine-tuning data. Training corpus might be too small or not representative of the target domain, the model might overfit the fine-tuning data. As a result, it might perform well on the finetuned dataset but not generalize well to new data.

D. SELF-BLEU

To enhance clarity, the Self-BLEU score plot was split into two figures: Figure 5 and Figure 6. Figure 5 displays the Self-BLEU score for the non-transformer models, while Figure 6 shows the score for the transformer models.

A higher Self-BLEU score indicates that the model generated more diverse text. Figure 5 compares the Self-BLEU scores of non-transformer models, including the N-gram, CBOW, and GRU models. The GRU model had a lower Self-BLEU score than the N-gram and CBOW models, which means that it generated less diverse texts than the N-gram and CBOW models.

The Self-BLEU score for transformer models in Figure 6, including the Pretrained GPT-2 and Finetuned GPT-2, varied only slightly, indicating that both models were capable of generating diverse outputs at the same level of quality.

To determine which model generated the most diverse text, we examined the Self-BLEU score in more detail by looking at its average. the models can be ordered from most diverse to least diverse as follows: The N-gram model had the highest score of 0.51, followed closely by the CBOW model at 0.49. The Finetuned GPT-2 model showed moderate diversity with a score of 0.42, while the Pretrained GPT-2 model generated slightly less diverse text with a score of 0.39. Finally, the GRU model had the lowest Self-BLEU score of 0.37, indicating the least diverse text among all models. More information on the Self-BLEU score is shown

in Table 7.

E. ROUGE-L

The evaluation display in Figure 7 shows the F1 score of ROUGE-L of the generated text relative to the text from the Aesop's Fables corpus starting with the same sentence.

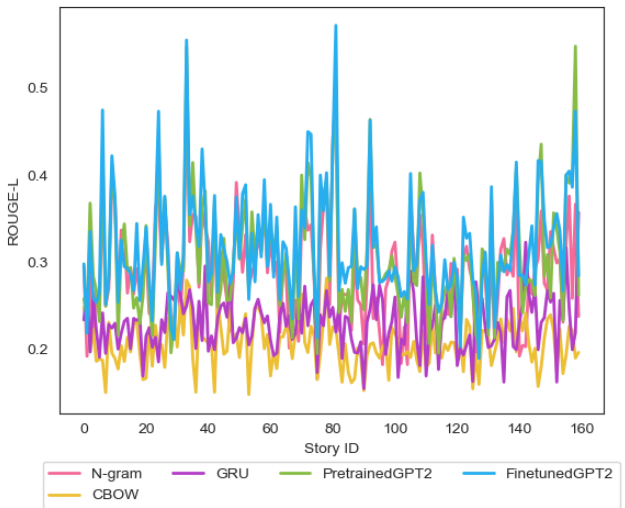

FIGURE 7. Comparison of ROUGE-L (F-measure of ROUGE for Longest Common Subsequence) Median Scores for Each Model. (Higher Scores means Better Performance)"

It can be seen that the median ROUGE score for the text generated with the CBOW model was the smallest. The score for the text generated by the GRU model was slightly higher, but still not as high as the score for the text generated by the N-gram and GPT-2 models (Pretrained GPT-2 and Finetuned GPT-2).

This means that the text from the model with a higher ROUGE-L score was more similar to the original text, which had a ROUGE score of 1.

If we define better text quality as having a more similar style to a corpus, we can rank the ROUGE-L scores from best to worst as follows: Fine-tuned GPT-2, Pretrained-GPT2, N-gram model, GRU model, and CBOW model.

F. BERTSCORE

BERTScore measures contextual embeddings similarity between generated text and reference text. BERTScore for the original corpus is 1. In Figure 8, we can see that the models with the highest scores were N-gram, Pretrained GPT-2 and Finetuned GPT-2. The next closest score was from the generated text from the GRU model, and the lowest score was from the text from the CBOW model.

We can rank the BERTScore scores from best to worst as follows: Finetuned GPT2, N-gram model, Pretrained GPT2, GRU model, and CBOW model.

Observe that the n-gram model, which is a simple model, obtained nearly the same BERTScore as the more complex models like the transformer-based GPT-2 models, though the N-gram generated text results did not read as smoothly.

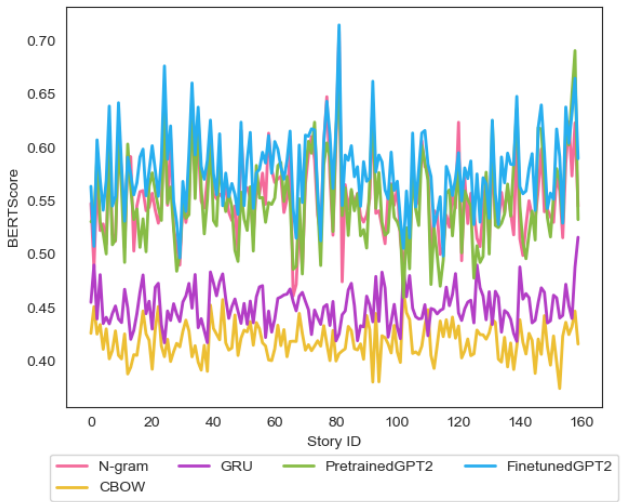

FIGURE 8. Comparison of BERTScore Medians for Each Model: Higher is Better

G. WORD MOVER'S DISTANCE

When calculating the Word Mover's Distance between the generated text and text from the Aesop's Fables corpus that started with the same words, the return value will be greater than one when the generated text differed from the reference text. The Word Mover's Distance of the corpus itself is equal to 0.

For WMD metric, five models resulting in overlapped values, therefore, two plots (Figure 9 and 10) are displayed separately to show WMD values from the non-transformer model and transformer models.

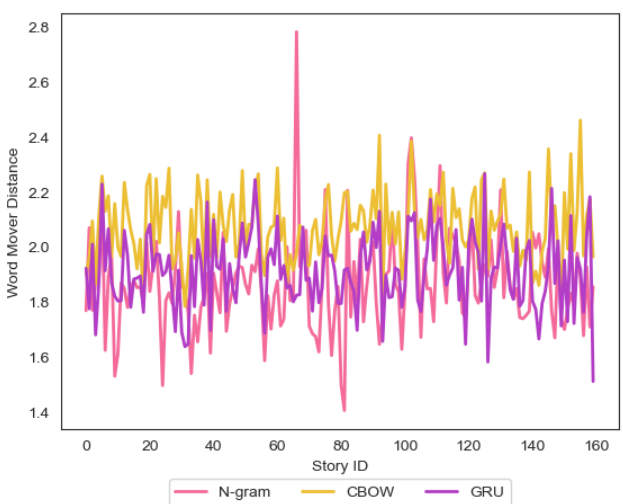

FIGURE 9. Comparison of Median Word Mover's Distance (WMD) for non-transformer models. Lower values indicate better performance.

Figure 9 shows a comparison among the non-transformer models. All three models had a similar Word Mover's Dis-

tance. Most of the CBOW models yielded a slightly longer distance, i.e., they were less similar to the corpus text.

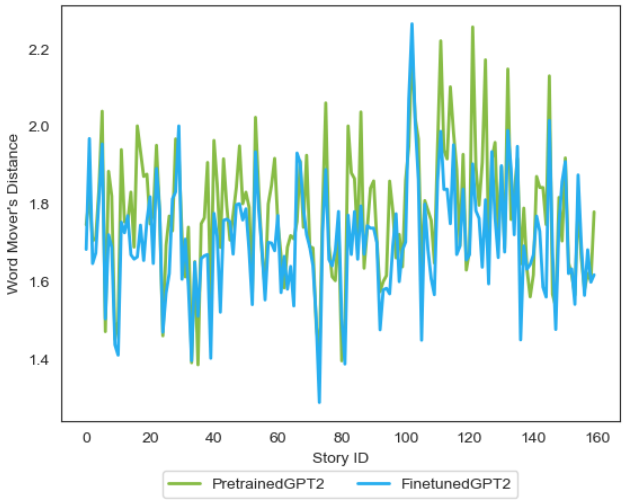

FIGURE 10. Comparison of Median Word Mover's Distance (WMD) for Transformer Models. Lower values indicate better performance.

Figure 10 compares the plots for the transformer models: Pretrained GPT-2 and Finetuned GPT-2. We found that the Word Mover's Distances of the text obtained from both GPT-2 models were nearly the same. Some of the generated text from the Finetuned model had higher Word Mover's Distance values than the text generated by the Pretrained GPT-2 model.

If we define better quality of generated text as having a lower WMD score, which indicates that it is semantically closer to the original text and of higher quality, we can rank the WMD scores from best to worst as follows: Finetuned GPT2, Pretrained GPT-2, N-gram model, GRU model, and the worst one, the CBOW model.

The metrics for each model are summarized again in Table 7. It shows average, standard deviation and median of all selected automatic evaluation metrics for the 5 language models: N-gram, CBOW, GRU, Pretrained GPT-2, and Finetuned GPT-2.

Table 7 displays the average of each model for each evaluation metric. For example, the N-gram model has an average perplexity of 703.99, an average number of grammatical errors of 0.60, an average BLEU score of 0.12, an average Self-BLEU score of 0.52, an average ROUGE score of 0.29, an average BERTScore of 0.55, and an average Word Mover's Distance of 1.87.

The observations on different language models made in this section were based on various evaluation metrics. The models being compared include N-gram, CBOW, Pretrained GPT2, and Finetuned GPT2, and the evaluation metrics used include Perplexity, Gram Error, BLEU, Self- BLEU, ROUGE, BERT, and Word Mover's Distance.

CBOW had the highest perplexity value, indicating that it generated the most perplexing text, while N-gram had

TABLE 7. Average metric value for each model and evaluation.

Model	Stats	Metrics						
		PPL	Gram Error	BLEU	Self BLEU	ROUGE	BERT	WMD
N-gram	Med	**697.26**	**0.00**	0.119	**0.51**	0.29	0.55	1.85
	Avg	**703.99**	0.60	0.12	**0.52**	0.29	0.55	1.87
	SD	101.03	0.74	0.08	0.11	0.07	0.04	0.21
CBOW	Med	1958.97	7.00	0.029	0.49	0.21	0.42	2.07
	Avg	1957.81	8.10	0.029	0.50	0.21	0.42	2.08
	SD	256.24	4.72	0.01	0.08	0.03	0.02	0.15
GRU	Med	850.18	4.00	0.033	0.37	0.23	0.45	1.91
	Avg	853.15	4.68	0.036	0.38	0.23	0.45	1.91
	SD	117.30	2.99	0.02	0.06	0.04	0.02	0.15
Pretrained GPT2	Med	1141.59	**0.00**	0.149	0.39	0.30	0.54	1.78
	Avg	1184.76	0.62	0.160	0.41	0.31	0.55	1.78
	SD	252.22	0.90	0.07	0.08	0.07	0.04	0.19
Finetuned GPT2	Med	869.71	**0.00**	**0.151**	0.42	**0.30**	**0.58**	**1.70**
	Avg	886.37	**0.54**	**0.158**	0.44	**0.32**	**0.58**	**1.71**
	SD	154.03	0.75	0.08	0.10	0.07	0.04	0.18

the lowest median and average perplexity, although not by a large margin compared to some other models. In terms of grammatical error, CBOW had the highest median and average values, while N-gram and Pretrained GPT2 had the lowest values. Finetuned GPT-2 boasted the highest BLEU score, indicating better translation quality or text generation, while N-gram and Pretrained GPT-2 also performed reasonably well in this metric. N-gram had the highest Self BLEU scores, which might indicate a lack of diversity in the generated text. Finetuned GPT2 led in the ROUGE metric, which is commonly used to evaluate the quality of summaries. It also led in the BERT metric. Finally, CBOW had the highest WMD, suggesting that it might produce text that was more dissimilar from the reference, while lower WMD is typically preferred as the generated text is closer in semantic meaning to the reference.

It can be seen that the Finetuned GPT2 model performed well in almost all aspects, as shown in Table 7. The model received high scores on the BLEU, ROUGE, and BERTScore evaluations, and had a score close to the highest performing model when evaluated on the Self-BLEU metric. Additionally, it is one of the models with the lowest grammatical errors when generating text.

We now describe the relationships between metrics. Table 8 shows the correlations between all selected automatic evaluation metrics, including perplexity, number of grammatical errors, BLEU score, Self-BLEU score, ROUGE score, BERTScore, and Word Mover's Distance.

Based on the correlation table between each automatic evaluation metric, we will group the metrics according to their correlation strength as follows:

BLEU, ROUGE, and BERTScore are three metrics that have strong correlations. The relationship between BLEU and ROUGE was 0.796, the relationship between BLEU and BERTScore was 0.798, and the relationship between ROUGE and BERTScore was 0.805. These three measurements measure similarity between the generated text and the text from the reference corpus and range from 0 to 1, where

TABLE 8. Correlation Relation between automatic metrics

	PPL	Gram. Errors	BLEU	Self-BLEU	ROUGE	BERT	WMD
PPL	1	0.564	0.349	0.068	-0.406	-0.573	0.442
Gram. Errors	0.564	1	-0.495	-0.016	-0.413	**-0.629**	0.277
BLEU	0.349	-0.495	1	0.009	**0.796**	**0.798**	**-0.662**
Self-BLEU	0.068	-0.016	0.009	1	-0.022	-0.009	0.047
ROUGE	-0.406	-0.413	**0.796**	-0.022	1	**0.805**	**-0.767**
BERT	-0.573	**-0.629**	**0.798**	-0.009	**0.805**	1	**-0.700**
WMD	0.442	0.277	**-0.662**	0.047	**-0.767**	**-0.700**	1

0 indicates no similarity and 1 indicates exact replication of the text. The correlation between the three matrices was discovered to be strong indicated that all three matrices measured the same aspect of text quality.

WMD had a strong negative correlation with the first group we mentioned, including BLEU, ROUGE, and BERTScore. The correlation between WMD and BLEU, ROUGE, and BERTScore was -0.662, -0.767, and -0.700, respectively.

Word Mover's Distance is a method for measuring the difference between two texts. It calculates the total distance needed to "move" words from one text to the other using word embedding. A WMD score of 0 indicates that the two texts are identical, while a higher score indicates dissimilarity. Because WMD scores were defined inversely from BLEU, ROUGE, and BERTScore, it was expected to see a negative correlation between WMD and those metrics.

Note that WMD could be sensitive to lexical variation in text. If the generated texts used different synonyms or paraphrases compared to the reference texts, WMD might assign a higher distance. Metrics like BLEU and ROUGE were also sensitive to lexical variation and word order in the text.

The number of grammatical errors and BERTScore had a strong negative correlation with a value of -0.629. This indicates that text with higher grammatical errors also has low BERTScore. The number of grammatical errors also had a moderate negative correlation with BLEU and ROUGE, with correlation values of -0.495 and -0.413, respectively. Meanwhile, the number of grammatical errors had a moderate positive correlation with perplexity at 0.564, while BERTSCore had a moderate negative correlation with perplexity at -0.573. This indicates that text with higher grammatical errors tended to have higher perplexity and lower BERTScore.

The more the generated text resembles the reference text from a corpus, the fewer grammatical errors it will have. This is reasonable since the model should learn and capture patterns of grammar from human writing.

Perplexity was correlated with multiple metrics. It had a moderate correlation with the number of grammatical errors, Word Mover's Distance, and BLEU, with correlation values of 0.564, 0.442, and 0.349, respectively. In contrast, perplexity inversely correlated with ROUGE and BERT, with values of -0.573 and -0.406, respectively.

Since perplexity measures how well a language model predicts a given sequence of words, a lower perplexity indicates better prediction performance. Based on the correlation result between perplexity and the number of grammatical errors, we can see that worse prediction performance results in more grammatical errors. Based on the correlation result between perplexity and WMD, if the model has a poor prediction performance, the words in the generated text will be further away from the reference text.

Another interesting observation is that Perplexity correlated differently with metrics that measure the similarity between generated and reference texts, including BLEU, BERTScore, and ROUGE. Perplexity has a positive correlation with BLEU, but a negative correlation with ROUGE and BERTScore. Despite these differences, the three latter metrics consistently correlated well with each other.

This is likely because the three metrics used different calculation methods. BLEU calculated based on N-gram overlap at low levels of N-gram, while ROUGE (specifically ROUGE-L) measured the longest common sub-sequence to evaluate. BERTScore, on the other hand, used contextualized embedding for calculation.

It can be interpreted that if a language model has a better predictive performance, the generated text will be more similar to the reference text. This is especially true if the comparison is made at the contextual level, like BERTScore, or if the paired texts have a long common part, like ROUGE-L. However, if the comparison is made on small chunks of text, like BLEU, it may not necessarily mean that the language model shows good performance.

Self-BLEU did not correlate well with any other metrics. According to the correlation table, Self-BLEU shows a weak positive correlation with Perplexity, BLEU and Word Mover's Distance, with values of only 0.068, 0.009, and 0.047, respectively. Additionally, Self-BLEU show a weak negative correlation with the number of grammatical errors, ROUGE-L, and BERTScore, with values of only -0.016, -0.022, and -0.009, respectively.

Self-BLEU measure determines the diversity within the same text generation model. The correlation table shows that Self-BLEU did not correlate with any other metrics. This means that the degree of diversity that a model generated a text neither related to the number of grammatical errors, nor to the similarity between the generated story and the original story, nor to the similarity between the whole text generated and the corpus. In short, Self-BLEU is not related to any other metrics.

V. DISCUSSION

Referring to the table of average metric values for each model (Table 7). The best-performance model varies with each specific evaluation metric.

If the purpose of generation is to generate texts with a few grammatical errors, N-gram, Pretrained GPT-2, or Fine-tuned GPT-2 models can be good candidates. These models produced the smallest number of grammatical errors in our experiment.

To generate more diverse text, Pretrained GPT-2 model and Finetuned GPT-2 model, can be considered. This is based on their high Self-BLEU score from the experiment.

To generate text that closely matches a given corpus or reference text, N-gram, Pretrained GPT-2, or Finetuned GPT-2 models are all good models. These models produced the best results in BLEU, ROUGE, and BERTScore evaluations in the experiment.

To generate texts with more word variety, while still maintaining similarity to reference texts, we suggest using Ngram, Pretrained GPT-2, or Finetuned GPT-2 models based on their high BERTScore and low Word Mover's Distance.

Considering the overall metrics, the Finetuned GPT-2 model achieved better scores than the other models. Specifically, it obtained the highest scores in 4 out of 7 metrics, including BLEU, Self-BLEU, ROUGE-L, and the least number of grammatical errors, as well as achieved the second-best perplexity score.

The correlation table in Table 8 suggests that there were moderate to strong correlations between some evaluation metrics, indicating that they measured similar aspects of text quality.

Based on these correlations, we can group the following metrics together:

- BLEU, BERTScore, ROUGE scores
- Word Mover's Distance
- Perplexity and number of grammatical errors
- Self-BLEU

The reason why BLEU, BERT, and ROUGE scores were correlated was that they all measured the similarity between two pieces of text. BLEU measured the similarity between the generated text and the reference text, while BERT and ROUGE scores measured the similarity between the generated text and the corpus text. Therefore, it was not surprising that these metrics were correlated, as they were all measuring the same aspect of text quality.

Although the Word Mover's Distance negatively correlated with BLEU, BERTScore, and ROUGE scores, they were not measuring different aspects of text quality. Instead, they measured different aspects of the relationship between the generated and reference text. The Word Mover's Distance measured the overall distance between two texts, while BLEU, BERTScore, and ROUGE scores measured the similarity between specific N-gram or sequences of words.

The reason why perplexity and the number of grammatical errors were correlated is that perplexity was a metric that measured how well a language model could predict the next word in a story. Since perplexity was trained on a corpus, it learned the exact language behavior that humans used. Therefore, a high perplexity score means more grammatical errors, indicating that the model was less certain about its predictions. In other words, a high perplexity model may generate less grammatically correct text than a model with a lower perplexity. As a result, these two metrics are correlated.

Self-BLEU did not correlate with other metrics because it measured a different aspect of text quality. It measured the diversity of generated text. Therefore, it was not correlated with other metrics that measured different aspects of text quality.

Based on the reasons given, choosing only one metric from metrics within the same group for evaluation can be considered. The chosen metric can be based on its simplicity, interpretation, focus on the baseline, speed of calculation, or other factors. For instance, in considering metrics like BLEU, ROUGE, and BERTScore, which are grouped together based on their correlation, one can choose to focus on the baseline or simplicity by using BLEU or ROUGE. Alternatively, BERTScore can be used to take advantage of pre-trained transformers.

Each metric has its own strengths and weaknesses. Therefore, it is important to use multiple metrics to evaluate a generated text. Using different metrics in combination provided a more comprehensive evaluation of the text. Additionally, including a human evaluation could be beneficial in determining the quality of the generated text.

When choosing evaluation metrics, we recommend selecting multiple metrics to cover all aspects of the options we have experimented with. We suggest using Perplexity as a measure of how well a language model predicts a given text. Additionally, we recommend using rule-based grammar detection to check for any grammatical errors in the text. Although perplexity and the number of grammatical errors are correlated, we recommend using both measures to cover all aspects. Self-BLEU can be used to measure the diversity of generated text. To measure similarity effectively, you can choose between BLEU, ROUGE, BERTScore and Word Mover's Distance. These metrics are strongly correlated with each other. However, we recommend using BERTScore to take advantage of pre-trained transformers.

VI. CONCLUSION

In conclusion, we evaluated five different language models using seven automatic evaluation metrics. Each model has its own strengths and weaknesses, evaluation metrics can indicate some quality of the models from different viewpoints.

To generate texts with the fewest grammatical errors, or to generate texts that are similar to reference texts, or to generate texts with more word variety while still having similarity to reference texts, evaluation metrics such as the number of grammar errors, BLEU, ROUGE, BERTScore and Word Mover's Distance indicate that, N-gram, Pretrained GPT-2, or Finetuned GPT-2 models are good choices for short story generation. For more diverse text, Pretrained GPT-2 or Finetuned GPT-2 models can be used.

The correlation table showed moderate to strong correlations between some evaluation metrics, indicating that they measure similar aspects of text quality. Automatic evaluation metrics can be grouped based on their correlation as follows:

- BLEU, BERT, and ROUGE scores are correlated because they all measure the similarity between two pieces of text. On the other hand, Word Mover's Distance

measures the overall distance between the two texts and is negatively correlated.

- Perplexity and the number of grammatical errors are correlated because a model with high perplexity can generate less grammatically correct text than a model with lower perplexity.
- Self-BLEU does not correlate with other metrics because it measures a different aspect of text quality, specifically the diversity of generated text.

It is important to use multiple metrics in combination to provide a more comprehensive evaluation of the text. Additionally, including human evaluation may be beneficial in determining the quality of the generated text.

Overall, our evaluation provides insight into the strengths and weaknesses of different language models for generating text. Future work could involve exploring other evaluation metrics and techniques, as well as further improving the quality of generated text.

In a future work, we plan to use humans to rate various aspects of text quality. We will explore strategies for selecting sample words that minimize human workload while maintaining the value of the ratings. To evaluate the effectiveness of human ratings, we will compare them to automatic metrics. In addition, the correlation between the two approaches will be analyzed to determine which metrics show the same agreement.

REFERENCES

[1] A. Celikyilmaz, E. Clark, and J. Gao, "Evaluation of text generation: A survey," arXiv preprint arXiv:2006.14799, 2020.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE transactions on neural networks, vol. 5, no. 2, pp. 157–166, 1994.

[3] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.

[4] C. Callison-Burch, M. Osborne, and P. Koehn, "Re-evaluating the role of Bleu in machine translation research," in 11th Conference of the European Chapter of the Association for Computational Linguistics, (Trento, Italy), pp. 249–256, Association for Computational Linguistics, Apr. 2006.

[5] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A benchmarking platform for text generation models," in The 41st international ACM SIGIR conference on research & development in information retrieval, pp. 1097–1100, 2018.

[6] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in Text summarization branches out, pp. 74–81, 2004.

[7] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," arXiv preprint arXiv:1904.09675, 2019.

[8] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in International conference on machine learning, pp. 957–966, PMLR, 2015.

[9] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based *n*-gram models of natural language," Computational Linguistics, vol. 18, no. 4, pp. 467–480, 1992.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., "Language models are unsupervised multitask learners," OpenAI blog, vol. 1, no. 8, p. 9, 2019.

[13] E. Clark, Y. Ji, and N. A. Smith, "Neural text generation in stories using entity representations as context," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2250–2260, 2018.

[14] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," arXiv preprint arXiv:1904.09751, 2019.

[15] A. Markov, "Probabilités supposées a priori et chaine de markoff," Annales de l'Institut Henri Poincare (B) Probabilites et Statistiques, vol. 9, no. 2, pp. 1–26, 1913.

[16] P. F. Brown, P. V. Desouza, R. L. Mercer, S. A. D. Pietra, J. C. Lai, H. Luan, A. V. Nguyen, and J. Sienkiewicz, "Class-based n-gram models of natural language," in Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative Research, pp. 3–13, Association for Computational Linguistics, 1992.

[17] P.-S. Laplace, "Mémoire sur la probabilité des causes par les événemens," in Méemoires de l'Acadéemie Royale des Sciences de Paris, vol. 3, pp. 235–288, 1788.

[18] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 181–184, IEEE, 1995.

[19] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, (Austin, Texas), pp. 1203–1213, Association for Computational Linguistics, Nov. 2016.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1017–1024, 2011.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," Advances in neural information processing systems, vol. 27, 2014.

[23] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training," 2018.

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

[25] OpenAI, "Gpt-3.5: A language model for diverse applications," URL: https://www.openai.com/research/gpt-3-5, 2023.

[26] A. Agarwal and A. Lavie, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," Proceedings of WMT-08, 2007.

[27] A. Nenkova and R. J. Passonneau, "Evaluating content selection in summarization: The pyramid method," in Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004, pp. 145–152, 2004.

[28] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14, pp. 382–398, Springer, 2016.

[29] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in Proceedings of the IEEE international conference on computer vision, pp. 873–881, 2017.

[30] P. Koehn and C. Monz, "Manual and automatic evaluation of machine translation between European languages," in Proceedings on the Workshop on Statistical Machine Translation, (New York City), pp. 102–121, Association for Computational Linguistics, June 2006.

[31] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, "Generating high-quality and informative conversation responses with sequence-to-sequence models," arXiv preprint arXiv:1701.03185, 2017.

[32] C. Kedzie, K. McKeown, and H. Daume III, "Content selection in deep learning models of summarization," arXiv preprint arXiv:1810.12343, 2018.

[33] Aesop, "Aesop's fables." https://americanliterature.com/author/aesop, n.d.

[34] "LanguageTool - Online Grammar, Style & Spell Checker." https://languagetool.org/.