Survey:- Natural Language Generation: Case Studies in Natural Answer Generation and High Compression Summary Evaluation

Dharmendra Thakur and **Pushpak Bhattacharyya**
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
{dharmendracse, pb}@cse.iitb.ac.in

Abstract

In the Question Answering domain, creating a "full-length answer" from a factual answer becomes crucial to elaborate a more conversational experience for the user. A reading comprehension system extracts a portion of text containing named entities and other information and serves as the response to a query (known as a "factoid answer"). In this survey paper, we explain the method takes as input a query as well as the extracted factoid answer and generates a full-length natural answer by using a pointer generator network model, sequence to sequence generation model, and a rule-based model. A rule-based model (RBM) that leverages a constituency and dependency parse tree of questions is developed. A transformer-based grammatical correction model GECToR, can be utilized as a post-processing step. This survey also includes the related work in the field of text summarization. Summary Evaluation is a critical task and becomes more critical when a summary is highly compressed. We also include various standard summary evaluation metrics, i.e., ROUGE, BLEU, BERTScore, LS-Score, etc. Summary evaluation metrics can be reference-free and reference-based.

1 Problem Statement

Natural Answer Generation

“Generate a Natural Response i.e., generate a full-length paraphrased natural answer, given a question and its factoid answer as input.”

- Sample Input 1:
  - Question : When were the normans in normandy?
  - Factoid Answer : 10th and 11th centuries
- Output 1: Any 1 of the 2 below
  - During the **10th and 11th centuries** , the normans were in normandy.
  - The normans were in normandy during the **10th and 11th centuries**.
- Sample Input 2:
  - Question : Who was the duke in the battle of hastings ?
  - Factoid Answer : william the conqueror
- Output 2: Any 1 of the 2 below
  - The duke in the battle of hastings was **william the conqueror**.
  - **William the conqueror** was the duke in the battle of hastings.

High Compression Summary Evaluation

Develop an algorithm to show the relevance of generated summary to the Source text, i.e., develop an evaluation metric to calculate the relevance score of the summary to the source text.

2 Motivation

QA systems are frequently used by applications like task-oriented conversational agents or chatbots to deliver factually accurate responses to queries, but they also need to create Natural Language Responses. QA systems often return a text span in the context of the question or a Knowledge Base fact triplet (Subject, Predicate, Object). It is a natural expansion of existing state-of-the-art QA systems to generate full-length natural responses. Exploration of hybrid neural methods that combine abstractive and extractive techniques and rule-based systems that use constituency and dependency parsing to answer the query. Unlike conversational chatbots that mimic human conversation without having to be factually correct or task-oriented dialogue systems that place the retrieved answer in a predefined template, our system generates accurate full-length paraphrased answers automatically, enhancing the system’s utility in these situations. This system, which blends template-based replies with neural-based responses not confined to a limited collection of templates, may be utilised in any task-specific scenario where natural answers are needed. For example, for every product, manuals are there in the maintenance domain, and it becomes a very tedious task to remember all the technical information provided in the user manual. Hence the QA systems extract the answer for the query asked by the users and give the answer, which is generally a factoid-type answer. To enhance the user experience, our system works as a post-processing step for the QA system. Instead of giving only a factoid answer as a response to the query, our system generates a full-length paraphrased and human-like response to the user.

The challenge of assessing the quality of a summary is quite difficult (Steinberger et al., 2009). There are still challenges with the best techniques and types of evaluation. The performance of summarization systems may be compared on several different grounds. A system summary can be compared to the original or source text, a human-given summary, or another system summary. There are two major kinds of summarization assessment techniques. In extrinsic evaluation, the summary quality is judged based on how helpful summaries are for a given task. In intrinsic evaluation, it is directly based on an analysis of the summary. The latter can include a content comparison with a human-written abstract or a comparison with the source material, assessing how many of the source document's significant ideas are covered by the summary. The difficulty in comparing the system summary to an "ideal summary" is that the ideal summary is difficult to define. The human summary might be from the author's article, by an expert asked to construct an abstract, or by an expert asked to extract sentences. A document may have many abstracts that may be used to summarise it. At the same time, content evaluations measure the ability to identify the key topics, text quality evaluations by an expert, and automatic summaries' readability, grammar, and coherence

3 Background Terminology

This section serves to acquaint the reader with the definitions, abbreviations, and phrases used in subsequent sections so that the context is apparent. Natural Answer Generation is the challenge of obtaining a full-length answer using a query and its factual answer as input. While some transformer-based models exist, such as the Modified pointer generator model, our goal in this paper is to discuss the various methods present to solve the task of Natural Answer Generation. The paraphrasing method includes the generation and selection of paraphrases using one of the methods mentioned in this section. Paraphrasing techniques identify, generate, or extract phrases, sentences, or longer natural language expressions that carry almost the same information, i.e., they may differ syntactically but are semantically similar to the target or expected response.

The style transfer is also included in this survey paper. According to the current NLP, the goal of style transfer is to change the style of a sentence without significantly changing its meaning, indicating that style transfer systems' outputs should be paraphrases of their inputs. The input to the third step is essentially the output from the second phase, converting the input text's style or tone from informal to formal. Other styles also exist, such as formal to casual, which might be helpful in the scenario where we are considering nature of the user(s) also.

To better comprehend all of the examples, it is necessary to grasp the colour-coding, which shows that the exact match is the 'black' colour. Note that exact match also considers case-sensitive cases, i.e., Nearly and nearly are separate tokens here. 'green' colour represents the corrected words in the generated answers. The 'brown' colour denotes the substitution of a word or the usage of a synonym, while the 'blue' colour denotes the text's/words converted to a formal style.

Question	what is the current strength of the winds ?
Factoid answer	140 mph
Target	the current strength of the winds is 140 mph .
Generated	the current strength of wind is 140 mph .
Paraphrased	the current speed of winds is 140 mph .

Following are the terms that need to be understood to get the context properly:

3.1 Parsing Methods

The task of generating a parse tree from a given sentence is known as parsing in computational linguistics. A parse tree is a tree that reveals the syntactical structure of a sentence using formal grammar, such as the connections between words or sub-sentences. The resulting tree will have distinct features depending on the sort of grammar we use. Constituency and dependency parsing are two separate types of grammar-based techniques. The resulting trees will be considerably different because they are based on very distinct assumptions. Although the eventual goal in both cases is to extract syntactic information.

Constituency Parser

The constituency parse tree is based on context-free grammars' formalism. The sentence is broken into constituents in this type of tree, which are sub-sentences that belong to a given grammar category. The grammar specifies how to construct proper sentences by following a set of rules. The rule VP –> V NP, for example, states that we can form a verb phrase (VP) from a verb (V) and then a noun phrase (NP). While these rules can be used to construct valid sentences, they can also be used to extract the syntactical structure of a given sentence using grammar. Let's start with a simple sentence: "I saw a fox." Here's an example of a constituency parse tree:

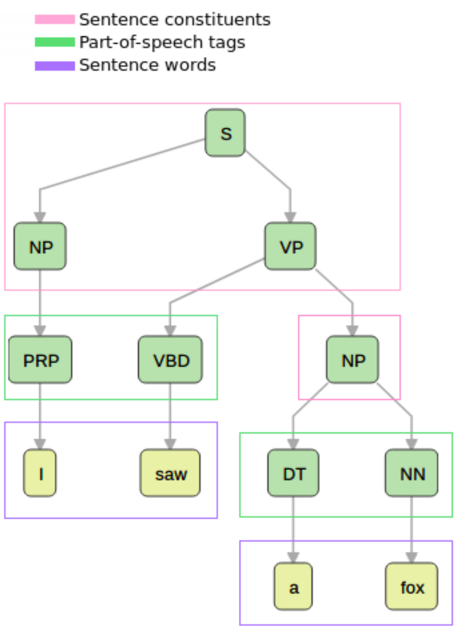

Figure 1: Constituency parser example illustration

This signifies that the grammar contains a rule like S –> NP VP, which means that a sentence can be formed by joining a noun phrase and a verb phrase. The verb phrase is also broken down into a verb and a noun phrase. As you may expect, this corresponds to another grammar rule.

Dependency Parser

Dependency parsing, unlike constituency parsing, does not use phrasal constituents or sub-phrases. The sentence's syntax is instead described in terms of word dependencies — that is, directed, typed edges between words in a graph.

A dependency parse tree is a graph G = (V, E) with the set of vertices V containing the sentence's words and each edge in E connecting two of them. Three requirements must be met by the graph:

- A single root node with no incoming edges is required.
- There must be a path from the root R to each node v in V.
- Except for the root, each node must have exactly one incoming edge.

Each edge in E also has a type, which specifies the grammatical relationship between the two words. Let's examine what happens if we conduct dependency parsing on the preceding example:

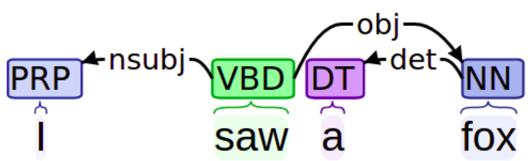

Figure 2: Dependency parser example illustration

As you can see, the outcome is quite different. The tree's root in this technique is the sentence's verb, while the edges between words represent their relationships.

The word "saw," for example, has a nsubj outgoing edge to the word "I," indicating that "I" is the nominal subject of the verb "saw." We say that "I" is dependent on "saw" in this example.

3.2 Question Types

Question Answering is a Natural Language Processing task in which a human/user asks questions in natural language and expects a relevant answer from the algorithm. There can be a different types of questions and based on that approaches may vary.

Different types of questions(Reddy et al. 2017) are :-

Factoid Type Questions

These questions are fact-based. Factoid questions frequently begin with "wh" words. for instance, What is the capital of India? In most cases, the answers are named entities.

List Type Questions

These are the queries that ask for a list of answers, as in Name ten comedic movies, please?. A list of the named entities will be provided to answer this query. Another illustration is: List the steps to reduce the freezer temperature? The response will be a list of statements in some sequence rather than a list of named things.

Bool Type Questions

These are the questions in which the answer is a boolean (either yes or no). An example of confirmation-type questions is: Does the sun rises in the east? The answer is simply yes or no.

Non Factoid Type Questions

These are open-ended questions that require complex answers to answer them. These can be opinions, descriptions, explanations. An example of non-factoid questions is How to read research papers? The answer to this question will contain some opinion and the answer will be descriptive.

3.3 Tools

There are numerous programmes available on the internet. However, in this study, we will explore three of the most important tools that are used to meet the task of natural answer creation. AllenNLP, SBERT, and Huggingface are the tools.

AllenNLP

In PyTorch, AllenNLP [1] provides a complete platform for handling natural language processing problems. They offer a diverse set of existing model implementations that are well-documented and developed to a high standard, making them an excellent starting point for further investigation. We are solely using this library's constituency and dependency parsing features.

SBERT

SBERT [2] is a platformdarkblue that offers a variety of pre-trained models for generating sentence embeddings that capture the semantic meaning of the text.

Hugging Face

Hugging Face[3] provides a transformer library. It's utilised extensively in practically every project that involves transformers. It's also significantly utilised in our project. Hugging Face's transformers library provides transformer-based architectures and pre-trained models. Transformers provides APIs that make it simple to download and train cutting-edge pre-trained models. Pretrained models can help you save money on compute, lower your carbon footprint, and save time over training a model from scratch. The models can be applied to a variety of modalities, including text, audio, video, and photos.

3.4 Transfer Learning

When a model is taught to predict the next word, researchers realised that by applying Transfer Learning in NLP, they could take the trained model, slice off the layer that predicts the next word, add a new layer, and train just that final layer — very quickly — to predict the sentiment of a sentence. Remember that the model was trained to predict the next word in the phrase. However, when it processes and converts into the rich representations put into the last layer to predict the next word, it looks to catch much relevant information in a sentence.

3.5 Evaluation Metrics for Summarization

Reference-based Metrics

Reference-based metrics are metrics that are based on human summaries and compare the expert-provided summary to the model-generated summary. The majority of the evaluation metrics for autonomous summarising compare a model-generated summary (i.e. the candidate) to a human-authored summary (i.e. the reference).

Reference-free Metrics

Reference-free metrics are those that are not dependent on human summaries and in which the model

[1]AllenNLP: http://docs.allennlp.org/v0.9.0/api/allennlp.models.constituency_parser.html

[2]SBERT Link: https://www.sbert.net/

[3]Huggingface Link : https://huggingface.co/models

generated summary is compared to the original text via some technique.

3.6 Correlation Coefficient

A correlation coefficient is a numerical measure of a statistical relationship between two variables. The variables could be two columns from a sample of observations or two components of a multivariate random variable with a known distribution. There are several forms of correlation coefficients, each with its own definition and set of features. They all use a scale of 1 to +1, with 1 denoting the strongest possible agreement and 0 denoting the strongest conceivable disagreement.

Spearman's Coefficient

The strength and direction of relationship between two ranked variables is measured using Spearman's rank correlation. It basically gives the measure of monotonicity of a relationship between two variables, i.e. how well a monotonic function can capture the relationship between two variables. The formula for Spearman's rank coefficient is:

$$\rho = 1 - \frac{6\Sigma\ \mathrm{d}_i^2}{n(n^2-1)}$$

Figure 3: Sparman's correlation coefficient formula

The Spearman Rank Correlation might be anywhere between +1 and -1.

- A value of +1 denotes a perfect rank relationship.
- There is no correlation between ranks if the value is 0.
- A value of -1 denotes a perfect negative rank relationship.

4 Related Work

In this paper, (Pal et al., 2019a) question answering and task-oriented conversation systems have attracted a lot of attention. In this paper, (Weston et al., 2015) author(s) presents a set of challenges for utilizing rule-based systems to infer and answer the question. Paraphrasing can be thought of as a type of bidirectional textual entailment, and the methods used in both fields are frequently quite similar. This (Gadag and Sagar, 2016) paper thesis focuses on paraphrase and textual entailment recognition, as well as paraphrase generation. They provide three methods for detecting para-textual and textual entailment, all of which have been evaluated against existing benchmarks. Back-Translation (Krishna et al., 2020) is particularly effective to get the paraphrase version of input text.

4.1 Natural Language Generation

Introduction

Natural Language Generation (nlg) encompasses both text-to-text and data-to-text conversions . In this paper (Dong et al., 2021) NLG is defined as "the subfield of artificial intelligence and computational linguistics concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information". Clearly, this definition matches data-to-text generation better than text-to-text generation, and it focuses solely on the former, presenting the rule-based approaches that dominated the area at the time in a helpful and clear manner. Natural language generation has various applications, however in this research we will focus on two well-known applications: Natural Answer Generation and Text Summarization. We will start with Natural Answer Generation and studies linked to it, then move on to Text Summarization.

Natural Answer Generation

In recent years, Natural Answer Generation (NAG), which generates natural answer sentences for a given topic, has gotten a lot of interest. NAG might offer specific entities fluently and intuitively, which is more user-friendly in the actual world than standard QA methods.

4.2 Natural Answer Generation from Factoid to Full length Answer Generation

Recently (Jain et al., 2021), QA and task-oriented conversation systems have attracted much attention. End-to-end memory networks employ a language modeling architecture that predicts a response by learning query embeddings and input and output memory representations from source sequences. (Weston et al., 2015) puts out a range of tasks for inferring and answering the question using rule-based systems. Introducing specific words into the vocabulary for each knowledge base entity type enhances memory networks and manages out-of-vocabulary (OOV) terms. To recreate facts, these systems rely on templates or specific heuristics. Dialogue systems such as those collect information from knowledge bases to generate a response. After extracting information from documents or external KBs, systems like (Fu and Feng, 2018) employ KB-based key-value memory. On the other hand, these systems are limited to the information described by the KB or slot-value memory. Our approach is general and may be utilised with any structured or unstructured information source, such as a knowledge base or a machine-comprehension dataset.

Modified Pointer Generator(MPG)

This strategy is based on (Jain et al., 2021). The following paragraphs list the key drawbacks of this strategy. Additionally, there were instances of model failure when the model simply produced the question itself. The reason could be because the model became biased toward adding more parts from the question than the factoid answers, which in some circumstances led to a complete copy of the question. The following are the primary categories of failure cases:-

- Incoherent sentence as a result of faulty logic
- Repetition of words item Only produces the factoids as the response
- produces clausal responses
- failure to take morphological differences into account

DialoGPT Model

The main drawback of this methodology is the issue of adding extraneous items to the final responses that are not included in the (Jain et al., 2021) question and the factual answer that is sometimes referred to as hallucination. In certain cases, the final response does not even contain the factoids. Additionally, the DialoGPT model frequently produces mistakes when copying numerical data, such as a year, number, or another item. The model makes a few mistakes when duplicating the appropriate nouns from the questions. In the final response, the names are present but are spelled differently. (For instance: Alexander - Alexanderrick; Elizabeth - Elizabetha). This is also seen by the example in Table 5 when DialoGPT altered the spelling of Arizona to "Arizona." Low BLEU and ROUGE scores are the results of this.

4.3 Answering Naturally : Factoid to Full length Answer Generation

In this (Pal et al., 2019b) paper TIn this paper he authors used two ways to turn the challenge of generating a full-length answer from the question and the factoid answer into a Neural Machine Translation (NMT) task. They developed a model based on the pointer-generator architecture presented in , with a few modifications. On the source side, they use two encoders to encode the question and factoid answer individually, as shown in Figure.

System Architecture

The two encoder pointer generator in the following system architecture diagram uses the question and factoid answer as input to generate a full-length answer in an end-to-end learning environment.

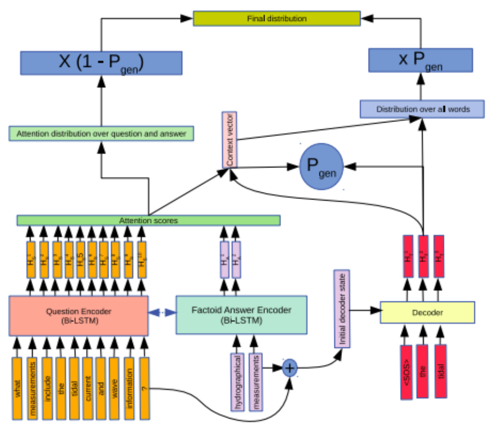

Figure 4: Pointer Generator Network

4.4 Paraphrase Generation Methods

Phrases, sentences, or longer natural language expressions that communicate almost the same information are recognized, generated, or extracted using paraphrasing approaches. On the other hand, Textual entailment techniques identify, create, or extract pairs of natural language phrases in such a way that a human reading (and trusting) the first element of a pair would infer that the other element is likewise true. Paraphrasing can be considered a type of bidirectional textual entailment, and the methods used in both fields are frequently quite similar. Both techniques are helpful in a wide range of natural language processing applications, including question answering, summarization, text creation, and so on, at least in theory.

In this (Gadag and Sagar, 2016) paper, They concentrate on paraphrase and textual entailment recognition, as well as paraphrase creation, in their thesis. They offer three approaches for recognizing paratextual and textual entailment, which have been tested on current benchmarks. The fundamental notion is that we can detect paraphrases and textual entailment quite effectively by capturing similarities at multiple abstractions of the inputs. Back translation, often known as reverse translation, is the process of re-translating material in literal terms from the destination language to the source language. For example, if you are translating material from English to Swedish, the translator will also produce a back translation in English to clarify the translated option's purpose. Back translations do not affect the translator's translation memory or other resources such as glossaries. Back translation (also known as double translation) is especially useful when the information at hand contains taglines, slogans, titles, product names, creative phrases, and puns, as the implicit meaning of the content in one language may not be the same in another. The reverse translation allows the content owner to see the creative license taken by the translators in adapting the text for their target market. Moreover, for sophisticated content, the translator will frequently provide numerous alternatives so that the source content owner may make the best selection for the brand.

Back-Translation-based Paraphrasing

Back-Translation (Krishna et al., 2020) is the process of re-translating content in literal terms from the destination language to the source language. The goal of employing the back-translation principle is to produce the paraphrases of the input text. We are utilizing the hugging-face-based translation model of English to Roman, English to Spanish, English to French, English to Russian, and their Back-Translated version.

4.5 Pre-Trained Model

BERT

Bidirectional Encoder Representations from Transformers, sometimes known as BERT, is a language representation model. BERT intends to pre-train deep bidirectional representations from the unlabeled text by concurrently conditioning both left and right context in all layers. As a result, without making significant task-specific architectural alterations, the pre-trained BERT model may be improved with just one extra output layer to provide cutting-edge models for various tasks, including question answering and language inference. In Figure 5, we display the pre-training.

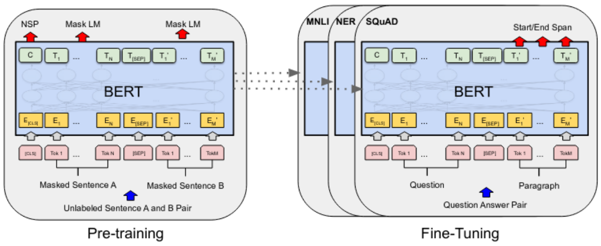

Figure 5: BERT Architecture

Pre-training of BERT includes two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM task, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. In NSP task, we pre-train the model for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus with an eye to understand sentence relationships.

RoBERTa

The RoBERTa (Liu et al., 2019) model improves on BERT by removing the next-sentence pretraining target and training with substantially bigger mini-batches and learning rates. Recently authors proposed adjustments to the BERT pretraining technique that increase end-task performance. They combined these enhancements and assessing their cumulative impact. This setup is known as RoBERTa, which stands for Robustly Optimized BERT Approach.

XLNET

The Transformer-XL model's pre-trained variant XLnet maximises the expected likelihood over all permutations of the input sequence factorization to learn bidirectional contexts using an autoregressive method.

T5

T5 gives a unified framework to solve all the text-based NLP problems (Raffel et al., 2020). T5 comes from the name "Text-to-Text Transfer Transformer". Here, all the problems are treated as text-to-text problems, which means the model takes text as input and produces text as output. We add a

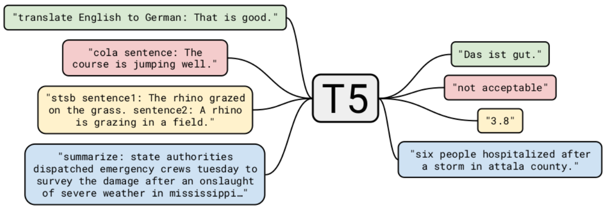

Figure 6: Diagram of T5 framework

text-specific text (which is called “prefix") to the original input sequence to specify the task to the model. Figure **??** shows some input/output examples for T5 framework. In the first example, model gets English sentence “That is good." as input and model generates “Das ist gut.". We can clearly see that we give a prefix “translate English to German:" in addition to the English sentence as input. The second example shows linguistic acceptability. The third example shows a regression problem, which predicts similarity between two sentences. The fourth examples shows summarization. T5 uses Transformer architecture. T5 is pre-trained on a masked language modeling objective, where consecutive spans of input tokens are replaced with a mask token and the model is trained to reconstruct the masked-out tokens. It uses C4 corpus (“Colossal Clean Crawled Corpus”) which contains natural and clean English text of nearly 750 GB size.

GPT-3

Recently (Brown et al., 2020) scaling up language models enhances task-independent, few-shot performance significantly. GPT-3, an autoregressive language model, is trained explicitly with 175 billion (175B) parameters, which is ten times more than any previous non-sparse language model. GPT-3 is used with no gradient updates (zero-shot) or fine-tuning with one-shot and few-shot examples specified solely through text interaction with the model. The Sparse Transformer uses the same model and architecture as GPT-2, except that the layers of the Transformer use alternating dense and locally banded sparse attention patterns, identical to GPT-2. The model sizes range from 125 million (125M) to 175 billion (175B), with the GPT-3 (Brown et al., 2020) model being the largest.

PEGASUS

Recently (Zhang et al., 2020) the authors of "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization" devised a self-supervised pre-training objective (called gap-sentence generation) for Transformer encoder-decoder models to improve fine-tuning performance on abstractive summarization, achieving state-of-the-art results on 12 different summarization datasets. Their theory is that the higher the fine-tuning performance, the closer the pre-training self-supervised target is to the final down-stream assignment. Several complete sentences are removed from documents during PEGASUS pre-training, and the model is tasked with recovering them. A document with missing sentences is an example of a pre-training input, with the output consisting of the missing sentences concatenated together.

The PEGASUS (Zhang et al., 2020) model has also been fine-tuned for the task of paraphrase generation. where the input is a single sentence and the output is a list of the input sentence’s paraphrases.

4.6 Generative Text Style Transfer

Natural language processing (NLP) advancements have sparked renewed interest in generative text models and style transfer challenges. While most research has concentrated on binary sentiment transfer, several recent studies have focused on text formality, a style that is more difficult to describe by particular keywords. In this (Schmidt and Braun) line, we look at the problem of generative text style transfer to improve language sophistication, with the goal of rewriting an input phrase to keep its sense while increasing its complexity to match a target-style text. Early research in the subject concentrated on situations where parallel literature is available, such as the classroom, Modern NLP defines the aim of style transfer as altering the style of a sentence without significantly affecting its meaning, implying that style transfer systems’ outputs should be paraphrases of their inputs. On the other hand, many existing systems are ostensibly built for style transfer, which naturally distorts the meaning of the input through attribute transfer, affecting semantic characteristics such as sentiment. In this article, we reformulate unsupervised style transfer as a para-generation issue and offer a straightforward technique based on fine-tuning pre-trained language models using autonomously generated para-data. Despite its straightforwardness,

4.7 Text Summarization

Automatic text summarising (Steinberger et al., 2009) is a method of extracting the most relevant information from a source text and presenting it in a condensed form tailored to the user's or task's needs. With the fast increase of information available on the internet, the significance of having a text summarising system has grown. Text understanding and production processes are directly linked to the generation of summaries. The original text is read first, and the content is identified. Following that, the main points are condensed into a succinct synopsis. Because the algorithm must grasp the point of a document, summarization is a difficult task. This requires semantic analysis and content categorization based on global knowledge. However, the system will be unable to do so without substantial global information. As a result, attempts at genuine abstraction have been mostly unsuccessful thus far. Fortunately, extraction, an approximation, is now more possible. To create an extract, the system only needs to identify the most significant parts of the text. The issue is that the summary is frequently incoherent. The reader can, nevertheless, develop a judgment on the original material. As a result, most automated systems only create extracts at the moment. Several theories ranging from text linguistics to artificial intelligence have been proposed.

Extractive Text Summarization

Extractive summarization methods work just like that. It takes the text, evaluates all of the sentences based on the text's understanding and relevancy, and then provides us with the most important sentences, basically ranking the sentences using one of the ranking algorithms Textrank (Mihalcea and Tarau, 2004) and Lexrank (Erkan and Radev, 2004). This approach does not generate new words or sentences. Instead, it simply presents the ones that already exist. Consider taking a page of text and using a highlighter to highlight the most significant sentences.

Abstractive Text Summarization

On the other hand, abstractive summarization tries to infer the meaning of the entire source text and then delivers it to us. It constructs words and sentences, assembles them meaningfully, and then adds the most significant facts from the text. Abstractive summarization approaches are more sophisticated and computationally expensive than extractive summarization techniques.

4.8 Standard Metrics for Summary Evaluation

In this (Steinberger et al., 2009) paper, we understood that The task of evaluating the quality of a summary is quite tough. There are still major disagreements concerning the appropriate assessment methodologies and kinds. Various factors may be used to compare the performance of summarization systems. The original text or source text, a human-generated summary, or another system summary can be compared to a system summary. There are two types of summary evaluation procedures.

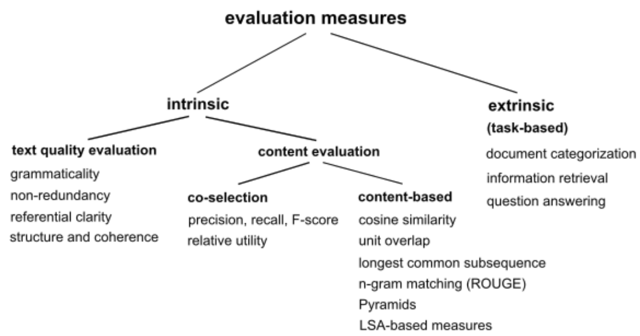

Figure 7: The taxonomy of summary evaluation measures (Steinberger et al., 2009)

The quality of a summary is assessed extrinsically based on how valuable summaries are for a particular job and intrinsically based on analysis of the summary. A comparison with a human-written abstract or a comparison with the source material can be used to determine how many of the original document's key themes are covered by the summary. Comparing the system summary to an "ideal summary" is challenging since the ideal summary is hard to define. The human summary might be from the author of the piece, a judge tasked with creating an abstract, or a judge tasked with extracting sentences. There may be a large number of abstracts that may be used to summarise a material. Text quality assessments examine automated summaries' readability, grammar, and coherence, whereas content evaluations assess the ability to identify significant themes.

ROUGE Scores

ROUGE. This metric has been the most commonly used automatic metric for summary evaluation. It assesses the quality of a summary by comparing it to a reference written by a human. The goal of the comparison is to see how many overlapping units (such as n-grams or word sequences) the summary and reference have (Lin and Och, 2004).

METEOR

METEOR. This metric, proposed by (Banerjee and Lavie, 2005), evaluates a candidate string by comparing its harmonic mean of unigram-precision and unigram-recall to a reference string.

BERTScore

BERTScore. (Zhang et al., 2019) presented this metric using token-level contextual embeddings generated by a pre-trained language model (here, we use BERT). The assessment score is determined by comparing the embeddings of the to-be-evaluated summary to those of the reference. R (recall), P (precision), and F (frequency) are the three measures that make up the BERTScore (F1 score).

WMS/SMS/S+WMS

WMS/SMS/S+WMS. The word mover's distance (WMD) was proposed by (Kusner et al., 2015) to compute the least cost of shifting a sequence into another. Each sequence is treated as a collection of words, with each word represented by its word embeddings. Afterward, the WMD can be converted into a similarity (WMS) (Clark et al., 2019). (Clark et al., 2019) developed a method for measuring the similarity of two sequences by computing the sentence mover's distance to improve the ability to evaluate multi-sentence texts based on WMS. The sentence mover's distance (SMS) and the sentence and word mover's distance (S+WMS) were introduced. S+WMS combines sentence and word embeddings and represents each sequence as a bag of both sentences and words. SMS employs sentence instead of word embeddings and represents each sequence as a bag of sentences.

MoverScore

MoverScore. Also inspired by WMD, (Zhao et al., 2019) encoded the reference and candidate texts as a sequence of n-gram embeddings and calculated the WMD between the two. We present the results of the best models reported in their work, which construct n-gram embeddings using a BERT pre-trained on the MNLI dataset with PMeans as the aggregator.

BERT+Cos+Ref.

BERT+Cos+Ref. The cosine similarity between the embeddings of the reference and the candidate summary is calculated using BERT as the encoder.

BERT+Cos+Doc.

BERT+Cos+Doc. This metric is similar to BERT+Cos+Ref, but it compares the source document to the candidate summary. In the baselines, this is the only statistic that does not have a reference.

4.9 Unsupervised Reference-Free Summary Quality Evaluation via Contrastive Learning

Automatic text summarization and generation have recently seen much success. Evaluation for such systems has been an issue of interest for better comparing and improving model performance. The choice of assessment metrics will significantly impact how well a generated summary is judged, which will impact how well summarization models are evaluated. Human judgment is an ideal measure frequently used as the gold standard. Human evaluation, however, requires a lot of time and energy. It is critical to have an automatic evaluation metric that saves time and simulates human judgment.

Dimension of Evaluation

The authors investigated a few summarization datasets. Figure 8 demonstrates how various datasets consider various evaluation dimensions. The authors found that these characteristics could be generally categorized into three classes: the semantic quality (Semantic), the linguistic quality (Linguistic), and other dimensions that are difficult to categorize (Else).

	Semantic	**Linguistic**	**Else**
DUC-05, DUC- 06 and DUC-07 (Xenouleas et al., 2019)	focus, non redundancy	grammaticality, structure & coherence	referential clarity
Newsroom 60 (Sun and Nenkova, 2019)	relevancy, informativeness, unnecessary content, verbosity	-	perfect surrogate, continue reading
*CNN/Daily Mail (Chaganty et al., 2018)	-	fluency, overall quality, redundancy	-
*Newsroom (Grusky et al., 2018)	informativeness, relevancy	coherence, fluency	-
NYT and CNN/Daily Mail (Sharma et al., 2019)	informativeness	grammaticality, coherence	-

Figure 8: Dimensions for Assessing Different Summarization Datasets. (Wu et al., 2020)

In this study, they build an approach to account for semantic and linguistic quality factors.

Methodology

Linguistic and semantic quality are the two most crucial elements influencing summary qualities. Linguistic quality, which comprises the fluency of each sentence, the coherence of entities/consecutive sentences, and the correctness of grammar, reflects how natural the generated summary is. Semantic quality, which typically comprises informativeness, relevance, redundancy, etc., determines whether a summary conveys the essential information from the source materials. In the sections that follow, consider both factors and outline our strategy. Figure 1 depicts the architecture of our model. The picture is divided into two sections. First, they show how our evaluator is set up to grade summaries using a BERT encoder. The evaluator is then trained using negative samples and a contrastive learning framework.

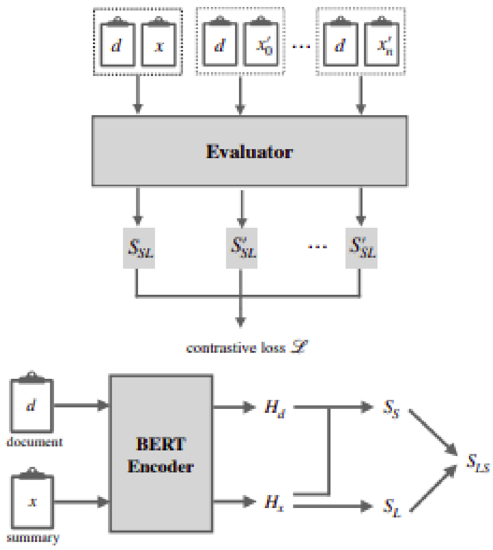

Figure 9: Model Framework. The architecture for contrastive learning is shown in the top picture, in which we generate various kinds of negative samples for each document x and compare them with x to determine a ranking loss. The evaluator, which determines the final evaluation score, is the figure at the bottom. Here, S, L, and SLS stand for S, L, and LS scores, respectively. (Wu et al., 2020)

Contrastive Training

We develop a new unsupervised training framework via contrastive learning. Intuitively, if we make some noise, e.g., disordering the words/sentences, for a given good summary, we can easily create a bad one with worse quality. We use human-generated summaries in the training data as "good" summaries, but they can also be replaced with other machine-generated ones. Since we evaluate the summaries from two different aspects, we create different types of noisy samples for each aspect. For example, one straightforward strategy is randomly removing some words or sentences in the original summary to get a new negative sample. We generate negative samples for various aspects of the summary quality. Negative samples can be generated by either disordering the words/sentences or deleting words. We do not delete entire sentences because most of the summaries have only very few sentences. In our experiments, we generate only one negative sample per type of operation for each base summary.

Datasets

On two benchmark datasets for single-document summarization, we perform empirical research. The original documents, the corresponding human-authored summaries (also known as references), and some model-generated summaries manually rated in several dimensions are all present. These datasets allow us to compare various evaluation techniques based on how well they correlate with human ratings.

	Newsroom	CNN/Daily
# of doc-ref pairs	108,802	10,932
# of sens in doc	31.08	34.20
# of words in doc	861.90	882.25
# of sens in ref	1.43	3.88
# of words in ref	34.90	64.87
# of systems	7	4
# of generated sums	420	1996

Figure 10: Dataset Statistics. (Wu et al., 2020)

NewsRoom

Newsroom. This summarising dataset, proposed by (Grusky et al., 2018), has 1.3 million documents and hand-written summaries. There are only 420 summaries with human evaluations in this collection. Seven different extractive or abstractive summarising systems produced these summaries. Three human raters assessed each document-summary pair in four dimensions (coherence, fluency, informativeness, and relevance).

We use the mean of three raters as the human score for each summary. These summaries with human assessments serve as the basis for our testing. We chose our training data (108,802 document-reference pairs) with no overlapped reference summaries with the test data in order to prevent information from leaking during the training process. This implies we do not use reference summaries when training with test data.

CNN/Daily Mail

CNN/Daily Mail, this dataset was initially developed by (Hermann et al., 2015) for question-answering research utilising newspapers, and it was then expanded to the area of summarization by including human scores for 2,513 references and system-generated summaries in three dimensions (overall, fluency and redundancy). For testing, we employ 1,996 summaries produced by four systems, and for training, 10,932 document-reference pairs. The reference summaries between the training and test sets do not overlap either. The data statistics for the training data are displayed in Table 3. We randomly chose 95

In this (Schmidt and Braun) We investigate the topic of generative text style transfer to increase language sophistication. GECToR (Omelianchuk et al., 2020) GEC sequence tagging system, which has three steps of training: synthetic data pretraining, errorful parallel corpus fine-tuning, and ultimately a mix of errorful and error-free parallel corpora fine-tuning. On the CoNLL-2014 and BEA-2019 datasets, this model produces state-of-the-art outcomes for the problem of grammar Error Correction.

5 Summary

The project's objective is to generate a full-length natural and paraphrased answer given a question and its factoid answer as an input. and along with this to develop a summary evaluation metric to show the relevance of the summary to the source text.

We discussed the Parsing methods, i.e., the constituency and dependency parsing methods, used to develop the rules for the natural answer generation problem. The helpful tools, like AllenNLP, SBERT, and Hugging face library, are also discussed. The idea of Transfer learning and its sub-approaches, for example, zero-shot learning and few-shot learning, are also discussed, which were helpful to fine-tune the GPT-3 model. We also discussed Spearman's correlation coefficient to develop the summary evaluation metric, which we will discuss later. In the literature survey, we explored recent Summarization and Machine Translation techniques used in Neural Natural Answer Generation, wherein we discussed the basic NMT model and attention model for summarization. Then we studied the Pointer Generator Network, covering the baseline and Pointer Generator models. Also, we discussed a very recent work related to our problem statement in detail. Our literature survey explored the recent Paraphrase generation and Style formation methods and Machine Translation based approaches useful to solve the problem of textual diversity in generated answers. We briefly reviewed the concept of a style transfer and used a T5-based style former model to convert the input text's style from casual to formal. The results were then presented professionally, considering all variants of the GECToR model and all paraphrasing approaches specified in the literature survey section for each GECToR variation, i.e., BERT, RoBERTa, and XLNET. We have shown the results for two types of datasets: NewsQA and Confirmatory, and we have done qualitative and error analysis on a few key cases.

We discussed about the various evaluation metric for example, ROUGE Score, BLEU Score, METEOR, BERTScore, WMS/SMS/S+WMS, MoverScore, BERTScore, BERT+Cos+Ref., BERT+Cos+Doc., and othres. We have

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760.

Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. 2021. A survey of natural language generation. *arXiv preprint arXiv:2112.11739*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Ashwini Gadag and BM Sagar. 2016. A review on different methods of paraphrasing. In *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pages 188–191. IEEE.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Manas Jain, Sriparna Saha, Pushpak Bhattacharyya, Gladvin Chinnadurai, and Manish Kumar Vatsa. 2021. Natural answer generation: From factoid answer to full-length answer using grammar correction. *arXiv preprint arXiv:2112.03849*.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: Tag, not rewrite. *arXiv preprint arXiv:2005.12592*.

Vaishali Pal, Manish Shrivastava, and Irshad Bhat. 2019a. Answering naturally: Factoid to full length answer generation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 1–9, Hong Kong, China. Association for Computational Linguistics.

Vaishali Pal, Manish Shrivastava, and Irshad Bhat. 2019b. Answering naturally: Factoid to full length answer generation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 1–9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Robert Schmidt and Spencer Braun. Generative text style transfer for improved language sophistication.

Josef Steinberger et al. 2009. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks.

Hanlu Wu, Tengfei Ma, Lingfei Wu, Tariro Manyumwa, and Shouling Ji. 2020. Unsupervised reference-free summary quality evaluation via contrastive learning. *arXiv preprint arXiv:2010.01781*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.