Inducing Constraints in Paraphrase Generation and Consistency in Paraphrase Detection

A THESIS
SUBMITTED FOR THE DEGREE OF
Doctor of Philosophy
IN THE
Faculty of Engineering

BY

Ashutosh Kumar

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

April, 2023

Declaration of Originality

I, **Ashutosh Kumar**, with SR No. **04-04-00-10-12-16-1-13962** hereby declare that the material presented in the thesis titled

Inducing Constraints in Paraphrase Generation and Consistency in Paraphrase Detection

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2016-2022**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Partha Talukdar: Advisor Signature

DEDICATED TO

My Parents, Suman and Arvind,

who supported me through thick and thin

Acknowledgements

The past six years have been some of the most adventurous years of my life thus far. What I imagined to be a smooth journey at the Indian Institute of Science (IISc) six years back was anything but that. I want to acknowledge the people that kept me sane on this rocky road and paved the way for me to complete the work that is this thesis. This work is a culmination of many technical and non-technical efforts by some of the most brilliant people I have met. I am no self-made man, and it would be untrue and outright arrogant to claim otherwise. I am grateful to be able to stand on the shoulders of giants.

I have been extremely fortunate to have Partha Talukdar as my thesis advisor. Partha is not only a brilliant researcher but also an amazing human being. I learned a lot from his profound understanding of research problems, his ability to break problems into bite-sized chunks, knack for seeing the bigger picture in research, and extreme patience in dealing with me and my questions. I am and will always be grateful for that.

I want to thank Arijit Biswas, and Yi Zhang, for hosting me as an applied research intern in the summer of 2017 and 2020 at Amazon, Bangalore, and AWS AI, Seattle (remote due to COVID), respectively. It was great collaborating with them, and I hope to keep that going in the near future.

I am deeply thankful to Bhuthesh R, who kept the lab infrastructure running smoothly without which doing this research would have been inconceivable. I have been extremely fortunate to share an office with my collaborators. I am thankful to Satwik Bhattamishra, Manik Bhandari, Kabir Ahuja, and Raghuram Vadapalli. They are undoubtedly the best bunch of researchers and collaborators anyone could have asked for. I had an amazing time with all the Machine and Language Learning (MALL) Lab members, IISc. Thanks Chandrahas, Madhav Nimishakavi, Naganand Yadati, Sawan Kumar, and Apoorv Umang Saxena, for hanging out with me and discussing life and research. I am indebted to my collaborator, Aditya Joshi, who guided and pushed me to the final stages of my Ph.D. I learned a lot about writing papers, researching, and life in general from Aditya.

I am blessed with a fantastic set of friends - Vaibhav Sharma, Monika Rana, Swapnil Pathak, Naveen Goel, Chandrahas (yes, again), Suvam Mukherjee, Arpita Biswas, Anirban Laha, Remish Minz, Saneem Ahmed Chemmengath, Rahul Biswas, and Nimisha Pahuja. They helped ensure that I was/am never stressed out beyond reason. I do not think they realize how valuable their encouragement, discussions, and contributions have been. I want to thank them for investing time in helping me be sane.

Most of all, I would like to thank my parents, Suman and Arvind, for encouraging me to follow my heart and giving me every possible privilege that a growing child can hope to have. There would, literally, have been no me and no thesis without them.

Abstract

Deep learning models typically require a large volume of data. Manual curation of datasets is time-consuming and limited by imagination. As a result, natural language generation (NLG) has been employed to automate the process. However, in their vanilla formulation, NLG models are prone to producing degenerate, uninteresting, and often hallucinated outputs [61]. Constrained generation aims to overcome these shortcomings by providing additional information to the generation process. Training data thus generated can help improve the robustness of other deep learning models. Therefore, the central research question of the thesis is:

*"How can we **constrain generation models**, especially in NLP, to produce **meaningful outputs** and utilize them for building better classification models?"*

To demonstrate how generation models can be constrained, we present two approaches for paraphrase generation. Paraphrase generation involves the generation of text that conveys the same meaning as a reference text. We propose two strategies for paraphrase generation:

1. DiPS (**Di**verse **P**araphraser using **S**ubmodularity): The first approach deals with constraining paraphrase generation to ensure diversity, i.e., ensuring that generated text(s) are sufficiently different from each other. We propose a decoding algorithm for obtaining diverse texts. We provide a novel formulation of the problem in terms of monotone submodular function maximization, specifically targeted toward the task of paraphrase generation. We demonstrate the effectiveness of our method for data augmentation on multiple tasks such as intent classification and paraphrase recognition.

2. SGCP (**S**yntax **G**uided **C**ontrolled **P**araphraser): The second approach deals with constraining paraphrase generation to ensure syntacticality, i.e., ensuring that the generated text is syntactically coherent with an exemplar sentence. We propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), an end-to-end framework for syntactic paraphrase generation without compromising relevance (fidelity). Through a battery of automated met-

rics and comprehensive human evaluation, we verify that this approach does better than prior works that utilize only limited syntactic information in the parse tree.

The second part of the research question pertains to ensuring that the generated output is meaningful. Towards this, we present an approach for paraphrase detection to ascertain that the generated output is semantically coherent with the reference text. Paraphrase Detection is the task of detecting whether or not the two input natural language statements are paraphrases of each other. Fine-tuning pre-trained models such as BERT and RoBERTa on paraphrastic datasets has become the go-to approach for such tasks. However, tasks like paraphrase detection are symmetric - they require the output to be invariant with the order of the inputs. In the traditional fine-tuned approach for paraphrase classification, inconsistency is often observed in the predicted labels or confidence scores based on the order of the inputs. We validate this shortcoming and apply a consistency loss function to alleviate inconsistency in symmetric classification. Our results show an improved consistency in predictions for three paraphrase detection datasets without a significant drop in the accuracy scores.

While these works address the research question via paraphrase generation and detection, the approaches presented here apply broadly to NLP-based deep learning models that require imposing constraints and ensuring consistency. The work on paraphrase generation can be extended to impose new kinds of constraints (for example, sentiment coherence) on generation, while paraphrase detection can be applied to ensure consistency in other symmetric classification tasks (for example, sarcasm interpretation) that use deep learning models.

Publications based on this Thesis

The work in this dissertation is based on the following peer-reviewed articles.

1. **Ashutosh Kumar***, Satwik Bhattamishra*, Manik Bhandari, and Partha Talukdar. *"Sub-modular optimization-based diverse paraphrasing and its effectiveness in data augmentation"*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1(Long and Short Papers), pages 3609–3619, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

2. **Ashutosh Kumar**, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar; *"Syntax-Guided Controlled Generation of Paraphrases"*. Transactions of the Association for Computational Linguistics 2020; 8 330–345.

3. **Ashutosh Kumar** and Aditya Joshi; *"Striking a Balance: Alleviating Inconsistency in Pre-trained Models for Symmetric Classification Tasks"*. Findings of the Association for Computational Linguistics 2022. Association for Computational Linguistics.

The following articles were also completed during the course of the Ph.D. but have not been discussed in the dissertation

1. **Ashutosh Kumar**, Arijit Biswas, Subhajit Sanyal. *"Ecommercegan: A generative adversarial network for e-commerce"*. In: 6th International Conference on Learning Representations - Workshop Track Proceedings, ICLR 2018, 30-3 May 2018, Vancouver; Canada.

2. Kaustubh D. Dhole, et. al. (includes **Ashutosh Kumar**). *"NL-Augmenter: A Framework for Task-Sensitive Natural Language Augmentation"*. arXiv preprint arXiv:2112.02721, 2021

3. **Ashutosh Kumar**. *"Discovering Non-Monotonic Autoregressive Ordering for Text Generation Models using Sinkhorn Distributions"*. ICLR Blog Track, 2022.

Softwares released based on this Thesis

1. **Diverse Paraphraser using Submodularity (DiPS)**.
   Paper - Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation: NAACL 2019.
   Source (LSTM) - https://github.com/malllabiisc/DiPS
   Source (Transformers) - https://github.com/GEM-benchmark/NL-Augmenter/tree/main/nlaugmenter/transformations/diverse_paraphrase

2. **Syntax-Guided Controlled Paraphraser (SGCP)** .
   Paper - Syntax-Guided Controlled Generation of Paraphrases
   TACL 2020.
   Source - https://github.com/malllabiisc/SGCP

3. **Alleviating Inconsistency in Pre-trained Paraphrase Detector**
   Paper - Striking a Balance: Alleviating Inconsistency in Pre-trained Models for Symmetric Classification Tasks.
   Findings of ACL 2022.
   Source - https://github.com/ashutoshml/alleviating-inconsistency

Contents

List of Figures

List of Tables

"It is difficult to reconstruct what it was that took us years, long hours of discussion, endless exchanges of drafts and hundreds of e-mails negotiating over words, and more than once almost giving up. But this is what always happens when a project ends reasonably well: once you understand the main conclusion, it seems it was always obvious."

- Daniel Kahneman

Thinking, Fast and Slow [62]

Chapter 1

Introduction

A paraphrase is a restatement (**Y**) of the meaning of a text or passage (**X**) using other words. For example, the sentence *'Giraffes like Acacia leaves and hay and they can consume 75 pounds of food a day.'* is a paraphrase of the text *'A giraffe can eat up to 75 pounds of Acacia leaves and hay every day.'* [51] The two sentences differ in three ways. Firstly, the text uses the singular *'a giraffe'* while the paraphrase generalizes it to the plural *'giraffes'*. Then, the text specifies *'up to 75 pounds'* whereas the paraphrase mentions *'75 pounds'*. Finally, the text refers to the ability of a giraffe to eat Acacia leaves (expressed through *'can eat'*) while the paraphrase conveys a giraffe's *'liking'* for the leaves. It follows that there is no unique paraphrase **Y** of a text **X**, although **X** and **Y** are expected to be semantically similar. As a result, linguistics accepts a more pragmatic definition of paraphrases known as '*quasi-paraphrases*'. Bhagat and Hovy [13] provide a comprehensive list of definitions for quasi-paraphrases.

Paraphrases are often used in the context of describing something in 'one's own words'. This is a technique commonly employed by humans to serve multiple functions. For example, a student who is asked to testify about their understanding of a theoretical concept may paraphrase what they have read in a book. Upon reading or listening to the paraphrase, the teacher is able to validate if the student has understood the concept. Similarly, when describing their research to a layperson, a researcher may eliminate jargon while conveying key ideas. The listener, based on their understanding of the subject, will interpret the paraphrase. It is evident that paraphrases are an important means of communication used to align a message with a listener's background knowledge.

Paraphrasing broadly refers to natural language processing (NLP) tasks related to paraphrases. In the context of **X** and **Y** above, **Y** may simplify the complexity of **X** or summarise the key content of **X**. In this thesis, we focus on two paraphrasing tasks: paraphrase generation (*i.e.*, generate **Y** from **X**) and paraphrase detection (*i.e.*, predict if **X** and **Y** are semantically similar). Paraphrasing finds applications in areas such as text simplification, conversation agents, abstractive summarisation, and more generally, data augmentation.

This thesis improves paraphrase generation and detection models by demonstrating how constraints can be induced in the former and consistency in the latter. In this chapter, we first motivate the problems in paraphrase generation and detection. Following that, we present the research statement of the thesis. In order to build a human-inspired view of the problems involved in paraphrasing, we discuss two case studies of humans performing paraphrasing. We then describe the contributions of the thesis.

1.1 Motivation

Deep learning models in NLP typically use sequential frameworks such as LSTMs and Transformers. In the forthcoming subsections, we first describe the challenges faced by deep learning models and then motivate the need for constraints in paraphrase generation and consistency in paraphrase detection.

1.1.1 Challenges

Deep learning approaches in NLP often face the following challenges:

1. Deep learning models typically require a large volume of data ('High Data Requirement' in Section 1.1.2).

2. In the context of text generation, deep learning models are prone to producing degenerate, uninteresting, and often hallucinated outputs [61] ('Poor Quality Output' in Section 1.1.2).

3. Evaluation of NLG output via automated metrics or human evaluation is fraught with errors. (discussed in Section 1.1.3).

We analyze each of these challenges in the subsequent sections.

1.1.2 Constraints in Paraphrase Generation

Paraphrase Generation may require constraints because of:

1. High Data Requirement: Conventional human-annotated paraphrase datasets are either too small for model training, **have limited variations**, or are domain specific. For instance, the Microsoft Research Paraphrase Corpus (MRPC) [31] is too small a dataset for deep learning generation models, while Quora Question Pairs (QQP) [1], and ParaSCI

[1]https://www.kaggle.com/c/quora-question-pairs

datasets [32] are domain-specific containing only questions and sentences extracted from scientific articles respectively. Paraphrase Adversaries from Word Scrambling (PAWS) [152] is built on top of QQP. Although it contains difficult examples for paraphrase and non-paraphrase pairs, the variations provided by them are still limited. It is, therefore, imperative to enable paraphrase generation models to produce **diverse** outputs using these available datasets.

2. Poor Quality Output: Since many paraphrase pairs in the datasets (as in the case of PAWS) contain only minor lexical variations, models built on top of them may generate sentences with repeating phrases, and limited syntactical differences. This necessitates the induction of constraints. Token and phrase-based constraints have been dealt with in previous works [2, 50, 106]. However, the induction of **syntax-based** constraints has been marginally explored and with limited quality.

1.1.3 Consistency in Paraphrase Detection

The motivation for consistency in paraphrase detection arises from natural language generation (NLG). NLG models are often evaluated either by computing metrics based on word overlap between $\mathbf{X}$ and $\mathbf{Y}$ or via human evaluation. While the former may result in incorrect conclusions, the latter is cost-intensive. Paraphrase detection is a task that can automate the evaluation process by predicting if the expected text $\mathbf{X}$ and the generated output $\mathbf{Y}$ are semantically similar. However, semantic similarity is an equivalent relationship: $\mathbf{X}$ is similar to $\mathbf{Y}$ is the same as $\mathbf{Y}$ is similar to $\mathbf{X}$. Approaches for paraphrase detection so far do not account for this. **Induction of consistency in paraphrase detection** aims to ensure that the equivalent relationship is retained.

1.2 Research Question

The central research question of the thesis is:

*"How can we **constrain generation models**, especially in NLP, to produce **meaningful outputs** and utilize them for building better classification models?"*

To address the question, the thesis considers three problems in paraphrasing: **(a)** Diversity in paraphrase generation (this refers to the '**constrain generation models**' part of the research question), **(b)** Syntacticality in paraphrase generation (this refers to the '**constrain generation models**' part of the research question), and **(c)** Consistency in paraphrase detection (this refers to the '**meaningful outputs**' part of the research question). Figure 1.1

summarises the scope of the thesis.

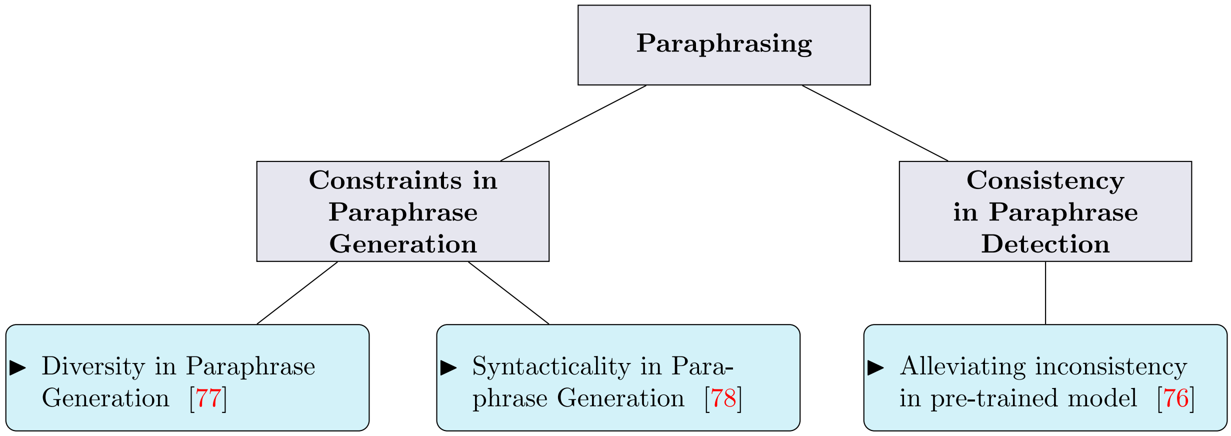


Figure 1.1: Scope of this thesis

Goal: This thesis presents approaches to induce constraints in paraphrase generation and consistency in paraphrase detection.

1.3 Human Perspective

While this thesis deals with automated approaches for paraphrase generation and detection as elicited in the research statement, it is useful to understand how humans perform and benefit from paraphrasing. Towards this, we present a related human perspective of paraphrasing. When humans generate or detect paraphrases, they must often deal with constraints and consistency. We demonstrate this through four case studies. For paraphrase generation, we use the case study of science journalism and advertisement generation, while for paraphrase detection, we describe claim verification and plagiarism detection.

1.3.1 Science Journalism

Science journalism [115] is a field that deals with converting scientific articles into easily comprehensible text that non-experts can consume. For example, in the context of machine learning, an effort towards science journalism was made in the tenth International Conference on Learning Representations (ICLR 22), called ICLR-Blog-Post-Track[1]. The main motive of science journalism is to disseminate accurate and jargon-free information about a scientific article to the masses for inclusion and initiation of a healthy dialogue between society and science. Science

[1] https://iclr-blog-track.github.io/

journalists face an increasing need to convey factually correct information through storytelling techniques, like stylizing sentences in a way that taps the emotional and rational sides of their audience. When writing an article catering to a general audience, a science journalist needs to **constrain** their text such that it is accurate, easily comprehensible, personalized for each demography, and engaging enough to achieve its objective.

While science journalism appears to be a paraphrasing task where a science journalist would simplify a scientific article, it poses peculiar challenges. The audience's demography needs to be considered before wording a certain text. Even within the scientific community, scientists from different areas find it difficult to engage in a healthy debate because of the lack of common vocabulary. The literary proficiency of the audience also plays an important role. In a study based on text simplification [121], it was observed that readers prefer sentences based on their literary skills. For example, consider the pair of sentences: **S1** *Because it is raining today, you should carry an umbrella.* and **S2**: *You should carry an umbrella today because it is raining.* While these are paraphrases of each other, it was found that fifth-grade readers preferred sentence **S2** where the cause follows the effect. In contrast, college students preferred sentence **S1** where the effect follows the cause.

Therefore, in most cases, it is helpful for science journalists to provide diverse (in terms of style and detail) paraphrases for the same text so that it can target the right audience appropriately.

1.3.2 Advertisement Generation

Advertisement Generation [19] is the art of creating taglines, slogans, and marketing messages that sell a product or service. A well-crafted advertisement can attract potential customers, build brand recognition, and increase sales. However, an ineffective or poorly designed advertisement can lead to a loss of sales and damage the company's reputation. It is important to note that advertisements should be sensitive and catchy enough to attract potential customers while also being *truthful to the product catalogue*. This truthfulness implies that the tagline must be a near-paraphrase of a specific component of the product catalogue.

For example, consider the following two slogans: *"Buy our colorful shoes now! They are the best"* and *"Experience comfort at your feet, step up your game and walk on cloud nine with our shoes! Available in multiple colors! Buy now!"* Although both advertisements have the same intent, it is easy to see how the second slogan could appeal more to users. Advertisement writing is a creative endeavor, and its authors need to consider the target demographic, market sentiments, and authenticity.

In most cases, advertisers try to provide structurally different yet meaning-preserving text to target the right population appropriately.

1.3.3 Claim Verification

Claim verification is the task of assessing the trustworthiness of information in a text. For example, social media posts during the COVID-19 pandemic often contained new information about the infection. Considering the constantly evolving understanding of the infection, when a post potentially contained information about COVID-19, several social media websites such as Instagram and Facebook would automatically add a tag urging readers to verify the claims made in the post separately. This is a case of paraphrase detection by humans.

Consider a social media post *'Drinking turmeric milk reduces your chances of catching COVID-19'*. A user wanting to verify the claim in the post may use a search engine with the keyphrases *'turmeric milk'* and *'COVID'*. Based on their level of expertise, they may read sources such as news articles or research papers. When they encounter a text in the source related to the claim in the social media post, they would check if the two are paraphrases of each other. If they are, the user would conclude that the claim is trustworthy. On a related note, multi-lingual speakers can verify claims using sources from multiple languages. For example, upon reading the code-mixed Hindi post *'haldi doodh peene se COVID nahi hoga (drinking turmeric milk will not give you COVID)'*, a multilingual speaker may still search for English terms *'turmeric milk'* and *'COVID'* to obtain English articles that are potentially related to the claim.

When performing claim verification, a human reader may read articles from multiple sources. Achieving **consistency** in paraphrase detection ensures that the reader arrives at the same conclusion regarding the claim, irrespective of the sequence in which the articles are read.

1.3.4 Plagiarism Detection

Plagiarism detection [40] is a critical task in maintaining academic integrity and ensuring legal compliance. It involves identifying instances where a section of text has been copied from another source without appropriate attribution. This task is crucial for protecting intellectual property rights and assessing the originality of work in academic settings. For instance, Turnitin[1] is a well-known software that is specifically designed to detect instances of text copying. However, to assess the originality of an article, a human requires sufficient knowledge about prior works and access to search engines. The typical approach involves analyzing key signals, such as sudden changes in tone or phrasing, and verifying the text's originality or attribution through search engines in cases where non-original or paraphrased content is suspected.

[1] https://www.turnitin.com/

Despite its importance, plagiarism detection is a labor-intensive and time-consuming task that involves assessing the consistency of the two texts under consideration.

1.4 Contributions of the Thesis

Pertaining to the three problems (Section 1.2), we now describe our work in terms of major findings and contributions to paraphrase generation and detection.

Inducing Diversity in Paraphrase Generation. The first problem deals with inducing diversity in the task of paraphrasing. This problem has applications in data augmentation and conversational agents. We find that previous paraphrasing approaches mainly focused on the issue of generating semantically similar paraphrases while paying little attention to diversity. In fact, most of the methods rely solely on top-k beam search sequences to obtain a set of paraphrases. However, the resulting set often contains many structurally similar sentences.In this work, we focus on the task of obtaining highly diverse paraphrases while not compromising on paraphrasing quality. We provide a novel formulation of the problem in terms of monotone submodular function maximization, specifically targeted to paraphrasing. Additionally, we demonstrate the effectiveness of our method for data augmentation on multiple tasks such as intent classification and paraphrase detection.

Inducing Syntacticality in Paraphrase Generation. We induce syntactical styles in paraphrases via controlled text generation. Syntax-guided paraphrasing deals with generating paraphrases that follow a reference syntactic style. Such syntactically coherent paraphrases find applications in tasks such as text simplification. Specifically, we look at problems where, in addition to the input sentence to be paraphrased, the syntactic guidance is sourced from a separate exemplar sentence. We find that prior works in syntax-guided paraphrasing have only utilized limited syntactic information available in the parse tree of the exemplar sentence. We address this limitation in the paper and propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), an end-to-end framework for syntactic paraphrase generation. We find that SGCP can generate syntax-conforming sentences without compromising relevance. We perform extensive automated and human evaluations over multiple real-world English language datasets to demonstrate the efficacy of SGCP over state-of-the-art baselines. In addition to these approaches, we also present a dataset: QQP-POS. This is a subset of the human-curated dataset - QQP, for syntactic paraphrase generation.

Inducing Consistency in Paraphrase Detection. Finally, we also show how consistency can be introduced in paraphrase detection, which is modeled as a classification task. While fine-tuning pre-trained models for downstream classification is the conventional paradigm in NLP, task-specific nuances may not get captured in the resultant models. Specifically, for tasks that take two inputs and require the output to be invariant of the order of the inputs, we observed inconsistencies in the predicted label or the confidence score. We propose a consistency loss function to alleviate inconsistency in symmetric classification. Our results show an improved consistency in predictions for three paraphrase detection datasets without a significant drop in the accuracy scores. Additionally, we examine the classification performance of three tasks (both symmetric and non-symmetric) to showcase the strengths and limitations of our approach.

While these works address the research question via paraphrase generation and detection, the approaches presented here apply broadly to NLP-based deep learning models that require imposing constraints and ensuring consistency.

1.5 Organization of Thesis

The thesis is organized as follows. In Chapter 2, we discuss some definitions related to paraphrases, highlight related works, and develop a technical background on sequence-to-sequence architectures, pre-trained model (BERT), and some subset selection strategies. We then begin Part 1 of the thesis by discussing a decoding time strategy for obtaining a diverse set of paraphrases (Chapter 3) and then obtaining syntax-guided paraphrases via controlled-text generation (Chapter 4). In Part 2, we elucidate the inconsistencies in the pre-trained paraphrase detection model and present an additional objective to alleviate the problems (Chapter 5). Finally, we summarize the key contributions of this thesis in Chapter 6 followed by potential future directions arising from this thesis.

Chapter 2

Background

In this chapter, we aim to provide the necessary background material for the following chapters. Given that the primary focus of this thesis is on **paraphrases**, we begin by defining what paraphrasing entails. We then delve into some prior work on paraphrase generation and detection before presenting technical details that will make it easier to understand the rest of the thesis.

2.1 What is a Paraphrase?

A paraphrase is a restatement ($\mathbf{Y}$) of the meaning of a text or passage ($\mathbf{X}$). The restatement can include, but not be limited to:

(a) Lexical Variation: Lexical variation involves basic edit operations like replacement with synonyms, swapping of words or phrases, insertion of informative content, and deletion of redundant content. For example, the sentences '*I don't want this.*' and '*I do not want this.*' are paraphrases of each other where word contraction pairs '*don't*' and '*do not*' are used in the place of each other.

(b) Semantic Variation: This is one of the primary requirements of a good paraphrase. The meaning of the rearrangement should not deviate too much from the original sentence that needs to be paraphrased. E.g., '*I ate a fruit for breakfast*' and '*I consumed a fruit for breakfast*' are paraphrases of each other because synonyms 'ate' and 'consumed' are used in place of each other.

(c) Syntactic Variation: Changing the structure of the sentence while preserving meaning refers to syntactic variation. The sentences '*What is the height of the table*' and '*What is the table height*' are syntactic variations. The first sentence uses a noun phrase, while the second uses a noun compound.

(d) Pragmatic Variation: These refer to two sentences that may appear different on the surface but carry the same implied meaning or potential impact. For example, the sentences '*Can you please get me some water?*' and '*Get me some water!*' are paraphrases for each other since the listener is expected to fetch water in both scenarios.

The concept of pragmatics encompasses bidirectional or two-way entailment between two sentences. If two sentences entail each other, they convey the same meaning. The only exception to this rule is the scenario where the two sentences are identical. In that case, they are not considered paraphrases.

Although the definition of paraphrases requires strict equivalence of semantics, linguistics accepts a more pragmatic alternative, allowing broader and approximate equivalence. Informally, this entails acceptance of more examples or *"quasi-paraphrases"*. However, because of the approximation, it is not easy to put-down a single fully-explanatory definition of paraphrases. Bhagat and Hovy [13] provide a comprehensive list of definitions for these *"quasi-paraphrases"*. We state some example "quasi-paraphrase"-pairs (**X**, **Y**) below.

1. **Text Simplification**
   (a) **X**: The conference will be held in the main auditorium of the university, which is located on the west side of campus.
   (b) **Y**: The meeting is at the university's big hall on the west side of campus.
2. **Intentions in Conversational Agents**
   (a) **X**: Kindly elucidate your statement for me.
   (b) **Y**: What do you mean?
3. **Active-Passive Voice with External knowledge**
   (a) **X**: Animal Farm was written by George Orwell.
   (b) **Y**: George Orwell who was born in the Bengal Presidency, British India, wrote the Animal Farm.
4. **Summarization**
   (a) **X**: The film was widely praised by critics for its visually impressive cinematography and the storyline that kept viewers engaged.
   (b) **Y**: The movie received critical acclaim for its stunning cinematography and compelling storyline.

5. **Approximate Numerical Equivalences**

    (a) **X**: Disneyland is 32 miles from here.

    (b) **Y**: Disneyland is around 30 minutes from here.

The above examples are utilized in different contexts depending on various factors such as the audience's demographic and preferences.
In the rest of the thesis, we will refer to all such "quasi"-variations as **Paraphrases**. Like the above examples, we have taken the broader, pragmatic, and ubiquitously accepted definition(s) of paraphrases.
We now discuss some prior works in context of paraphrase generation and detection, highlight the strengths and the shortcomings of the related approaches, before building the necessary technical foundations essential for understanding the rest of this thesis.

2.2 Related Work

Paraphrasing a given sentence is an important problem and numerous approaches have been proposed to address it. Recently *sequence-to-sequence* based data-driven deep learning models have been proposed, which try to address the limitations of earlier traditional rule-based [98] methods. Prakash et al. [107] employ residual connections in LSTM to enhance the traditional sequence-to-sequence model. Gupta et al. [45] provide a variational auto-encoder (VAE) [65] based framework to improve the quality of generated paraphrases. Li et al. [86] propose a reinforcement learning based model which uses pointer-generator [116] for generating paraphrases and an evaluator based on [104] to penalize non-paraphrastic generations. Several other works [17, 58] exist for paraphrasing, though they have either been superseded by newer models or are not directly applicable to our settings. However, most of these methods focus on the issue of generating semantically similar paraphrases, while paying little attention to diversity.

Diversity in paraphrasing models was first explored by [45] where they propose to generate variations based on different samples from the latent space in a deep generative framework. Although diversity in paraphrasing models has not been explored extensively, methods have been proposed to address diversity in other NLP tasks [83, 82, 44]. Diverse beam search proposed by [133] generates k-diverse sequences by dividing the candidate subsequences at each time step into several groups and penalizing subsequences which are similar to prior groups. The most relevant to our approach is the method proposed by [124] for neural conversation models where they incorporate diversity by using DPP to select diverse subsequences at each time step. Although their work is addressed in the scenario of neural conversation models, it could be naturally adapted to paraphrasing models and thus we use it as a baseline.

Submodular functions have been applied to a wide variety of problems in machine learning [56, 60, 71, 69] and have recently attracted much attention in several NLP tasks including document summarization [90], data selection in machine translation [67] and goal-oriented chatbot training [30]. However, their application to sequence generation is largely unexplored.

Data augmentation is a technique for increasing the size of labeled training sets by leveraging task-specific transformations which preserve class labels. While the technique is ubiquitous in the vision community [72, 110], data augmentation in NLP is largely under-explored. Most current augmentation schemes involve thesaurus-based synonym replacement [151, 140], and replacement by words with paradigmatic relations [68]. Both of these approaches try to boost the generalization abilities of downstream classification models through word-level substitutions. However, they are inherently restrictive in terms of the diversity they can offer. Our work offers a data-augmentation scheme via high-quality paraphrases.

Controllable Text Generation is an important problem in NLP which has received significant attention in recent times. Prior works include generating text using models conditioned on attributes like formality, sentiment or tense [52, 119, 149] as well as on syntactical templates [58, 20]. These systems find applications in adversarial sample generation [58], text summarization, and table-to-text generation [105]. While achieving state-of-the-art in their respective domains, these systems typically rely on a known finite set of attributes thereby making them quite restrictive in terms of the styles they can offer.

Paraphrase generation. While the generation of paraphrases has been addressed in the past using traditional methods [98, 10, 108, 47, 153, 95, 145], they have recently been superseded by deep learning-based approaches [107, 45, 88, 87, 77]. The primary task of all these methods [107, 46, 87] is to generate the most semantically similar sentence and they typically rely on beam search to obtain any kind of lexical diversity. Kumar et al. [77] try to tackle the problem of achieving lexical, and limited syntactical diversity using submodular optimization but do not provide any syntactic control over the type of utterance that might be desired. These methods are therefore restrictive in terms of the syntactical diversity that they can offer.

Controlled Paraphrase Generation. Our task is similar in spirit to Iyyer et al. [58], Chen et al. [20], which also deals with the task of syntactic paraphrase generation. However, the approach taken by them is different from ours in at least two aspects. Firstly, SCPN [58] uses attention [6] based pointer-generator network [116] to encode input sentences and a linearised constituency tree to produce paraphrases. Due to the linearization of the syntactic tree, a lot of dependency-based information is generally lost. Our model, instead, directly encodes the tree structure to produce a paraphrase. Secondly, the inference (or generation) process in SCPN is computationally very expensive, since it involves a two-stage generation process. In the first stage, they generate full parse trees from incomplete templates, and then from full parse trees to final generations. In contrast, the inference in our method involves a single-stage process, wherein our model takes as input a semantic source, a syntactic tree and the level of syntactic style that needs to be transferred, to obtain the generations. Additionally, we also observed that the model does not perform well in low-resource settings. This, again, can be attributed to the compounding implicit noise in the training due to linearised trees and the generation of full linearised trees before obtaining the final paraphrases.

Chen et al. [20] propose a syntactic exemplar-based method for controlled paraphrase generation using an approach based on latent variable probabilistic modeling, neural variational inference, and multi-task learning. This, in principle, is very similar to Chen et al. [21]. As opposed to our model which provides different levels of syntactic control of the exemplar-based generation, this approach is restrictive in terms of the flexibility it can offer. Also, as noted in Shi et al. [120], an auto-encoder-based approach might not offer rich enough syntactic information as offered by actual constituency parse trees. Additionally, VAEs [66] are generally unstable and harder to train [15, 46] than seq2seq-based approaches.

Pre-trained Classification Models like BERT [28], and RoBERTa [91] are typically fine-tuned for classification tasks using a low-capacity neural network classifier connected to the pre-trained model on its first token (typically `[CLS]` token). We demonstrate the inconsistency in the case of symmetric classification tasks for pairs of inputs, depending on the order of inputs. To the best of our knowledge, this is the first work that incorporates task-specific nuances to ensure consistency in symmetric classification.

Consistency Loss has been used in style transfer tasks to minimize the distance between round-trip generation of candidates for image-to-image translation [155] or text style transfer [53]. In a similar vein, we apply consistency loss (formulated as either the Kullback-Leibler or the Jensen-Shannon divergence loss) to alleviate the inconsistency problem in symmetric tasks.

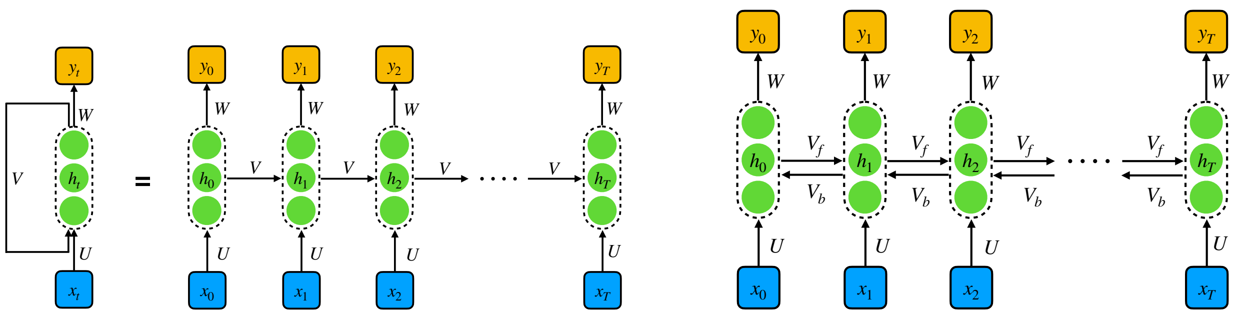
Embedding-based Semantic Similarity Scores based on BERT-based models like SBERT [111, 130] can map surface form realizations to embeddings. Their performance is worse than directly using BERT-style cross-encoder models for tasks such as semantic similarity [130]. However, the primary aim of such embedding-based scorers is orthogonal and, at best, complementary to the goal of our work since we want to ensure high-performing, consistent classifiers. Similarly, an alternative for symmetric classification is to separately obtain predictions for (X, Y) and (Y, X), and then average the confidence scores during test time. But, this is a weakly grounded, heuristic-driven approach. In general, *averaging does not rectify the mistakes made by the model, only masks it.*

2.3 Technical Fundamentals

We now describe some technical fundamentals for understanding the methods presented in the thesis. Specifically, we first present NLP sequence frameworks that are used in approaches in this thesis. We then discuss the broad categorization of decoding approaches. Following that, we describe the notions of representative subset selection through submodularity and determinantal point processes.

2.3.1 NLP Sequence Frameworks

Recurrent Neural Networks: To model sequential information, recurrent neural networks (RNN) were one of the first neural network structures to be invented. As the name suggests, they are standard neural networks with a time-based looping. The output from the previous time-step acts as an input to the next time step, and an RNN allows persistence of information across time. One of the best ways to understand the workings of an RNN, is through unrolling the RNN module across time-steps as seen in Figure 2.1a.



(a) Recurrent Neural Network - Rolled (left) and Unrolled (right) version. Please refer to Equation 2.1 for details.

(b) Bidirectional Recurrent Neural Network - Unrolled version. Please refer to Equation 2.2 for details.

Figure 2.1: Recurrent Neural Networks. Illustration inspired by Christopher Olah [23].

As with any neural network, the states (h_t) and inputs (x_t) are passed through layers of linear maps and non-linearity. The final equations for obtaining the output y_t are:

$$\begin{aligned} h_t &= g_1(Ux_t + Vh_{t-1} + b_1) \\ y_t &= g_2(Wh_t + b_2), \end{aligned} \tag{2.1}$$

where g_1, g_2 are activation functions (non-linearities) and U, V, W, b_1, b_2 are the learnable weights that are shared temporally. h_t and y_t are the RNN hidden states and output vectors, respectively. This type of modelling results in the possibility of processing inputs of variable lengths,

while being parameter efficient. However, the main drawback of this system is that the model has difficulty accessing information from distant past (also termed long-term dependency problem - due to vanishing gradients) and, in its vanilla formulation, does not consider any future input information for building the current state.
One simple approach to address the absence of future input data in an RNN is to utilize a Bidirectional RNN or BiRNN. (**BiRNN**)(Figure 2.1b).

The following are the equations that govern a Bidirectional RNN:

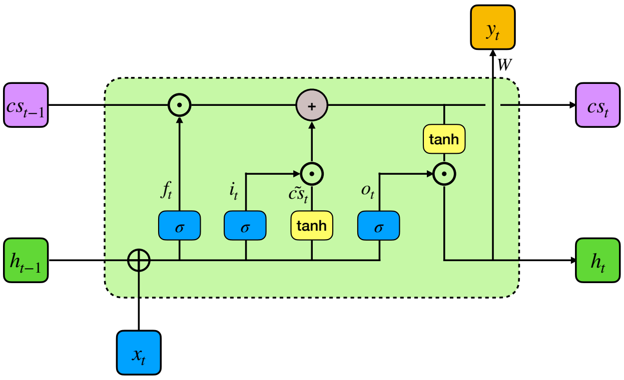
$$
\begin{aligned}
h_t^f &= g_1(Ux_t + V_f h_{t-1}^f + b_f) \\
h_t^b &= g_1(Ux_t + V_b h_{t-1}^b + b_b) \\
h_t &= h_t^f \oplus h_t^b \\
y_t &= g_2(Wh_t + b_2),
\end{aligned}
\tag{2.2}
$$

where g_1, g_2 are activation functions (non-linearities), $U, V_f, V_b, W, b_b, b_f, b_2$ are the learnable weights that are shared temporally and $\oplus$ is the concatenation operator. h_t^b, h_t^f are the backward and forward hidden states, respectively. h_t is a concatenation of h_t^b, h_t^f and y_t is the output vector similar to a vanilla RNN.
To overcome the drawback of long-term dependency, Hochreiter and Schmidhuber [49] proposed Long-short Term Memory (LSTMs) and Cho et al. [22] proposed Gated Recurrent Units (GRUs) that introduce *gates* for allowing selective information to flow in an recurrent network.

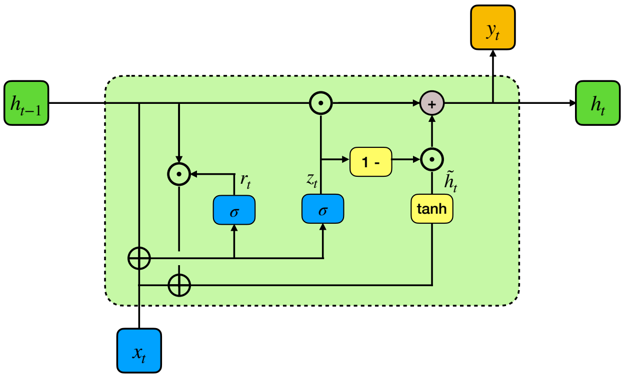
LSTM/GRU: Long-short Term Memory builds on top of standard RNNs, and helps mitigate long-term dependency problems. This, however, comes at the cost of heavier computation. The following diagram (Figure 2.2a) illustrates one cell (one time-step) of an LSTM network.

The following are the equations that govern the computation in an LSTM:

$$
\begin{aligned}
f_t &= \sigma(W_f(h_{t-1} \oplus x_t) + b_f) \\
i_t &= \sigma(W_i(h_{t-1} \oplus x_t) + b_i) \\
o_t &= \sigma(W_o(h_{t-1} \oplus x_t) + b_o) \\
\tilde{cs}_t &= tanh(W_o(h_{t-1} \oplus x_t) + b_o) \\
cs_t &= f_t \odot cs_{t-1} + i_t \odot \tilde{cs}_t \\
h_t &= o_t \odot tanh(cs_t) \\
y_t &= g(Wh_t + b),
\end{aligned}
\tag{2.3}
$$



(a) Single Unit in an LSTM. Also called an LSTM cell. Please refer to Equation 2.3 for details.



(b) Single Unit in an GRU. Also called a GRU cell. Please refer to Equation 2.4 for details.

Figure 2.2: Single Unit of an LSTM and a GRU. Illustration inspired by Christopher Olah [23].

where g is an activation function, $\odot$ is the Hadamard product operator, and f_t, i_t and o_t serve as *forget, input and output gates*, respectively with the associated learnable weights $W_f, b_f, W_i, b_i, W_o, b_o$ that are shared temporally. cs_t is called the cell state, which allows for selective long-term information flow. Like RNNs, h_t, y_t are the hidden representation, and output representations, respectively, and W_t and b are the learnable weights.
Gated recurrent units (GRU) (Figure 2.2b) simplify the LSTM models while achieving similar empirical results. In a GRU, the cell state is eliminated, thereby reducing memory footprint for storing the same.

The equations governing the computation of a GRU model are as follows:

$$\begin{aligned}
r_t &= \sigma(W_r(h_{t-1} \oplus x_t) + b_r) \\
z_t &= \sigma(W_z(h_{t-1} \oplus x_t) + b_z) \\
\tilde{h_t} &= tanh(W_h(x_t \oplus (r_t \odot h_{t-1})) + b_h) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t} \\
y_t &= g(Wh_t + b),
\end{aligned} \tag{2.4}$$

where r_t, z_t are called the reset and update gates respectively and W_r, b_r, W_z, b_z are their associated learnable weights shared temporally. Like RNNs, h_t, y_t are the hidden representation, and output representations, respectively, and W_t and b are the learnable weights.

Note that bidirectional versions of both GRU and LSTM are possible, and are referred to as BiGRU and BiLSTM model, respectively, in the later chapters.

In the first part of this thesis (Chapter 3, 4), we will primarily be focusing on sequence-to-sequence (Seq2Seq networks or encoder-decoder networks) [128, 22] that take a text as input, encode it into representational vectors (h_t) and sequentially (auto-regressively) generated the target text using a decoder network. We call the input as the **source** text and the expected output as the **target** text. Each text comprises a sequence of **tokens**, which can either be words or subwords [118].

In the second part (Chapter 5), we will only be considering the encoder-only model, specifically BERT [28], and RoBERTa [91].

Attention in Sequence-to-Sequence Networks: Theoretically, a high-capacity RNN (or GRU/LSTM) can carry all the information needed for a Seq2Seq generation task in its hidden state(s). Pragmatically, however, such a system is difficult, if not impossible, to train. To allow models to automatically (soft-)search for parts of a source text that are relevant to predicting a target token, without having to explicitly form these parts as a hard segment, Bahdanau et al. [6] proposed Attention-based RNNs.

Consider the following figure (Figure 2.3) for an encoder-decoder attention based model.

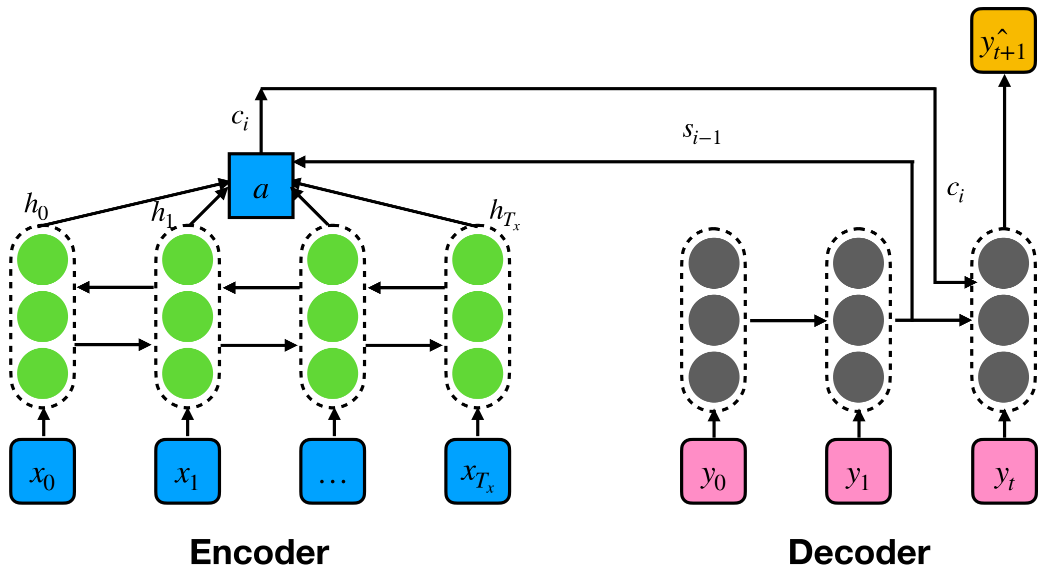


Figure 2.3: Sequence to Sequence based Attention Model Architecture. Please refer to Equation 2.5 for details.

We index encoder sequences using subscript j and the decoder sequences using subscript i. The hidden states h_j for the encoder are determined through the BiRNN equations (Equation 2.2).

$$
\begin{aligned}
c_i &= \sum_{j=1}^{T_x} \alpha_{ij} h_j \\
\alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \\
e_{ij} &= a(s_{i-1}, h_j),
\end{aligned}
\tag{2.5}
$$

where c_i denotes the context vector useful for decoding token at time-step i, α_{ij} is the j^{th} co-ordinate of the softmax-vector that assigns soft signals (weights) to the encoder hidden state representations h_j. e_{ij} is the j^{th} co-ordinate of the alignment vector that can be modeled using any neural network a, or simply dot products [94].

Attention based models have been pivotal for capturing relevant information for the decoder; however the utility of the softmax-vectors α_{ij} as interpretable signals is still (2022) hotly debated in the research community [59, 142].

Transformer Networks, and BERT Despite their promising advantage of addressing long-term-dependency as well as aligning contextual information correctly, attention based gated recurrent units suffer from slow training primarily due to non-parallelizability of the structure across GPU nodes.

Transformer Network: To address the computational issues, Vaswani et al. [132] proposed Transformer Networks. Transformer networks are, arguably, one of the most significant contributions in the Deep learning research community. A vanilla transformer network is a Seq2Seq network where both the encoder and the decoder use self-attention to encode inputs and generate relevant outputs.

Like traditional Seq2Seq models, the encoder takes the token-level vectorized representation of the source sequence as input, and the decoder (through training) learns to output the tokenized representation of the target. Transformer networks remove the recurrence formulation from the encoder and rely on self-attention. To capture time information Transformer Networks typically use positional embeddings that are augmented (or added) with the token representations.

Vaswani et al. [132] propose an elegant formulation of attention mechanism using dot products (called scaled dot product attention) of token key $k \in \mathbb{R}^{d_k}$, query $q \in \mathbb{R}^{d_q}$ and values $v \in \mathbb{R}^{d_v}$. In practice, the key, query, and value of all computable tokens are packed together in a matrix represented K, Q, V given below,

$$\begin{aligned} \text{Attention}(Q, K, V) &= \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\ \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \end{aligned} \tag{2.6}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are the associated learnable weights. And head represents one attention network. There are h such heads. We combine multiple such heads to get a high capacity representation, which is then projected back onto a lower dimensional subspace to give each token d_{model} dimensional representation. In practice, $d_k = d_v = d_{model}/h$.

BERT: Using just the encoder component of the traditional transformer networks, Devlin et al. [28] proposed Bidirectional Encoder Representations from Transformers (BERT). BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both the left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. We will look at this modification in Chapter 5.

2.3.2 Sequence Decoding/Generation

In this section, we will discuss the broad strategies for decoding using a Seq2Seq model or any other Natural Language Generation (NLG) model. There are two main types:

A. Autoregressive Generation: The most conventional approach for obtaining outputs from a decoder is through something called Autoregressive Generation. In this, the decoder utilizes information obtained from the encoder as well as the previous sequentially generated tokens, to produce a new token. Early works [131, 42, 136] showed that the order in which the tokens are generated is critical for determining the best autoregressive sequence. However, owing to the simplicity and intuitiveness of using the standard left-to-right order, they became ubiquitous. The two main ordering schemes under autoregressive decoding are:

1. **Monotonic Ordering**: A pre-determined order of sequence generation, be it *left-to-right* or *right-to-left*, is referred to as monotonic ordering. Mathematically, the modeling objective is:

$$p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x}) \prod_{i=2}^{n} p(y_i|y_{<i}, \mathbf{x}),$$

where $\mathbf{y}$ is the target sequence and $\mathbf{x}$ is the source sequence. y_i are the tokens of the target sequence $\mathbf{y}$. Note that the ordering information is implicit in this formulation. It is assumed that the best ordering of sequence to be generated is known.

2. **Adaptive Non-Monotonic Ordering**: In the most general cases, determining the order of prediction of generated tokens helps obtain an optimal final sequence $\mathbf{y}$. More formally, the modeling objective for such an approach is:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = p(y_{z_1}|\mathbf{x})p(z_1|y_{z_1}, \mathbf{x}) \prod_{i=2}^{n} p(y_{z_i}|z_{<i}, y_{z_{<i}}\mathbf{x})p(z_i|z_{<i}, y_{z_{\leq i}}\mathbf{x}),$$

where $\mathbf{y}$ is the target sequence, $\mathbf{x}$ is the source sequence and $\mathbf{z}$ determines the order in which the target sequence should be decoded. y_i, z_j are the generated tokens, and the positions (or order) at which they need to be placed in the sequence, respectively. Although mathematically sound, determining the *best order* $\mathbf{z}$ using previously known approaches is empirically challenging, especially in situations where the domain of the data is not provided. The model must infer that *adaptively* from the data [75].

B. Non-Autoregressive Generation: These approaches decode multiple tokens in parallel but under certain conditional-independent constraints. While being time-efficient, the generation quality of these approaches is lower compared to *auto-regressive* approaches. Most of the work focusing on Non-Autoregressive approaches typically tries to alleviate the inaccuracies introduced due to the conditional independence between tokens constraint.

Although pragmatic in their rights, we will only focus on **Monotonic ordering** in the first part of the thesis. However, we should note that the work in this thesis is either independent of the decoding strategy (DiPS: Chapter 3) or requires trivial changes in the way the output is processed (SGCP: Chapter 4).

2.3.3 Representative Subset Selection

A foundation towards the DiPS model (Chapter 3) is the notion of subset selection. As the name suggests, this involves selecting a subset of points from a larger set of data-points (called the **ground-set**) such that the selected points can effectively *describe or represent* the data collection.

More formally, given a set of points or data collection (ground set) $\mathbf{V}$, the objective is to find a subset of points $\mathbf{X} \subset \mathbf{V}$, such that $\mathbf{X}$ (argument set), *represents* the data collection. The *representation* of data-collection is measured using a set function $f : 2^{\mathbf{V}} \rightarrow \mathbb{R}$, where $2^{\mathbf{V}}$ is the power set of $\mathbf{V}$ - the domain of the function f. The extremely trivial case is of selecting all the points. However, we are interested in selecting a smaller set *i.e.*, $k = |\mathbf{X}| \ll |\mathbf{V}|$. Therefore, the overall objective is a constrained maximization problem where we try to maximize the functional value f under the cardinality constraint. We explore the fundamentals of two main strategies involved in understanding that chapter.

(A) Submodular Functions

Let $\mathbf{V}$ be a set. The following are equivalent definitions of submodular set functions $\mathcal{F} : 2^{\mathbf{V}} \to \mathbb{R}$:

- **Definition 2.1** $\forall \mathbf{X} \subset \mathbf{Y} \subset \mathbf{V}$ *and* $t \notin \mathbf{Y}$*:* $\mathcal{F}(\mathbf{X} \cup t) - \mathcal{F}(\mathbf{X}) \geq \mathcal{F}(\mathbf{Y} \cup t) - \mathcal{F}(\mathbf{Y})$
- **Definition 2.2** $\forall \mathbf{X}, \mathbf{Y} \subset \mathbf{V}$*:* $\mathcal{F}(\mathbf{X} \cup \mathbf{Y}) + \mathcal{F}(\mathbf{X} \cap \mathbf{Y}) \leq \mathcal{F}(\mathbf{X}) + \mathcal{F}(\mathbf{Y})$
- **Definition 2.3** $\forall \mathbf{X} \subset \mathbf{V}$ *and* $s, t \in \mathbf{V} \setminus \mathbf{X}, s \neq t$*:* $\mathcal{F}(\mathbf{X} \cup \{s\}) + \mathcal{F}(\mathbf{X} \cup \{t\}) \geq \mathcal{F}(\mathbf{X} \cup \{s, t\}) + \mathcal{F}(\mathbf{X})$

Moreover, $\mathcal{F} : \{0,1\}^n \to \mathbb{R}$ is submodular if $\forall i$ the discrete derivative: $\partial_i \mathcal{F}(x) = \mathcal{F}(x+e_i) - \mathcal{F}(x)$ is **non-increasing** in x. It is interesting to note that submodular functions are in some sense close to both, concave as well as convex functions [137, 70]. The definition given above is more like the definition for concave function - non-increasing discrete derivative, while submodular functions find more utility in function minimization akin to convex functions. In both, convex and submodular minimization, exact polynomial-time algorithms exist. However, maximizing a submodular function is NP-Hard (We discuss submodular maximization in details in Section 2.3.3).

It is natural to question: why is it possible to minimize submodular functions? While the exact algorithms are more involved, it can be simply explained using *the Lovász extension*

Definition 2.4 *(Lovász Extension) Assume* $\mathcal{F} : \{0,1\}^n \to \mathbb{R}$*, the* ***Lovász Extension****,* $\mathcal{F}^L : [0,1]^n \to \mathbb{R}$ *is given as:*

$$\mathcal{F}^L(x) = \sum_{i=0}^{n} \alpha_i \mathcal{F}(X_i),$$

where $x = \sum \alpha_i \mathbf{1}_{X_i}, \sum \alpha_i = 1, \alpha_i \geq 0$ *and* $\emptyset = X_0 \subset X_1 \subset \ldots \subset X_n$.

It can be observed $\mathcal{F}^L$ is an extension for $\mathcal{F}$ since $\mathcal{F}^L(x) = \mathcal{F}(x)$ for $x \in \{0,1\}^n$. An interesting property of this extension is that $\mathcal{F}^L$ is convex $\Leftrightarrow$ $\mathcal{F}$ is submodular. We know that convex functions can be minimized in polynomial time, using ellipsoid methods. Once the minimizer for $\mathcal{F}^L$ is obtained, we get a convex combination $\mathcal{F}^L = \sum_{i=0}^{n} \alpha_i \mathcal{F}(T_i)$ and one of the T_i is the solution of the submodular function $\mathcal{F}(X)$.

Based on the utility, theoretical underpinning, and computational-efficiency we describe two such methods in the following subsections.

Selection through Submodular Optimization: As alluded to earlier, we are interested in maximization of the overall functional value under the cardinality constraint. An important subclass of submodular functions is that of **Monotonic Submodular Functions** [70], where if we enlarge the argument set X, the functional value will never decrease. More formally,

Definition 2.5 *(Monotonic function) A function $f : 2^{\mathbf{V}} \rightarrow \mathbb{R}$ is called a monotonic function, if $\forall \mathbf{X} \subseteq \mathbf{Y} \subseteq \mathbf{V}$, $f(\mathbf{X}) \leq f(\mathbf{Y})$*

Having defined monotonic functions, let us take a look at some examples of Monotonic Submodular Functions.

1. **Modular Functions**. A class of functions where the inequalities characterizing the submodularity (Definition 2.1) hold true with equality are defined as modular functions *i.e.,* $\forall \mathbf{X} \subset \mathbf{Y} \subset \mathbf{V}$ and $t \notin \mathbf{Y}$: $\mathcal{F}(\mathbf{X} \cup t) - \mathcal{F}(\mathbf{X}) = \mathcal{F}.(\mathbf{Y} \cup t) - \mathcal{F}(\mathbf{Y})$. They can always be expressed in terms of summations over weight functions $w : \mathbf{V} \rightarrow \mathbb{R}$: $\mathcal{F}(\mathbf{X}) = \sum_{x \in \mathbf{X}} w(x)$. Moreover, submodularity in also preserved in the case of the composition of any concave function $h : \mathbb{R} \rightarrow \mathbb{R}$ with a monotone modular function $g : 2^{\mathbf{V}} \rightarrow \mathbb{R}$. An example of such a submodular function is $\mathcal{F}(X) = h \circ g(X) = \sqrt{|X|}$, where $h(x) = \sqrt{x}$, a known concave function, and g is the cardinality function counting the elements in set $\mathbf{X}$, which is a monotone modular function.

2. **Weighted Coverage**. A class of submodular functions is weighted coverage functions. Consider a universal set $\mathbf{U}$, a non-negative submodular function (may or may not be monotone) $\mathcal{G} : 2^{\mathbf{U}} \rightarrow \mathbb{R}$, and $\mathcal{V}$ as a collection of subsets of $\mathbf{U}$. Now, for any subcollection, $\mathcal{X} \subseteq \mathcal{V}$, the following function is monotone submodular.

$$\mathcal{F}(\mathcal{X}) := \mathcal{G}\left(\bigcup_{v \in \mathcal{X}} v\right) = \sum_{u \in \bigcup_{v \in \mathcal{X}} v} w(u) \tag{2.7}$$

   where $w : \mathbf{U} \rightarrow \mathbb{R}_{+}$ is a non-negative weight function for $\mathcal{G}$. Note that $\mathcal{F}(X)$ is monotone $\Leftrightarrow$ $\mathcal{G}$ is monotone, and $\mathcal{F}(X)$ is a submodular function for any arbitrary submodular objectives $\mathcal{G}$.

We will primarily look at a modification of the scoring of the candidates in the decoding objective of our sequence-to-sequence models, such that they satisfy the submodular and monotonicity conditions. This is done so that we can use a simple greedy approach to select a subset

Algorithm 1: Greedy selection for submodular optimization (Cardinality constraint)

Input: Ground Set: V
Budget: k
Submodular Function: $\mathcal{F}$

$\mathbf{X} \leftarrow \emptyset$
$N \leftarrow V$
while $|\mathbf{X}| < k$ **do**
  $x^* \leftarrow \text{argmax}_{x \in N} \mathcal{F}(\mathbf{X} \cup \{x\})$
  $\mathbf{X} \leftarrow \mathbf{X} \cup \{x^*\}$
  $N \leftarrow N \setminus \{x^*\}$
end
return X

of candidates from our ground set.

Greedy Maximization of Monotone Submodular Functions Consider the following problem:

$$\underset{\mathbf{X} \subset \mathbf{V}}{\text{argmax}}\, \mathcal{F}(\mathbf{X}) \text{ s.t. } |\mathbf{X}| = k, \tag{2.8}$$

where $\mathcal{F}$ is a monotonic-submodular function.

While the above problem is NP-Hard, it can be solved approximately. A simple approach for solving the approximate maximization problem (in the case of the given cardinality constraint) is a greedy algorithm (Refer Algorithm 1). This algorithm starts with an empty set $\mathbf{X}_0$ and iteratively adds elements x which maximize the updated functional value $\mathcal{F}(\mathbf{X}_i \cup \{x\})$ the most. In other terms, find elements x s.t. $\mathcal{F}(\mathbf{X}_i \cup \{x\}) - \mathcal{F}(\mathbf{X}_i)$ is maximized at each iteration i. More formally,

$$\mathbf{X}_{i+1} = \mathbf{X}_i \cup \{\underset{x}{\text{argmax}}\, \mathcal{F}(\mathbf{X}_i \cup \{x\}) - \mathcal{F}(\mathbf{X}_i)\}. \tag{2.9}$$

This provides a good approximation to the optimal solution of the NP-Hard problem. This is based on the following theorem due to Nemhauser et al. [101].

Theorem 2.1 *(Nemhauser et al. [101]) Given a non-negative submodular monotone function $\mathcal{F} : 2^{\mathbf{V}} \rightarrow \mathbb{R}_+$, and let $\{\mathbf{X}_i\}_{i \geq 0}$ be the greedily selected sets defined in Equation 2.9, and $\mathcal{F}(\mathbf{X}^*) = \max_{\mathbf{X}:|\mathbf{X}| \leq k} \mathcal{F}(X)$. Then $\forall k, l > 0$,*

$$\mathcal{F}(\mathbf{X}_l) \geq \left(1 - e^{-\frac{l}{k}}\right) \mathcal{F}(\mathbf{X}^*) \tag{2.10}$$

In particular, for $l = k, \mathcal{F}(\mathbf{X}_k) \geq \left(1 - \frac{1}{e}\right) \mathcal{F}(\mathbf{X}^*)$

Proof: Fix l and k. Let $\mathbf{X}^* \in \text{argmax}\{\mathcal{F}(\mathbf{X}) : |\mathbf{X}| \leq k\}$. In case the functional value $\mathcal{F}$ is

maximized at size $< k$, we can always add elements in the resultant set $\mathbf{X}^*$ such that $|\mathbf{X}^*| = k$ since it will either increase the functional value or keep it the same. This is because $\mathcal{F}$ is a monotonic function. Now, label the resultant points in $\mathbf{X}^*$ *arbitrarily*, as $\{x_1^*, \ldots, x_k^*\}$.
For a set function $\mathcal{F} : 2^{\mathbf{V}} \to \mathbb{R}, \mathbf{X} \subseteq \mathbf{V}$, and $e \in \mathbf{V}$, let $\Delta(e|\mathbf{X}) := \mathcal{F}(\mathbf{X} \cup \{e\}) - \mathcal{F}(\mathbf{X})$. Δ is also called the discrete derivative. For $i < l$,

$$\begin{aligned}
\mathcal{F}(\mathbf{X}^*) &\leq \mathcal{F}(\mathbf{X}^* \cup \mathbf{X}_i) && (\because \mathcal{F} \text{ is a monotonic function}) \quad (2.11)\\
&= \mathcal{F}(\mathbf{X}^* \cup \mathbf{X}_i) - \mathcal{F}(\{x_1^*, \ldots, x_{k-1}^*\} \cup \mathbf{X}_i)\\
&+ \mathcal{F}(\{x_1^*, \ldots, x_{k-1}^*\} \cup \mathbf{X}_i) - \ldots && (\text{Alternate terms cancel each other}) \quad (2.12)\\
&+ \mathcal{F}(\{x_1^* \cup \mathbf{X}_i) - \mathcal{F}(\mathbf{X}_i)\\
&+ \mathcal{F}(\mathbf{X}_i)\\
&= \mathcal{F}(\mathbf{X}_i) + \sum_{j=1}^{k} \Delta(x_j^* | \mathbf{X}_i \cup \{x_1^*, \ldots, x_{j-1}^*\}) && (\text{Rewriting previous equation}) \quad (2.13)\\
&\leq \mathcal{F}(\mathbf{X}_i) + \sum_{x \in \mathbf{X}^*} \Delta(x|\mathbf{X}_i) && (\because \mathcal{F} \text{ is submodular}) \quad (2.14)\\
&\leq \mathcal{F}(\mathbf{X}_i) + \sum_{x \in \mathbf{X}^*} (\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}_i)) && (\because \text{The greedy algorithm}) \quad (2.15)\\
&\leq \mathcal{F}(\mathbf{X}_i) + k(\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}_i)) && (\because k \text{ is the maximum size and } \mathcal{F} \text{ is submodular}) \quad (2.16)\\
\mathcal{F}(\mathbf{X}^*) &\leq \mathcal{F}(\mathbf{X}_i) + k(\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}_i)) && (2.17)
\end{aligned}$$

Re-arranging Equation 2.17, we get

$$\begin{aligned}
\mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_i) &\leq k(\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}_i))\\
&= k(\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}^*) + \mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_i)) \quad (2.18)\\
\mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_i) &\leq k(\mathcal{F}(\mathbf{X}_{i+1}) - \mathcal{F}(\mathbf{X}^*) + \mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_i))
\end{aligned}$$

Let $\delta_i = \mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_i)$, hence we have $\delta_i \leq k(\delta_i - \delta_{i+1})$. This implies that $\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$. Hence $\delta_l \leq (1 - \frac{1}{k})^l \delta_0$, where $\delta_0 = \mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\emptyset) \leq \mathcal{F}(\mathbf{X}^*)(\because \mathcal{F}$ is non-negative). We also know that $\forall x \in \mathbb{R}, 1 - x \leq e^{-x}$. Therefore,

$$\delta_l \leq e^{-\frac{l}{k}} \mathcal{F}(\mathbf{X}^*) \quad (2.19)$$

$$\mathcal{F}(\mathbf{X}^*) - \mathcal{F}(\mathbf{X}_l) \leq e^{-\frac{l}{k}} \mathcal{F}(\mathbf{X}^*) \quad (2.20)$$

$$\mathcal{F}(\mathbf{X}_l) \geq (1 - e^{-\frac{l}{k}}) \mathcal{F}(\mathbf{X}^*) \quad (2.21)$$

For the special case of $l = k$, $\mathcal{F}(\mathbf{X}_k) \geq (1 - \frac{1}{e})\mathcal{F}(\mathbf{X}^*)$ $\square$

(B) Determinantal Point Processes (DDP)

Another relevant subset selection strategy is obtained through Determinantal Point Processes (DPP). Determinantal point processes are probabilistic models of configurations that favour diversity [74]. A DPP offers a distribution over subsets of a fixed ground set. Aligned with our goal of subset selection, in presence of negative correlations between elements of the set, DPP offers an elegant, efficient and exact algorithm for sampling, marginalization, conditioning and other inference tasks. A DPP assigns higher probability to sets of items that are diverse.

Definition 2.6 *(Kulesza et al. [74]) A point process $\mathcal{P}$ on a ground set $\mathcal{V}$ is a probability measure over "point patterns" of $\mathcal{V}$, which are finite subsets of $\mathcal{V}$. In the discrete case, a point process is simply a probability measure on the power set of $\mathcal{V}$ i.e., $2^{\mathcal{V}}$. A sample from $\mathcal{P}$ might be the empty set, the entirety of $\mathcal{V}$, or anything in between. It is called a* ***determinantal point process*** *if, when $\mathbf{V}$ is a random subset drawn according to $\mathcal{P}$, we have, $\forall X \subseteq \mathcal{V}$:*

$$\mathcal{P}(X \subseteq \mathbf{V}) = det(K_X) \tag{2.22}$$

for some real, symmetric $N \times N$ matrix K indexed by the elements of $\mathcal{V}$. Here, $K_X \equiv [K_{ij}]_{i,j \in A}$ denotes the restriction of K to the entries indexed by elements of X, and $det(K_\emptyset) = 1$

Since $\mathcal{P}$ is a probability measure, any principal minor of K must be non-negative. In other words, K must be positive semi-definite.

Let us understand why a DPP favours diversity. K contains all the information needed to compute the probability of any subset $A \in \mathbf{V}$. If $X = \{i\}$, we have $\mathcal{P}(i \in \mathbf{V}) = K_{ii}$ i.e., the diagonal entries of K gives the marginal probabilities of including individual elements of $\mathcal{V}$. If the element is 1 then it is almost always selected by the DPP. For a two-element set $X = \{i, j\}$,

$$\begin{aligned}\mathcal{P}(\{i,j\} \in \mathbf{V}) &= \begin{vmatrix} K_{ii} & Kij \\ Kji & K_{jj} \end{vmatrix} \\ &= K_{ii}K_{jj} - K_{ij}Kji \\ &= \mathcal{P}(i \in \mathbf{V})\end{aligned} \tag{2.23}$$

The off-diagonal elements determine the negative correlations between pairs of elements. Large values of K_{ij} imply that i, j tend to not co-occur. If we think of the entries of the matrix

K as measurements of similarity between pairs of elements of $\mathcal{V}$, the highly similar elements are unlikely to appear together. If $K_{ij} = \sqrt{K_{ii}K_{jj}}$, the i, j are "perfectly similar" and do not appear together almost surely. However, if there is no correlation i.e., K is diagonal then the elements appear independently. Note that DPPs cannot represent distributions where elements are *more* likely to co-occur than if they were independent: correlations are always non-positive. The efficient sampling strategy for determining a k-sized subset using DPP is described in Kulesza and Taskar [73]. The resulting algorithm requires $O(|V|k^2)$ time overall.

Part I
Inducing Constraints in Paraphrase Generation

Chapter 3

Diverse Paraphrase Generation

In this chapter, we focus on the task of generating highly diverse paraphrases while not compromising on paraphrasing quality.

3.1 Introduction

As stated earlier, paraphrase generation is the task of rephrasing a given text in multiple ways such that the semantics of the generated sentences remain unaltered. Paraphrasing *Quality* can be attributed to two key characteristics - *fidelity* which measures the semantic similarity between the input text and generated text, and *diversity*, which measures the lexical dissimilarity between generated sentences.

Many previous works [107, 45, 86] address the task of obtaining semantically similar paraphrases. While it is essential to produce paraphrases with high fidelity, it is equally important, and in many cases desirable, to produce lexically diverse ones. Diversity in paraphrase generation finds applications in text simplification [102, 146], document summarization [84, 100], QA systems [36, 12], data augmentation [151, 140], conversational agents [83] and information retrieval [4]. Some examples for the above can be found in Section 2.1.

To obtain a set of multiple paraphrases, most of the current paraphrasing models rely solely on top-k beam search sequences (Table 3.1). The resulting set, however, contains many structurally similar sentences with only minor, word-level changes.

There have been some prior works [81, 34] which address the notion of diversity in NLP, including in sequence learning frameworks [124, 133]. Although Song et al. [124] addresses the issue of diversity in the scenario of neural conversation models using determinantal point processes (DPP), it could be naturally used for paraphrasing. Along similar lines, subset selection based on Simultaneous Sparse Recovery (SSR) [34] can also be easily adapted for the

SOURCE REFERENCE	– how do i increase body height ? – what do i do to increase my height ?
BEAM SEARCH	– how do i increase my height ? – how do i increase my body height ? – how do i increase the height ? – how would i increase my body height ?
DIPS (OURS)	– how could i increase my height ? – what should i do to increase my height ? – what are the fastest ways to increase my height ? – is there any proven method to increase height ?

Table 3.1: Sample paraphrases generated by Beam search and our method. It can be seen that our approach offers lexically diverse paraphrases without compromising on fidelity

same task.

Though these methods are helpful in maximizing diversity, they are restrictive in terms of retaining fidelity with respect to the source sentence. Addressing the task of diverse paraphrasing through the lens of monotone submodular function maximization [41, 70, 5] alleviates this problem and also provides a few additional benefits. Firstly, the submodular objective offers better flexibility in terms of controlling diversity as well as fidelity. Secondly, there exists a simple greedy algorithm for solving monotone submodular function maximization [101], which guarantees the diverse solution to be almost as good as the optimal solution. Finally, many submodular programs are fast and scalable to large datasets.

Below, we list the main contributions of this chapter.

1. We introduce **Di**verse **P**araphraser using **S**ubmodularity (DiPS). DiPS maximizes a novel submodular objective function specifically targeted toward paraphrasing.

2. We perform extensive experiments to show the effectiveness of our method in generating structurally diverse paraphrases without compromising on fidelity. We also compare against several possible diversity-inducing schemes.

3. We demonstrate the utility of diverse paraphrases generated via DiPS as data augmentation schemes on multiple tasks such as intent and question classification.

We have made DiPS's source code available at https://github.com/malllabiisc/DiPS

3.2 Methodology

Similar to Prakash et al. [107], Gupta et al. [45], Li et al. [86], we formulate the task of paraphrase generation as a sequence-to-sequence learning problem. Previous SEQ2SEQ based approaches

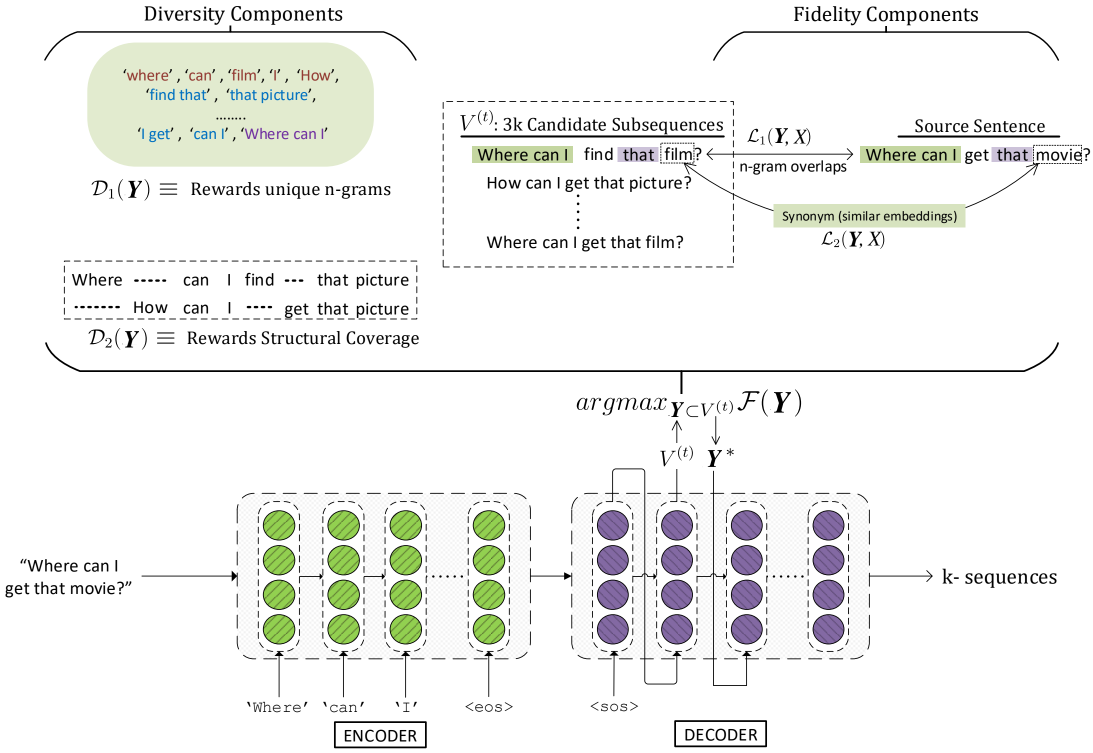


Figure 3.1: Overview of DiPS during decoding to generate k paraphrases. At each time step, a set of N sequences ($V^{(t)}$) is used to determine $k < N$ sequences (Y^*) via submodular maximization . The above figure illustrates the motivation behind each submodular component. Please see Section 3.2 for details.

depend entirely on the standard cross-entropy loss to produce semantically similar sentences and greedy decoding during generation. However, this does not guarantee lexical variety in the generated paraphrases. To incorporate some form of diversity, most prior approaches rely solely on top-k beam search sequences. The k-best list generated by standard beam search is a poor surrogate for the entire search space [39]. In fact, most of the sentences in the resulting set are structurally similar, differing only by punctuations or minor morphological variations.

While being similar in the encoding scheme, our work adopts a different approach for the final decoding. We propose a framework that organically combines a sentence encoder with a diversity-inducing decoder.

Algorithm 2: DiPS

Input: Input Sentence: S_{in}
Max. decoding length: T
Submodular objective: $\mathcal{F}$
No. of paraphrases required: k

Process S_{in} using the encoder of Seq2Seq
Start the decoder with input symbol `sos`
$t \leftarrow 0$; $P \leftarrow \emptyset$
while $t < T$ **do**
    Generate top $3k$ most probable subsequences
    $P \leftarrow$ Select k based on $\text{argmax}_{\mathbf{Y} \subseteq V^{(t)}} \mathcal{F}(\mathbf{Y})$ using **Algorithm** 1
    $t = t + 1$
end
return P

3.2.1 Overview

Our approach is built upon Seq2Seq framework. We first feed the tokenized source sentence to the encoder. The task of the decoder is to take as input the encoded representation and produce the respective paraphrase. To achieve this, we train the model using standard cross-entropy loss between the generated sequence and the target paraphrase. Upon completion of training, instead of using greedy decoding or standard beam search, we use a modified decoder where we incorporate a submodular objective to obtain high-quality paraphrases. Please refer to Figure 3.1 for an overview of the proposed method.

During the generation phase, the encoder takes the source sentence as input and feeds its representation to the decoder to initiate the decoding process. At each time-step t, we consider N most probable subsequences since they are likely to be well-formed. Based on the optimization of our submodular objective, a subset of size $k < N$ is selected and sent as input to the next time step $t + 1$ for further generation. The process is repeated until desired output length T or `<eos>` token, whichever comes first.

3.2.2 Monotone Submodular Objectives

With the technical foundation of submodularity in subset selection(Section 2.3.3) in mind, we introduce the formal notations and propose the modified decoding objective in terms of submodular and modular functions.

Let X be a sentence to be paraphrased, $\mathbf{Y}$ be the set of selected/generated candidates and $Y \in \mathbf{Y}$ be one such candidate subsequence. Note that at each time-step t, $\mathbf{Y}$, and subsequently Y, will be overwritten.

We design a parameterized class of submodular functions tailored toward the task of paraphrase generation. Let $V^{(t)}$ be the ground set of possible subsequences at time step t. We aim to determine a set $\mathbf{Y} \subseteq V^{(t)}$ that retains certain *fidelity* as well as *diversity*. Hence, we model our submodular objective function as follows:

$$Y^* = \underset{\mathbf{Y} \subseteq V^{(t)}}{\operatorname{argmax}} \, \mathcal{F}(\mathbf{Y}) \quad s.t. \; |Y| \leq k \tag{3.1}$$

where k is our budget (desired number of paraphrases) and $\mathcal{F}$ is defined as:

$$\mathcal{F}(\mathbf{Y}) = \lambda \mathcal{L}(\mathbf{Y}, X) + (1 - \lambda)\mathcal{D}(\mathbf{Y}) \tag{3.2}$$

Here X is the source sentence, $\mathcal{L}(\mathbf{Y}, X)$ and $\mathcal{D}(\mathbf{Y})$ measure *fidelity* and *diversity*, respectively. $\lambda \in [0, 1]$ is the trade-off coefficient. This formulation clearly brings out the trade-off between the two key characteristics.

Fidelity

It is imperative to design functions that exploit the decoder search space to maximize the semantic similarity between the generated and the source sentence. To achieve this we build upon a known class of monotone submodular functions [126]

$$f(X) = \sum_{i \in U} \mu_i \phi_i(m_i(\mathbf{Y})) \tag{3.3}$$

where U is the set of features to be defined later, $\mu_i \geq 0$ is the feature weight, $m_i(\mathbf{Y}) = \sum_{Y \in \mathbf{Y}} m_i(Y)$ is non-negative modular function and ϕ_i is a non-negative non-decreasing concave function. Based on the analysis of concave functions in [67], we use the simple square root function as ϕ ($\phi(a) = \sqrt{a}$) in both of our fidelity objectives defined below.

We consider two complementary notions of sentence similarity namely syntactic and semantic. To capture syntactic information we define the following function:

$$\mathcal{L}_1(\mathbf{Y}, X) = \mu_1 \sqrt{\sum_{Y \in \mathbf{Y}} \sum_{n=1}^{N} \beta^n \left| Y_{\text{n-gram}} \cap X_{\text{n-gram}} \right|} \tag{3.4}$$

where $|Y_{\text{n-gram}} \cap X_{\text{n-gram}}|$ represents the number of overlapping n-grams between the source and the candidate sequence Y for different values of $n \in \{1, \ldots, N\}$(we use $N = 3$). Since longer n-gram overlaps are more valuable, we set $\beta > 1$. This function inherently increases the BLEU score between the source and the generated sentences.

We address the semantic aspect of fidelity by devising a function based on the word embeddings of source and generated sentences. We define embedding-based similarity between two sentences as,

$$\mathcal{S}(Y, X) = \frac{1}{|Y|} \sum_{w_i \in Y} \underset{w_j \in X}{\operatorname{argmax}} \; \psi(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) \tag{3.5}$$

where $\mathbf{v}_{w_i}$ is the word embedding for a token w_i and $\psi(\mathbf{v}_{w_i}, \mathbf{v}_{w_j})$ is the gaussian radial basis function (RBF)[1]. For each word in the candidate sequence Y, we find the best matching word in the source sentence using word-level similarity. Using the above-mentioned measure for embedding similarity we use the following submodular function:

$$\mathcal{L}_2(\mathbf{Y}, X) = \mu_2 \sqrt{\sum_{Y \in \mathbf{Y}} \mathcal{S}(Y, X)} \tag{3.6}$$

This function helps increase the semantic homogeneity between the source and generated sequences. The above-defined functions (Equation 3.4, 3.6) are compositions of non-decreasing concave functions and modular functions. Thus, staying in the realm of the class of monotone submodular functions mentioned in Equation 3.3, we define fidelity function $\mathcal{L}(\mathbf{Y}, X) = \mathcal{L}_1(\mathbf{Y}, X) + \mathcal{L}_2(\mathbf{Y}, X)$

Diversity

Ensuring high fidelity often comes at the cost of producing sequences that only slightly differ from each other. To encourage diversity in the generation process it is desirable to reward sequences with a higher number of distinct n-grams as compared to others in the ground set $V^{(t)}$. Accordingly, we propose to use the following function:

$$\mathcal{D}_1(\mathbf{Y}) = \mu_3 \sum_{n=1}^{N} \beta^n \left| \bigcup_{Y \in \mathbf{Y}} Y_{n-gram} \right| \tag{3.7}$$

For $\beta = 1$, $\mathcal{D}_1(\mathbf{Y})$ denotes the number of distinct n-grams present in the set $\mathbf{Y}$. Since shorter n-grams contribute more towards diversity, we set $\beta < 1$, thereby giving more value to shorter n-grams. It is easy to see that this function is monotone non-decreasing as the number of distinct n-grams can only increase with the addition of more sequences. To see that $\mathcal{D}_1(\mathbf{Y})$ is submodular, consider adding a new sequence to two sets of sequences, one a subset of the other. Intuitively, the increment in the number of distinct n-grams when adding a new sequence to the smaller set should be larger than the increment when adding it to the larger set, as the

[1]We find gaussian RBF to work better than other similarity metrics such as cosine similarity

distinct n-grams in the new sequence might have already been covered by the sequences in the larger set.

Apart from distinct n-gram overlaps, we also wish to obtain sequence candidates that are not only diverse but also cover all major structural variations. It is reasonable to expect sentences that are structurally different to have a lower degree of word/phrase alignment as compared to sentences with minor lexical variations. Edit distance (Levenshtein) is a widely accepted measure to determine such dissimilarities between two sentences. To incorporate this notion of diversity, a formulation in terms of edit distance seems like a natural fit for the problem. To do so, we use the coverage function which measures the similarity of the candidate sequences $\mathbf{Y}$ with the ground set $V^{(t)}$. The coverage function is naturally monotone submodular and is defined as:

$$\mathcal{D}_2(\mathbf{Y}) = \mu_4 \sum_{x_i \in V^{(t)}} \sum_{Y_j \in \mathbf{Y}} \mathcal{R}(Y_i, Y_j) \tag{3.8}$$

where $\mathcal{R}(Y_i, Y_j)$ is an alignment based similarity measure between two sequences Y_i and Y_j given by:

$$\mathcal{R}(Y_i, Y_j) = 1 - \frac{\texttt{EditDistance}(Y_i, Y_j)}{|Y_i| + |Y_j|} \tag{3.9}$$

Note that $\mathcal{R}(Y_i, Y_j)$ will always lie in the range $[0, 1]$.

Evidently, this method allows flexibility in terms of controlling diversity and fidelity. Our goal is to strike a balance between these two to obtain high-quality generations.

3.3 Experiments

3.3.1 Datasets

In this section, we outline the datasets used for evaluating our proposed method. We specify the actual splits in Table 3.2. Based on the task, we categorize them into the following:

1. **Intrinsic evaluation**: To demonstrate the efficacy of our method on fidelity and diversity, we use the positive subset (pairs with the label=1, indicating that they are paraphrases) of the existing *Quora question pair*[1] dataset, called *Quora-Div* and the existing *Twitter URL paraphrasing* [79] dataset, referred to as just *Twitter*.

   We additionally perform in-domain data augmentation for the task of paraphrase recognition. For that, we augment sentences generated through different paraphrasing model

[1]https://www.kaggle.com/c/quora-question-pairs

Dataset	Task	Train	Val.	Test	Classes
Quora-Div	Intrinsic	120K	20K	5K	N/A
Twitter	Intrinsic	100K	15K	3K	N/A
Quora-PR	Intrinsic	40K	10K	40K	2
		Data augmentation			
SNIPS	Intent	10k	1k	700	7
Yahoo-L31	Intent	4K	1K	1K	2
TREC	Question	1K	200	500	6

Table 3.2: Dataset Statistics for Paraphrase Generation, and Data Augmentation Tasks (Detection and Classification). Please see Section 3.3.1

as positive samples to the *Quora-PR* dataset. *Quora-PR* is a subset of *Quora question pair* dataset which contains positive and negative examples.

2. **Data augmentation**: We exhibit the importance of samples generated through our method on the task of Data augmentation using three existing datasets. *SNIPS [26]*, *Yahoo-L31*[1] is used for intent classification and *TREC* [85] is used for question classification. Each dataset is balanced in terms of the number of samples per class.

3.3.2 Baseline

Several models have sought to increase diversity, albeit with different goals and techniques. However, majority of the prior works in this area have focused on the task of producing diverse responses in dialog systems [83, 113] and not paraphrasing. Given the lack of relevant baselines, we compare our model against the following methods:

1. **SBS**: Decoder which performs standard beam search during generation.
2. **DBS**: An alternative of beam search to incorporate diversity. [133]
3. **DPP** (Section 3.3.4): Decoder based on Determinantal Point Processes [74]
4. **SSR** (Section 3.3.5): Decoder based on Subset Selection using Simultaneous Sparse Recovery [35]

   We additionally evaluate against the following paraphrase generation models:

5. **VAE-SVG**: VAE-based generative framework for paraphrase generation. [45]
6. **RbM**: Deep Reinforcement learning based paraphrase generation model. [86]

[1] https://webscope.sandbox.yahoo.com/

Note that the first four baselines are trained using the same Seq2Seq network and differ only in the decoding phase.

3.3.3 Model Details - Seq2Seq Models

Parameter	Value
Max grad norm	1.0
Batch size	16
Cell type	LSTM
LSTM Layers (Depth)	2
Hidden size	256
Embedding size	300
Vocabulary size	20,000
Dropout	None
Attention Model	Luong-general
Bidirectional Encoder	True
Max length	20
Learning Rate (Optimizer)	0.0002
Desired Paraphrases (k)	20

Table 3.3: Hyper-parameter settings for DiPS

Given a sequence of inputs $X = (x_1, \ldots, x_T)$, where T is the input sequence length, the goal of the sequence-to-sequence model is to estimate the conditional probability $\mathbb{P}(Y|X)$, where Y is the corresponding output sequence $Y = (y_1, \ldots, y_{T'})$. The input sequence length T may differ from the output sequence length T'. We choose the attention model [94, 6], which is based on the encoder-decoder framework proposed by [22, 128]. The encoder as well as the decoder is modeled using a recurrent neural network (RNN). We use a Long-short term memory unit (LSTM) [49] as it helps in learning problems with long-range temporal dependencies. The encoder LSTM takes as input the tokens of the sentence whose paraphrase needs to be generated and produces a sequence of encoder hidden states $h_i : i \in \{1 \ldots T\}$. At each time step, the decoder receives the word embedding of the previous word, a decoder state s_t and the attention distribution is calculated using the weighted sum of encoder states:

$$c_t = \sum_{i=1}^{T} \alpha_{t_i} h_i,\ \ \alpha_{t_i} = \frac{\exp \eta(s_{t-1}, h_i)}{\sum_{j=1}^{T} \exp \eta(s_{t-1}, h_j)}$$

to produce the corresponding paraphrase token y'_t

3.3.4 Baseline (DPP) - Determinantal Point Processes

Consider the problem of sampling $\mathbf{Y}$ points from V associated with a similarity matrix $K \in \mathbb{R}^{n\times n}$, that is symmetric, real and positive semi-definite (PSD). Determinantal point processes (DPP) [74] are elegant probabilistic models that capture negative correlation and help in efficient sampling which follow the distribution given by:

$$P(\mathbf{Y} \subseteq V) = \det(K_{\mathbf{Y}})$$

Assume the following q and ϕ functions:

$$q(Y, X) = \frac{1}{|Y|} \sum_{w_i \in Y} \operatorname*{argmax}_{w_j \in X} \psi(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) \tag{3.10}$$

$$\phi(Y_i, Y_j) = \frac{1}{|Y_i|} \sum_{w_k \in Y_i} \operatorname*{argmax}_{w_m \in Y_j} \psi(\mathbf{v}_{w_k}, \mathbf{v}_{w_m}) \tag{3.11}$$

Note that X is the source sentence, Y_i, Y_j are generated candidates and w_i, w_j, w_k, w_m are the tokens in the respective sentences, while v denotes the embeddings of those tokens. ψ is the same as given in Equation 3.5. The most relevant construction of DPPs is not through K but through L-ensembles kernel matrix [14, 73]. We calculate the matrix, $L(Y_i, Y_j, X) = q(Y_i, X)\phi(Y_i, Y_j)q(Y_j)$ Note that this function is not symmetric. In order to make it symmetric we operate on the final L-ensembles kernel matrix $\mathcal{L} = \frac{1}{2}(L + L^\top)$. We then use the sampling algorithm described in Kulesza and Taskar [73] to select the k candidates.

3.3.5 Baseline (SSR) - Subset selection via Simultaneous Sparse Recovery

Consider the problem of finding k points from a collection of $|V| = N$ data points which preserve the essential characteristics of the set $V = \{v_1, \ldots, v_N\}$. Assume that we can form a non-negative dissimilarity matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ such that each element d_{ij} is indicative of how well a data point i is suited to be a representative of data point j. Elhamifar et al. [34] propose

a method to select a subset of points from V that can well encode all the data points based on the dissimilarity matrix $\mathbf{D}$.
To do so, consider latent variables $z_{ij} \in \mathbf{Z}$ associated with dissimilarities d_{ij}. Each element z_{ij} can be interpreted as the *probability* that data point i is a representative of j.They formulate the problem as the following row-sparsity regularized trace minimization program on $\mathbf{Z} \in \mathbb{R}^{N \times N}$:

$$\begin{aligned} &\min \operatorname{tr}(\mathbf{D}^\top \mathbf{Z}) + \lambda \|\mathbf{Z}\|_{1,q} \\ &\text{s.t } \mathbf{Z} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{Z} = \mathbf{1}^\top, \|\mathbf{Z}\|_{1,\infty} \leq k \end{aligned} \tag{3.12}$$

where k denotes the cardinality constraint, $\operatorname{tr}(\cdot)$ denotes the trace operator, $\|Z\|_{1,q} \triangleq \sum_{i=1}^{N} \|z_i\|_q$ and $\mathbf{1}$ denotes an all-one N-dimensional vector. A set of representative points can be obtained by optimizing the above function and selecting indices corresponding to the non-zero rows of the sparse matrix $\mathbf{Z}^*$.

We start with selecting the top $3k$ most probable subsequences in each time step and then we use sparse subset selection to select k diverse subsequences which are fed into the decoder for the next time step. To use sparse subset selection we need to form a dissimilarity matrix $\mathbf{D}$. In contrast to DPP, the matrix need not be positive semi-definite. In addition, elements d_{ij}, need not necessarily satisfy the triangle inequality, and the matrix $\mathbf{D}$ can be asymmetric as well. We use an alternate formulation of Sparse subset selection [35] to select k-samples from a given ground set:

$$\begin{aligned} &\min \operatorname{tr}(\mathbf{D}^\top \mathbf{Z}) \\ &\text{s.t } \|\mathbf{Z}\|_{1,\infty} \leq k, \ \mathbf{Z} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{Z} = \mathbf{1}^\top, \end{aligned} \tag{3.13}$$

We use the following equation to compute dissimilarity between two sequences:

$$\mathbf{D}_{ij} = 1 - \phi(Y_i, Y_j),$$

where ϕ is given through Equation 3.11.

3.3.6 Intrinsic Evaluation

It is important to recognize that assessing paraphrase generation models, as well as other natural language generation models, involves several dimensions and has gained considerable attention in recent times. Given its complexity, it is essential to evaluate all metrics in tandem rather than treating them as independent entities.

1. **Fidelity**: To evaluate our method for the fidelity of generated paraphrases, we use

three machine translation metrics which are suitable for paraphrase evaluation task [145]: BLEU [103](upto bigrams), METEOR [8] and TER-Plus [122]. While all of these essentially look at surface level forms, METEOR and TER-Plus encourage the use of synonym replacements

2. **Diversity**: We report the degree of diversity by calculating the number of distinct n-grams ($n \in \{1, 2, 3, 4\}$). The value is scaled by the number of generated tokens to avoid favoring long sequences.

In addition to fidelity and diversity, we evaluate the efficacy of our method via accuracy by using the generated paraphrases as augmented samples in the task of paraphrase recognition on the *Quora-PR* dataset. We enrich the samples with DiPS generated samples. We perform experiments with multiple augmentation settings for the following classifiers:

1. **LogReg**: Simple Logistic Regression model. We use the Tf-idf vectors as feature vectors [96].
2. **SiameseLSTM**: Siamese adaptation of LSTM to measure quality between two sentences [99]

We also perform ablation testing to highlight the importance of each submodular component.

3.3.7 Extrinsic Evaluation via Data-Augmentation

We evaluate the importance of using high-quality paraphrases for data-augmentation in two downstream classification tasks, namely intent classification and question classification. Our original generation model is trained on *Quora-Div* question pairs. Since the intent classification and question classification contain questions, this setting seems like a good fit for performing transfer learning. We perform experiments on the following standard classifier models:

1. **LogRegDA**: Simple logistic regression model trained using Tf-idf vectors as feature vectors [96].
2. **LSTM**: Single layered LSTM classification model.

In addition to SBS and DBS, we use the following data augmentation baselines for comparison:

1. **SynRep** : Simple synonym replacement
2. **ContAug**: Data augmentation scheme based on replacing words with similar paradigmatic relations. [68]

Model	Quora-Div		
	BLEU↑	**METEOR↑**	**TERp↓**
SBS	33.1	28.2	55.6
DBS [133]	30.9	28.3	57.5
VAE-SVG [45]	33.4	25.6	63.2
RbM [86]	29.4	29.5	62.5
DPP	30.5	27.9	57.3
SSR	28.7	26.8	58.7
DiPS (Ours)	**35.1**	**29.7**	**53.2**

Model	Twitter		
	BLEU↑	**METEOR↑**	**TERp↓**
SBS	51.1	23.5	67.9
DBS [133]	47.1	22.1	69.0
VAE-SVG [45]	46.7	25.2	67.1.
RbM [86]	47.7	29.3	68.7
DPP	44.8	21.4	71.4
SSR	41.3	20.0	74.4
DiPS (Ours)	**55.3**	**30.1**	**63.5**

Table 3.4: Results on **Quora-Div** and **Twitter** dataset. Higher↑ BLEU and METEOR score is better whereas lower↓ TERp score is better. Please see Section 3.4 for details.

3.3.8 Setup

We train our SEQ2SEQ model with attention [6] for up to 50 epochs using the adam optimizer [64] with an initial learning rate set to 2e-4. During the generation phase, we follow standard beam search till the number of generated tokens is nearly half the source sequence length (token level) to avoid possible erroneous sentences. We then apply submodular maximization stochastically with probability p at each time step. Since each candidate subsequence is extended by a single token at every time step, the information added might not necessarily be useful as our submodular components work on the sentence level. This approach is time efficient and avoids redundant computations.

For each augmentation setting, we randomly select sentences from the training data and generate their paraphrases. We then add them to the training data with the same label as that of the source sentence. We evaluate the performance of different classification models in terms of accuracy.

Based on the formulation of the objective function, it should be clear that diversity would attain maximum value at (or around) $\lambda = 0$ albeit at the cost of fidelity. This is certainly not a desirable property for paraphrasing systems. To address this, we perform hyperparameter tuning for λ value by analyzing the trade-off between diversity and fidelity based on varying λ values. In practice, the diversity metric attains saturation at a certain λ range (usually 0.2 - 0.5).

3.4 Results

Our experiments were geared toward answering the following primary questions:

	Quora-Div			
Model	**1-distinct**	**2-distinct**	**3-distinct**	**4-distinct**
SBS	12.8	24.8	35.3	46.6
VAE-SVG [45]	15.8	22.5	27.6	31.8
DBS [133]	17.9	33.7	44.8	54.9
DPP	17.1	34.4	49.1	62.6
SSR	16.6	32.8	47.1	60.7
DiPS (Ours)	**18.1**	**37.2**	**52.3**	**65.3**
	Twitter			
Model	**1-distinct**	**2-distinct**	**3-distinct**	**4-distinct**
SBS	20.0	30.9	38.1	44.6
VAE-SVG [45]	19.3	28.2	33.3	37.2
DBS [133]	25.8	40.7	48.2	53.9
DPP	25.6	41.4	51.1	59.0
SSR	26.6	43.7	54.0	62.4
DiPS (Ours)	**28.3**	**46.6**	**56.7**	**64.5**

Table 3.5: Results on **Quora-Div** and **Twitter** dataset. Higher distinct scores imply better lexical diversity. Please see Section 3.4 for details.

Q1. Is DiPS able to generate diverse paraphrases without compromising on fidelity? (Section 3.4.1)

Q2. Are paraphrase generated by DiPS useful in data augmentation? (Section 3.4.2)

Q3. How does varying λ affect the performance of DiPS? (Section 3.4.3)

Q4. How important are the different submodular objective components in DiPS? (Section 3.4.3)

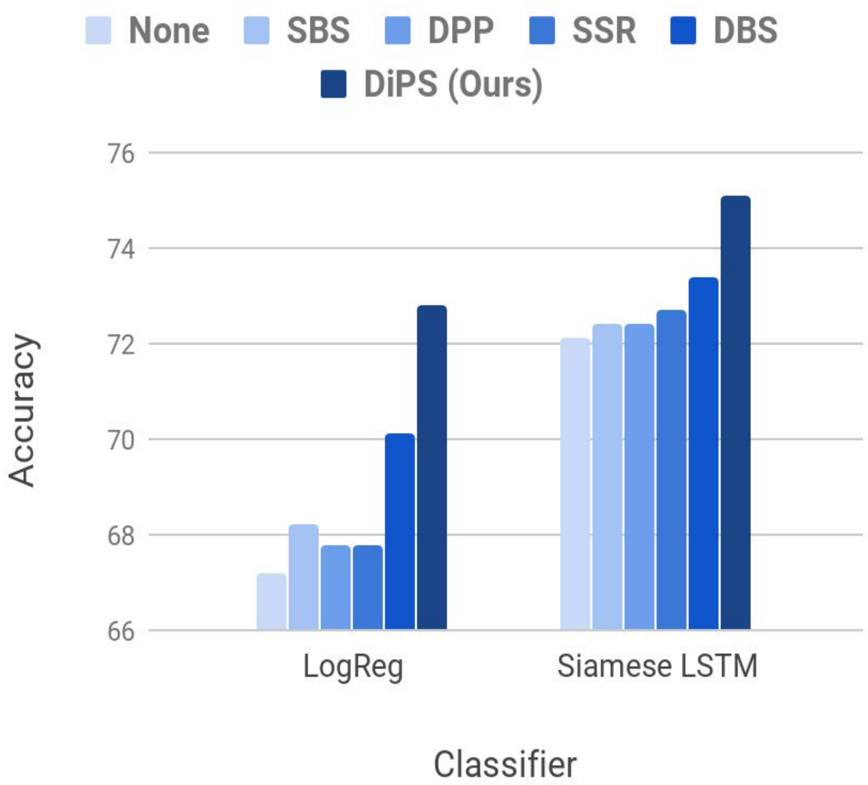


Figure 3.2: Comparison of accuracy scores of two paraphrase recognition models using different augmentation schemes (Quora-PR). Both LogReg and SiameseLSTM achieve the highest boost in performance when augmented with samples generated using DiPS

3.4.1 Intrinsic Evaluation

We compare our method against recent paraphrasing models as well as multiple diversity-inducing schemes. DiPS outperforms these baseline models in terms of fidelity metrics namely BLEU, METEOR, and TERp. A high METEOR score and a low TERp score indicate the presence of not only exact words but also synonyms and semantically similar phrases. Notably, our model is not only able to achieve substantial gains over other diversity-inducing schemes but is also able to do so without compromising on fidelity. Diversity and fidelity scores are reported in Table 3.5 and Table 3.4, respectively.

As described in Section 3.3.6, we evaluate the accuracy of paraphrase recognition models when provided with training data augmented using different schemes. It is reasonable to expect that high-quality paraphrases would tend to yield better results on in-domain paraphrase recognition tasks. We observe that using the paraphrases generated by DiPS helps in achieving substantial gains in accuracy over other baseline schemes. Figure 3.2 showcases the effect of using paraphrases generated by our method as compared to other competitive paraphrasing methods.

3.4.2 Data augmentation

	LogRegDA			LSTM		
Model	**YahooL31**	**TREC**	**SNIPS**	**YahooL31**	**TREC**	**SNIPS**
NoAug	62.7	82.2	93.4	64.8	94.2	94.7
SBS	63.6	84.6	93.8	65.4	94.4	94.7
DBS	63.3	84.2	94.1	65.6	95.2	96.1
SynRep	63.7	85.2	93.9	65.3	93.6	95.5
ContAug	63.8	86.0	95.3	66.3	95.8	96.4
DiPS(Ours)	**64.9**	**86.6**	**96.0**	**66.7**	**96.4**	**97.1**

Table 3.6: Accuracy scores of two classification models on various data augmentation schemes. Please see Section 3.4 for details

Data Augmentation results for intent and question classification are shown in Table 3.6. While SBS does not offer much lexical variability, DBS offers high diversity at the cost of fidelity. SynRep and ContAug are augmentation schemes that are limited by the number of structural variations they can offer. DiPS on the other hand provides generation having high structural variations without compromising on fidelity. The boost in accuracy scores on both types of classification models is indicative of the importance of using high-quality paraphrases for data augmentation.

3.4.3 Analysis

In this section, we perform extensive analysis to investigate the importance of the trade-off coefficient λ and each of the submodular components.

(1) Importance of the trade-off coefficient λ

We conduct experiments with varying values of the trade-off coefficient λ to analyse the fidelity performance of DiPS (Figure 3.3, Figure 3.4). As expected, we observe that the word-overlap metric BLEU increases with the increase in λ values. This trend is consistent across both datasets. We also evaluate the diversity using n-distinct metrics and observe that as λ decreases or $1 - \lambda$ increases the diversity increases (Figure 3.5, Figure 3.6). Overall there is a tradeoff in the component and λ acts as a control knob that can be tweaked as per the user.

(2) Ablation Study

In this section, we highlight the importance of using each submodular component for the generation of high-quality paraphrases.

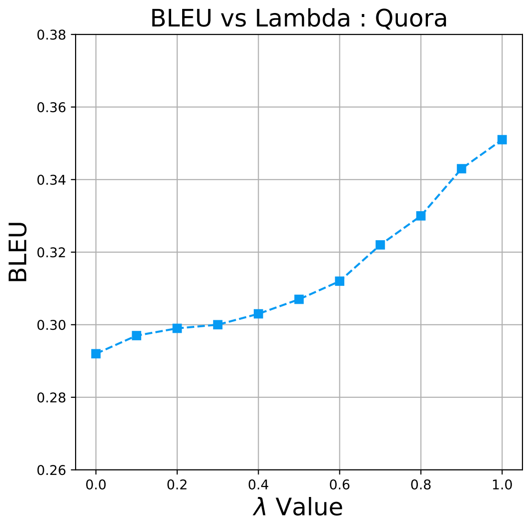


Figure 3.3: Effect of varying the trade-off coefficient λ in DiPS on BLEU score for quora dataset. Please see Section 3.4.3 for details.

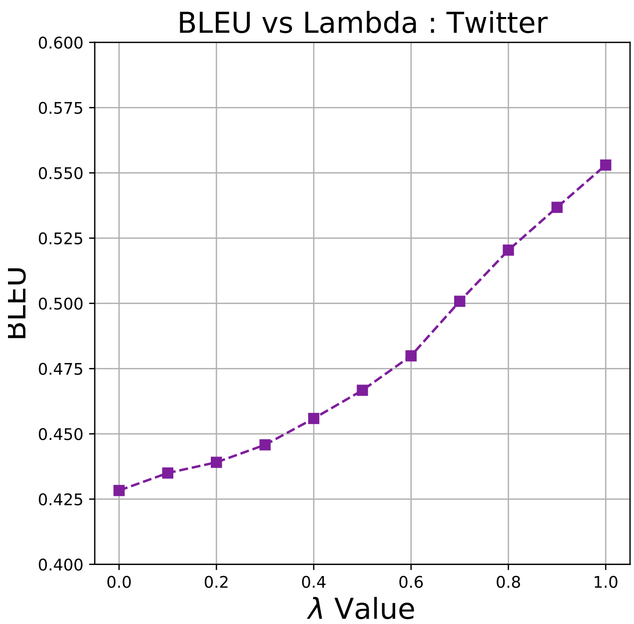


Figure 3.4: Effect of varying the trade-off coefficient λ in DiPS on BLEU score for twitter dataset. Please see Section 3.4.3 for details.

(2.a.) Fixed λ: We fix the trade-off coefficient value at $\lambda = 0.7$ and provide BLEU and the corresponding 2-distinct score for each of the component combinations (Table 3.7). $\mathcal{D}_1$ provides

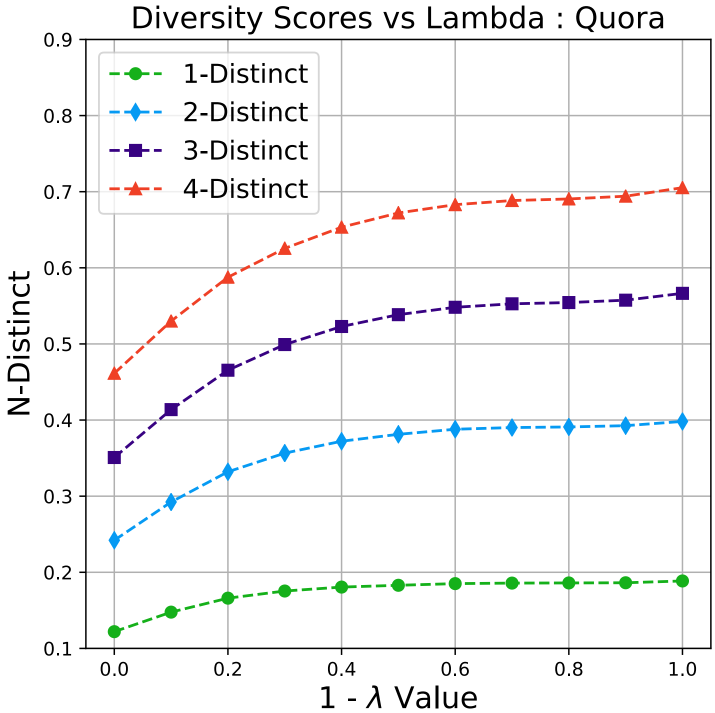


Figure 3.5: Effect of varying the trade-off coefficient λ in DiPS on various diversity metrics on the Quora dataset. Please see Section 3.4.3 for details.

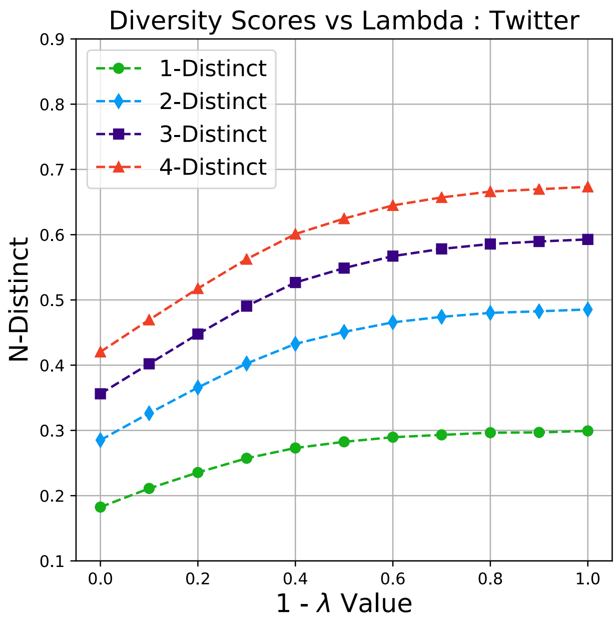


Figure 3.6: Effect of varying the trade-off coefficient λ in DiPS on various diversity metrics. Please see Section 3.4.3 for details.

higher diversity than $\mathcal{D}_2$, whereas $\mathcal{L}_1$ provides a marginally higher fidelity than $\mathcal{L}_2$.

Submodular Components	BLEU	2-distinct
$\mathcal{L}_1 + \mathcal{D}_1$	48.7	48.0
$\mathcal{L}_1 + \mathcal{D}_2$	52.3	35.4
$\mathcal{L}_2 + \mathcal{D}_1$	46.0	46.5
$\mathcal{L}_2 + \mathcal{D}_2$	51.6	35.5

Table 3.7: Results of ablation testing at fixed $\lambda = 0.7$ - Twitter Dataset. Please see Section 3.4.3 for details.

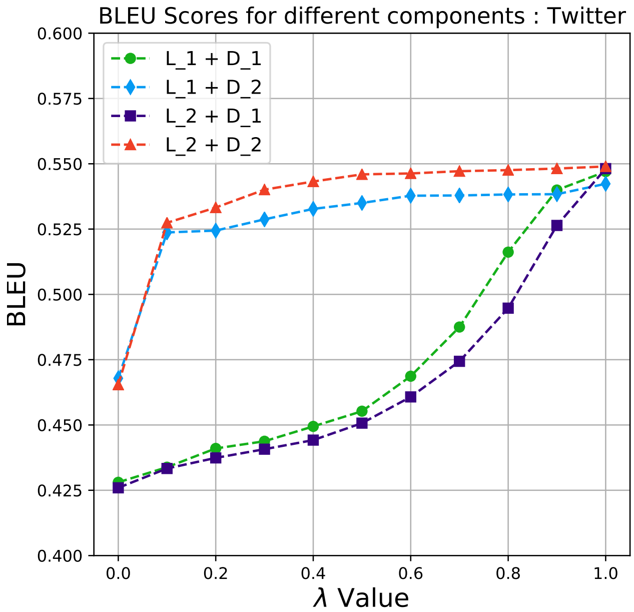


Figure 3.7: Effect of varying the trade-off coefficient λ in DiPS for individual combinations of submodular components - twitter dataset. Please see Section 3.4.3 for details.

(2.b.) Varying λ: We also analyse the effect of λ for individual components of submodular formulation. We vary the coefficient on the different combinations. As can be seen in Figure 3.7, while the fidelity components $\mathcal{L}_i$ provide similar levels of performance gains, it is the diversity component $\mathcal{D}_1$, that affects the diversity and therefore fidelity the most. Although, as is expected, at higher levels of λ, both fidelity components are able to provide similar progressive gains.

3.5 Summary

In this chapter, we have proposed DiPS, a model which generates high-quality paraphrases by maximization of a novel submodular objective function explicitly designed for paraphrasing. In contrast to prior works focusing exclusively on fidelity or diversity, a submodular function-based approach offers a significant degree of freedom to control fidelity and variety. Through extensive experiments on multiple standard datasets, we have demonstrated the effectiveness of our approach over numerous baselines. We observe that the diverse paraphrases generated are not only interesting and meaning-preserving but are also helpful in data augmentation. We showcase using multiple settings on the task of intent and question classification. We hope our approach will impact paraphrase generation, and data-augmentation, and other NLG problems in conversational agents and text summarization.

Chapter 4

Syntax-Guided Paraphrase Generation

In the previous chapter, we introduced a diversity-driven mechanism for paraphrase generation. It should be noted that although DiPS produces lexically diverse paraphrases, it might lack syntactical variations.

The goal of this chapter is to fill in that gap and introduce a method for syntax-guided paraphrase generation via controlled text generation. Given a sentence (e.g., *"I like mangoes"*) and a constraint (e.g., sentiment flip), the goal of controlled text generation is to produce a sentence that adapts the input sentence to meet the requirements of the constraint (e.g., *"I hate mangoes"*). Going beyond such simple constraints, recent works have started exploring the incorporation of complex syntactic guidance as constraints in the task of controlled paraphrase generation. In these methods, syntactic guidance is sourced from a separate exemplar sentence. However, these prior works have only utilized limited syntactic information available in the parse tree of the exemplar sentence. In this chapter, we address this limitation in the paper and propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), an end-to-end framework for syntactic paraphrase generation. We find that SGCP can generate syntax-conforming sentences while not compromising on relevance.

4.1 Introduction

Controlled text generation is the task of producing a sequence of coherent words based on given constraints. These constraints can range from simple attributes like tense, sentiment polarity and word-reordering [52, 119, 149] to more complex syntactic information. For example, given a sentence *"The movie is awful!"* and a *simple* constraint like flip sentiment to positive, a controlled text generator is expected to produce the sentence *"The movie is fantastic!"*.

These constraints are important in not only providing information about *what to say* but

SOURCE	– *how do i predict the stock market ?*
EXEMPLAR	– *can a brain transplant be done ?*
SCPN	– *how can the stock and start ?*
CGEN	– *can the stock market actually happen ?*
SGCP (Ours)	– *can i predict the stock market ?*
SOURCE	– *what are some of the mobile apps you ca n't live without and why ?*
EXEMPLAR	– *which is the best resume you have come across ?*
SCPN	– *what are the best ways to lose weight ?*
CGEN	– *which is the best mobile app you ca n't ?*
SGCP (Ours)	– *which is the best app you ca n't live without and why ?*

Table 4.1: Sample syntactic paraphrases generated by SCPN [58], CGEN [20], SGCP (Ours). We observe that SGCP is able to generate syntax-conforming paraphrases without compromising much on relevance.

also *how to say it*. Without any constraint, the ubiquitous sequence-to-sequence neural models often tend to produce degenerate outputs and favour generic utterances [134, 83]. While simple attributes are helpful in addressing *what to say*, they provide very little information about *how to say it*. Syntactic control over generation helps in filling this gap by providing that missing information.

Incorporating complex syntactic information has shown promising results in neural machine translation [125, 1, 148], data-to-text generation [105], abstractive text-summarization [18] and adversarial text generation [58]. Additionally, recent work [58, 77] has shown that augmenting lexical and syntactical variations in the training set can help build better-performing and more robust models.

In this chapter, we focus on the task of syntactically controlled paraphrase generation, i.e., given an input sentence and a syntactic exemplar, produce a sentence that conforms to the syntax of the exemplar while retaining the meaning of the original input sentence. While the syntactically controlled generation of paraphrases finds applications in multiple domains like data augmentation and text passivization, we highlight its importance in the particular task of Text simplification. As pointed out in Siddharthan [121], depending on the literacy skill of an individual, certain syntactical forms of English sentences are more straightforward to comprehend than others. As an example, consider the following two sentences:

S1 *Because it is raining today, you should carry an umbrella.*

S2 *You should carry an umbrella today because it is raining.*

Connectives that permit pre-posed adverbial clauses have been found to be difficult for third to fifth-grade readers, even when the order of mention coincides with the causal (and temporal) order [3, 80]. Hence, they prefer sentence **S2**. However, various other studies [25, 63, 54] have suggested that for older school children, college students, and adults, comprehension is better for the cause-effect presentation, hence sentence **S1**. Thus, modifying a sentence syntactically would help in better comprehension based on literacy skills.

Prior work in syntactically controlled paraphrase generation addressed this task by conditioning the semantic input on either the features learned from a linearized constituency-based parse tree [58] or the *latent* syntactic information [20] learned from exemplars through variational auto-encoders. Linearizing parse trees typically results in the loss of essential dependency information. On the other hand, as noted in [120], an auto-encoder-based approach might not offer rich enough syntactic information as guaranteed by actual constituency parse trees. Moreover, as noted in Chen et al. [20], SCPN [58] and CGEN [20] tend to generate sentences of the same length as the exemplar. This is undesirable because it often produces sentences that end abruptly, compromising grammaticality and semantics. Please see Table 4.1 for sample generations using each model.

To address these gaps, we propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), which uses *complete* exemplar syntactic tree information. Additionally, our model provides an easy mechanism to incorporate different levels of syntactic control (granularity) based on the height of the tree being considered. The decoder in our framework is augmented with rich enough syntactical information to produce syntax-conforming sentences while not losing out on semantics and grammaticality.

The main contributions of this work are as follows:

- We propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), an end-to-end model to generate syntactically controlled paraphrases at different levels of granularity using a parsed exemplar.
- We provide a new decoding mechanism to incorporate syntactic information from the exemplar sentence's syntactic parse.
- We provide a dataset formed from Quora Question Pairs [1] for evaluating the models. We also perform extensive experiments to demonstrate the efficacy of our model using multiple automated metrics and human evaluations.

[1] https://www.kaggle.com/c/quora-question-pairs

4.2 SGCP: Proposed Method

This section describes the inputs and various architectural components essential for building SGCP, an end-to-end trainable model. Our model, as shown in Figure 4.1, comprises a sentence encoder (4.2.2), syntactic tree encoder (4.2.3), and a syntactic-paraphrase-decoder (4.2.4).

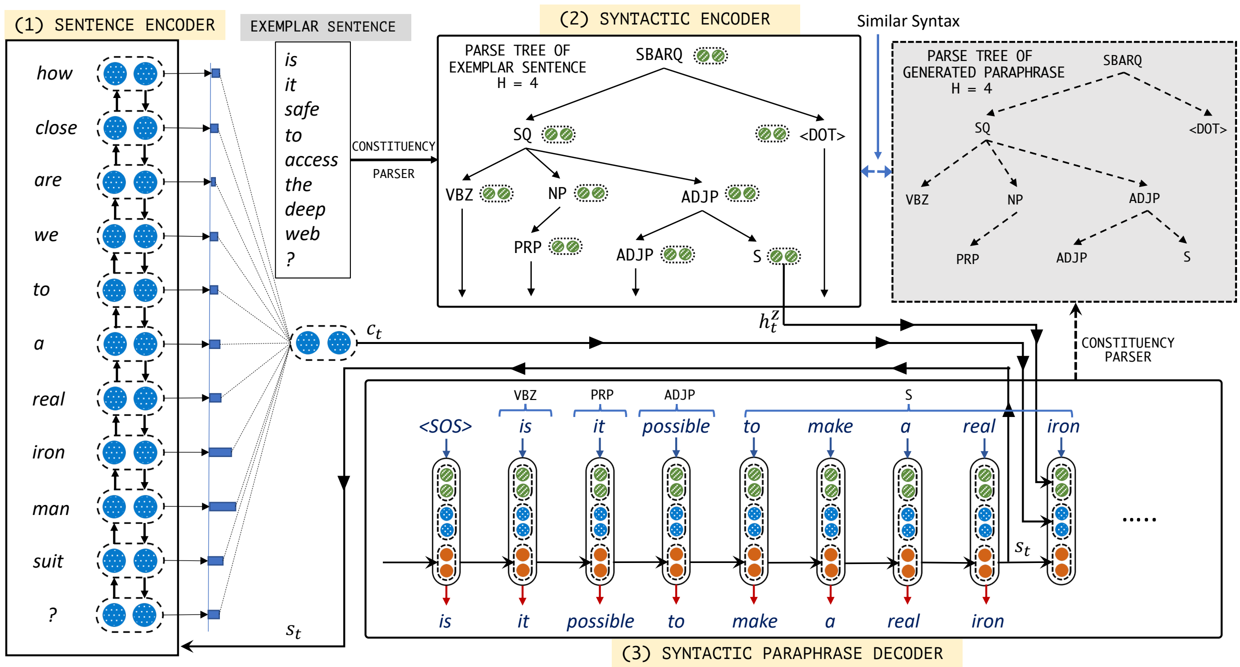


Figure 4.1: Architecture of SGCP (proposed method). SGCP aims to paraphrase an input sentence while conforming to the syntax of an exemplar sentence (provided along with the input). The input sentence is encoded using the Sentence Encoder (Section 4.2.2) to obtain a semantic signal c_t. The Syntactic Encoder (Section 4.2.3) takes a constituency parse tree (pruned at height H) of the exemplar sentence as an input and produces representations for all the nodes in the pruned tree. Once both of these are encoded, the Syntactic Paraphrase Decoder (Section 4.2.4) uses pointer-generator network, and at each time step takes the semantic signal c_t, the decoder recurrent state s_t, embedding of the previous token and syntactic signal h_t^Z to generate a new token. Note that the syntactic signal remains the same for each token in a span (shown in the figure above curly braces; please see Figure 4.2 for more details). The gray-shaded region (not part of the model) illustrates a qualitative comparison of the exemplar syntax tree and the syntax tree obtained from the generated paraphrase. Please refer to Section 4.2 for details.

4.2.1 Inputs

Given an input sentence X and a syntactic exemplar Z, our goal is to generate a sentence Y that conforms to the syntax of Z while retaining the meaning of X.

While the semantic encoder (Section 4.2.2) works on sequence of input tokens, the syntactic encoder (Section 4.2.3) operates on constituency-based parse trees. We parse the syntactic exemplar Z[1] to obtain its constituency-based parse tree. The leaf nodes of the constituency-based parse tree consist of tokens for sentence Z. However, we only need the syntacticality of exemplar Z to generate a syntax-guided paraphrase of sentence X. Therefore, the information in leaf nodes of Z is not necessary for the task at hand. To prevent any *meaning* propagation from exemplar sentence Z into the generation, we remove these leaf/terminal nodes from its constituency parse. The tree thus obtained is denoted as $\mathcal{C}^Z$.

The syntactic encoder, additionally, takes as input H, which governs the level of syntactic control needed to be induced. The utility of H will be described in Section 4.2.3.

4.2.2 Semantic Encoder

The semantic encoder, a multi-layered Gated Recurrent Unit (GRU), receives tokenized sentence $X = \{x_1, \ldots, x_{T_X}\}$ as input and computes the contextualized hidden state representation h_t^X for each token using:

$$h_t^X = \mathrm{GRU}(h_{t-1}^X, e(x_t)), \tag{4.1}$$

where $e(x_t)$ represents the learnable embedding of the token x_t and $t \in \{1, \ldots, T_X\}$. Note that we use byte-pair encoding [118] for word/token segmentation.

4.2.3 Syntactic Encoder

This encoder provides the necessary syntactic guidance for the generation of paraphrases. Formally, let constituency tree $\mathcal{C}^Z = \{\mathcal{V}, \mathcal{E}, \mathcal{Z}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ the set of edges and $\mathcal{Z}$ the labels associated with each node.

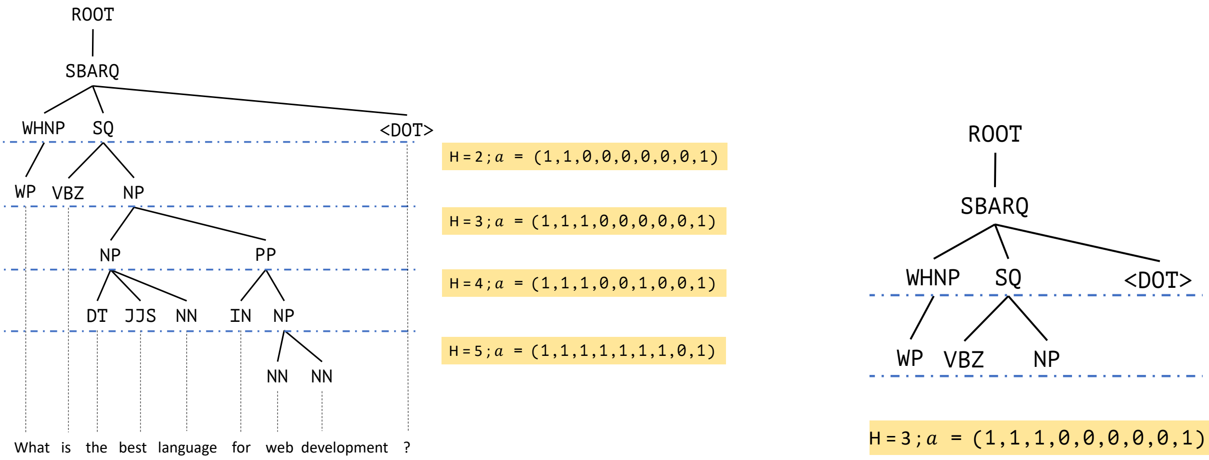
We calculate the hidden-state representation h_v^Z of each node $v \in \mathcal{V}$ using the hidden-state representation of its parent node $pa(v)$ and the embedding associated with its label z_v as follows:

$$h_v^Z = \mathrm{GeLU}(W_{pa} h_{pa(v)}^Z + W_v e(z_v) + b_v), \tag{4.2}$$

where $e(z_v)$ is the embedding of the node label z_v, and W_{pa}, W_v, b_v are learnable parameters. This approach can be considered similar to TreeLSTM [129]. We use `GeLU` activation function [48] rather than the standard `tanh` or `relu`, because of superior empirical performance.

As indicated in Section 4.2.1, syntactic encoder takes as input the height H, which governs the level of syntactic control. We randomly prune the tree $\mathcal{C}^Z$ to height $H \in \{3, \ldots, H_{\mathtt{max}}\}$, where $H_{\mathtt{max}}$ is the height of the full constituency tree $\mathcal{C}^Z$. The minimum value of 3 is a heuristic

[1] Obtained using the Stanford CoreNLP toolkit [97]



(a) Full Constituency Parse Tree (b) Pruned tree at height H = 3

Figure 4.2: The constituency parse tree serves as an input to the syntactic encoder (Section 4.2.3). The first step is to remove the leaf nodes which contain *meaning representative tokens* (Here: What is the best language ...). H denotes the height to which the tree can be pruned and is an input to the model. Figure (a) shows the full constituency parse tree annotated with vector $\boldsymbol{a}$ for different heights. Figure (b) shows the same tree pruned at height H = 3 with its corresponding $\boldsymbol{a}$ vector. The vector $\boldsymbol{a}$ serves as an *signalling* vector (Section 4.2.4) which helps in deciding the syntactic signal to be passed on to the decoder. Please refer Section 4.2 for details.

that the pruned sub-tree is at a sufficiently deep level. The purpose of pruning is twofold. Firstly, pruning results in generation of alternate sub-trees. These sub-trees can increase the training data by adding more samples relevant to the sub-trees. Secondly, pruning prevents inference-time distribution shift. The reason to do this is as follows. Not all English sentences can be converted into the desirable syntax of the exemplar sentence. For example, consider the exemplar is '*What is the best language for web development?*' and the source sentence is '*How do I go from Bengaluru to Hyderabad?*'. The syntax and word length of the source and exemplar are different. Also, the two sentences are questions of different types ('what' and 'how' respectively). In this case, the expected output could be '*What is a good way to go from Bengaluru to Hyderabad?*' This situation represents the scenario where the full-syntax tree of the exemplar is incompatible in terms of syntactic generation with the source sentence. In such cases, it may be necessary to prune the exemplar tree to a level where it becomes compatible. However, note that all trees are trivially compatible at the root since all constituency parse trees have the root node `ROOT`. Since pruning during training has made it easier for the model to understand the distribution, it should handle such cases effectively. Consider the sentence '*What is the best language for web development?*'. We prune the constituency-based parse tree of the exemplar sentence to height $H = 3$, where the leaf nodes have the labels `WP, VBZ, NP`, and `<DOT>`, as shown in Figure 4.2b. While we calculate the hidden-state representation of all nodes, only the pruned tree terminal nodes provide the decoder's syntactic signal (Section 4.2.4).

We maintain a queue $\mathbb{L}_H^Z$ of such terminal node representations where elements are inserted from left to right for a given H. Specifically, for the particular example given in Figure 4.2b,

$$\mathbb{L}_H^Z = [h^Z_{\texttt{WP}}, h^Z_{\texttt{VBZ}}, h^Z_{\texttt{NP}}, h^Z_{\texttt{<DOT>}}]$$

We emphasize that the length of the queue $|\mathbb{L}_H^Z|$ is a function of height H.

4.2.4 Syntactic Paraphrase Decoder

Having obtained the semantic and syntactic representations, the decoder is tasked with the generation of syntactic paraphrases. This can be modeled as finding the best $Y = Y^*$ that maximizes the probability $\mathbb{P}(Y|X, Z)$, which can further be factorized as:

$$Y^* = \underset{y}{\operatorname{argmax}} \prod_{t=1}^{T_Y} (y_t | y_1, \ldots, y_{t-1}, X, Z), \tag{4.3}$$

where T_Y is the maximum length up to which decoding is required.

In the subsequent sections, we use t to denote the decoder time step.

(a) Using Semantic Information

At each decoder time step t, the attention distribution α^t is calculated over the encoder hidden states h_i^X, obtained using Equation 4.1, as:

$$\begin{aligned} e_i^t &= v^\intercal \tanh(W_h h_i^X + W_s s_t + b_{\text{attn}}) \\ \alpha^t &= \text{softmax}(e^t), \end{aligned} \tag{4.4}$$

where s_t is the decoder cell-state and $v, W_h, W_s, b_{\text{attn}}$ are learnable parameters.

The attention distribution provides a way to jointly align and train sequence-to-sequence models by producing a weighted sum of the semantic encoder hidden states, known as context-vector c_t given by:

$$c_t = \sum_i \alpha_i^t h_i^X \tag{4.5}$$

c_t is the semantic signal essential for generating meaning-preserving sentences.

(b) Using Syntactic Information

During *training*, we use $Z = Y$ and each terminal node in the tree $\mathcal{C}^Z$, pruned at H, is equipped with information about the span of words it needs to generate. This is because during training, by having $Z = Y$, we specifically know the constituency tags associated with the spans in the final syntactic paraphrase to be generated. At each time step t, only *one* terminal node representation $h_v^Z \in \mathbb{L}_H^Z$ is responsible for providing the syntactic signal, which we call h_t^Z. This hidden-state representation to be used is governed through an *signalling* vector $\boldsymbol{a} = (a_1, \ldots, a_{T_y})$, where each $a_i \in \{0, 1\}$. **0** indicates that the decoder should keep on using the same hidden-representation $h_v^Z \in \mathbb{L}_H^Z$ that is currently being used, and **1** indicates that the next element (hidden-representation) in the queue $\mathbb{L}_H^Z$ should be used for decoding.

(c) Example

The utility of $\boldsymbol{a}$ can be best understood through Figure 4.2. Consider the syntactic tree $\mathcal{C}^Z$ for the sentence *"What is the best language for web development ?"*, pruned at height $H = 3$. For this example, the terminal nodes `WP, VBZ, NP`, and `<DOT>` in the pruned tree are considered to provide the syntactic signal. We process the tree tokens using the syntactic encoder and obtain a queue of terminal node representations:

$$\mathbb{L}_H^Z = [h_{\texttt{WP}}^Z, h_{\texttt{VBZ}}^Z, h_{\texttt{NP}}^Z, h_{\texttt{<DOT>}}^Z]$$

and a corresponding *signalling vector*

$$\boldsymbol{a} = (1, 1, 1, 0, 0, 0, 0, 0, 1)$$

The length of this vector is equal to the number of tokens in the sentence. The value at each position corresponds to the operation to be performed on the queue. We show the working of the process as follows.

$a_i = 1$ provides a signal to pop an element from the queue $\mathbb{L}_H^Z$ while $a_i = 0$ provides a signal to keep using the last popped element. This element is then used to guide the decoder *syntactically* by giving a signal in the form of hidden-state representation (Equation 4.8).

Specifically, in this example, the $a_1 = 1$ signals $\mathbb{L}_H^Z$ to pop $h_{\texttt{WP}}^Z$ to provide syntactic guidance to the decoder for generating the first token, $y_1 =$ *"What"*. $a_2 = 1$ signals $\mathbb{L}_H^Z$ to pop $h_{\texttt{VBZ}}^Z$ to provide syntactic guidance to the decoder for generating the second token, $y_2 =$ *"is"* . $a_3 = 1$ helps in obtaining $h_{\texttt{NP}}^Z$ from $\mathbb{L}_H^Z$ to provide guidance to generate the third token. As described earlier, $a_4, \ldots, a_8 = 0$ indicate that the same representation $h_{\texttt{NP}}^Z$ should be used for syntactically guiding tokens $y_3, \ldots, y_8$. Therefore, $h_{\texttt{NP}}^Z$ helps in generating *"the best language for web development"* Finally $a_9 = 1$ helps in retrieving $h_{\texttt{<DOT>}}^Z$ for guiding decoder to generate token y_9,

"?". Note that $|\mathbb{L}_H^Z| = \sum_{i=1}^{T_y} a_i$

While $\boldsymbol{a}$ is provided to the model during training, this information might not be available during inference. Providing $\boldsymbol{a}$ during generation makes the model restrictive and might produce ungrammatical sentences. SGCP is tasked to learn a proxy for the *signalling* vector $\boldsymbol{a}$, using *transition probability vector* $\boldsymbol{p}$.

At each time step t, we calculate $p_t \in (0, 1)$, which determines the probability of changing the syntactic signal using:

$$p_t = \sigma(W_{\text{bop}}([c_t; h_t^Z; s_t; e(y_t')]) + b_{\text{bop}}), \tag{4.6}$$

$$h_{t+1}^Z = \begin{cases} h_t^Z & p_t < 0.5 \\ \texttt{pop}(\mathbb{L}_H^Z) & \text{otherwise} \end{cases} \tag{4.7}$$

where `pop` removes and returns the next element in the queue, s_t is the decoder state, and $e(y_t')$ is the embedding of the input token at time t during decoding.

(d) Overall

The semantic signal c_t, together with decoder state s_t, the embedding of the input token $e(y_t')$ and the syntactic signal h_t^Z is fed through a GRU followed by softmax of the output to produce a vocabulary distribution as:

$$\mathbb{P}_{\text{vocab}} = \text{softmax}(W([c_t; h_t^Z; s_t; e(y_t')]) + b), \tag{4.8}$$

where $[;]$ represents concatenation of constituent elements, and W, b are trainable parameters.

We augment this with the copying mechanism [135] as in the pointer-generator network [116]. Usage of such a mechanism offers a probability distribution over the extended vocabulary (the union of vocabulary words and words present in the source sentence) as follows:

$$\begin{aligned} \mathbb{P}(y) &= p_{\text{gen}}\mathbb{P}_{\text{vocab}}(y) + (1 - p_{\text{gen}}) \sum_{i:z_i=z} \alpha_i^t \\ p_{\text{gen}} &= \sigma(w_c^\intercal c_t + w_s^\intercal s_t + w_x^\intercal e(y_t') + b_{gen}) \end{aligned} \tag{4.9}$$

where w_c, w_s, w_x and b_{gen} are learnable parameters, $e(y_t')$ is the input token embedding to the decoder at time step t, and α_i^t is the element corresponding to the i^{th} co-ordinate in the attention distribution as defined in Equation 4.4

The overall objective can be obtained by taking the negative log-likelihood of the distributions obtained in Equation 4.6 and Equation 4.9.

$$\begin{aligned}\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T}&[\log \mathbb{P}(y_t^*) \\ &+ a_t \log(p_t) \\ &+ (1-a_t)\log(1-p_t)]\end{aligned} \tag{4.10}$$

where a_t is the t^{th} element of the vector $\boldsymbol{a}$.

4.3 Experiments

Our experiments are geared toward answering the following questions:

Q1. Is SGCP able to generate syntax-conforming sentences without losing out on meaning? (Section 4.4.1, 4.4.4)

Q2. What level of syntactic control does SGCP offer? (Section 4.4.2, 4.4.3, 4.4.2)

Q3. How does SGCP compare against prior models, qualitatively? (Section 4.4.4)

Q4. Are the improvements achieved by SGCP statistically significant? (Section 4.4.1)

Based on these questions, we outline the methods compared (Section 4.3.1), along with the datasets (Section 4.3.2) used, evaluation criteria (Section 4.3.3) and the experimental setup (Section 4.3.4).

4.3.1 Methods Compared

As in Chen et al. [20], we first highlight the results of the two direct return-input baselines.

1. **Source-as-Output**: Baseline where the output is the semantic input.
2. **Exemplar-as-Output**: Baseline where the output is the syntactic exemplar.

We compare the following competitive methods:

3. **SCPN** [58] is a sequence-to-sequence based model comprising two encoders built with LSTM [49] to encode semantics and syntax respectively. Once the encoding is obtained, it serves as an input to the LSTM-based decoder which is augmented with soft-attention [6] over encoded states as well as a *copying* mechanism [116] to deal with out-of-vocabulary tokens. [1]

[1]Note that the results for SCPN differ from the ones shown in [58]. This is because the dataset used in [58] is at least 50 times larger than the largest dataset (ParaNMT-small) in this work.

4. **CGEN** [20] is a VAE [66] model with two encoders to project semantic input and syntactic input to a latent space. They obtain a syntactic embedding from one encoder, using a standard Gaussian prior. To obtain the semantic representation, they use von Mises-Fisher prior, which can be thought of as a Gaussian distribution on a hypersphere. They train the model using a multi-task paradigm, incorporating paraphrase generation loss and word position loss. We considered their best model, VGVAE + LC + WN + WPL, which incorporates the above objectives.

5. **SGCP (Section 4.2)** is a sequence-and-tree-to-sequence-based model which encodes semantics and tree-level syntax to produce paraphrases. It uses a GRU [24] based decoder with soft attention on semantic encodings and a *begin of phrase* (bop) gate to select a leaf node in the exemplar syntax tree. We compare the following two variants of SGCP:
   (a) SGCP-F : Uses full constituency parse tree information of the exemplar for generating paraphrases.
   (a) SGCP-R : SGCP can produce multiple paraphrases by pruning the exemplar tree at various heights. This variant first generates 5 candidate generations, corresponding to 5 different heights of the exemplar tree namely $\{H_{\texttt{max}}, H_{\texttt{max}}-1, H_{\texttt{max}}-2, H_{\texttt{max}}-3, H_{\texttt{max}}-4\}$, for each (source, exemplar) pair. The one with the highest ROUGE-1 score with the source sentence is selected as the final generation from these candidates.

Note that, except for the return-input baselines, all methods use beam search during inference.

4.3.2 Datasets

We train the models and evaluate them on the following datasets:
(1) ParaNMT-small [20] is an **existing** dataset that contains 500K sentence-paraphrase pairs for training, and 1300 manually labeled sentence-exemplar-reference which is further split into 800 test data points and 500 dev. data points respectively.
As in Chen et al. [20], our model uses only (sentence, paraphrase) during training. The paraphrase itself serves as the exemplar input during training.

This dataset is a subset of the original ParaNMT-50M dataset [143]. ParaNMT-50M is a data set generated automatically through back translation of original English sentences. It is inherently noisy due to imperfect neural machine translation quality, with many sentences being non-grammatical and some even non-English sentences. Because of such noisy data points, it is optimistic to assume that the corresponding constituency parse tree would be well aligned.

To that end, we **propose** to use the following additional dataset, which is more well-formed and has more human intervention than the ParaNMT-50M dataset.

(2) QQP-Pos: (Newly curated dataset) The original Quora Question Pairs (QQP) dataset contains about 400K sentence pairs labeled positive if they are duplicates of each other and negative otherwise. The dataset is composed of about 150K positive and 250K negative pairs. We select those positive pairs which contain both sentences with a maximum token length of 30, leaving us with ˜146K pairs. We call this dataset as QQP-Pos.

Similar to ParaNMT-small, we use only the sentence-paraphrase pairs as a training set and sentence-exemplar-reference triples for testing and validation. We randomly choose 140K sentence-paraphrase pairs as the training set $\mathbb{T}_{train}$, and the remaining 6K pairs $\mathbb{T}_{eval}$ are used to form the evaluation set $\mathbb{E}$. Additionally, let $\mathbb{T}_{eset} = \bigcup\{\{X, Y\} : (X, Y) \in \mathbb{T}_{eval}\}$. Note that $\mathbb{T}_{eset}$ is a set of sentences while $\mathbb{T}_{eval}$ is a set of sentence-paraphrase pairs.
Let $\mathbb{E} = \phi$ be the initial evaluation set. For selecting exemplar for each *each sentence-paraphrase pair* $(X, Y) \in \mathbb{T}_{eval}$, we adopt the following procedure:

Step 1: For a given $(X, Y) \in \mathbb{T}_{eval}$, construct an exemplar candidate set $\mathbb{C} = \mathbb{T}_{eset} - \{X, Y\}$. $|\mathbb{C}| \approx 12,000$.

Step 2: Retain only those sentences $C \in \mathbb{C}$ whose sentence length (= number of tokens) differ by at most 2 when compared to the paraphrase Y. This is done since sentences with similar constituency-based parse tree structures tend to have similar token lengths.

Step 3: Remove those candidates $C \in \mathbb{C}$, which are very similar to the source sentence X, i.e. $\texttt{BLEU}(X, C) > 0.6$.

Step 4: From the remaining instances in $\mathbb{C}$, choose that sentence C as the exemplar Z which has the least Tree-Edit distance with the paraphrase Y of the selected pair i.e. $Z = \underset{C \in \mathbb{C}}{\text{argmin}}\ \texttt{TED}(Y, C)$. This ensures that the constituency-based parse tree of the exemplar Z is quite similar to that of Y, in terms of Tree-Edit distance.

Step 5: $\mathbb{E} := \mathbb{E} \cup (X, Z, Y)$

Step 6: Repeat procedure for all other pairs in $\mathbb{T}_{eval}$.

From the obtained evaluation set $\mathbb{E}$, we randomly choose 3K triplets for the test set $\mathbb{T}_{test}$, and the remaining 3K for the validation set $\mathbb{V}$.

QQP-Pos								
Model	**BLEU↑**	**MET.↑**	**R-1↑**	**R-2↑**	**R-L↑**	**TED-R↓**	**TED-E↓**	**PDS↑**
Source-as-Output	17.2	31.1	51.9	26.2	52.9	16.2	16.6	99.8
Exemplar-as-Output	16.8	17.6	38.2	20.5	43.2	4.8	0.0	10.7
SCPN [58]	15.6	19.6	40.6	20.5	44.6	9.1	8.0	27.0
CGEN [20]	34.9	37.4	62.6	42.7	65.4	6.7	6.0	65.4
SGCP-F	**36.7**	**39.8**	**66.9**	**45.0**	**69.6**	**4.8**	**1.8**	**75.0**
SGCP-R	$\boxed{38.0}$	$\boxed{41.3}$	$\boxed{68.1}$	$\boxed{45.7}$	$\boxed{70.2}$	6.8	5.9	$\boxed{87.7}$
ParaNMT-small								
Source-as-Output	18.5	28.8	50.6	23.1	47.7	12.0	13.0	99.0
Exemplar-as-Output	3.3	12.1	24.4	7.5	29.1	5.9	0.0	14.0
SCPN [58]	6.4	14.6	30.3	11.2	34.6	6.2	**1.4**	15.4
CGEN [20]	13.6	24.8	44.8	21.0	48.3	6.7	3.3	70.2
SGCP-F	**15.3**	**25.9**	**46.6**	<u>21.8</u>	**49.7**	**6.1**	**1.4**	**76.6**
SGCP-R	$\boxed{16.4}$	$\boxed{27.2}$	$\boxed{49.6}$	$\boxed{22.9}$	$\boxed{50.5}$	8.7	7.0	$\boxed{83.5}$

Table 4.2: Results on QQP and ParaNMT-small dataset. Higher↑ BLEU, METEOR (MET.), ROUGE (R-) and PDS is better whereas lower↓ TED score is better. SGCP-R selects the best candidate out of many, resulting in a performance boost for semantic preservation (shown in box). We bold the statistically significant results of SGCP-F, only, for a fair comparison with the baselines. Note that Source-as-Output and Exemplar-as-Output are only dataset quality indicators and not competitive baselines. Please see Section 4.4 for details.

4.3.3 Evaluation

It should be noted that there is no single fully-reliable metric for evaluating syntactic paraphrase generation. Therefore, we evaluate on the following metrics to showcase the efficacy of syntactic paraphrasing models.

1. **Automated Evaluation**.

   (i) Alignment based metrics: We compute BLEU [103], METEOR [8], ROUGE-1, ROUGE-2, ROUGE-L [89] scores between the generated and the reference paraphrases in the test set.

   (ii) Syntactic Transfer: We evaluate the syntactic transfer using Tree-edit distance [150] between the parse trees of:

   (a) the generated and the syntactic exemplar in the test set - **TED-E**

   (b) the generated and the reference paraphrase in the test set - **TED-R**

   (iii) Model-based evaluation: Since our goal is to generate paraphrases of the input sentences, we need some measure to determine if the generations indeed convey the same meaning as the original text. To achieve this, we adopt a model-based evaluation metric as used by Shen et al. [119] for Text Style Transfer and Isola et al. [55] for Image Transfer. Specifically, classifiers are trained on the task of Paraphrase Detection and then used as Oracles to evaluate the generations of our model and the baselines. We fine-tune two RoBERTa [91] based sentence pair classifiers, one on Quora Question Pairs (*Classifier-1*) and the other on ParaNMT + PAWS[1] datasets (*Classifier-2*) which achieve accuracies of 90.2% and 94.0% on their respective test sets[2].

Once trained, we use *Classifier-1* to evaluate generations on QQP-Pos and *Classifier-2* on ParaNMT-small.

We first generate syntactic paraphrases using all the models (Section 4.3.1) on the test splits of QQP-Pos and ParaNMT-small datasets. We then pair the source sentence with their corresponding generated paraphrases and send them as input to the classifiers. The Paraphrase Detection score, denoted as *PDS* in Table 4.2, is defined as, the ratio of the number of generations predicted as paraphrases of their corresponding source sentences by the classifier to the total number of generations.

2. **Human Evaluation.**
   While `TED` is sufficient to highlight syntactic transfer, there has been some skepticism regarding automated metrics for paraphrase quality [112]. To address this issue, we perform a human evaluation on 100 randomly selected data points from the test set. In the evaluation, 3 judges (non-researchers proficient in the English language) were asked to assign scores to generated sentences based on the semantic similarity with the given source sentence. The annotators were shown a source sentence and the corresponding outputs of the systems in random order. The scores ranged from 1 (doesn't capture meaning at all) to 4 (perfectly captures the meaning of the source sentence).

4.3.4 Setup

(a) Pre-processing. Since our model needs access to constituency parse trees, we tokenize and parse all our data points using the fully parallelizable Stanford CoreNLP Parser [97] to obtain their respective parse trees. This is done prior to training in order to prevent any additional computational costs that might be incurred because of repeated parsing of the same data points during different epochs.

[1]Since the ParaNMT dataset only contains paraphrase pairs, we augment it with PAWS [152] dataset to acquire negative samples.

[2]Since the test set of QQP is not public, the 90.2% number was computed on the available dev set (not used for model selection)

Source	*what should be done to get rid of laziness ?*
Template Exemplar	*how can i manage my anger ?*
SCPN [58]	*how can i get rid ?*
CGEN [20]	*how can i get rid of ?*
SGCP-F (Ours)	*how can i stop my laziness ?*
SGCP-R (Ours)	*how do i get rid of laziness ?*
Source	*what books should entrepreneurs read on entrepreneurship ?*
Template Exemplar	*what is the best programming language for beginners to learn ?*
SCPN [58]	*what are the best books books to read to read ?*
CGEN [20]	*what 's the best book for entrepreneurs read to entrepreneurs ?*
SGCP-F (Ours)	*what is a best book idea that entrepreneurs to read ?*
SGCP-R (Ours)	*what is a good book that entrepreneurs should read ?*
Source	*how do i get on the board of directors of a non profit or a for profit organisation ?*
Template Exemplar	*what is the best way to travel around the world for free ?*
SCPN [58]	*what is the best way to prepare for a girl of a ?*
CGEN [20]	*what is the best way to get a non profit on directors ?*
SGCP-F (Ours)	*what is the best way to get on the board of directors ?*
SGCP-R (Ours)	*what is the best way to get on the board of directors of a non profit or a for profit organisation ?*

Table 4.3: Sample generations of the competitive models. Please refer to Section 4.4.5 for details

(b) Implementation details. We train both our models using the Adam Optimizer [64] with an initial learning rate of 7e-5. We use a bidirectional 3-layered GRU for encoding the tokenized semantic input and a standard pointer-generator network with GRU for decoding. The token embedding is learnable with dimension 300. To reduce the training complexity of the model, the maximum sequence length is kept at 60. The vocabulary size is kept at 24K for QQP and 50K for ParaNMT-small.

SGCP needs access to the level of syntactic granularity for decoding, depicted as H in Figure 4.2. During *training*, we keep on varying it randomly from 3 to $H_{\mathtt{max}}$, changing it with each training epoch. This ensures that our model is able to generalize because of an implicit regularization attained using this procedure. At each time step of the decoding process, we keep a teacher-forcing ratio of 0.9.

4.4 SGCP Results

4.4.1 Semantic Preservation and Syntactic transfer

1. Automated Metrics: As can be observed in Table 4.2, our method(s) (SGCP-F/R (Section 4.3.1)) are able to outperform the existing baselines on both the datasets. Source-as-Output is independent of the exemplar sentence being used and since a sentence is a paraphrase of itself, the *paraphrastic scores* are generally high while the *syntactic scores* are below par. The opposite is true for Exemplar-as-Output. These baselines also serve as *dataset quality* indicators. It can be seen that the source is semantically similar while being syntactically different from the target sentence whereas the opposite is true when the exemplar is compared to target sentences. Additionally, source sentences are syntactically and semantically different from exemplar sentences as can be observed from TED-E and PDS scores. This helps in showing that the dataset has rich enough syntactic diversity to learn from.

Through TED-E scores it can be seen that SGCP-F is able to adhere to the syntax of the exemplar template to a much larger degree than the baseline models. This verifies that our model is able to generate meaning-preserving sentences while conforming to the syntax of the exemplars when measured using standard metrics.

It can also be seen that SGCP-R tends to perform better than SGCP-F in terms of *paraphrastic scores* while taking a hit on the *syntactic scores*. This makes sense, intuitively, because in some cases SGCP-R tends to select lower H values for syntactic granularity. This can also be observed from the example given in Table 4.6 where $H = 6$ is more favourable than $H = 7$, because of better meaning retention.

Although CGEN performs close to our model in terms of BLEU, ROUGE and METEOR scores on the ParaNMT-small dataset, its PDS is still much lower than that of our model, suggesting that our model is better at capturing the original meaning of the source sentence. In order to show that the results are not coincidental, we test the statistical significance of our model. We follow the non-parametric Pitman's permutation test [33] and observe that our model is statistically significant when the significance level (α) is taken to be 0.05. Note that this holds true for all metrics on both the datasets except ROUGE-2 on ParaNMT-small.

2. Human Evaluation: Table 4.4 shows the results of human assessment. It can be seen that annotators, generally tend to rate SGCP-F and SGCP-R (Section 4.3.1) higher than the baseline models, thereby highlighting the efficacy of our models. This evaluation additionally shows that automated metrics are somewhat consistent with human evaluation scores.

	SCPN	CGEN	SGCP-F	SGCP-R
QQP-Pos	1.63	2.47	**2.70**	$\boxed{2.99}$
ParaNMT-small	1.24	1.89	**2.07**	$\boxed{2.26}$

Table 4.4: A comparison of human evaluation scores for comparing the quality of paraphrases generated using all models. A higher score is better. Please refer to Section 4.4.1 for details.

SOURCE : *how do i develop my career in software ?*	
SYNTACTIC EXEMPLAR	**SGCP GENERATIONS**
how can i get a domain for free ?	*how can i develop a career in software ?*
what is the best way to register a company ?	*what is the best way to develop career in software ?*
what are good places to visit in new york ?	*what are good ways to develop my career in software ?*
can i make 800,000 a month betting on horses ?	*can i develop my career in software ?*
what is chromosomal mutation ? what are some examples ?	*what is good career ? what are some of the ways to develop my career in software ?*
is delivery free on quikr ?	*is career useful in software ?*
is it possible to mute a question on quora ?	*is it possible to develop my career in software ?*

Table 4.5: Sample SGCP-R generations with a single source sentence and multiple syntactic exemplars. Please refer to Section 4.4.4 for details.

4.4.2 Syntactic Control

1. Syntactical Granularity : Our model can work with different levels of granularity for the exemplar syntax, i.e., different tree heights of the exemplar tree can be used for decoding the output.

As can been seen in Table 4.6, at height 4 the syntax tree provided to the model is not enough to generate the full sentence that captures the meaning of the original sentence. As we increase the height to 5, it is able to capture the semantics better by predicting *some of* in the sentence. We see that at heights 6 and 7 SGCP is able to capture both semantics and syntax of the source and exemplar respectively. However, as we provide the complete height of the tree i.e., 7, it further tries to follow the syntactic input more closely leading to sacrifice in the overall relevance since the original sentence is about *pure substances* and not *a pure substance*. It can be inferred from this example that since a source sentence and exemplar's syntax might not be fully compatible with each other, using the complete syntax tree can potentially lead to a loss

S	*what are pure substances ? what are some examples ?*
E	*what are the characteristics of the elizabethan theater ?*
H = 4	*what are pure substances ?*
H = 5	*what are some of pure substances ?*
H = 6	*what are some examples of pure substances ?*
H = 7	*what are some examples of a pure substance ?*

Table 4.6: Sample generations with different levels of syntactic control. S and E stand for source and exemplar, respectively. Please refer to Section 4.4.2 for details.

of relevance and grammaticality. Hence by choosing different levels of syntactic granularity, one can address the issue of compatibility to a certain extent.

2. Syntactic Variety : Table 4.5 shows sample generations of our model on multiple exemplars for a given source sentence. It can be observed that SGCP can generate high-quality outputs for a variety of different template exemplars even the ones which differ a lot from the original sentence in terms of their syntax. A particularly interesting exemplar is *what is chromosomal mutation ? what are some examples ?*. Here, SGCP is able to generate a sentence with two question marks while preserving the essence of the source sentence. It should also be noted that the exemplars used in Table 4.5, were selected manually from the test sets, considering only their *qualitative compatibility* with the source sentence. Unlike the procedure used for the creation of QQP-Pos dataset, the final *paraphrases* were not kept in hand while selecting the *exemplars*. In real-world settings, where a *gold paraphrase* won't be present, these results are indicative of the qualitative efficacy of our method.

4.4.3 SGCP-R Analysis

ROUGE-based selection from the candidates favour paraphrases that have higher n-gram overlap with their respective source sentences and hence may capture the source's meaning better. This hypothesis can be directly observed from the results in Table 4.2 and Table 4.4 where we see higher values on automated semantic and human evaluation scores. While this helps in getting better semantic generations, it tends to result in higher TED values. One possible reason is that, when provided with the complete tree, fine-grained information is available to the model for decoding and it forces the generations to adhere to the syntactic structure. In contrast, at lower heights, the model is provided with lesser syntactic information but equivalent semantic information.

4.4.4 Qualitative Analysis

	Single-Pass	Syntactic Signal	Granularity
SCPN	✗	Linearized Tree	✓
CGEN	✓	POS Tags (During training)	✗
SGCP	✓	Constituency Parse Tree	✓

Table 4.7: Comparison of different syntactically controlled paraphrasing methods. Please refer to Section 4.4.4 for details.

As can be seen from Table 4.7, SGCP not only incorporates the best aspects of both the prior models, namely SCPN and CGEN, but also utilizes the complete syntactic information obtained using the constituency-based parse trees of the exemplar.

From the generations in Table 4.3, it can be observed that our model is able to capture both, the semantics of the source text as well as the syntax of the template. SCPN, evidently, can produce outputs with the template syntax, but it does so at the cost of the semantics of the source sentence. This can also be verified from the results in Table 4.2 where SCPN performs poorly on PDS as compared to other models. In contrast, CGEN and SGCP retain much better semantic information, as is desirable. While generating sentences, CGEN often abruptly ends the sentence as in example 1 in Table 4.3, truncating the penultimate token with *of*. The problem of abrupt ending due to insufficient syntactic input length was highlighted in Chen et al. [20] and we observe similar trends. SGCP on the other hand generates more relevant and grammatical sentences.

Based on empirical evidence, SGCP alleviates this shortcoming, possibly due to dynamic syntactic control and decoding. This can be seen in e.g., 3 in Table 4.3 where CGEN truncates the sentence abruptly (penultimate token = *directors*) but SGCP is able to generate relevant sentence without compromising on grammaticality.

4.4.5 Limitations and Future directions

All natural language English sentences cannot necessarily be converted to desirable syntax. We note that SGCP does not take into account the compatibility of the source sentence and template exemplars and can freely generate syntax-conforming paraphrases. This at times, leads to imperfect paraphrase conversion and nonsensical sentences like example 6 in Table 4.5 (*is career useful in software ?*). Identifying compatible exemplars is an important but separate task in itself, which we defer to future work.

Another important aspect is that the task of paraphrase generation is inherently domain-agnostic. It is easy for humans to adapt to new domains for paraphrasing. However, due to the nature of the formulation of the problem in NLP, all the baselines as well as our model(s), suffer from dataset bias and are not directly applicable to new domains. A prospective future direction can be to explore it from the lens of domain independence.

Analyzing the utility of controlled paraphrase generations for the task of data augmentation is another interesting possible direction.

4.5 Summary

In this chapter, we proposed SGCP, an end-to-end framework for syntactically controlled paraphrase generation. SGCP generates a paraphrase of an input sentence while conforming to the syntax of an exemplar sentence provided along with the input. SGCP comprises a GRU-based sentence encoder, a modified RNN-based tree encoder, and a pointer-generator-based novel decoder. In contrast to previous works that focus on a limited amount of syntactic control, our model can generate paraphrases at different levels of granularity of syntactic control without compromising on relevance. Through extensive evaluations of real-world datasets, we demonstrate SGCP's efficacy over state-of-the-art baselines.

We believe that the above approach can be helpful in various text generation tasks, including syntactic exemplar-based abstractive summarization, text simplification, and data-to-text generation.

Part II
Inducing Consistency in Paraphrase Detection

Chapter 5

Consistency in Paraphrase Detection

In the previous chapters of the thesis, we discussed how constraints can be induced in paraphrase generation in order to ensure diversity and syntacticality. In this chapter, we focus on paraphrase detection. As described in the introduction chapter (Chapter 1), we look at the problem of inconsistencies in fine-tuned pre-trained models for paraphrase detection, and aim to address that using a simple objective function.

While fine-tuning pre-trained models for downstream classification is the conventional paradigm in NLP, task-specific nuances may not get captured in the resultant models. Specifically, for tasks that take two inputs and require the output to be invariant of the order of the inputs, inconsistency is often observed in the predicted label or confidence score. In this chapter, we propose a consistency loss function to alleviate inconsistency in symmetric classification.

5.1 Introduction

Symmetric classification tasks are tasks involving two inputs where the model output should be independent of the order in which the two input texts are given. In other words, *the output of the classifier should be the same and the confidence score must not be significantly different*, if the inputs X and Y are instead supplied as Y and X. Examples of symmetric classification are paraphrase detection, multi-lingual semantic similarity and so on. Although attention-based [7, 132] pre-trained language models have led to significant performance gains in multiple text classification tasks; they demonstrate erratic behavior on symmetric classification. An example[1] of *inconsistency* for paraphrase detection is shown in Figure 5.1. While it is natural to use simple rules for paraphrase detection, the need to use a model-based metric for the task becomes evident when dealing with complex pairs. Consider the sarcastic paraphrase pair:

[1]Note that while this particular example is based on our fine-tuned model, it will change depending on the trained model. The overall argument is valid, nonetheless.

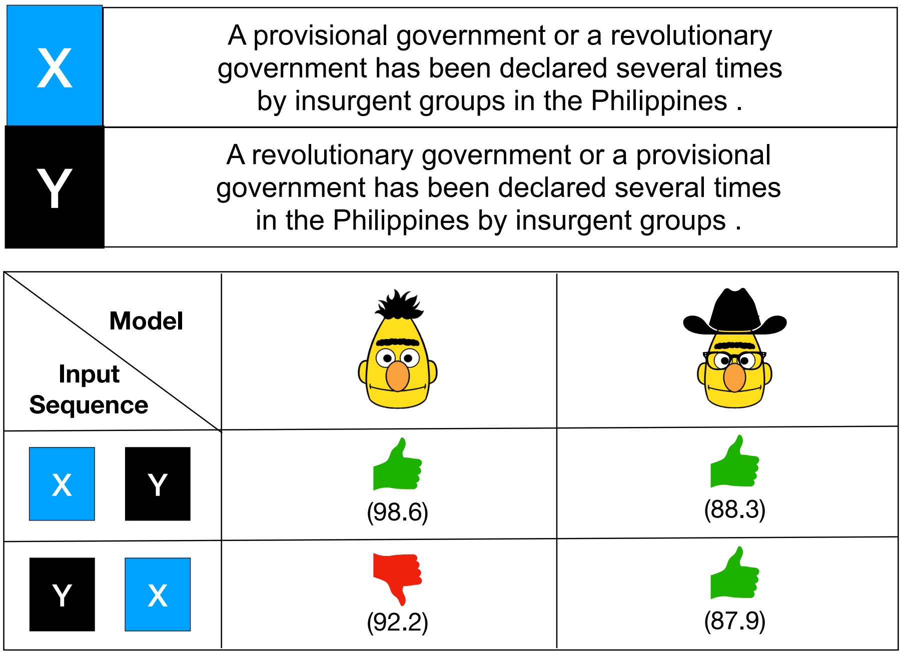


Figure 5.1: Impact of reordering an example input pair (X and Y) on standard fine-tuned BERT and BERT-with-consistency-loss . The pair are true paraphrases. and denote that the model predicted them to be paraphrases and not-paraphrases, respectively. Confidence scores are reported in brackets. Details in Section 5.1.

"Wow! You are really short!" and *"Wow! You aren't even able to reach the lowest bookshelf!"*. These sentences are, in a sense, functionally similar to each other since they convey similar intents but are not exactly *paraphrases* in the strict sense of the word because of implicit negative sentiment in the first sentence. Additional examples can be found in Table 5.3. To alleviate such inconsistency for symmetric classification tasks, we propose a simple additional *drop-in* fine-tuning objective, based on either the Kullback-Leibler (KL) or Jensen-Shannon (JS) divergence (or any f-divergence [114]), to the cross-entropy loss for symmetric tasks. We refer to this as the *consistency loss*.

The main contributions of this chapter are:

(a) Highlight inconsistency issues in symmetric tasks,

(b) Describe a consistency loss function to alleviate inconsistency, and

(c) Demonstrate the applicability and limitations of the loss function via qualitative and quantitative analyses on tasks from the GLUE benchmark.

Additionally, we have made the data and code public[1] to drive future research.

Note: The inconsistency problem can be attributed partly to the positional embedding. How-

[1] https://github.com/ashutoshml/alleviating-inconsistency

Category	Datasets	Train	Val.	Test
Pairwise Symmetric	**QQP**	327462	40430	36384
	PAWS	49401	8000	8000
	MRPC	3302	408	366
Single Sentence	**SST2**	60615	6872	6734
Pairwise Non-symmetric	**QNLI**	99506	5463	5237
	RTE	2241	277	249

Table 5.1: Datasets Statistics. Please refer to Section 5.3.

ever, it has been shown that eliminating positional embedding results in poor performance of the model [141, 139].

5.2 Method

5.2.1 Problem Description

A. Given a pair of input sentences (X, Y), label $l_{(X,Y)}$, and a pre-trained BERT-based model $\mathcal{M}_{\text{PRE}}$, output a *reliable model* $\mathcal{M}_{\text{REL}}$ for predicting an output label for a new input pair (X_{test}, Y_{test}) such that the *inconsistency* between its different ordering is minimized. While we only experiment with semantic similarity (or paraphrasing), the description holds for other symmetric relations, too (like if two sentences have the same polarity or not).
B. Given a model fine-tuned on the task above $\mathcal{M}_{\text{REL}}$, can it help in providing a better initialization for transfer learning an empirically superior model $\mathcal{M}'$ on other downstream tasks?

5.2.2 Setup

For problem **A** (Section 5.2.1), the input is a concatenation of tokenized strings $X = x_1, \ldots, x_m$ and $Y = y_1, \ldots, y_n$ separated using a special token (`[SEP]` in the case of BERT). The concatenated inputs with the special token are passed through multiple self-attention layers [132]. In the traditional approach, the representation of the first token (`<s>` or `[CLS]`) is passed through a fully connected classifier layer (the exact final representation is used irrespective of the arity of the task inputs). Our approach uses the `[CLSPara]` representation for symmetric classification tasks. In contrast, we use the standard first token (`<s>` or `[CLS]`) representation for single input and non-symmetric classification tasks (Section 5.3). Since we first fine-tune the model on `[CLSPara]` representation, our approach allows for pair-wise knowledge to be transferred to other downstream classification tasks (problem **B** (Section 5.2.1)).

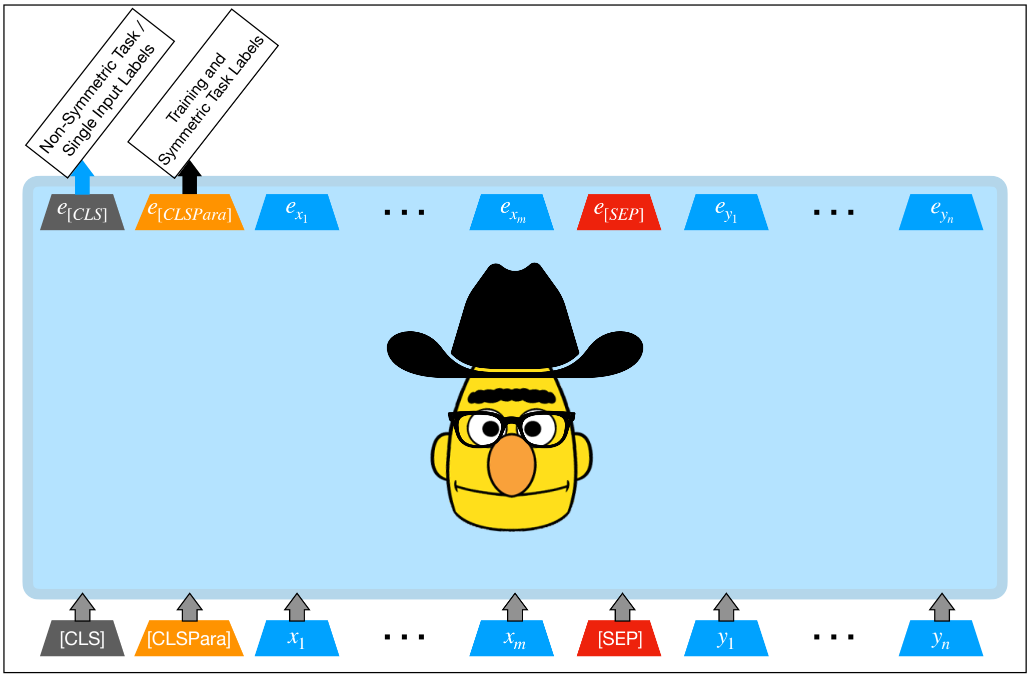


Figure 5.2: BERT-with-consistency-loss. We use an additional classification token: `[CLSPara]` for our input, upon which the consistency objective is applied. Please refer to Section 5.2.2 for details.

Let us contrast this method that we call BERT-with-consistency-loss as shown in Figure 5.2. In the traditional BERT-based approach, the input is pre-pended with a special symbol (`[CLS]` in case of BERT and `<s>` in case of RoBERTa). In our approach, we concatenate the special symbol with an extra symbol. We call the extra symbol `[CLSPara]`. This extra token is used explicitly for symmetric classification tasks to ensure prediction consistency. We also experimented with sharing the `[CLS]` representation and a different classification layer for each task but observed a substantial degradation in the performance of the other classification task. We speculate that this was because of the negative knowledge transfer owing to excessive parameter sharing.
The common objective used for fine-tuning BERT-based models is the cross-entropy loss, which maximizes the probability of predicting the correct output class for a given input, given as:

$$\mathcal{L}_{ce}(l, \hat{l}) = -\sum_i l_i \log \hat{l}_i, \tag{5.1}$$

where l is the one-hot representation of the target class, $\hat{l}$ is the softmax output of the model, and i is the associated co-ordinate. As described earlier, this objective may produce an inconsistent prediction based on the order of the two inputs. To overcome this weakness, we propose an additional consistency loss formulated in terms of either the KL or the JS Divergence. We pass the inputs X and Y through the same model twice, once as a pair (X, Y) (called $L2R$) and then as the pair (Y, X) (called $R2L$). Having obtained the outputs from the model for $L2R$ and $R2L$, the final objective function for is as follows:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{ce}(l, \hat{l}_{L2R}) + \mathcal{L}_{ce}(l, \hat{l}_{R2L}) \\ &+ \lambda * \mathcal{D}(p_{L2R} || p_{R2L}),\end{aligned} \tag{5.2}$$

where λ is the weight assigned to the consistency loss. Empirically, adding this multiplicative term λ with annealing helped stabilize the objective and achieve faster convergence. It also ensured that the model had developed the capability to classify the sentence pair correctly (the primary goal) before making it adhere to appropriate symmetric confidence scores. p_{L2R} and p_{R2L} are the associated confidence/softmax vectors assigned by the model for $L2R$ and $R2L$ sentence pairs, and $\mathcal{D}$ is one of the following:

1. $KL(p||q) = \sum_{x \in X} p(x) \log(\frac{p(x)}{q(x)})$
2. $JS(p||q) = \frac{1}{2} KL(p||m) + \frac{1}{2} KL(q||m)$,

Here p, q are probability distributions and $m = \frac{1}{2}(p + q)$. Minimizing divergences between two distributions brings them closer to each other.

5.3 Experimental Setup

5.3.1 Datasets

We experiment with five existing datasets from the GLUE benchmark [138] as well as the PAWS dataset [152][1]. We categorize them under the following headings:

A. For Symmetric Tasks:

1. **QQP:** Quora Question Pairs [57] data set contains pairs of questions marked with either 1 (paraphrases) or 0 (not paraphrases).

2. **PAWS:** Paraphrase Adv. from Word Scrambling [152], contains human-labeled sentence pairs annotated in line with QQP. The uniqueness of this dataset is the creation procedure which involves back-translation and word swapping.

[1]Since the test split of these datasets is not available in the GLUE benchmark[138], we use splits as given in Table 5.1. The validation dataset is kept as original, and the new train and test sets are created by randomly splitting initial train data into train and test sets.

Models	(A) *L2R* and *R2L* Prediction Consistency Mean ± stddev (Section 5.3.2: Evaluation [1])		
	QQP	**PAWS**	**MRPC**
BERT-BASE	96.6 ± 0.15	96.0 ± 0.54	91.1 ± 1.41
BERT-BASE W/ KL	**99.3 ± 0.02**	**98.1 ± 0.12**	**97.7 ± 0.82**
BERT-BASE W/ JS	**98.9 ± 0.05**	**98.1 ± 0.22**	**96.9 ± 0.93**
RoBERTa-BASE	97.0 ± 0.14	96.7 ± 0.25	91.5 ± 0.22
RoBERTa-BASE W/ KL	**99.3 ± 0.03**	**98.9 ± 0.11**	**97.4 ± 0.78**
RoBERTa-BASE W/ JS	**99.1 ± 0.05**	**98.7 ± 0.23**	**96.7 ± 1.11**

Models	(B) *L2R* and *R2L* Confidence Consistency Pearson Correlation [MSE * 1000] (Section 5.3.2: Evaluation [2])		
	QQP	**PAWS**	**MRPC**
BERT-BASE	98.2 [5.89]	96.5 [14.2]	92.7 [17.0]
BERT-BASE W/ KL	<u>99.9 [0.12]</u>	<u>99.6 [0.5]</u>	<u>99.5 [0.3]</u>
BERT-BASE W/ JS	<u>99.8 [0.48]</u>	<u>99.3 [1.9]</u>	<u>99.0 [1.1]</u>
RoBERTa-BASE	98.3 [5.90]	97.4 [10.8]	94.1 [16.3]
RoBERTa-BASE W/ KL	<u>99.3 [0.10]</u>	<u>99.7 [0.4]</u>	<u>99.5 [0.3]</u>
RoBERTa-BASE W/ JS	<u>99.8 [0.40]</u>	<u>99.6 [1.5]</u>	<u>99.0 [1.3]</u>

Models	**(C) Classification Performance Metrics** (Section 5.3.2: Evaluation [3])		
	QQP (Acc/F1)	**PAWS (Acc/F1)**	**MRPC (Acc/F1)**
BERT-BASE	89.5 / 85.7	91.1 / 90.1	78.3 / 82.7
BERT-BASE W/ KL	87.1 / 82.3	88.0 / 86.8	73.0 / 80.7
BERT-BASE W/ JS	89.7 / 86.0	90.5 / 89.5	76.6 / 82.6
RoBERTa-BASE	90.2 / 87.2	92.6 / 91.7	82.4 / 86.0
RoBERTa-BASE W/ KL	87.2 / 82.7	91.5 / 90.5	74.7 / 81.0
RoBERTa-BASE W/ JS	90.0 / 86.6	92.3 / 91.6	79.2 / 84.9

Models	**(C) Classification Performance Metrics** (Section 5.3.2: Evaluation [3])		
	SST2 (Acc)	**QNLI (Acc)**	**RTE (Acc)**
BERT-BASE	94.0 ± 0.10	87.9 ± 0.13	63.0 ± 1.33
BERT-BASE W/ KL	94.1 ± 0.20	71.2 ± 4.15	51.6 ± 1.50
BERT-BASE W/ JS	94.2 ± 0.42	74.5 ± 0.80	50.2 ± 16.90
RoBERTa-BASE	94.4 ± 0.39	89.9 ± 0.47	70.6 ± 2.35
RoBERTa-BASE W/ KL	94.5 ± 0.36	85.3 ± 1.62	58.7 ± 5.40
RoBERTa-BASE W/ JS	95.1 ± 0.12	86.8 ± 1.51	61.4 ± 1.06

Table 5.2: **Parts (A) & (B)**: *L2R* and *R2L* Prediction and Confidence Consistency. **Part (C) Classification Metrics**. (*-BASE) indicate , (*- *W/* *) indicate . Higher Accuracy, Higher Pearson Correlation and lower MSE are better. Numbers in **bold** are statistically significant. <u>Underlined</u> numbers are better on average than baselines. Please refer to Section 5.4.1 for a discussion.

3. **MRPC:** Microsoft Research Paraphrase Corpus [31] comprises human annotated sentence pairs collected from newswire articles.

B. For Single Input Task:

1. **SST2:** Stanford Sentiment Treebank [123]. This is a collection of human-annotated movie reviews. We work with the standard two-class setting where the annotations have

opposite polarities (1 for positive sentiment and 0 otherwise).

C. For Non-symmetric tasks:

1. **QNLI:** Natural Language Inference dataset constructed from SQuAD [109] related to a two-class classification problem to determine if the premise entails a hypothesis or not.
2. **RTE:** Recognizing Textual Entailment [27, 9, 43, 11] Corpus is a combination of multiple RTE datasets containing one of two labels (1 for entailment and 0 for non-entailment).

5.3.2 Evaluation

We analyze the results of the traditional objective as well as our approach on BERT-BASE and RoBERTA-BASE across four different seeds under the following categories:

1. **Prediction Consistency:** This evaluation is done only for the symmetric task. Score = $\frac{\mathbb{1}_{(l_{L2R}=l_{R2L})}}{(\#\text{ of } L2R \text{ Samples})} * 100$, where l_{L2R}, l_{R2L} denote labels for $L2R$ and $R2L$, respectively. Note that this is not related to the ground truth labels.
2. **Confidence Consistency:** We perform these evaluations specifically for the symmetric tasks. This is to analyze how aligned the confidence (softmax output associated with label 1) is predicted by the model for $L2R$ and $R2L$ settings. The metrics used are the Pearson correlation (scaled by 100) and the mean squared error (MSE - scaled by 1000) between the two confidence scores of the test data.
3. **Standard Classification Metrics:** These are task-specific metrics (accuracy/F1) used in the standard GLUE tasks [138]

5.3.3 Implementation Details

To fine-tune the model for symmetric tasks, we club together three paraphrase detection datasets (a) QQP, (b) PAWS, and (c) MRPC. To ensure that all the models see the same data, we augment the dataset with its reverse samples during training. The model is then trained by passing the `[CLSPara]` (Section 5.2.2) representation through a low-capacity classifier and optimized using Equation 5.1 for baseline models and Equation 5.2 for the consistency inducing models (Ours). We then use these models to conduct two sets of evaluations. We individually evaluate the paraphrase detection results on QQP, PAWS, and MRPC. We then take the fine-tuned model obtained above and additionally fine-tune (`[CLS]` or `<s>` token) on the single input task (SST-2) and non-symmetric tasks (QNLI, RTE).

We use the hugging-face library [144] for tokenizing the input, and the pytorch-lightning framework [37] for loading the pre-trained models and fine-tuning them. We optimize the objective using the AdamW [93] optimizer with a learning rate of 2e-5 (obtained through hyperparameter tuning {2e-4, 2e-5, 4e-5, 2e-6}). Since the input contains an additional token `[CLSPara]`, we extend the tokenizer vocabulary for each model. Each model was fine-tuned on a single Nvidia 1080Ti GPU (12 GB) for a maximum of 3 epochs ($\approx$ 6hrs/experiment). In the case of BERT [28], we use the `bert-base-cased` model, while for RoBERTa [91], we use the `RoBERTa-base` model. For training stability, we perform lambda-annealing, i.e., increase the λ parameter from 0.0 to 100.0 as the training progresses. This ensures that the model has developed the capability to classify the sentence pairs with some degree of correctness before making it adhere to the appropriate symmetric confidence scores. We also experimented with fixed λ, but the resultant models converged slowly ($\approx$ 15 epochs).

5.4 Results

Our experiments address three questions:

Q1. What are the shortcomings of the current objective function for symmetric tasks? (Section 5.1, Section 5.4.2)

Q2. Does adding the consistency loss alleviate the inconsistency problem? (Section 5.4.1)

Q3. Can consistency-based fine-tuning improve other downstream tasks? (Section 5.4.1)

5.4.1 Quantitative Analysis

Table 5.2 presents our results. **Parts (A) & (B)** compare $L2R$ and $R2L$ models in terms of prediction consistency and confidence consistency. Models trained with the consistency loss (indicated by $W/*$) assign more similar predictions (indicated by higher scores in (A)) and confidence scores (indicated by higher correlation in (B)) as compared to the base model (indicated by $-BASE$), for both the base models (BERT-BASE/RoBERTA-BASE) and all symmetric test data sets (QQP, PAWS, MRPC). Moreover, the MSE (indicated within square brackets in part (B)) with consistency training is an order of magnitude smaller than without it. The improvements in part (A) are statistically significant at a significance level (α) of 0.01 according to McNemar's statistical test [33].

Part (C) shows the results on **downstream fine-tuning.** Our models (indicated by $W/*$) do not compromise significantly (statistically evaluated) on the classification metrics for QQP, PAWS, and MRPC (F1/accuracy). The consistency loss does not change the accuracy scores of single sentence input tasks (SST-2), but affects the non-symmetric tasks (QNLI, RTE) negatively. This seems natural since the final objective of both tasks is quite different and, in many cases, uncorrelated or negatively correlated. Incorporating consistency loss before fine-tuning on non-symmetric tasks (such as entailment) should, therefore, be avoided.

Limitations: Our goal is to increase the reliability (measured in terms of confidence scores) of the model and not specifically target classification performance metrics like accuracy and F1. Cases where they increase can only partially be attributed to a stricter consistency constraint.

5.4.2 Qualitative Analysis

We sample 30 instances that were assigned opposite labels for *L2R* and *R2L* by the BERT-Base models (majority voting) for QQP, MRPC, and PAWS. An evaluator with NLP expertise analysed these examples and grouped them into recall error types. We then check the predictions for the same set of instances from BERT + JS (recall). Counts for these error types (defined in Section 5.4.3) are shown in Table 5.4. Out of those 30 examples for QQP, MRPC, and PAWS, 26, 26, and 23 respectively get corrected by . In general, the numbers reduce for all error types.

Dataset	Example pair	True label	*L2R* Label	*R2L* Label
MRPC	**(1)** *Shares in Wal-Mart closed at $ 58.28 , up 16 cents , in Tuesday trading on the New York Stock Exchange.* **(2)** *Wal-Mart shares rose 16 cents to close at $ 58.28 on the New York Stock Exchange.*	1	0	1
	(1) *Darren Dopp , a Spitzer spokesman , declined to comment late Thursday.* **(2)** *John Heine , a spokesman for the commission in Washington , declined to comment on Mr. Spitzer 's criticism.*	0	0	1
QQP	**(1)** *How do I retrieve my deleted history from Google chrome?* **(2)** *Can history be retrieved after deleting Google chrome?*	1	0	1
	(1) *Is consciousness possible without self-awareness?* **(2)** *Is self-awareness possible without consciousness?*	0	1	0
PAWS	**(1)** *This iteration is larger and has a smaller storage capacity than its previous versions.* **(2)** *This iteration is smaller and has a greater storage capacity than its previous versions*	0	0	1
	(1) *To get there , take Marine Drive west from the Lions Gate Bridge past Horseshoe Bay to Lighthouse Park and then continue on to 7100 Block Marine Drive.* **(2)** *To get there , take the Marine Drive from the Lions Gate Bridge to the west , past the Horseshoe Bay , Lighthouse Park and continue on to the 7100 Marine Drive block.*	1	1	0

Table 5.3: Sample pairs which are classified differently by the fine-tuned model based on their input order in the standard classification setting in each of the paraphrase dataset. Please refer Section 5.1, Section 5.2.2 for details.

5.4.3 Recall Error Types in Qualitative Analysis

The qualitative analysis compares types of errors with and without consistency loss. The recall error types can be described as follows:

A. QQP:

1. **Different expected answer**: This error is said to occur in the case of QQP when the two input questions have a different expected answer. An example of such a pair is: '*Is consciousness possible without self-awareness?*' and '*Is self-awareness possible without consciousness?*'. The two questions are essential complements of each other.

2. **Different answer type + Additional details**: This error is said to occur when one of the inputs is structured in a way that the answer would solicit additional details. For example, the input pair '*How do I structure a big PHP project?*' and '*How do I build a perfect PHP project?*' are similar - but nuances between 'structuring' and 'building' a project may result in different answers.

3. **Additional details and/or pronoun change**: The input pair '*What are the best ways to get thick and wavy hair?*' and '*How can I get thick, wavy hair (as a guy)?*' is similar - although the latter uses the first-person pronoun.

B. MRPC:

1. **Additional details missing**: One of the inputs contains information (*i.e.*, details) that are not present in the other input. For example, '*The caretaker, identified by church officials as Jorge Manzon, was believed to be among the nine missing - some of them children*' contains the number of missing persons that are not present in '*The caretaker, identified by church officials as Jorge Monzon, was believed to be among the missing, who are presumed dead*'.

2. **Reordering of phrases**: The two inputs contain the same information although the information may be represented using different phrasal structures. For example, '*Shares in Wal-Mart closed at $ 58.28 , up 16 cents , in Tuesday trading on the New York Stock Exchange.*' conveys the same information as '*Wal-Mart shares rose 16 cents to close at $ 58.28 on the New York Stock Exchange .*' The former uses passive voice while the latter uses 'shares' as the main verb.

3. **Named entities and pronouns**: One input replaces entities with pronouns, as in the case of '*The bonds traded to below 60 percent of face value earlier this year*' and '*They traded down early this year to 60 percent of face value on fears Aquila may default .*'

4. **Focus of sentences is different**: While information in one input is subsumed by the other, the latter might focus on a broader context. For example, '*A power cut in New York in 1977 left 9 million people without electricity for up to 25 hours*' is implied in the sentence '*The outage resurrected memories of other massive power blackouts , including one in 1977 that left about 9 million people without electricity for 25 hours .*' However, the latter describes a resurrection of memories of the event in 1977.

5. **Synonyms**: One or more words in an input may be replaced by its synonyms in the other input. For example, '*In 2001 , the number of death row inmates nationally fell for the first time in a generation*' can be converted to '*In 2001 , the number of people on death row dropped for the first time in a decade.*' by replacing the word 'fell' with 'dropped'.

C. PAWS

1. **Nouns/adjectives are changed**: In the case of these errors, adjectives are replaced. An example pair is '*This iteration is larger and has a smaller storage capacity than its previous versions*' and '*This iteration is smaller and has a greater storage capacity than its previous versions .*'

2. **Named entities are changed**: This refers to pairs where named entities (locations/people) are different. An example is the pair '*When Mexico was within Los Angeles , Botello was chief of staff for Mexican General Ramirez y Sesma . His two brothers also married daughters of the general*' and '*When Los Angeles was within Mexico , Botello was Chief of Staff of the Mexican General Ramirez y Sesma , his two brothers also married the general 's daughters .*'

5.5 Summary

In this chapter, we proposed an additional objective: consistency loss between $L2R$ and $R2L$ predictions to alleviate the problem of input order-sensitive inconsistency in the case of symmetric classification tasks. For three symmetric classification tasks, our proposed solution improves consistency in terms of Pearson's correlation and MSE. As expected, consistency loss results in a drop in the performance of non-symmetric tasks such as QNLI and RTE. Surprisingly, KL divergence results in marginally higher consistency than the JS counterpart. We leave this

Error type		
QQP		
Different expected answer	4	0
Different answer type + Additional details	8	1
Different answer type + Additional details + Pronoun change	1	0
Additional details and/or pronoun change	17	3
MRPC		
Additional details missing	13	2
Reordering of phrases	3	0
Named entities and pronouns	6	1
Focus of sentences is different	6	0
Synonyms	2	1
PAWS		
Phrases are changed	10	4
Nouns/adjectives are changed	12	1
Nouns/adjectives and phrases are changed	4	0
Named entities are changed	3	1
Names entities and nouns/adjectives are changed	1	1

Table 5.4: Recall errors in QQP, MRPC & PAWS: BERT ( ) and BERT with JS ( ). Please refer to Section 5.4.2.

analysis for future work. Our qualitative analysis shows that all error types, including changes in phrases or addition/deletion of details, are reduced when the consistency loss is incorporated.

While consistency loss ensures that the predicted labels are the same even if the order of inputs is swapped, it can be used in the future to ensure expected outputs for anti-symmetric classification tasks (where $\mathbb{P}(L2R) = 1 - \mathbb{P}(R2L)$) like next and previous sentence prediction, where reordering the inputs must result in an opposite predicted label. In addition, the proposed method can be applied to evaluate paraphrase generation models [77, 78] as well. To validate that paraphrasing models are indeed generating semantically similar outputs, BERT-with-consistency can be used to either evaluate and filter out incorrect generations or be used as an objective to train learned metrics like BLEURT [117].

Chapter 6

Summary, Conclusions and Future Work

This thesis deals with paraphrase generation and detection. Paraphrase generation involves generating a text that conveys the same meaning as the source text but is expressed in different words or structures. Paraphrase detection identifies whether a pair of texts have the same meaning or intent. This thesis discussed approaches to induce constraints and consistency in paraphrase generation and detection. Specifically, we focused on constraints in the form of diversity and syntax. Toward this, we presented approaches for diversity-aware and syntax-aware paraphrase generation. Following that, we presented an approach for consistency-aware paraphrase detection. Table 6.1 summarises the problems, examples, techniques, and key takeaways presented in this thesis. The examples refer to the expected input/outputs for each problem.

In general, the approaches presented in this thesis enhance the applicability of paraphrase generation and detection models in various natural language processing tasks.

6.1 Diversity in paraphrase generation

Diversity refers to the ability of a paraphrase generation model to generate sentences that differ in their surface forms while retaining the meaning of the input sentence. Therefore, if the input is '*how do i increase body height ?*', the expected outputs would be '*how could i increase my height ?*', '*what should I do to increase my height ?*' and so on. These output sentences convey the meaning of the input sentence but vary in their expressions. Prior works had limited abilities to address this issue. We proposed a method that adds diversity constraint to the decoding objective to handle this problem. We call this method DiPS and it outperforms past approaches. DiPS maximises a novel submodular objective function designed for paraphrasing.

Problems in Paraphrasing	Examples	Technique	Key Takeaways
Diversity in paraphrase generation	**Input (X):** - how do i increase body height ? **Output (Y):** - how could I increase my height ? - what should I do to increase my height ? - what are the fastest ways to increase my height ? - is there any proven method to increase height ?	Monotone submodular function maximisation	DiPS model offers high diversity without compromising on fidelity Useful for data augmentation
Syntacticality in paraphrase generation	**Input** **(X):** What are pure substances ? What are some examples ? **Exemplar sentence** **(Z)**: What are the characteristics of the Elizabeth theatre ? **Output** **(Y):** What are the examples of a pure substance ?	TreeLSTM-based paraphrase generation	SGCP was the state-of-the-art syntax-guided paraphrase generation model [154]
Consistency in paraphrase detection	**Input** **X:** a provision government or a revolutionary government has been declared several times by insurgent groups in philippines . **Y:** a provision government or a revolutionary government has been declared several times in philippines by insurgent groups . **Output** **For (X, Y) as input:** 1 (88.3) **For (Y, X) as input:** 1 (87.9)	Minimise f-divergence between L2R and R2L label scores	Reduced inconsistency in confidence scores predicted by pre-trained models

Table 6.1: Summary of problems and approaches presented in this thesis.

The function allows a high degree of freedom to control fidelity and diversity of paraphrases. We applied this to multiple data-augmentation settings on intent and question classification tasks and observed consistent performance improvements.

6.2 Syntacticality in paraphrase generation

SGCP allows generation of syntactically controlled paraphrases from two sentences: input and exemplar. The input sentence provides the content, while the exemplar sentence provides the syntax. For example, assume that the input sentence is '*what are pure substances ? what are some examples ?*' and the exemplar sentence is '*what are the characteristics of the elizabeth theatre ?*'. The paraphrase generator must be able to discover the structure of the exemplar sentence that is closest to the content of the input sentence: '*what are the A of B ?*' and extract the values of A and B as '*examples*' and '*pure substance*' respectively to produce the output sentence '*what are some examples of a pure substance ?*'. Prior works have used a standard LSTM-based approach with linearized constituency-based parse tree to capture syntactic information. Deriving from this, we modify the LSTM architecture for the case of non-linearized constituency based parse tree. The modified LSTM-based tree encoder learns the syntax of the exemplar sentence. During decoding, a pointer-generator-based decoder attends to relevant tokens in the input sentence for generating the paraphrase. As a result, the output contains the semantics of the input sentence cast into the structure of the exemplar sentence. This approach, known as SGCP, generated multiple paraphrases according to different tree-level granularity. This granularity depends on the height of the constituency tree extracted from the exemplar. We experimented with two datasets, QQP-Pos and ParaNMT-small, and showed that SGCP outperforms state-of-the-art baselines.

6.3 Consistency in paraphrase detection

Finally, we looked at the problem of inconsistencies in fine-tuned embedding-based pre-trained cross-encoder models for paraphrase detection. Prior cross-encoder-based methods were sensitive to input order in symmetric classification tasks such as paraphrase detection. This was inconsistent with the definitions of semantic similarity, which is an equivalent relationship: if $\mathbf{X}$ is similar to $\mathbf{Y}$, then $\mathbf{Y}$ is similar to $\mathbf{X}$. We proposed an additional objective: consistency loss between $Left - to - right$ and $Right - to - left$ predictions to alleviate the problem of input order-sensitive inconsistency in the case of symmetric classification tasks such as paraphrase detection. For three symmetric classification datasets: QQP, MRPC, and PAWS, our proposed solution improved consistency in terms of Pearson's correlation and Mean Squared Error.

6.4 Future Work

We now discuss how our approaches may impact constrained paraphrase generation, data-augmentation, and other NLG problems in conversational agents and text summarization.

While we have shown the application of using DiPS in augmenting samples for simple classification tasks, we anticipate its utility in self-training for generative models as well. Diversity-aware paraphrasing has been applied to Transformer models for obtaining multiple candidates [29, 38]. A simple modification in the objective of DiPS opens up avenues for other NLG setups which require generations to possess diversity while not lacking the other required qualities. For example, goal-oriented conversational agents need diverse utterances without losing their context. Similarly, summarization systems must remove redundant clauses while incorporating diverse key information in the original article. This can be achieved by augmenting diverse paraphrases.

We anticipate that syntax-aware paraphrasing will help provide additional information to the NLG systems, help build competitive test sets for assessing the robustness of general NLP models, and aid in science journalism [115]. Syntax-guided paraphrasing generates candidates with diverse structures, potentially enriching test sets after thorough human evaluation for assessing other natural language generation (NLG) systems.

Although we explored an LSTM-based architecture for both DiPS and SGCP, we anticipate better results using current state-of-the-art self-attention-based models[1]. Investigating these approaches in guiding multilingual NLG models also seems like a natural next step [92, 147]. Scaling these methods to larger datasets could provide insights into their performance on more diverse sentence structures. Furthermore, exploring the approach in other languages could extend its applicability and usefulness. Additionally, investigating the potential of different tree encoders to provide additional syntactic information to guide the paraphrase generator could further improve the quality and diversity of the generated paraphrases.

While building generation models for NLP caters to a wide variety of tasks, the texts generated by those models must be adequately assessed. To validate that NLG models are generating semantically correct outputs, BERT-with-consistency can either evaluate and filter out incorrect generations or as an additional objective for training learned metrics like BLEURT [117]. Another interesting direction is to apply the objective to anti-symmetric classification tasks (where $\mathbb{P}(L2R) = 1 - \mathbb{P}(R2L)$) like next and previous sentence prediction, where reordering the inputs must result in an opposite predicted label to obtain the expected order-based outputs. We anticipate that such a formulation may help the models make more informed predictions.

Broadly speaking, techniques presented in this thesis will directly impact at least two NLG tasks: summarization and conversational agents.

As mentioned earlier, incorporating diversity and syntacticality in paraphrase generation can have significant benefits in building better summarization systems and conversation agents. We discuss them below.

(a) By generating a diverse set of paraphrases, summarization systems can select the most suitable alternative expression or phrase that conveys the intended meaning while considering overall summary coherence, clarity, brevity, and relevance. This can also help prevent over-reliance on specific phrasing patterns that can result in monotonous and uninteresting summaries.

(b) In conversation agents, paraphrase generation systems can facilitate generating more varied and natural-sounding responses. By incorporating diversity, the conversation agent can generate a broader range of responses that incorporate different stylistic variations tailored to the user's preferences and conversational style. This can help enhance the overall user experience by enabling the conversation agent to provide more personalized and engaging interactions.

[1] Subsequent works [16, 127] explored the methods through the pre-trained models and found the resulting generations useful for data augmentation.

Bibliography

[1] Roee Aharoni and Yoav Goldberg. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2021. URL https://www.aclweb.org/anthology/P17-2021. 50

[2] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1098. URL https://aclanthology.org/D17-1098. 4

[3] Richard Chase Anderson and Alice Davison. Conceptual and empirical bases of readability formulas. *Center for the Study of Reading Technical Report; no. 392*, 1986. 51

[4] Peter G Anick and Suresh Tipirneni. The paraphrase search assistant: terminological feedback for iterative information seeking. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 153–159. ACM, 1999. 29

[5] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013. 30

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 13, 18, 37, 41, 58

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. 70

[8] Satanjeev Banerjee and Alon Lavie. 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, 2005. 40, 61

[9] Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4. Venice, 2006. 76

[10] Regina Barzilay and Lillian Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 16–23. Association for Computational Linguistics, 2003. 13

[11] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009. 76

[12] Delphine Bernhard and Iryna Gurevych. Answering learners' questions by retrieving question paraphrases from social q&a sites. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 44–52. Association for Computational Linguistics, 2008. 29

[13] Rahul Bhagat and Eduard Hovy. Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472, September 2013. doi: 10.1162/COLI_a_00166. URL https://aclanthology.org/J13-3001. 2, 11

[14] Alexei Borodin. Determinantal point processes, 2009. URL https://arxiv.org/abs/0911.1153. 38

[15] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. URL https://www.aclweb.org/anthology/K16-1002. 14

[16] Tien-Cuong Bui, Van-Duc Le, Hai-Thien To, and Sang Kyun Cha. Generative pre-training for paraphrase generation by representing and predicting spans in exemplars.

In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 83–90, 2021. doi: 10.1109/BigComp51126.2021.00025. 85

[17] Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3152–3158. AAAI Press, 2017. 12

[18] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, 2018. 50

[19] Zhangming Chan, Yuchi Zhang, Xiuying Chen, Shen Gao, Zhiqiang Zhang, Dongyan Zhao, and Rui Yan. Selection and generation: Learning towards multi-product advertisement post generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3818–3829, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.313. URL `https://aclanthology.org/2020.emnlp-main.313`. 6

[20] Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1599. URL `https://www.aclweb.org/anthology/P19-1599`. xii, 13, 14, 50, 51, 58, 59, 61, 63, 67

[21] Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. A multi-task approach for disentangling syntax and semantics in sentence representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2453–2464, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1254. URL `https://www.aclweb.org/anthology/N19-1254`. 14

[22] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014. 16, 18, 37

[23] Christopher Olah. Understanding lstm networks, 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Online; accessed 15-July-2022]. x, 15, 17

[24] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014. 59

[25] Herbert H Clark and Eve V Clark. Semantic distinctions and memory for complex sentences. *Quarterly journal of experimental psychology*, 20(2):129–138, 1968. 51

[26] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190, 2018. 36

[27] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005. 76

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 14, 18, 20, 77

[29] Kaustubh D. Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Srivastava, Samson Tan, Tongshuang Wu, Jascha Sohl-Dickstein, Jinho D. Choi, Eduard H. Hovy, Ondrej Dusek, Sebastian Ruder, Sajant Anand, Nagender Aneja, Rabin Banjade, Lisa Barthe, Hanna Behnke, Ian Berlot-Attwell, Connor Boyle, Caroline Brun, Marco Antonio Sobrevilla Cabezudo, Samuel Cahyawijaya, Emile Chapuis, Wanxiang Che, Mukund Choudhary, Christian Clauss, Pierre Colombo, Filip Cornell, Gautier Dagan, Mayukh Das, Tanay Dixit, Thomas Dopierre, Paul-Alexis Dray, Suchitra Dubey, Tatiana Ekeinhor, Marco Di Giovanni, Rishabh Gupta, Rishabh Gupta, Louanes Hamla, Sang Han, Fabrice Harel-Canada, Antoine Honore, Ishan Jindal, Przemyslaw K. Joniak, Denis Kleyko, Venelin

Kovatchev, Ashutosh Kumar, and et al. Nl-augmenter: A framework for task-sensitive natural language augmentation. *CoRR*, abs/2112.02721, 2021. URL https://arxiv.org/abs/2112.02721. 84

[30] Mladen Dimovski, Claudiu Musat, Vladimir Ilievski, Andreea Hossman, and Michael Baeriswyl. Submodularity-inspired data selection for goal-oriented chatbot training based on sentence embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4019–4025. International Joint Conferences on Artificial Intelligence Organization, 7 2018. 13

[31] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. 3, 75

[32] Qingxiu Dong, Xiaojun Wan, and Yue Cao. ParaSCI: A large scientific paraphrase dataset for longer paraphrase generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 424–434, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.33. URL https://aclanthology.org/2021.eacl-main.33. 4

[33] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1128. URL https://aclanthology.org/P18-1128. 64, 77

[34] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery. In *Advances in Neural Information Processing Systems*, pages 19–27, 2012. 29, 38

[35] Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. Dissimilarity-based sparse subset selection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11): 2182–2197, 2016. 36, 39

[36] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1608–1618, 2013. 29

[37] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3:6, 2019. 77

[38] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.84. URL https://aclanthology.org/2021.findings-acl.84. 84

[39] Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics, 2006. 31

[40] Tomáš Foltýnek, Norman Meuschke, and Bela Gipp. Academic plagiarism detection: A systematic literature review. *ACM Comput. Surv.*, 52(6), oct 2019. ISSN 0360-0300. doi: 10.1145/3345317. URL https://doi.org/10.1145/3345317. 7

[41] S Fujishige. Submodular functions and optimization. *Annals of Discrete Mathematics*, 58, 2005. 30

[42] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/germain15.html. 20

[43] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007. 76

[44] Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, 2013. 12

[45] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *AAAI Conference on Artificial Intelligence*, 2018. 12, 13, 29, 30, 36, 41, 42

[46] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 13, 14

[47] Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics, 2007. 13

[48] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 53

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 16, 37, 58

[50] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL https://aclanthology.org/P17-1141. 4

[51] http://www.ocw.upj.ac.id/files/Slide LSE-04.pdf. Examples of paraphrasing, 2022. URL http://www.ocw.upj.ac.id/files/Slide-LSE-04.pdf. [Online; accessed 15-July-2022]. 2

[52] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org, 2017. 13, 49

[53] Yufang Huang, Wentao Zhu, Deyi Xiong, Yiye Zhang, Changjian Hu, and Feiyu Xu. Cycle-consistent adversarial autoencoders for unsupervised text style transfer. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2213–2223, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.201. URL https://www.aclweb.org/anthology/2020.coling-main.201. 14

[54] Judith W Irwin. The effects of explicitness and clause order on the comprehension of reversible causal relationships. *Reading Research Quarterly*, pages 477–488, 1980. 51

[55] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 62

[56] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013. 13

[57] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs, 2017. URL https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs. 74

[58] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1170. URL https://www.aclweb.org/anthology/N18-1170. xii, 12, 13, 50, 51, 58, 61, 63

[59] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357. 19

[60] Stefanie Jegelka and Jeff Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *CVPR 2011*, pages 1897–1904, 2011. doi: 10.1109/CVPR.2011.5995589. 13

[61] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation, 2022. URL https://arxiv.org/abs/2202.03629. iii, 3

[62] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 978-0-374-27563-1. 1

[63] Evelyn Walker Katz and Sandor B Brent. Understanding connectives. *Journal of Memory and Language*, 7(2):501, 1968. 51

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 41, 63

[65] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 12

[66] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *In Proceedings of ICLR*, 2014. 14, 59

[67] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 131–141, 2014. 13, 33

[68] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 452–457, 2018. 13, 40

[69] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *European conference on computer vision*, pages 65–81. Springer, 2002. 13

[70] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014. 22, 23, 30

[71] Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):32, 2011. 13

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 13

[73] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011. 27, 38

[74] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. 26, 36, 38

[75] Ashutosh Kumar. Discovering non-monotonic autoregressive ordering for text generation models using sinkhorn distributions. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/non-monotonic-autoregressive-ordering/. https://iclr-blog-track.github.io/2022/03/25/non-monotonic-autoregressive-ordering/. 21

[76] Ashutosh Kumar and Aditya Joshi. Striking a balance: Alleviating inconsistency in pre-trained models for symmetric classification tasks. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1887–1895, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.148. URL https://aclanthology.org/2022.findings-acl.148. 5

[77] Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1363. URL https://www.aclweb.org/anthology/N19-1363. 5, 13, 50, 81

[78] Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:330–345, 2020. 5, 81

[79] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, 2017. 35

[80] Elena T Levy. The roots of coherence in discourse. *Human Development*, 46(4):169–188, 2003. 51

[81] Jiwei Li and Dan Jurafsky. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*, 2016. 29

[82] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015. 12

[83] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, 2016. 12, 29, 36, 50

[84] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th international conference on World wide web*, pages 71–80. ACM, 2009. 29

[85] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002. 36

[86] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878. Association for Computational Linguistics, 2018. 12, 29, 30, 36, 41

[87] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1421. URL `https://www.aclweb.org/anthology/D18-1421`. 13

[88] Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1332. URL `https://www.aclweb.org/anthology/P19-1332`. 13

[89] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. 61

[90] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011. 13

[91] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 14, 18, 62, 77

[92] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020. doi: 10.1162/tacl_a_00343. URL https://aclanthology.org/2020.tacl-1.47. 85

[93] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 77

[94] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166. 19, 37

[95] Nitin Madnani and Bonnie J Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387, 2010. 13

[96] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5. URL http://nlp.stanford.edu/IR-book/information-retrieval-book.html. 40

[97] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010. 53, 62

[98] Kathleen R McKeown. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10, 1983. 12, 13

[99] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10350. URL https://ojs.aaai.org/index.php/AAAI/article/view/10350. 40

[100] Preksha Nema, Mitesh M Khapra, Anirban Laha, and Balaraman Ravindran. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1063–1072, 2017. 29

[101] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1): 265–294, 1978. 24, 30

[102] Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91, 2017. 29

[103] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`. 40, 61

[104] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016. 12

[105] Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1263. URL `https://www.aclweb.org/anthology/N19-1263`. 13, 50

[106] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1119. URL `https://aclanthology.org/N18-1119`. 4

[107] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1275. 12, 13, 29, 30

[108] Chris Quirk, Chris Brockett, and William Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 142–149, 2004. 13

[109] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 76

[110] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems*, pages 3239–3249, 2017. 13

[111] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410. 14

[112] Ehud Reiter. A structured review of the validity of bleu. *Computational Linguistics*, 44 (3):393–401, 2018. doi: 10.1162/coli_a_00322. 62

[113] Alan Ritter, Colin Cherry, and William B Dolan. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593, 2011. 36

[114] Paul Rubenstein, Olivier Bousquet, Josip Djolonga, Carlos Riquelme, and Ilya O Tolstikhin. Practical and consistent estimation of f-divergences. *Advances in Neural Information Processing Systems*, 32:4070–4080, 2019. 71

[115] David M. Secko, Elyse Amend, and Terrine Friday. Four models of science journalism. *Journalism Practice*, 7(1):62–80, 2013. doi: 10.1080/17512786.2012.691351. URL https://doi.org/10.1080/17512786.2012.691351. 5, 84

[116] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099. 12, 13, 57, 58

[117] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL https://aclanthology.org/2020.acl-main.704. 81, 85

[118] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162. 18, 53

[119] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841, 2017. 13, 49, 62

[120] Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1159. URL https://www.aclweb.org/anthology/D16-1159. 14, 51

[121] Advaith Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014. 6, 50

[122] Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268. Association for Computational Linguistics, 2009. 40

[123] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality

over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. 75

[124] Yiping Song, Rui Yan, Yansong Feng, Yaoyuan Zhang, Dongyan Zhao, and Ming Zhang. Towards a neural conversation model with diversity net using determinantal point processes. In *AAAI*, 2018. 12, 29

[125] Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, 2016. 50

[126] Peter Stobbe and Andreas Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems*, pages 2208–2216, 2010. 33

[127] Jiao Sun, Xuezhe Ma, and Nanyun Peng. AESOP: Paraphrase generation with adaptive syntactic control. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5176–5189, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.420. URL https://aclanthology.org/2021.emnlp-main.420. 85

[128] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 18, 37

[129] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015. 53

[130] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online, June 2021. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.naacl-main.28. 14

[131] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17 (205):1–37, 2016. URL http://jmlr.org/papers/v17/16-272.html. 20

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. 19, 70, 72

[133] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.12340. URL https://ojs.aaai.org/index.php/AAAI/article/view/12340. 12, 29, 36, 41, 42

[134] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015. 50

[135] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015. 57

[136] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *In Proceedings of ICLR*, 2016. URL https://arxiv.org/abs/1511.06391. 20

[137] Jan Vondrák. Optimization of submodular functions tutorial, 2009. https://theory.stanford.edu/~jvondrak/data/submod-tutorial-1.pdf. 22

[138] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446. 74, 76

[139] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in {bert}. In *International Confer-*

ence on Learning Representations, 2021. URL https://openreview.net/forum?id=onxoVA9FxMw. 72

[140] William Yang Wang and Diyi Yang. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, 2015. 13, 29

[141] Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.555. URL https://www.aclweb.org/anthology/2020.emnlp-main.555. 72

[142] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1002. URL https://aclanthology.org/D19-1002. 19

[143] John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, 2018. 59

[144] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. 77

[145] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics, 2010. 13, 40

[146] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297, 2015. 29

[147] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41. 85

[148] Xuewen Yang, Yingru Liu, Dongliang Xie, Xin Wang, and Niranjan Balasubramanian. Latent part-of-speech sequences for neural machine translation, 2019. 50

[149] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298, 2018. 13, 49

[150] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989. 61

[151] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015. 13, 29

[152] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1131. URL https://www.aclweb.org/anthology/N19-1131. 4, 62, 74

[153] Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. Combining multiple resources to improve smt-based paraphrasing model. *Proceedings of ACL-08: HLT*, pages 1021–1029, 2008. 13

[154] Jianing Zhou and Suma Bhat. Paraphrase generation: A survey of the state of the art. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language*

Processing, pages 5075–5086, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.414. URL https://aclanthology.org/2021.emnlp-main.414. 83

[155] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 14