MONASH University

Context-aware Natural Language Understanding and Generation

Xuanli He

Doctor of Philosophy

A thesis submitted for the degree of Doctor of Philosophy at
Monash University in 2022
School of Information Technology

Copyright notice

Abstract Natural language understanding and generation are two critical components of natural language processing. Natural language understanding seeks to describe the semantics and syntax of human text, while natural language generation aims to automate the generation of human-like text. Although both have benefited from the recent flood of research on deep learning, they have also been criticized for lacking context modeling. Since human languages are context-dependent by nature, there are many context-related system errors in current natural language understanding and generation models. This research considers contextual information and studies four context-dependent tasks to fill this gap.

The context contains information that helps alleviate uncertainty in understanding the human text. In natural language processing, the context is usually associated with the surrounding words or phrases. However, the definition of context goes beyond this basic meaning. In this work, we expand the scope of the context and explore its effectiveness in four tasks of natural language understanding and generation.

We first investigate the role of short-range context in subword segmentation for neural machine translation. Existing works segment uncommon words so abruptly that the resultant subwords do not respect linguistic rules. We propose a novel mixed character-subword Transformer that conditions source- and target-side contexts, and leverages dynamic programming to split words into subwords. Our empirical results indicate that context-conditioned segmentation can produce morphologically plausible subwords, leading to significant improvements in translation quality.

Then we explore long-range context-dependent natural language understanding in dialogue systems. We start by illustrating that one can modify a scene graph via a language command. To achieve this goal, we create three datasets and describe this modification task as a conditional generation problem. The desired scene graph can be produced depending on the context, *i.e.,* the original graph and the modification command. We also propose a novel architecture to address this modification problem. Our comprehensive studies demonstrate that the proposed architecture can meet the desideratum and surpass all baseline models.

Next, a conversation involves back-and-forth communication. Contexts in a dialogue system can go beyond an utterance. Thus, we examine the ability to model a long-range dependency in a conversational semantic parsing task. In addition, we find that a standard parser can place incorrect arguments into a logical form. We superimpose a copy mechanism onto the context-dependent parser to solve these problems. Our experiments show that the proposed model can deliver sizeable improvements over the baselines.

Finally, we argue that the definition of context is not restricted to the neighboring words or sentences. Instead, given a specific task, the format, wording, and genre of the input sentences must conform to the specified rules. Hence, we denote such meta information as the task-specific context. We fine-tune or condition generic generative models on in-distribution examples to capture this high-level context. Then we harvest a large quantity of synthetic in-domain unlabeled data from the tailored models. Finally, we demonstrate that one can significantly advance self-training, knowledge distillation, and few-shot learning using the synthetic in-domain data.

Declaration

This thesis is an original work of research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: _____

Print Name: Xuanli He

Date: _____

Publications during candidature

The publications arising from this thesis are:

- **Xuanli He**, Quan Tran, and Gholamreza Haffari. 2019. A Pointer Network Architecture for Context-Dependent Semantic Parsing. In Proceedings of the 17th Annual Workshop of the Australasian Language Technology Association, pages 94–99, Sydney, Australia. Australasian Language Technology Association.

- **Xuanli He**, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic Programming Encoding for Subword Segmentation in Neural Machine Translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3042–3051, Online. Association for Computational Linguistics.

- **Xuanli He**, Quan Hung Tran, Gholamreza Haffari, Walter Chang, Zhe Lin, Trung Bui, Franck Dernoncourt, and Nhan Dam. 2020. Scene Graph Modification Based on Natural Language Commands. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 972–990, Online. Association for Computational Linguistics.

- **Xuanli He**, Gholamreza Haffari, and Mohammad Norouzi. 2022. Generate, Annotate, and Learn: NLP with Synthetic Text, Accepted to Transactions of the Association for Computational Linguistics 2022. Association for Computational Linguistics.

Other publications arising from my research degree candidature are:

- **Xuanli He**, Quan Tran, William Havard, Laurent Besacier, Ingrid Zukerman, and Gholamreza Haffari. 2018. Exploring Textual and Speech information in Dialogue Act Classification with Speaker Domain Adaptation. In Proceedings of the Australasian Language Technology Association Workshop 2018, pages 61–65, Dunedin, New Zealand. Australasian Language Technology Association.

- **Xuanli He**, Gholamreza Haffari, and Mohammad Norouzi. 2018. Sequence to Sequence Mixture Model for Diverse Machine Translation. In Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 583–592, Brussels, Belgium. Association for Computational Linguistics.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Gholamreza Haffari and Dr Mohammad Norouzi, for opening the research door to the field of NLP for me. Without their dedicated and generous supervision, I would not have had the chance to foster my research skills, develop critical thinking, and eventually complete my PhD path and become a qualified researcher. In addition to their scientific knowledge and initiative conduct, they have also helped me shape valuable attitudes toward research and life.

I would like to specially thank all members of my panel committee, namely, Prof. Graham Farr, Dr Lan Du, and Dr Shirui Pan, for their invaluable feedback and support during my candidature. This thesis has significantly benefited from the thorough comments and valuable suggestions made by my examiners, particularly Prof. Hal Daumé (University of Maryland) and Assoc. Prof. Shafiq Joty (Nanyang Technological University).

I would like to express my great appreciation to my fantastic internship hosts, Dr Quan Tran (Adobe), Dr Walter Chang (Adobe), Dr Trung Bui (Adobe), Dr Zhe Lin (Adobe), Dr Yi Xu (Amazon), Dr Iman Keivanloo (Amazon), and Dr Xiang He (Amazon) for their kind support and inspiring motivation throughout my internship programs. I would also like to thank Monash University for providing me with financial support during my PhD, without which it would have been impossible for me to start my research career. Also thanks to the support team at Monash Advanced Research Computing Hybrid (MonARCH) and Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) for all their help with using the GPU nodes for running my experiments.

This thesis would not have been possible without the companionship and mental support of my peers and colleagues. First, I would like to thank Quan for the fruitful discussion on my first submission in my PhD; Sameen Maruf and Poorya Zaremoodi for the technical help; Narjes Askarian, Fahimeh Saleh, Snow Situ, Trang Vu, Najam Zaidi, and Jinming Zhao for always listening to my chores, and last but not least, Bhagya Hettige, Zhuang Li, Islam Nassar, Mostafa Rizk, Ruangsak Trakunphutthirak, Lizhen Qu, Qiongkai Xu, Laksri Wijerathna, Tong Wu, and Haolan Zhan, it is a pleasure to have known you. I also developed strong bonds with Mahmoud Ahmed Hossameldeen and Srinibas Swain, with whom I shared my sorrow and happiness, bringing me a sense of belonging.

I would also like to express my sincere thanks to my life-long, beloved friends, Xue Bai, Weijia Chen, Hao Duan, Lingxiao Jiang, Lingjuan Lyu, Xiaoping Li, Ruinan Wan,

Xiaodan Yang, Yiming Yang, and Maggie Zhang, whose friendship has consolidated as the time flew.

Finally, I cannot express enough gratitude to my parents, who brought me into this beautiful world. Your endless love, encouragement, and moral and emotional support have shaped me and helped me find my life.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
GRU	**G**ated **R**ecurrent **U**nit
KD	**K**nowledge **D**istillation
LLMs	**L**arge **L**anguage **M**odels
LM	**L**anguage **M**odel
LSTM	**L**ong **S**hort-**T**erm **M**emory
NLG	**N**atural **L**anguage **G**eneration
NLP	**N**atural **L**anguage **P**rocessing
NLU	**N**atural **L**anguage **U**nderstanding
NMT	**N**atural **M**achine **T**ranslation
PLMs	**P**re-trained **L**anguage **M**odels
RNN	**R**ecurrent **N**eural **N**etwork

List of Notations

$\boldsymbol{x}$	The source/input sentence
$\boldsymbol{y}$	The target/output sentence/label
$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\lvert\mathcal{D}\rvert}$	Training data
$\boldsymbol{x}_i, \boldsymbol{y}_i$	The ith word in the source/target sentence
$\boldsymbol{h}_i, \boldsymbol{s}_i$	The hidden representation of ith word in the source/target sentence
$\boldsymbol{W}, \boldsymbol{U}$	Parameter matrices
$\boldsymbol{b}$	Parameter vector

Chapter 1

Introduction

Natural language generation (NLG) is an essential subfield of natural language processing (NLP). It aims to automatically produce fluent text from various forms of input. There has been a surge of interest in NLG from both academia and industry (Reiter and Dale, 2000). Researchers have highlighted that the study of NLG has fostered the development of artificial intelligence, cognitive science, and human-computer interaction. From the perspective of the application, many companies have leveraged NLG technology to automate tasks involving routine document creation. Moreover, technology companies have introduced numerous and various text-generation application programming interfaces (APIs), such as machine translation, document summarization, automatic speech recognition, *etc.*, to serve millions of end-users.

Natural language understanding (NLU) is another critical area of the NLP field. As opposed to NLG, NLU focuses on capturing the correlations between linguistic units and understanding the high-level encoded information. NLU can be considered akin to reading comprehension, whereas NLG is associated with composition. However, NLG and NLU are not separate. In text-to-text generation, NLU is responsible for converting discrete input tokens into linguistic-aware representations, which will eventually be decomposed into another set of discrete symbols via NLG. Moreover, NLU can be leveraged to convert human-written text into machine-understandable representation to facilitate communication between humans and machines.

The basis of NLG is to learn a function mapping the source signals to target outputs. Early research in this area approached the generation problem via a list of hand-crafted

rules. Such labor-intensive approaches limited the development of NLG to simple applications such as weather summary (Goldberg, 1993), activity summary (McKeown et al., 1994), software manuals (Paris and Vander Linden, 1996), *etc.* These systems are comprised of multiple brittle, pipelined sub-components. Consequently, they suffer from compounding errors when any module raises an issue. Due to the complexity of hand-crafted engineering, statistical approaches were introduced to reduce the need for arduous manual effort. It has been shown that sentences generated via statistical approaches are less fluent, although these approaches achieve partial success (Bentivogli et al., 2016; Koehn and Knowles, 2017; Toral and Sánchez-Cartagena, 2017). In addition, statistical approaches focus on the statistical alignment between source and target elements, but ignore semantic and syntactic understanding of the source text. Finally, sophisticated statistical models require multiple tunable modules to deliver decent results.

As with NLG, parsing of a natural sentence into a structured representation via NLU was initiated using rule-based algorithms (Woods, 1973; Johnson, 1984). However, since rule-based approaches require domain-specific knowledge and do not adapt quickly to multiple domains, statistical approaches were suggested in order to eliminate this lack of flexibility (Zelle and Mooney, 1996; Wong and Mooney, 2006). Nevertheless, the weaknesses of statistical approaches in NLG apply to NLU as well.

Thanks to the algorithmic and computational breakthrough of deep neural networks (DNNs), NLG and NLU performances have experienced significant improvements, including in architectural design, generation quality, and inference speed (Bahdanau et al., 2014; Cho et al., 2014; Rush et al., 2015; Xu et al., 2015; Dong and Lapata, 2016; Ling et al., 2016; Nallapati et al., 2016; Jia and Liang, 2017; Vaswani et al., 2017). Despite the success of DNNs, we believe that unleashing the full potential for NLU and NLG is yet to be seen. In this thesis, we improve DNNs for a range of NLU and NLG tasks by exploiting the "context" in which the text appears. We consider a broad notion of context consisting of (1) short-range context, (2) long-range context, and (3) meta context, in which the abstractness is gradually increasing. For the short-range context, we investigate its impact on the characteristic of a source language in machine translation. We explore the history of conversations in a dialogue setting as the study of the long-range context. We define the high-level context, such as domain, style, genre,

etc. as meta context. Then we examine the significance of meta context in various text understanding and classification tasks.

Short-range Context for Machine Translation In machine translation, discovering the units of text used for generating the translation is very important. Following humans convention, we had been working on word-level translations (Koehn, 2009; Bahdanau et al., 2014; Cho et al., 2014; Jean et al., 2015). To reduce the memory footprint and the computational cost, subword segmentation was devised and has become an integral component of NMT systems (Sennrich et al., 2015; Kudo, 2018). It aims to handle the representation of rare words by splitting them into subwords (*e.g.,* 'multilateralism'→ 'multilater'+'alism'). The German word 'carts' can be segmented differently: (1) 'carts', (2) 'cart' + 's', (3) 'car' + 'ts', *etc.* Some of these are not morphologically plausible. Existing subword algorithms consider each word separately and employ a greedy algorithm to conduct the segmentation, leading to implausible splits such as 'car' + 'ts'. If we have access to the corresponding English context: "The railway system was equipped with electrically powered carts", we can produce subwords like 'cart' + 's', respecting linguistic rules.

Long-range Context for Dialogue/Search A search engine can convert a search query to a machine-understandable structure for efficient and effective image search (Schuster et al., 2015; Anderson et al., 2016). Existing image search engines usually employ a single-turn search. However, users' demands tend to be dynamically changing. They usually start with the main object and gradually expand their queries with more details. Hence, users must type the complete queries multiple times when working on the single-turn search. Instead, a conversational search can track the changing intent of users and update the retrieval results accordingly. In addition, most semantic parsing approaches target an utterance-level parsing, while the data is derived from a conversation (Dong and Lapata, 2016; Ling et al., 2016; Jia and Liang, 2017). Table 1.1 shows that omitting the history of the conversation can lead to an improper conversion to the machine-executable representation. In summary, in dialogues, back-and-forth interaction always exists; thus, a long-range context-dependency is present in conversations by nature. We state that incorporating context-aware modeling is crucial to dialogue-driven NLU tasks.

Meta Context for Text Understanding and Classification For text classification problems, having access to more in-domain unlabeled data can significantly advance the

TABLE 1.1: An error made by the utterance-level parsing. It is clear that without the contextual information, the NLU model cannot infer the correct logical form.

dialog history
...
user: go to the next email and read it
user: read the next email
...
current utterance:
go next
logical form
reference: (next email)
utterance-level parsing: (unknown command)

task performance. However, such unlabeled data is usually missing. A simple solution is that one can retrieve unlabeled examples from a large and diverse open-domain dataset (Du et al., 2021). Nevertheless, due to the intellectual property and privacy concerns, it is not feasible to find relevant examples for professional domains, such as medical, legal, and financial domains. A domain is always linked to a specific style or genre, affecting the wording and phrasing (Hu et al., 2017; Prabhumoye et al., 2018). It has shown that one can suffer from performance degradation when using mismatched domains (Kouw and Loog, 2018; Wilson and Cook, 2020). Moreover, Table 1.2 shows that some problems conform to complex input schemes, *e.g.,* sentence pairs with certain relations. Such a characteristic also exacerbates the difficulty of retrieving in-domain examples. Therefore, we believe the retrieval solution is suboptimal. We coin the high-level information of downstream tasks, such as domain, genre, style, *etc.*, meta context. Finally, we argue that modeling the meta context can promote the generation of in-domain unlabeled data.

The aforementioned issues have motivated this research to investigate the effectiveness of using various context-aware approaches to boost the performance of context-dependent NLG and NLU.

1.1 Problem Statement

In summary, we have pinpointed the necessity and importance of contextual information when dealing with human text. We argue that the current models of natural language

TABLE 1.2: Examples with complex input schemes. The first sentence pair is entailed, where as the second pair is not entailed.

Sentence 1: A place of sorrow, after Pope John Paul II died, became a place of celebration, as Roman Catholic faithful gathered in downtown Chicago to mark the installation of new Pope Benedict XVI. Sentence 2: Pope Benedict XVI is the new leader of the Roman Catholic Church. Relation: entailment
Sentence 1: When did the third Digimon series begin? Sentence 2: Unlike the two seasons before it and most of the seasons that followed, Digimon Tamers takes a darker and more realistic approach to its story featuring Digimon who do not reincarnate after their deaths and more complex character development in the original Japanese. Relation: not entailment

understanding and generation fail to handle contextual information adequately. This thesis focuses on three major weaknesses related to context. These are below:

- **Problem 1:** Context-agnostic subword segmentation tends to produce suboptimal subwords which do not abide by linguistic rules.
- **Problem 2:** Sentence-level language understanding models suffer from performance degradation when dealing with context-dependent inputs.
- **Problem 3:** A particular task requires input sentences to satisfy specific attributes. However, this meta context has been under-studied in the literature.

1.2 Research Objectives

The main research objective is to leverage the context to alleviate the uncertainty and ambiguity within the text such that we can enhance the utility of deep learning models for natural language understanding and generation tasks.

In order to achieve this overarching objective, we pose three areas of investigation, divided across three parts of the thesis.

- **Objective 1:** Enhance subword segmentation in neural machine translation by considering the source and target context

- **Objective 2:** Advance context-dependent natural language understanding by incorporating historical context into simple context-independent models
- **Objective 3:** Model the task-specific context and generating in-domain unlabeled data such that the performance of the task of interest can be boosted

1.3 Contributions

Guided by the objectives, this study makes three essential contributions. We summarize them as follows:

- **Incorporating source and target context into subword segmentation**: We propose a novel mixed character-subword transformer to conduct context-dependent subword segmentation in NMT tasks. Since there are multiple plausible segmentations, we leverage a dynamic programming approach to address the intractable marginalization. We also employ dynamic programming to find more morphologically plausible segmentations. We experiment with the proposed method on five language pairs. Our empirical results suggest that this context-dependent approach can produce subwords that respect linguistic features, leading to significant improvements in translation quality over the context-agnostic counterparts. This work was published in ACL 2019.
- **Context-dependent natural language understanding in dialogue systems**: Understanding human-written text and generating structured text are crucial to dialogue systems. Since conversations involve back-and-forth communication, context can significantly drive conversation. We focus on two language understanding tasks, *scene graph modification* and *semantic parsing*. The former aims to update a scene graph via a textual command, while the latter converts a natural sentence into a machine-understandable representation. Since scene graph modification is a novel task, we create three datasets for it, then we use several existing models to perform this task. These models are not designed for this task, and all demonstrate poor performance. Thus, we cast the original scene graph and the command as the context. Then we devise a task-tailored model to encourage interactions within the context. Finally, our empirical study indicates that the proposed model

can fulfill the modification goal and outperform the baselines significantly across the crafted datasets. This work was published in EMNLP2020 as a findings paper.

Next, we study the effect of long-range context in dialogue-oriented semantic parsing. It has been shown that a context-independent parser has difficulty managing context-dependent parsing. We integrate a context-aware module into the context-independent parser to alleviate the parsing errors caused by the lack of context modeling. We additionally note that the standard parser can describe false arguments for the input utterance due to the memorization of frequent phrases. To resolve this issue, we propose utilizing a copy mechanism to copy the arguments to the appropriate target positions. Finally, our experiments indicate that the augmented parser can deliver superior results over the baseline model, corroborating the necessity of the context-aware and copy mechanisms. This work was published in ALTA2019.

- **Modeling task-specific context**: To the best of our knowledge, we first propose leveraging unlabeled synthetic data to advance self-training, knowledge distillation, and few-shot learning. We claim that context goes beyond surrounding words. A particular task tends to have a unique characteristic, which we name the task-specific context. Such context has a noticeable impact on the style and wording. In order to model such a meta context, we tailor a generic generative model trained on generic data to a task-specific model. To meet this requirement, we fine-tune or condition the generic model on the in-domain data. Then we can synthesize a lot of in-domain unlabeled data. Finally, we show that using the synthetic data can drastically boost the performance of the classification task of interest. This work has been accepted to *Transactions of the Association for Computational Linguistics 2022*.

1.4 Thesis Outline

In this section, we provide an outline of the rest of the thesis. The major contributions of this thesis are presented in Chapters 3, 4, 5 and 6, where the first three chapters address the uncertainty issues related to the context, as defined as the surrounding words and sentences. Chapter 6 focuses on the alleviation of uncertainty via the task-specific context. A summary of each chapter is as follows:

- **Chapter 2: Background** This chapter provides a thorough review of the foundations described in the thesis, covering the up-to-date achievements in natural language understanding and generation, and the development of pre-trained large language models.

- **Chapter 3: Subword Segmentation with Contextual Information** In this chapter, we first present our novel subword segmentation model for translation tasks, which considers the contexts of both source and target sentences when segmenting a word. Then we describe thorough experiments on five language pairs with ten translation directions. Our empirical study demonstrates the effectiveness of the proposed approach in comparison to word-bounded solutions in terms of the quality of segmentation and translation.

- **Chapter 4: Context-conditional Scene Graph Modification** We introduce a novel task in this chapter. This task aims to produce an updated scene graph by conditioning an existing graph and a modification command. We first craft three datasets for this task. Then we tailor the existing graph-generation models to examine the feasibility of updating a given scene graph from a descriptive sentence. Next, we propose a novel cross-attention mechanism to understand the context effectively and generate high-quality scene graphs. The proposed approach significantly outperforms the baseline across the three datasets.

- **Chapter 5: Context-dependent Semantic Parsing** This chapter investigates the effectiveness of incorporating long-range context into semantic parsing in a dialogue system. We identify that the baseline model suffers from two major flaws: (1) lack of context modeling and (2) argument misplacement. We integrate *context-aware* and *copy* mechanisms into the baseline model to overcome these drawbacks. Our comprehensive empirical study shows that the context-dependent method outperforms its context-agnostic counterpart under different neural architectures. The copy mechanism is orthogonal to the context-aware architecture, leading to additional boosts in performance.

- **Chapter 6: Advancing Text Classification by Modeling Task-specific Context** We argue that context is also associated with a particular task. The text style and wording are strongly affected by this high-level context. This chapter reports our theoretical and empirical study of the importance of modeling the

task-specific context. Our experiments highlight that we can guide the generic generative model to concentrate on task-specific context and synthesize high-quality unlabeled in-domain data. We advance self-training, knowledge distillation, and few-shot learning by incorporating the synthetic data into the learning objective.

- **Chapter 7: Conclusion** This chapter summarizes our findings and contributions in this thesis and also outlines potential directions for future work.

Chapter 2

Background

This chapter will provide a thorough review of the foundations described in the thesis, including the up-to-date achievements made in natural language understanding and generation and a detailed description of pre-trained large language models.

We initiate this chapter with a description of the advancement of NLG in Section 2.1. We first depict the basics of rule-based text generation systems. Then a statistical solution will be reviewed before introducing the state-of-the-art neural models. Since these neural approaches will lay a foundation for our models described in later chapters, we will spotlight every detail, including architecture, training and decoding procedures, and evaluation metrics.

Then, in Section 2.2, we shift the attention to NLU, which also has experienced an involution from rule-based systems to neural models. We will explain them one by one and portray the entire length of the neural solutions.

Section 2.3 will review the recent progress in pre-trained language models and stress two dominant directions: (1) unidirectional model and (2) bidirectional model. Moreover, we will discuss the concepts and applications of these models.

2.1 Natural Language Generation

Automating text generation is a prominent branch of NLP. The ultimate goal of NLG is not limited to text generation, but also aims to produce human-like text. As a

rudimentary trial, researchers developed a modularized system to meet this desideratum, where a list of rules codes each module (Reiter and Dale, 2000). A typical rule-based NLG system consists of three pieces:

- Document planner: determining the content and structure of a document;
- Microplanner: deciding words and syntactic structures for expressing the content and structure from the document planner; and
- Surface realizer: producing actual text by using the abstract representations from the microplanner.

Note that these components are not exclusive to rule-based NLG systems. Designing rules requires domain knowledge, so this approach cannot massively scale to various domains. As a remedy, a statistical method was first introduced for machine translation (Brown et al., 1993), and quickly applied to broad NLG tasks (Oh and Rudnicky, 2000; Belz and Kow, 2009; Langner and Black, 2009). This approach can model the probability of generating a target sentence given a source sentence. The probabilistic model is obtained automatically by learning the underlying distribution of the training data. Since the statistical solution is data-driven, one can smoothly adapt it to any domain whenever the training data is available. Besides, due to the efficacy, the statistical model had dominated NLG for decades until the advent of neural approaches.

Mathematically, the statistical model seeks to find the most probable target sequence $\hat{\boldsymbol{y}}$ given a source sentence like:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \tag{2.1}$$

After applying Bayes' rule, one can transform the conditional probability to:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y})P(\boldsymbol{x}|\boldsymbol{y})$$

where $P(\boldsymbol{x}|\boldsymbol{y})$ and $P(\boldsymbol{y})$ are a transition model and a language model respectively. The former is responsible for an alignment between source tokens (or phrases) and the most probable target tokens (or phrases) (Brown et al., 1993; Koehn, 2009), whereas the latter rewards a fluent and grammatically plausible sentence evaluated on n-gram probabilities (Vogel et al., 2000; Koehn, 2009).

Although the statistical approach fulfilled a triumph in academia and industry, it was criticized for two major flaws. First, the markovian property hinders the capability of capturing long-distance dependency. Second, statistical systems also comprise multiple sub-modules. Since each module is tuned separately, any intermediate brittleness can lead to an unexpected collapse.

Due to the breakthrough of deep learning in computer vision and natural language processing (Mikolov et al., 2010; Krizhevsky et al., 2012; Mikolov et al., 2013; Simonyan and Zisserman, 2014), neural approaches overtook their statistical counterparts at the NLG field since 2014. The neural NLG aims to conduct text generation via a neural network, which can be optimized end-to-end. This neural model consists of two core parts: *encoder* and *decoder*. As depicted in Figure 2.1, the encoder reads the input source words and projects them into a vector representation. On the other hand, the decoder leverages this vector to emit the target sentence word by word. As both source and target sides are comprised of a sequence of words, this model is coined as *sequence-to-sequence* (SEQ2SEQ) model.

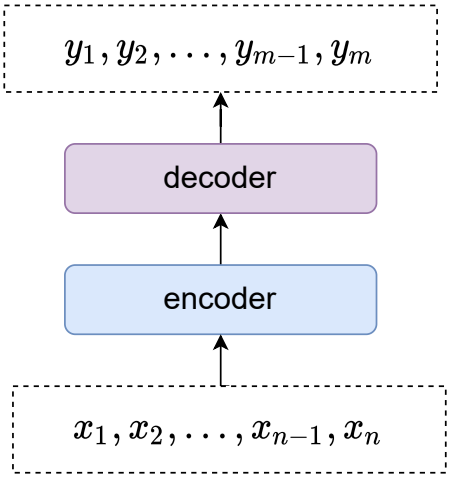


FIGURE 2.1: A general workflow of a sequence-to-sequence model.

In the emerging SEQ2SEQ model, a fixed representation of the source sentence is entrusted for a content representation. Afterward, the decoder takes this representation as input and sequentially generates the target words (Sutskever et al., 2014). However, compressing the holistic source sentence into a content vector tends to suffer from dramatic performance degradation, when the length of the source sentence grows (Cho et al., 2014). Thus, an attention-based SEQ2SEQ model was proposed to compute the context representation dynamically. These models employ recurrent neural networks

(RNNs) (described shortly) (Elman, 1990) to model the sequential information. Despite the success of RNN-based models in NLG tasks (Krizhevsky et al., 2012; Bahdanau et al., 2014; Cho et al., 2014; Rush et al., 2015; Nallapati et al., 2016; Paulus et al., 2017; See et al., 2017), *etc.*, the recurrence significantly impedes the parallelism for training and cause a computational bottleneck when handling longer sequences, such as document summarization and document translation. A novel architecture named Transformer (Vaswani et al., 2017) was proposed to parallelize the computation by utilizing a complete attention-based mechanism[1]. Due to the fast computation and state-of-the-art generation performance, Transformer has become the de facto architecture for NLG tasks.

The rest of this section will describe the RNN-based attentional SEQ2SEQ model (Section 2.1.1) and the Transformer-based encoder-decoder model (Section 2.1.2), which are the architectural backbone for our research. Finally, two decoding approaches and popular automatic evaluation metrics will be elaborated.

2.1.1 RNN-based Seq2seq Architecture

2.1.1.1 Recurrent Neural Networks

The cornerstone of RNN-based architecture is recurrent neural networks designed to model temporal signals. As shown in Figure 2.2, RNNs read a sequence of vector inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{n-1}, \boldsymbol{x}_n\}$ one by one. At each time step t, RNNs first update the *hidden representation* $\boldsymbol{h}_t$ from the previous state $\boldsymbol{h}_{t-1}$ and the input $\boldsymbol{x}_t$. Then, the output $\boldsymbol{y}_t$ can be generated from $\boldsymbol{h}_t$. The beauty of using RNNs is $\boldsymbol{h}_t$ can represent the partial sequence $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$, which manages to remove the markovian constraint. Formally, one can model a one-step vanilla RNN as:

$$\begin{aligned} \boldsymbol{h}_t &= \text{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) \\ &= f(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b}_h) \end{aligned}$$

where $\boldsymbol{W}_{hh}$, $\boldsymbol{W}_{xh}$ and $\boldsymbol{b}_h$ are learnable parameters, which are shared across timesteps, f is a non-linear activation function (*e.g.,* sigmoid). $\boldsymbol{h}_0$ is often initialize with zeros. The

[1] Although a convolution-based model (Gehring et al., 2017) can achieve the same purpose, we do not discuss it since we have not used it in our research.

discrete symbol y_t is emitted via:

$$y_t \sim \text{softmax}(\boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

where $\boldsymbol{W}_{hy}$ and $\boldsymbol{b}_y$ are shared and learnable parameters as well. $\boldsymbol{h}_t$ is first projected into a score vector through a linear transformation. Then the softmax function ($\text{softmax}(\boldsymbol{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K}\exp(z_j)}$) converts the score vector to a probability vector.

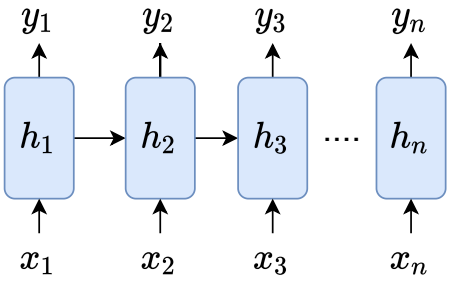


FIGURE 2.2: An overview of recurrent neural networks.

Theoretically, the vanilla RNNs can handle long-range dependency. Practically, due to the gradient-based back-propagation, the vanilla RNNs experience two notorious issues: *exploding* and *vanishing* gradients (Bengio et al., 1994). These issues are ascribed to the multiplicative nature of gradient updates. Particularly, when we back-propagate the gradients through time, the accumulative gradients will become either exponentially large (exploding gradient) or exponentially small (vanishing gradient). Consequently, the vanilla RNNs can fall short of capturing the long-range dependency.

To mitigate the aforementioned issues, a special variant of RNN, called long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), was devised as a replacement for the vanilla RNNs. According to Figure 2.3, the gist of LSTM is to leverage three gating functions to control the information flows of the input signals $\boldsymbol{x}_t$, previous hidden states $\boldsymbol{h}_{t-1}$ and the current hidden states $\boldsymbol{h}_t$. The gates can be formulated like:

$$\begin{aligned}
\boldsymbol{g}_t^i &= \sigma(\boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{b}_i) \\
\boldsymbol{g}_t^f &= \sigma(\boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{b}_f) \\
\boldsymbol{g}_t^o &= \sigma(\boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{b}_o)
\end{aligned}$$

where σ is a sigmoid function, and $\{\boldsymbol{W}, \boldsymbol{b}\}$ are shared across timesteps. In addition to

the hidden state $\boldsymbol{h}_t$, LSTM introduces a cell state $\boldsymbol{c}_t$ to store or remove the temporal information as follows:

$$\begin{aligned} \tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{b}_c) \\ \boldsymbol{c} &= \boldsymbol{g}_t^f \odot \boldsymbol{c}_{t_1} + \boldsymbol{g}_t^i \odot \tilde{\boldsymbol{c}}_t \\ \boldsymbol{h}_t &= \boldsymbol{g}_t^o \odot \boldsymbol{c} \end{aligned}$$

where $\odot$ denotes element-wise multiplication. With the aid of LSTM, one can train robust and stable RNNs to model long-range dependency. Because of the superior performance (Bahdanau et al., 2014), LSTM has become the default setting for RNNs.

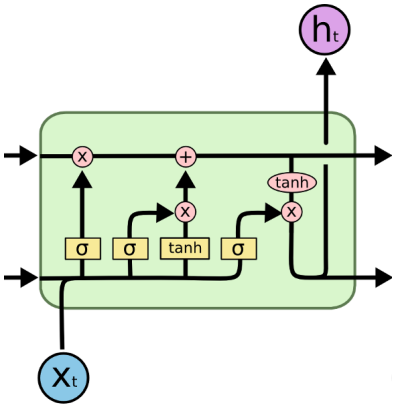


FIGURE 2.3: A schematic illustration of LSTM cell (Image source: Olah (2015)).

2.1.1.2 Neural Text Generation with RNNs

Bear the foundation in mind, we can delineate the RNN-based SEQ2SEQ model. Mathematically, a SEQ2SEQ architecture aims to model a conditional probability $P(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{x} = \{x1, x2, ..., x_{m-1}, x_m\}$ is the source sequence and $\boldsymbol{y} = \{y1, y2, ..., y_{n-1}, y_n\}$ is the target sequence. Note that x and y can be either a word or a subword (described in Section 2.1.5). To allow the model to serve sequences with arbitrary lengths, <s> and </s> are added at the beginning and end of each sentence to signal the *start* and *end* of a sentence respectively. The conditional probability $P(\boldsymbol{y}|\boldsymbol{x})$ can be decomposed as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{|\boldsymbol{y}|} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \tag{2.2}$$

where θ indicates the learnable parameters of the SEQ2SEQ model, y_t is the current target tokens and $\boldsymbol{y}_{<t}$ is the generated tokens so far. Figure 2.4 provides a schematic illustration

of RNN-based attentional SEQ2SEQ model. We will explain the salient components below.

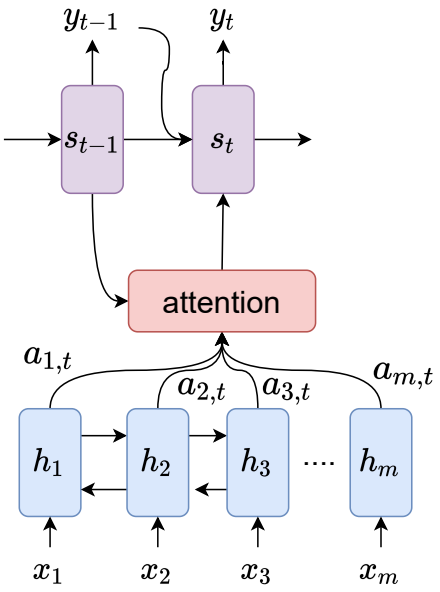


FIGURE 2.4: A schematic illustration of RNN-based attentional SEQ2SEQ model when generating y_t.

Word Embeddings Neural approaches can only manipulate a real-valued representation. Hence, one must convert the discrete tokens into vector representations via word embeddings to work with discrete symbols. Conventionally, we maintain two embedding tables, one for the source sequences $\boldsymbol{E}_S^{H\times|V_S|}$ and one for the target sequences $\boldsymbol{E}_T^{H\times|V_T|}$. H is a pre-defined value, and $|V_S|$ and $|V_T|$ are source and target vocabulary sizes respectively. $\boldsymbol{E}_S$ and $\boldsymbol{E}_T$ are learnable as well.

Encoder A bidirectional RNN is used to construct a "random access memory" of source hidden states as the context-aware representations of the source tokens. Specifically, a source sentence is traversed by one left-to-right RNN and one right-to-left RNN. Then, at each time step j, the forward and backward hidden states can be computed as:

$$\begin{aligned}\overrightarrow{\boldsymbol{h}}_j &= \overrightarrow{\mathrm{RNN}}(\boldsymbol{E}_S(x_j), \overrightarrow{\boldsymbol{h}}_{j-1}),\\ \overleftarrow{\boldsymbol{h}}_j &= \overleftarrow{\mathrm{RNN}}(\boldsymbol{E}_S(x_j), \overleftarrow{\boldsymbol{h}}_{j-1})\end{aligned}$$

where $\boldsymbol{E}_S(x_j)$ is the embedding of x_j. Afterwards, the contextual representation of the source token $\boldsymbol{x}_j$, referred to as $\boldsymbol{h}_j$, can be obtained by a concatenation of $\overrightarrow{\boldsymbol{h}}_j$ and $\overleftarrow{\boldsymbol{h}}_j$, *i.e.*, $\boldsymbol{h}_j = [\overrightarrow{\boldsymbol{h}}_j : \overleftarrow{\boldsymbol{h}}_j]$.

Attentional Decoder The decoder is a uni-directional RNN and sequentially generates the target tokens in a left-to-right manner. Moreover, it is paired with an attention mechanism to update its internal states at each time step t, which can be formulated as:

$$\boldsymbol{s}_t = \text{RNN}(\boldsymbol{E}_T(y_t), \boldsymbol{s}_{t-1}, \boldsymbol{c}_t) \tag{2.3}$$

where $\boldsymbol{E}_T(y_t)$ is the embedding of y_t, $\boldsymbol{s}_{t-1}$ is the previous hidden states and $\boldsymbol{c}_t$ denotes a context vector computed as a weighted sum of source memories:

$$\boldsymbol{c}_t = \sum_{j=1}^{m} \alpha_{jt} \boldsymbol{h}_j \tag{2.4}$$

The weight α_{jt} of each $\boldsymbol{h}_j$ can be calculated by:

$$\alpha_{jt} = \frac{\exp(\text{score}(\boldsymbol{h}_j, \boldsymbol{s}_{t-1}))}{\sum_{k=1}^{m} \exp(\text{score}(\boldsymbol{h}_k, \boldsymbol{s}_{t-1}))}$$

Here, score is referred to as a *content-based* function, and it can be one of the following three formats (Luong et al., 2015):

$$\text{score}(\boldsymbol{h}, \boldsymbol{s}) = \begin{cases} \boldsymbol{h}^T \boldsymbol{s} \\ \boldsymbol{h}^T \boldsymbol{W} \boldsymbol{s} \\ \boldsymbol{v}^T \tanh(\boldsymbol{W}_a \boldsymbol{h} + \boldsymbol{W}_b \boldsymbol{s}) \end{cases}$$

The attention mechanism is crucial to SEQ2SEQ model, enabling the decoder to access the source information at each step selectively. This selection aims to establish a soft alignment between source and target tokens. A higher α_{jt} suggests the source token x_j strongly correlates to the target token y_t.

Finally, one can generate the target token y_t by conditioning on all previously generated tokens $\boldsymbol{y}_{<t}$ and source tokens $\boldsymbol{x}$ via:

$$\begin{aligned} \boldsymbol{u}_t &= \tanh(\boldsymbol{s}_t + \boldsymbol{W}_{cu}\boldsymbol{c}_t + \boldsymbol{W}_{nu}\boldsymbol{E}_T(y_{t-1})) \\ P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) &= \text{softmax}(\boldsymbol{W}_y\boldsymbol{u}_t + \boldsymbol{b}_y) \\ y_t &\sim P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \end{aligned}$$

where $\{\boldsymbol{W}, \boldsymbol{b}\}$ are also learnable parameters of the SEQ2SEQ model. Now, we have formulated Equation 2.2 in a RNN-based SEQ2SEQ model.

2.1.2 Transformer-based Seq2seq Architecture

Since RNNs were born to model temporal dynamics, RNN-based SEQ2SEQ models have dominated NLG tasks for a while (Bahdanau et al., 2014; Rush et al., 2015; Xu et al., 2015; Li and Jurafsky, 2016b). However, the recurrence also brings two limitations to SEQ2SEQ models. The first limitation is its recurrent property. Particularly, one has to process each token sequentially, which causes a computational bottle for long sequences, especially for the encoder, where all tokens are given and accessible. Second, as shown in Section 2.1.1.1, the history information is stored in a recurrent unit, which is subject to an update. In other words, once the memory is removed (partially or thoroughly), one cannot resume it. Thus, RNNs, including LSTM, have difficulty maintaining long-distance relations. To remedy these shortcomings, Vaswani et al. (2017) proposed to replace recurrent units with self-attention and feedforward networks for the SEQ2SEQ model, which is termed as *Transformer*.

The model architecture is illustrated in Figure 2.5, and we will dissect the core components in the rest of this subsection.

Word Embeddings Like RNNs-based SEQ2SEQ models, Transformer projects the discrete tokens into real-valued representations via the source ($\boldsymbol{E}_S^w$) and the target embeddings ($\boldsymbol{E}_T^w$). Moreover, since Transformer dismisses the recurrence mechanism for the sake of parallelism, the ordering information is discarded, which treats a sentence as a bag of words. In order to compensate for the lack of ordering information, a positional

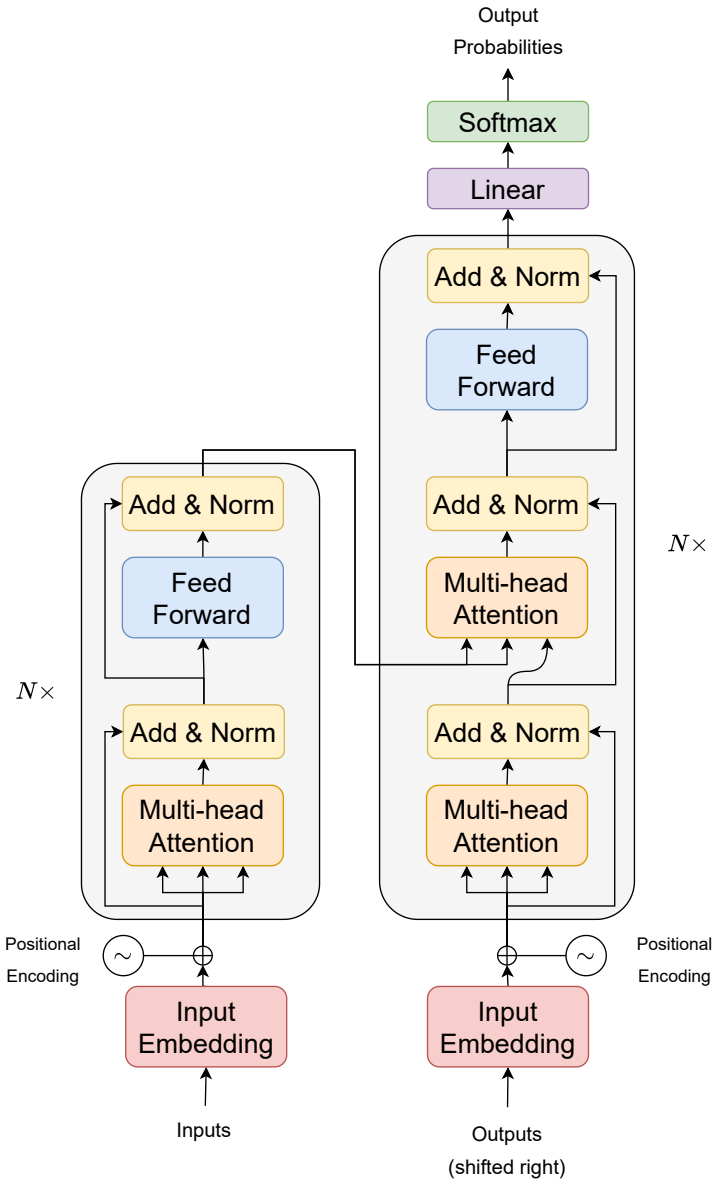


FIGURE 2.5: A schematic illustration of Transformer model.

encoding is incorporated into the Transformer. The source and target positional encoding denote $\boldsymbol{E}_S^p$ and $\boldsymbol{E}_T^p$ respectively. Given the ith source token x_i and the jth target token, their embeddings can be formulated as:

$$\boldsymbol{E}_S(x_i) = \boldsymbol{E}_S^w(x_i) + \boldsymbol{E}_S^p(i) \tag{2.5}$$

$$\boldsymbol{E}_T(y_j) = \boldsymbol{E}_T^w(y_j) + \boldsymbol{E}_T^p(j) \tag{2.6}$$

Regarding the positional encodings, Vaswani et al. (2017) proposed to use a static sinusoidal encoding. For a token at the position k, its positional encodings can be computed based on sine and cosine functions of different frequencies:

$$\begin{aligned} \boldsymbol{E}^p(k) &= \begin{bmatrix} \vdots \\ PE(k,2i) \\ PE(k,2i+1) \\ \vdots \end{bmatrix} \\ PE(k,2i) &= \sin\Big(\frac{k}{10000^{2i/H}}\Big) \\ PE(k,2i+1) &= \cos\Big(\frac{k}{10000^{2i/H}}\Big) \end{aligned}$$

where $i \in [0, \lfloor \frac{H-1}{2} \rfloor]$ indicates the ith dimension, H is the pre-defined embedding size, and the wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. One advantage of using the sinusoidal positional encodings is since the sinusoid has a frequency, it may manage to extrapolate to sequence lengths longer than the ones encountered during training. Alternatively, Gehring et al. (2017) leveraged learnable position encodings to inject the positional information. These encodings are randomly initialized, and can be learnt via the training process.

Encoder The encoder is a stack of N identical blocks, with each consisting of two sub-layers. The base of the first sub-layer is a multi-head self-attention module (denoted by MULTIHEAD_{self}), whereas the primary module of the second sub-layer is a feed-forward network (denoted by FFN). These two sub-layers will be described shortly. A residual connection (He et al., 2016) is inserted around the two sub-layers, followed by a layer normalization (Ba et al., 2016). Therefore, each encoder block can be formulated as:

$$\hat{\boldsymbol{X}}^n = \text{LayerNorm}(\boldsymbol{X}^{n-1} + \text{MULTIHEAD}_{self}(\boldsymbol{X}^{n-1})) \tag{2.7}$$

$$\boldsymbol{X}^n = \text{LayerNorm}(\hat{\boldsymbol{X}}^n + \text{FFN}(\hat{\boldsymbol{X}}^n)) \tag{2.8}$$

where $\boldsymbol{X}^0$ is the output of the encoder embedding layer and $n \in [1, N]$

Decoder The decoder also comprises a stack of N identical blocks. In addition to the two sub-layers in the encoder, each decoder block inserts another sub-layer between

MULTIHEAD$_{self}$ and FFN. The sub-layer conducts a multi-head inter-attention (denoted by MULTIHEAD$_{cross}$) between the output of the encoder and the current decoder block. Note that the self-attention sub-layers in decoder blocks employ a triangular mask to prevent positions from attending to subsequent positions, unseen at the inference stage. Hence, the computation of nth encoder block can be conducted via:

$$\hat{\boldsymbol{Y}}^n = \text{LayerNorm}(\boldsymbol{Y}^{n-1} + \text{MULTIHEAD}_{self}(\boldsymbol{Y}^{n-1})) \tag{2.9}$$

$$\widetilde{\boldsymbol{Y}}^n = \text{LayerNorm}(\hat{\boldsymbol{Y}}^n + \text{MULTIHEAD}_{cross}(\boldsymbol{X}^N, \hat{\boldsymbol{Y}}^n)) \tag{2.10}$$

$$\boldsymbol{Y}^n = \text{LayerNorm}(\widetilde{\boldsymbol{Y}}^n + \text{FFN}(\widetilde{\boldsymbol{Y}}^n)) \tag{2.11}$$

where $\boldsymbol{Y}^0$ is the output of the decoder embedding layer, $\boldsymbol{X}^N$ is the final output from the encoder.

Regarding the conditional probability of the next target token y_t, it can be calculated via:

$$P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_y \boldsymbol{Y}_t^N + \boldsymbol{b}_y)$$

where $\boldsymbol{Y}_t^N$ is the final output from the decoder at the timestep t.

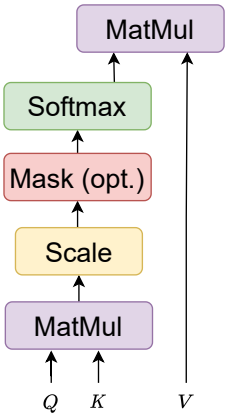


FIGURE 2.6: Scaled dot-production attention.

Multi-head Attention The core of Transformer is an attention mechanism. Briefly, the attention mechanism aims to represent a vector via a list of vectors. Under the scope of Transformer, an attention module takes queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$), and values ($\boldsymbol{V}$) as inputs. Then, the corresponding outputs can be produced, where each of them is a

weighted sum of the values. The weight assigned to each value is a scaled dot product between a query and a key as shown in Figure 2.6. One can formulate this operation as:

$$\text{ATTENTION}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{2.12}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are matrices, and $\sqrt{d_k}$ is the second dimension of $\boldsymbol{K}$. Since the dot-product attention is a matrix multiplication by nature, which has been highly optimized by the modern GPU architecture, it is much faster and more space-efficient than the additive attention.

In order to encourage the model to jointly attend to information from the distinctive representation subspaces at different positions, instead of using a single attention function, Transformer employs a novel multi-head attention, where the inputs are linearly projected into $\mathcal{K}$ subspaces. In each of these subspaces of the inputs, the attention operation, *i.e.,* Equation 2.12, will be executed in parallel. Eventually, one can concatenate the $\mathcal{K}$ intermediate outputs and project them into the final values as:

$$\begin{aligned} \text{MULTIHEAD}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, ..., \text{head}_{\mathcal{H}})\boldsymbol{W}^O \\ \text{where head}_\text{i} &= \text{ATTENTION}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V) \end{aligned}$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{H \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{H \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{H \times d_k}$ and $\boldsymbol{W}^O \in \mathbb{R}^{H \times H}$ are projection matrices. Note that $d_k = \frac{H}{\mathcal{K}}$.

As shown in Equation 2.8, 2.10 and 2.11, there exist three places of using multi-head attention within Transformer:

- **Multi-head Self-attention in Encoder** This self-attention works on the output of the previous layer in the encoder, denoted by $\text{MULTIHEAD}_{self}(\boldsymbol{X}^{n-1})$. Here, the keys, values and queries are sourced from the same place.

$$\text{MULTIHEAD}_{self}(\boldsymbol{X}^{n-1}) = \text{MULTIHEAD}(\boldsymbol{X}^{n-1}, \boldsymbol{X}^{n-1}, \boldsymbol{X}^{n-1})$$

- **Multi-head Self-attention in Decoder** Self-attention is applied to the output of the previous layer in the decoder, denoted by $\text{MULTIHEAD}_{self}(\boldsymbol{Y}^{n-1})$ as well. Since we can only access the generated tokens during the inference time, the self-attention forces each position to attend to its preceding positions and itself by

masking out the future tokens.

$$\textsc{MultiHead}_{self}(\boldsymbol{Y}^{n-1}) \quad = \quad \textsc{MultiHead}(\boldsymbol{Y}^{n-1}, \boldsymbol{Y}^{n-1}, \boldsymbol{Y}^{n-1})$$

- **Encoder-decoder Multi-head Attention** This attention mirrors the attention mechanism in the RNN-based SEQ2SEQ. Thus, the keys and values come from the output of the last layer of the encoder, whereas the output of the previous layer of the decoder is treated as the queries.

Feed-Forward Network (FFN) Both encoder and decoder layers append a fully connected feed-forward network (FFN) to the attentive sub-layer. This is formulated as two linear transformations with a ReLU activation in between:

$$\text{FFN}(\boldsymbol{X}) \quad = \quad \text{ReLU}(\boldsymbol{X}\boldsymbol{W}_{ff1} + \boldsymbol{b}_{ff1})\boldsymbol{W}_{ff2} + \boldsymbol{b}_{ff2}$$

where $\{\boldsymbol{W}, \boldsymbol{b}\}$ are parameters exclusive to each layer, and $\text{ReLU}(x) = \max(0, x)$

2.1.3 Training and Decoding

Training One can use backpropagation to update all parameters of a SEQ2SEQ model (Rumelhart et al., 1985; Lecun, 1988), where the *objective* function is a negative log-likelihood (conditional) over the training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i^{|\mathcal{D}|}$. The conditional log-likelihood is formulated as the sum of the log-probability of predicting a correct symbol, conditioning the prefix $\boldsymbol{y}_{<t}$ and source inputs $\boldsymbol{x}$. Thus, one can seek the optimum set of parameters $\boldsymbol{\theta}^*$ as follows:

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} -\log P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \sum_{t=1}^{|\boldsymbol{y}|} -\log P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})
\end{aligned}$$

In order to find $\boldsymbol{\theta}^*$, one usually employs the gradient descent (GD) algorithm, especially for deep learning models (Goodfellow et al., 2016). GD aims to update the parameters in the opposite direction of the gradient of the objective function *w.r.t.* to the parameters.

Hence, for each update i, one can update the parameters by:

$$\boldsymbol{\theta}_i \quad = \quad \boldsymbol{\theta}_i - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i) \tag{2.13}$$

where $J(\boldsymbol{\theta}) \triangleq \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} -\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ is the objective function, and η is the learning rate determining the size of the steps we take to reach a (local) minimum. Since vanilla gradient descent or batch gradient descent (BGD) (*c.f.,* Equation 2.13) has to compute the gradient of the entire training set before making *one* update, it is prolonged and intractable for the large dataset. On the contrary, stochastic gradient descent (SGD) allows a parameter update for each training example. However, since SGD relies on the gradient of a single example, it can introduce a high variance and cause a severe fluctuation (Ruder, 2016). As a middle ground, the mini-batch gradient descent manages to update the parameters based on the gradient of a small batch of training examples. This variant can accelerate the training procedure and reduce the variance, leading to a stable convergence; thus it has been used as the default setting for numerous deep learning models and also called SGD because of the stochasticity (Ruder, 2016).

Notwithstanding its popularity, the mini-batch gradient descent faces some challenges. First, the choice of the learning rate is burdensome. Small learning rates can cause painfully slow convergence, whereas a large learning rate can cause an overshoot on the minimum or even divergence. Second, SGD uses a fixed learning rate throughout the training process. However, due to the non-convex surfaces, the fixed learning rate cannot handle the complicated dynamics of DNNs. For instance, we need a large η to find the vicinity of local optima quickly at the early stage. Nevertheless, a smaller η should be employed to avoid overshooting the local optima.

To address the these challenges, many adaptive approaches have been proposed, such as Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Adam (Kingma and Ba, 2014), to name a few. Adam quickly gained credence among these approaches due to its effectiveness in training DNNs.

Decoding Once an NLG system is well-trained, we can utilize it to process unseen source sentences. We summarize three popular decoding strategies below.

According to Equation 2.1, the NLG targets the maximum conditional probability. A simple solution is to use *greedy search*. Similar to other greedy algorithms, one can

simply select the token with the highest probability from $P_{\boldsymbol{\theta}}(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x})$ at each time step, and add it into the final sequence. Obviously greedy search is not guaranteed to find the generation with the highest probability. This failure can be seen from Figure 2.7, where the best sequence is "<s> a b </s> ", while greedy search gives us a sub-optimal solution: "<s> b b </s> ".

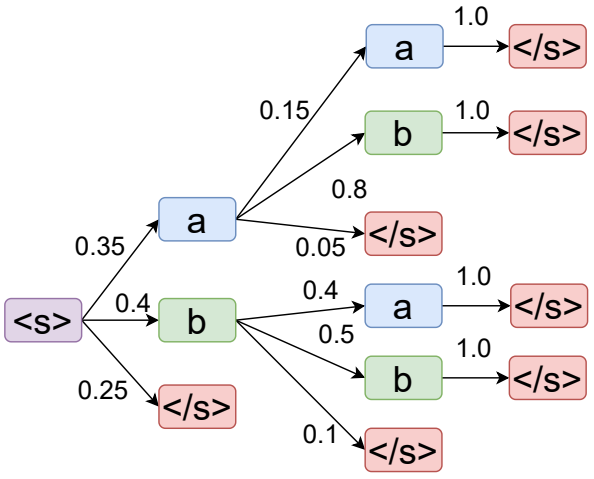


FIGURE 2.7: An example where greedy search fails (Neubig, 2017).

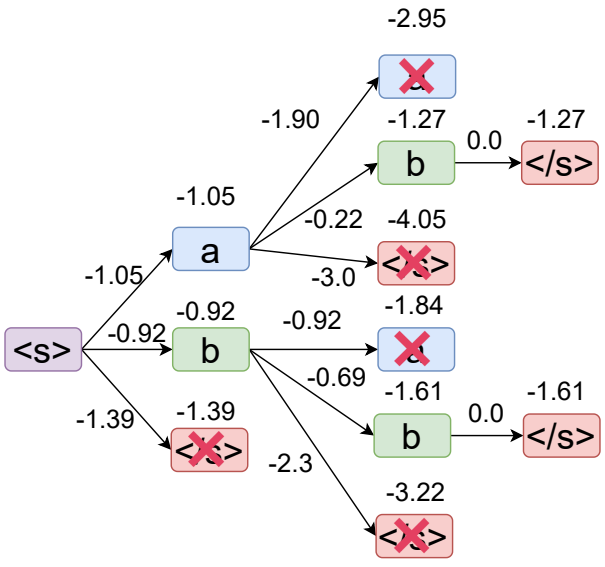


FIGURE 2.8: An example of beam search with b=3 (Neubig, 2017).

Of course, we can use exhaustive search to find the best solution, but the magnitude of the vocabulary of the decoder makes this search intractable. Thus, we resort to *beam search*, an optimized best-first search. Unlike greedy search, beam search considers b best hypotheses at each time step, where b is the width of the beam. As shown in

Figure 2.8, we expand $b = 3$ candidates, with each candidate selecting the top b tokens out of size of $|\mathcal{V}|$, resulting in $b * b$ active hypotheses. The score of a hypothesis at time step t is formulated as:

$$s(\boldsymbol{y}_t^{i,j}, \boldsymbol{x}) = s(\boldsymbol{y}_{t-1}^{i}, \boldsymbol{x}) + \log P_{\boldsymbol{\theta}}(y_t^j \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \tag{2.14}$$

where $i, j \in [1, b]$. Then we prune them into the top b candidates according the scores.

$$s(\boldsymbol{y}_t^{i}, \boldsymbol{x}) = \max_j s(\boldsymbol{y}_t^{i,j}, \boldsymbol{x}) \quad \text{for each } i \in [1, b]$$

We denote the b candidates as $Y_t = [\boldsymbol{y}_t^1, ..., \boldsymbol{y}_t^b]$. It is worth noting that we usually use the log-likelihood to score the candidates, because it is numerically stable on computers. If a special token $\langle \text{eos} \rangle$ is generated, we will add this candidate to a candidate pool, meanwhile removing it from the search tree. The expansion and pruning are repeated until we collect enough candidates in the pool. Finally, we return the candidate with the highest log-likelihood as the best generation.

The naive beam search will favor a short sentence because the log-likelihood will decrease every time a new word is added to the candidate. Even though they might be better than their shorter competitors, long sentences are at a disadvantage when the re-scoring criterion is based on the total log-likelihood. Consequently, as the increase of beam size, beam search is more likely to have a significant *length bias* towards the shorter candidates and demonstrate performance degradation (Koehn and Knowles, 2017).

There are several different methods fixing this length bias problem. A simple but effective approach is to normalize the log likelihood by the length of the candidate:

$$s(\boldsymbol{y}, \boldsymbol{x}) = \log P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) / |\boldsymbol{y}| \tag{2.15}$$

Then, we can use this re-scoring function $s(\boldsymbol{y}, \boldsymbol{x})$ to select the target sentence among a pool of candidates.

In addition to the best-first solutions, the community has studied sampling generation, where the next token y_t is sampled from the conditional probability $P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{y}_{<t}, \boldsymbol{x})$. Such randomness can introduce diversity into the generated sentences, compared to the best-first counterparts. One use case is dialogue response, which requires the system

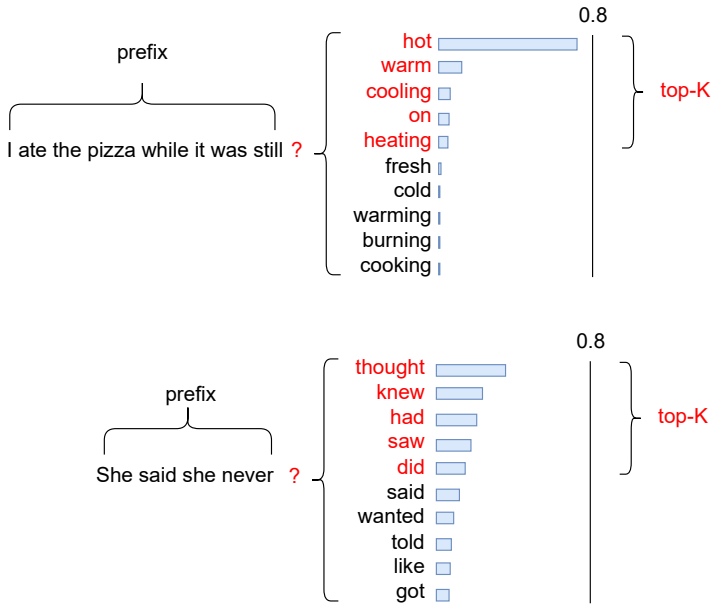


FIGURE 2.9: Two examples of using top-K sampling when generating the next token. Candidate tokens are in red.

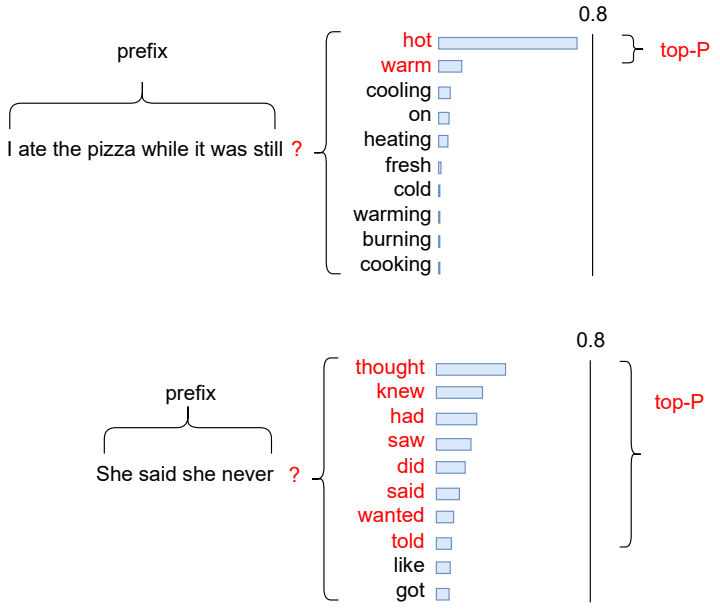


FIGURE 2.10: Two examples of using top-P sampling when generating the next token. Candidate tokens are in red.

to produce various replies to a particular user input to prevent monotony (Li et al., 2016a). Furthermore, for any source sentence, usually there exist multiple valid targets (Vijayakumar et al., 2016; He et al., 2018; Shen et al., 2019). However, a pure sampling approach can cause incoherence and degeneration (Radford et al., 2019). To

overcome this issue, two heuristic approaches have been introduced. The first one is top-K sampling, which samples the next word from the top-K most probable choices (Fan et al., 2018; Radford et al., 2019). Orthogonal to the top-K approach, one can sample a token from the top-P portion of the probability mass. According to Figure 2.9 and 2.10, the number of candidate tokens is fixed for the top-K sampling, whereas the top-P technique demonstrates certain elasticity when selecting words. Empirical studies indicate that pruning-based sampling strategies can emit high-quality and diverse texts (Radford et al., 2015; Holtzman et al., 2019).

2.1.4 Evaluation

After generating target sentences from given inputs, we should assess our NLG performance. Hiring professional evaluators to examine the outputs is the best choice, as they can give a comprehensive judgment. However, this labor-intensive service is not feasible. Since the source inputs are paired with reference generation, we can compare a system generation against the reference one. The closer they are, the better system generation is (Papineni et al., 2002).

BLEU[2] was designed by Papineni et al. (2002) for the evaluation of MT systems. The cornerstone of this metric is to compute the ratio of common n-grams between reference translations and system ones. For example, if we have:

Example 1

Candidate: I am a student

Reference: I am a CS professor

In this example, the unigram BLEU is 3/4, while the bigram BLEU is 2/3. However, this precision-based property can lead to irrationally high scores, if we observe the following example:

Example 2

Candidate: the the the the the the the

Reference: the cat is on the mat

[2]BiLingual Evaluation Understudy

Now the unigram BLEU is 7/7, which is egregious. Therefore, BLEU scores should be calculated as:

$$p_n = \frac{\sum_{\text{n-gram} \in \{\text{system translation} \cap \text{reference translation}\}} Count_{clip}(\text{n-gram})}{\sum_{\text{n-gram}' \in \{\text{system translation}\}} Count(\text{n-gram}')}$$

where $Count_{clip} = \min(count, Max\_Ref\_Count)$. In other words, if the number of a n-gram exceeds its occurrence in a reference translation, it will be clipped to the occurrence. Hence, in example 2, the correct unigram BLEU is 2/7.

More common unigrams between a system translation and reference one indicate a higher *adequacy*, while longer n-gram matches account for *fluency*. A good translation should take both adequacy and fluency into consideration. Furthermore, the length between system translation and reference one should also be matched. Since the clipped n-gram precision measure already penalizes a longer system translation, a multiplicative *brevity penalty* factor is introduced to punish a shorter system translation. Finally, a BLEU score between a system translation and the reference one is computed as:

$$\text{BLEU} = \text{BP} \cdot \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big)$$

where w_n is the weight for a clipped n-gram precision and usually is uniform. BP is the brevity penalty factor:

$$\text{BP} = \begin{cases} 1 & c > r \\ e^{1-r/c} & otherwise \end{cases}$$

where c and r are the lengths of the system translation and the reference one respectively.

Although various evaluation metrics have been proposed in the literature, we only mention the BLEU, as we only use it for this thesis.

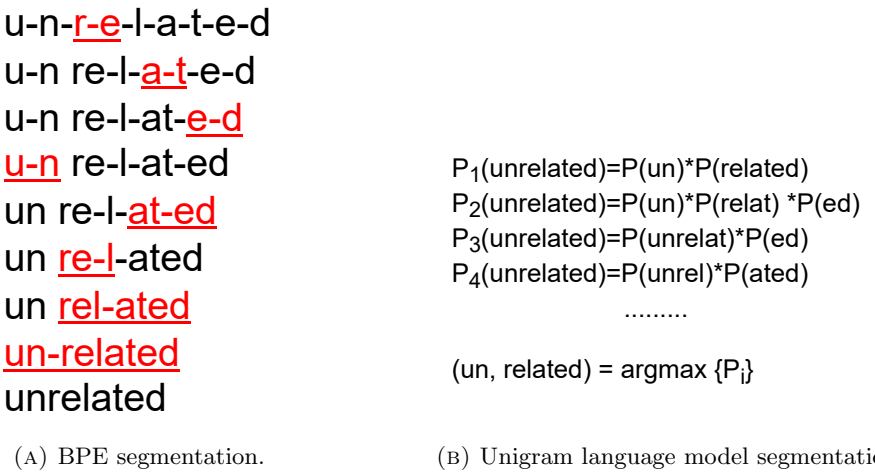
2.1.5 Subword Segmentation

Neural networks have revolutionized machine translation (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014). Early neural machine translation (NMT) systems used words as the atomic element of sentences. They used vocabularies with tens of thousands words, resulting in prohibitive training and inference complexity. While learning can be sped up using sampling techniques (Jean et al., 2015), word based NMT models

have a difficult time handling rare words, especially in morphologically rich languages such as Romanian, Estonian, and Finnish. The size of the word vocabulary should increase dramatically to capture the compositionality of morphemes in such languages.

More recently, many NMT models have been developed based on characters and a combination of characters and words (Ling et al., 2015; Luong and Manning, 2016; Lee et al., 2017; Vylomova et al., 2017; Cherry et al., 2018). Fully character based models (Lee et al., 2017; Cherry et al., 2018) demonstrate a significant improvement over word based models on morphologically rich languages. Nevertheless, owing to the lack of morphological information, deeper models are often required to obtain a good translation quality. Moreover, elongated sequences brought by a character representation drastically increases the inference latency.

In order to maintain a good balance between the vocabulary size and decoding speed, subword units are introduced in NMT (Sennrich et al., 2015; Wu et al., 2016). These segmentation approaches are data-driven and unsupervised. Therefore, with a negligible pre-processing overhead, subword models can be applied to any NLP task (Vaswani et al., 2017; Devlin et al., 2019). Meanwhile, since subword vocabularies are generated based on word frequencies, only the rare words are split into subword units and common words remain intact. For instance, "unrelated" is segmented as "un" and "related".



(A) BPE segmentation. (B) Unigram language model segmentation.

FIGURE 2.11: Two subword segmentation algorithms on 'unrelated". Hyphens indicate possible merges according the merge table.

Byte Pair Encoding (BPE) (Sennrich et al., 2015) starts from individual characters then utilizes a compression algorithm to merge the most frequent units until reaching the

predefined limit, while wordpiece (Schuster and Nakajima, 2012) and unigram language model (Kudo, 2018) apply language model to characters to form their subwords. Figure 2.11 illustrates the segmentation procedures of BPE and unigram language model. These approaches can dramatically reduce vocabulary size to moderate volumes such as 32K or 16K. Meanwhile, there are nearly no OOVs because of subword segmentation.

Since the unigram model can segment a word differently via the probabilistic model, it can regularize segmentation errors introduced by a deterministic segmentation, such as BPE. Provilkov et al. (2019) proposed applying dropout to BPE segmentation. Figure 2.12 depicts that at each merge iteration, one can randomly disallow a small fraction of mergeable pairs. Now, BPE can produce multiple different segmentations to fulfill the regularization effect.

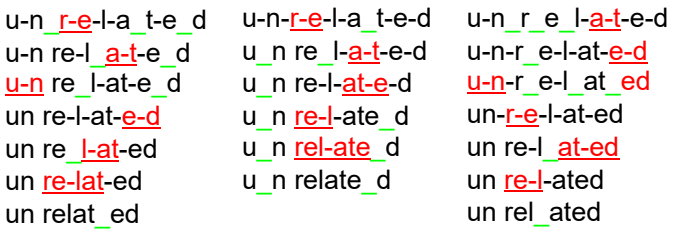


FIGURE 2.12: Segmentation process of the word 'unrelated' using BPE dropout. Hyphens indicate possible merges according the merge table. Merges performed at each iteration are shown in red, dropped underline in green.

2.2 Natural Language Understanding

Due to the breakthrough of deep learning, numerous and various tasks within the field of NLP have made impressive achievements (Vaswani et al., 2017; Devlin et al., 2018; Edunov et al., 2018). However, most of these achievements are assessed by automatic metrics, which are relatively superficial and brittle, and can be easily tricked (Jia and Liang, 2017; Paulus et al., 2017; Läubli et al., 2018). Hence, understanding the underlying meaning of natural language sentences is crucial when deploying NLP systems to serve human users. NLU aims to translate an unstructured human-written text into a structured machine-understandable representation, which enables the recognition of the intents of user inputs.

The development of NLU has experienced three stages. First, primitive NLU systems employed rule-based techniques to perform the conversion from a sentence to a machine-understandable form (Woods, 1973; Johnson, 1984). Thus, these systems are inevitable to be brittle and laborious. To reduce the engagement of human activities, researchers shifted the attention to statistical techniques. A seminal work from Zelle and Mooney (1996) proposed to automatically transform a sentence to a database query via statistical learning. This line of work utilized a deterministic shift reduce parser to produce the structured form from the natural sentence. One advantage of this approach is that as the generated query must be correctly executed, the evaluation is straightforward. The data-driven method can be applied to other datasets and even multiple languages without the demand for domain knowledge. Mathematically, one can view the conversion task as a conditional probability problem, *i.e.,* $P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{x}$ is the human sentence, and $\boldsymbol{y}$ is the corresponding structural form. Inspired by this, Wong and Mooney (2006) performed semantic parsing, a type of NLU problem, via the statistical NLG technique. With the advancement of the deep learning models, $P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ has been gradually modeled by the SEQ2SEQ framework because of its outstanding performance.

The rest of this section will describe two popular directions in NLU: (1) semantic parsing and (2) scene graph parsing.

2.2.1 Semantic Parsing

Semantic parsing is a task that aims to convert human utterances to machine-executable representations, such as logical forms, programming snippets, SQL queries, *etc.* (see Table 2.1). Due to its significance in NLU, semantic parsing has gained dramatic attention. Early data-driven approaches utilized high-quality lexicons and hand-crafted feature engineering to parse the human sentences. Although these methods achieved impressive performance over some datasets, using hand-crafted features impedes the further development of semantic parsing and knowledge transferring across different tasks.

Motivated by the triumph of SEQ2SEQ models in different generation tasks (Bahdanau et al., 2014; Rush et al., 2015; Vinyals et al., 2015; Xu et al., 2015), researchers have investigated applying the end-to-end framework to semantic parsing (Dong and Lapata, 2016; Ling et al., 2016; Jia and Liang, 2017). Since this framework lessens the need for high-quality lexicons, fabricated templates, and hand-engineered features, it has gained

TABLE 2.1: Examples of semantic parsing.

Example 1:
English: which state has the most rivers running through it? *logical form*: (argmax $0 (state:t $0) (count $1 (and (river:t $1) (loc:t $1 $0))))
Example 2:
English: if length of bits is lesser than integer 3 or second element of bits is not equal to string 'as' *code snippets*: if len(bits) <3 or bits[1] != 'as':
Example 3:
English: What record company did conductor Mikhail Snitko record for after 1996? *code snippets*: : SELECT Record Company WHERE (Year of Recording >1996) AND (Conductor = Mikhail Snitko)

credence among the community. We will present the SEQ2SEQ model and its variant for semantic parsing below.

Seq2seq Architecture As both texts $\boldsymbol{x}$ and machine-interpretable meaning representations $\boldsymbol{y}$ consist of a sequence of tokens, one can model any semantic parsing problem via a standard SEQ2SEQ model whenever the annotated data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_i^{|\mathcal{D}|}$ is available. The training and decoding suites are identical to NLG.

Seq2Tree Architecture As SEQ2SEQ models were initially proposed to address the text generation tasks, they excel at producing unstructured outputs without a rigid hierarchy. However, as demonstrated in Table 2.1, machine-executable representations are hierarchically structured. Memorizing various pieces of auxiliary information (*e.g.,* parenthesis pairs) imposes a burden on the long-range dependency of the SEQ2SEQ model. Therefore, Dong and Lapata (2016) proposed a sequence-to-tree (SEQ2TREE) model which is specific for the compositional nature of meaning representations.

According to Figure 2.13, given a logical form "*lambda $0 e (and (>(departure time $0) 1600:ti) (from $0 dallas:ci))*", the tokens inside pairs of brackets are replaced with a special token <n>, denoted as a non-terminal token. Then one can organize the transformed logical form in a tree structure, where each level is an expansion of <n>. At each level, a special token </s> signifies the termination of this level. The SEQ2TREE model leverages a tree-like decoder to model the hierarchical representation.

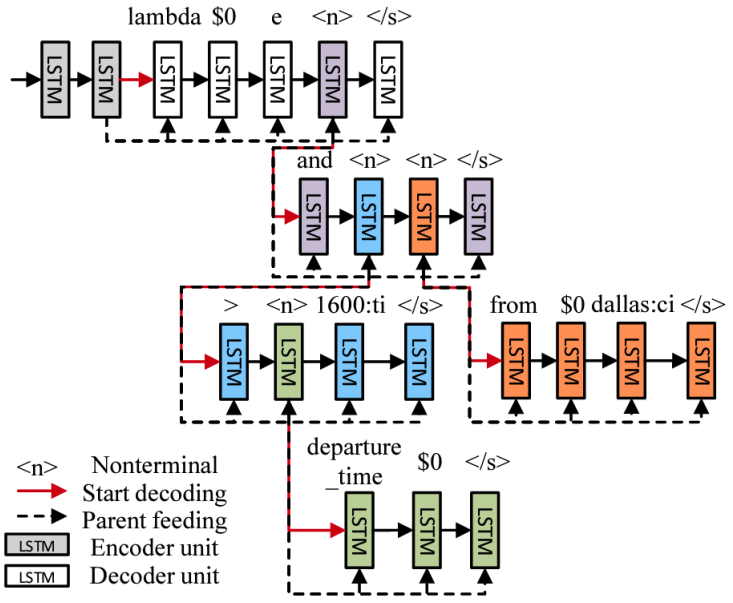


FIGURE 2.13: The schematic illustration of the hierarchical decoder of the SEQ2TREE model (Dong and Lapata, 2016).

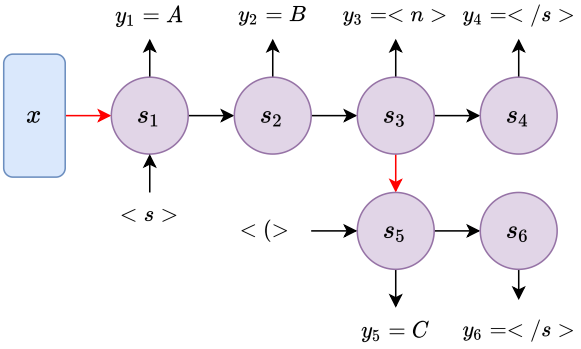


FIGURE 2.14: A SEQ2TREE decoding example for the logical form "A B (C)" (Dong and Lapata, 2016).

During the inference time, a logical form can be generated in a top-down manner as well (*c.f.,* Figure 2.14). Specifically, SEQ2TREE produces a logical form sequentially, until encountering </s>. Whenever <n> is produced, a sub-tree generation will be triggered after the previous sequence generation terminates. This recursion will suspend when no <n> appears. Special tokens <s> and <(> indicate the beginning of a sequence and a non-terminal sequence respectively, while </s> represents the end of any type of sequence. The sequence is decoded during the sub-tree generation by conditioning the non-terminal hidden vector. In addition, to encourage connection between a parental node and its sub-tree, the input of the sub-tree generation will be concatenated with the non-terminal hidden vector before being fed into the decoder.

Formally, given $\boldsymbol{x}$, one can express the probability of $\boldsymbol{y}$ = "$A\,B\,(C)$" as the product of two sequences (*c.f.*, Figure 2.14):

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{y}_{1:4}|\boldsymbol{x})p(\boldsymbol{y}_{5:6}|\boldsymbol{y}_{\leq 3}, \boldsymbol{x}) \tag{2.16}$$

Since the generated structural outputs must be executable, strict string matching is utilized to evaluate the model performance.

2.2.2 Scene Graph Parsing

The computer vision community originally introduced *scene graph* representation to represent the rich, structured knowledge within an image via graphs. As shown in Figure 2.15, a node in a scene graph can represent either an object, an attribute for an object, or a relationship between two objects. Because of the effectiveness in knowledge encoding and expression, this structured representation has been successfully used for various visual tasks (Chang et al., 2014; Johnson et al., 2015; Teney et al., 2017). However, the costs of annotating scene graphs are prohibitive. A list of works (Dai et al., 2017; Xu et al., 2017) proposes automatically generating scene graphs from images. Given the success of using scene graphs in image retrieval, there has been an attempt to parse an image description into a scene graph to facilitate precise image matching (Schuster et al., 2015; Anderson et al., 2016). This thesis targets natural language understanding, so this section shifts attention to the latter.

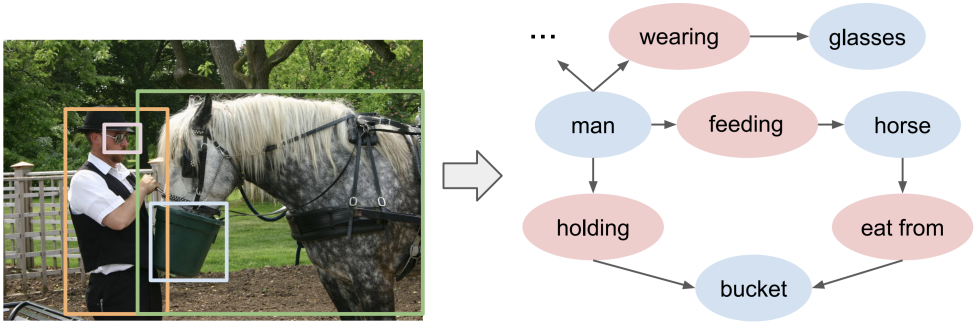


FIGURE 2.15: An example of scene graph representation for an image (Johnson et al., 2015).

The task of transforming a descriptive sentence into a scene graph is defined as follows. Given a set of object classes C, a set of relation types R, a set of attribute types A, and

a sentence s, we want to parse s to a scene graph:

$$G =< O, E >$$

where $O = \{o_1, ..., o_n\} \subseteq C$ is a set of objects mentioned in s and $E \subseteq O \times R \times O$ is a set of relations between two objects in the graph. Each o_i is a tuple (c_i, A_i), where $c_i \in C$ is the class of o_i, and $A_i \subseteq A$ are the associated attributes. For example, given a sentence $s =$"A young boy in a black shirt", we can extract two objects $o_1 = (\text{boy}, \{\text{young}\})$, and $o_2 = (\text{shirt}, \{\text{black}\})$, and a relation $e_1 = (o_1, \text{in}, o_2)$. A visualization of this conversion is depicted in Figure 2.16.

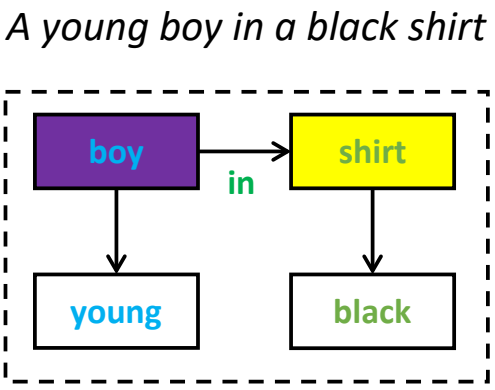


FIGURE 2.16: An example of parsing a sentence into a scene graph.

Due to the lack of annotated data, most works employ rule-based avenues to address the parsing problem (Schuster et al., 2015; Anderson et al., 2016). Until very recently, Wang et al. (2018) creatively developed annotated data by aligning scene graphs from Visual Genome (Krishna et al., 2017) and MS COCO (Lin et al., 2014) to text descriptions. Building upon this labeled data, Andrews et al. (2019) devices an end-to-end parsing system surpassing the rule-based models by a large margin.

The gist of this end-to-end model is an *Attention Graph* mechanism, which is designed to predict the node type of a token and the relation between two tokens. Given a token, there are six node types:

- **SUBJ** The node label for an object in O; It is linked to a virtual node, denoted as *root*;

- **PRED** The node label for a relationship R; The arc of this node points to a SUBJ node;
- **OBJT** The node label for an object in O. It is connected to the relevant PRED node;
- **ATTR** The node label for an attribute in A. The arc of this node points to either SUBJ or OBJT node;
- **same** This label is used for phrasal nodes. For instance, "in front of" is a phrase indicating a relation between two object nodes. Thus, *same* means that the token shares the same node type with the preceding node;
- **none** There is no node type attached to this token.

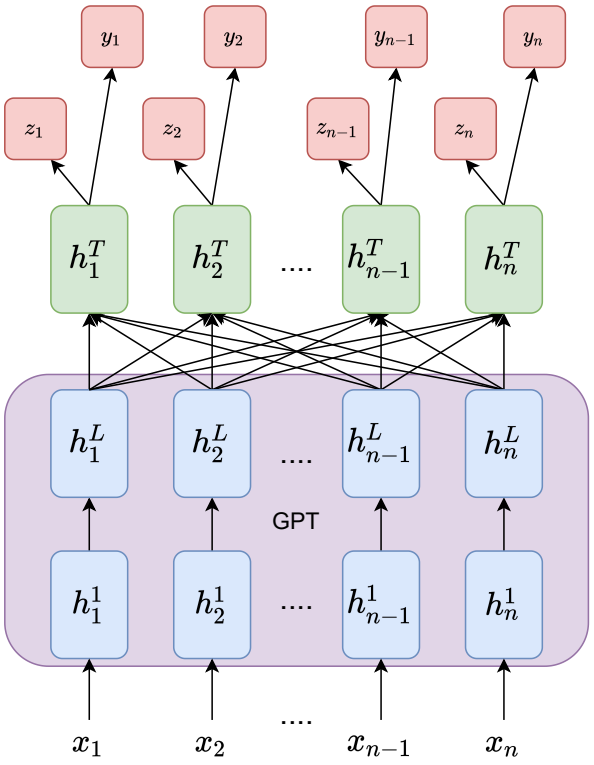


FIGURE 2.17: An example of predicting the node and relations from a sentence input. z_i is the prediction of the parental node, where as y_i is the predicted node type.

Bear the definitions in mind. We can formulate the parsing model as a conditional probability:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) = P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) * P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$$

where $\boldsymbol{x}$ is a sequence of input tokens, $\boldsymbol{y}$ is the corresponding node types, and $\boldsymbol{z}$ indicates the assigned parental tokens. As shown in Figure 2.17, The sentence $\boldsymbol{x} = \{x_1, ..., x_n\}$ can be fed into GPT model (will be discussed in Section 2.3.1) to obtain the unidirectional contextualized representation $\{\boldsymbol{h}_1^L, ..., \boldsymbol{h}_n^L\}$. Then, a self-attention module is employed to produce the bidirectional contextualized representation $\{\boldsymbol{h}_1^T, ..., \boldsymbol{h}_n^T\}$. One can emit the probabilities of node types and parental tokens from the outputs of the self-attention layer. Finally, the model can be trained via the standard cross-entropy losses on $\boldsymbol{y}$ and $\boldsymbol{z}$. We can predict the node type and the parental node for each token at the inference stage.

2.3 Pre-trained Language Models

Pre-training a model on large-scaled data (*e.g.,* ImageNet (Russakovsky et al., 2015)) has been considered as the groundbreaking milestone for this wave of advancement in computer vision (CV). Nowadays, using the pre-trained model as a backbone has become the default setting for downstream CV tasks. Inspired by such achievements, the NLP community has been trying to mirror the triumph. First, it has been shown that the convergence speed for NLP models can be accelerated, if we initialize the word embeddings with pre-trained ones, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). As highlighted in Section 1, languages are context-dependent by nature. However, the pre-trained word embeddings do not consider this salient information. Moreover, except the word embedding layer, other components are still randomly initialized. To encourage the modeling of the contextual message, there are some attempts to use a pre-trained RNN model as the backbone for various text classification tasks (Dai and Le, 2015; Howard and Ruder, 2018; Peters et al., 2018). These models reveal the potential of pre-trained models and lay a foundation for the prosperity of large pre-trained language models (LLMs). Eventually, the revolution on the LLMs for NLP tasks happened in 2018. Devlin et al. (2018) and Radford et al. (2018) independently proposed pre-training a large Transformer model on massive text corpora in a self-supervised manner. Their experiments showed that fine-tuning the LLMs on downstream NLP tasks could drastically surpass the previous state-of-the-art (SOTA) systems. Thus, since then, the pretraining-and-finetuning paradigm has monopolized the learning scheme in NLP.

The rest of this section will explain two prototypical LLMs, *i.e.,* GPT (Radford et al., 2018) and BERT (Devlin et al., 2018).

2.3.1 Unidirectional Pre-trained Language Models

Language models (LMs) have a long history and play a crucial role in the NLP field. The primary goal of LMs is to appraise the naturalness of a sentence $\boldsymbol{y} = \{y_1, ..., y_n\}$, *i.e.,* whether a sentence looks like a natural sentence used by humans. In order to achieve this goal, we can estimate the probability of $\boldsymbol{y}$. The estimation can be formulated as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}) \;=\; \prod_{t=1}^{|\boldsymbol{y}|} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}) \tag{2.17}$$

The higher $P_{\boldsymbol{\theta}}(\boldsymbol{y})$ suggests a more plausible sentence. Since $P_{\boldsymbol{\theta}}(\boldsymbol{y})$ is the generic case of $P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, we can model it via RNNs or Transformer, but without the encoder. Similarly, we can optimize θ on a large corpus (like Penn TreeBank (Marcus et al., 1994)) via minimizing the negative log-likelihood as described in Section 2.1.3. Once the LMs are well-trained, we can score a sentence $\boldsymbol{y}$ by feeding it into the LMs. In convention, we use perplexity (PPL) for the scoring, which is calculated as:

$$\text{PPL} = \sqrt[T]{\frac{1}{\prod_{t=1}^{T} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t})}} \tag{2.18}$$

where $T = |\boldsymbol{y}|$. Since these LMs comprehend the text from left to right only, they are referred to as *unidirectional* LMs or *autoregressive* LMs.

Although LMs were devised to assess the quality of sentences, recent works have shown that pre-trained LMs can be endowed with language understanding capability. As such, they can enhance a range of downstream NLP tasks through a fine-tuning process (Dai and Le, 2015; Howard and Ruder, 2018).

GPT Albeit the effectiveness of these approaches, their usage of shallow RNNs and middle-sized datasets cannot fully exploit the potential of LMs. Thus, Radford et al. (2018) proposed training a 12-layer Transformer decoder on the BooksCorpus dataset (Zhu et al., 2015), which contains over 7,000 unique unpublished books from a variety of genres. In addition, as the corpus contains long stretches of contiguous text, it can encourage

the model to learn to capture long-range information, which is crucial for document-level downstream tasks. This model is coined as *GPT*.

Radford et al. (2018) conduct comprehensive experiments to examine the power of GPT on various tasks, including natural language inference, question answering, commonsense reasoning, semantic similarity, text classification, *etc.* They found that with the help of the pre-training, GPT could significantly surpass the previous SOTA systems.

GPT-2 Because of the achievements made by GPT, Radford et al. (2019) aims to further advance the capability of LMs by scaling the model size and training data size to a 48-layer Transformer decoder and 40GB text. Since this model is derived from GPT, it is termed as *GPT-2*. As the training data is sourced from numerous and diverse websites and GPT-2 have a large capacity to learn any signals, Radford et al. (2019) believe that GPT-2 can acquire world knowledge to some extent via self-supervised learning.

In order to validate their hypothesis, in contrast to previous works emphasizing minimal supervision, they decided to test GPT-2 on the zero-shot setting, where no supervision is provided. According to their experiments, even without any supervised training, GPT-2 can complete multiple challenging NLP tasks, such as reading comprehension, summarization, translation, *etc.* Furthermore, GPT-2 achieved SOTA performance on 7 out of the 8 datasets in a zero-shot setting. Finally, it has shown that GPT-2 can generate realistic and coherent continuations about their chosen topic as shown in Table 2.2

TABLE 2.2: An example of synthetic sentences generated by GPT-2.

Human-written prompts: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
Machine-generated completion: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved .....

GPT-3 After seeing the triumph of the first two generations of GPT models, the GPT team is ambitious for a more grandiose goal. Specifically, they train a model, so-called

GPT-3, with 175 billion parameters on 570GB data and test its performance in the few-shot setting (Brown et al., 2020).

Since GPT-3 contains 175 billion parameters, the fine-tuning process will be computationally prohibitive. As a byproduct, it has shown that GPT-2 can generate topic-relevant text from a given prompt. Inspired by this, GPT-3 introduces an innovative paradigm to perform few-shot learning by sidestepping the parameter-updating process. As shown in Figure 2.18, for a sentiment analysis task on tweet, one can craft a prompt by concatenating a task description, a few labeled examples, and a test instance. Then GPT-3 can take this prompt as an input $\boldsymbol{x}$, and produce the correct label y via a conditional probability $P_{\boldsymbol{\theta}}(y|\boldsymbol{x})$. The logic behind this magic is GPT-3 manages to learn the pattern from the provided prompt such that it can complete the unlabeled instance by generating its label.

FIGURE 2.18: An example of prompt-based few-shot learning by GPT-3.

To demonstrate the generality of the prompt-based few-shot learning, Brown et al. (2020) studied a total of 51 few-shot learning tasks. Surprisingly, GPT-3 not only achieves strong performance in few-shot settings, but also matches, in some cases, the SOTA fine-tuned results. Thanks to GPT-3, the prompt-based few-shot learning has revolutionized the learning paradigm in machine learning.

2.3.2 Bidirectional Pre-trained Language Models

The unidirectional LMs are disadvantageous because the representation of each token only considers the leftward context. Such restrictions cause difficulty in addressing tasks requiring context from both directions, such as question answering. Bahdanau et al. (2014) and Peters et al. (2018) have identified that bidirectional contextual representation can significantly boost machine translation and representation learning. These works leverage two RNNs to encode left-to-right and right-to-left contexts, respectively. Nevertheless, the Transformer encoder can model the two-side context via the self-attention module by nature, which is superior to the shallow concatenation of a left-to-right and a right-to-left model. Hence, there have been works expressing interest in training the Transformer encoder in a self-supervised manner.

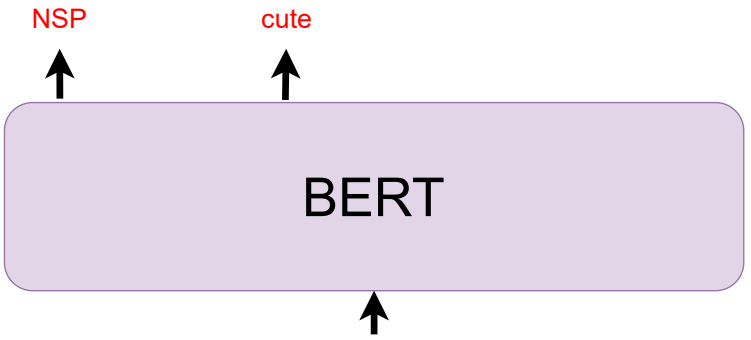


FIGURE 2.19: A schematic illustration of the mask prediction and the next sentence prediction for BERT.

BERT The pioneering work is *BERT*, attempting to pre-train a 24-layer Transformer encoder on the BooksCorpus and English Wikipedia. As the self-attention module would allow each word to access itself, the standard token prediction used by the unidirectional LMs is trivial. Inspired by Taylor (1953), Devlin et al. (2018) devised a masked language model (MLM) to effectively advance the representation learning and the acquisition of linguistic knowledge. Concretely, as shown in Figure 2.19, one can randomly mask some percentage of tokens, and feed the imprinted sentence into BERT. Finally, one can consider the prediction of the masked tokens as the objective.

However, the [MASK] token can cause a discrepancy between the pre-training and fine-tuning, as the [MASK] token does not appear during fine-tuning. To mitigate this, instead of replacing all selected tokens with [MASK], if the i-th token is chosen, one can replace it with (1) the [MASK] symbol 80% time; (2) a random token from the vocabulary 10% of time; (3) the original i-th token 10% of time. According to the empirical study, this avenue works the best.

In addition to the MLM, Devlin et al. (2018) introduces another objective focusing on predicting the relationship between two sentences. Specifically, Figure 2.19 shows that the input consists of two sentences, separated by a special token [SEP]. 50% of the time, the second sentence is immediately after the first one, whereas 50% of the time is a random sentence from the corpus. A [CLS] token is prepended to the input to denote whether the two sentences are adjacent. This objective is beneficial to downstream tasks inferring a relationship between two sentences, such as question answering and natural language inference.

Unlike GPT models emphasizing generation power, BERT put more effort into discriminative tasks. One can feed a sentence or a sentence pair into BERT during fine-tuning. Then the hidden states $\boldsymbol{h}^L_{[\text{CLS}]}$ of [CLS] at the last layer are used to represent the input. Finally, we can pass $\boldsymbol{h}^L_{[\text{CLS}]}$ into an output layer for classification. Despite its simplicity, fine-tuning from the pre-trained BERT can outperform its contemporary works, like GPT, by a sizeable margin. Although one can use BERT for sequence labeling tasks, such as named entity recognition, we limit the discussion to the classification tasks, as we only touch them in this thesis.

RoBERTa The efficacy of BERT has attracted lots of attention from the community and led to the emergence of multiple post-BERT variants. All these variants manage to outperform the original BERT significantly. Out of these variants, RoBERTa noticed that the underperformance of BERT is mainly attributed to insufficient training. Thus, Liu et al. (2019) proposed to make several modifications to BERT, including:

1. Training the model longer, with bigger batches, over more data;
2. Removing the next sentence prediction objective;
3. Training on longer sequences;

4. Dynamically changing the masking pattern applied to the training data.

The first three tactics are straightforward and have been corroborated in previous works (Lample and Conneau, 2019; Yang et al., 2019b). Liu et al. (2019) has found that the original BERT conducted the masking operation in the data pre-processing stage, resulting in a *static* mask. In order to alleviate the potential memorization caused by such a static strategy, BERT repeated an input instance multiple times and applied different masking operations to each of them. However, this brute-force approach can drastically increase storage when scaling to 10x larger training data. As a remedy, RoBERTa designed a *dynamic masking* to randomly mask an input on the fly before feeding it into the model. With the aid of these modifications, RoBERTa further advanced the performance of BERT and marked new SOTA results, compared to the previously published methods.

2.4 Summary

This chapter laid a foundation for the comprehension of this thesis through a review of prior relevant works. We first overviewed the concepts, development, and up-to-date techniques in natural language generation and understanding. Then we reviewed the two mainstreams of pre-trained large language models and described their architectures, objectives, training data, and applications. Bearing the background knowledge in mind, we aim to explore and extend these lines of works by modeling context-dependent knowledge.

Part I

Context-dependent Subword Segmentation

Chapter 3

Subword Segmentation with Contextual Information

This chapter is based on:

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic Programming Encoding for Subword Segmentation in Neural Machine Translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3042–3051, Online. Association for Computational Linguistics.

Due to computational and memory constraints, NMT cannot handle all words in the studied languages. A straightforward avenue is to select uncommon words and treat them as unknown words. This solution can cause mistranslation because of the loss of information. To sidestep this issue, Sennrich et al. (2015), Wu et al. (2016), and Kudo (2018) proposed segmenting words into subword units, which can maximally retain the rare words and drastically reduce the memory footprint.

These approaches consider words as separate and independent pieces. However, people articulate their thoughts via a sequence of words and the choices of words must precisely reflect the actual meaning. Nevertheless, a text is not a combination of expressive words and phrases. Instead, the inter-connection among words forms the context and affects the expressions of words.

This chapter aims to produce linguistically plausible subwords by conditioning the source and target-side contexts. Specifically, we employ a mixed character-subword transformer architecture to conduct subword re-segmentation of the target text using a pre-built subword vocabulary. As the source text is provided for machine translation systems, one can access it and obtain global contextual information by using the encoder. Moreover, we believe subword segmentation can benefit from the left-side local context when decoding the target text. To validate this proposal, we examine it in English $\leftrightarrow$ (German, Romanian, Estonian, Finnish, Hungarian) translations. The empirical studies demonstrate the superiority of our approach over the previous context-independent approaches.

3.1 Introduction

The segmentation of rare words into subword units (Sennrich et al., 2015; Wu et al., 2016) has become a critical component of neural machine translation (Vaswani et al., 2017) and natural language understanding (Devlin et al., 2019; Liu et al., 2019). Subword units enable *open vocabulary* text processing with a negligible pre-processing cost and help maintain a desirable balance between the vocabulary size and decoding speed. Since subword vocabularies are built in an unsupervised manner (Sennrich et al., 2015; Wu et al., 2016), they can be easily adopted for any language.

Given a fixed vocabulary of subword units, rare words can be segmented into a sequence of subword units in different ways. For instance, "un+conscious" and "uncon+scious" are both suitable segmentations for the word "unconscious". There has been a surge of interest in the exploration of subword segmentation. Seeking an optimal subword segmentation is still an open research question. We identify two popular families of subword segmentation algorithms in neural machine translation:

1. Greedy algorithms: Wu et al. (2016) segment words by recursively selecting the longest subword prefix. Sennrich et al. (2015) recursively combine adjacent word fragments that co-occur most frequently, starting from characters.

2. Stochastic algorithms (Kudo, 2018; Provilkov et al., 2019) draw multiple segmentations for source and target sequences resorting to randomization to improve robustness and generalization of translation models.

However, since priors works omit the source context, the resultant segmentation is suboptimal, causing the lack of respect for linguistic properties. We argue that the segmentation of source sentences can be thought of as input features to enhance the segmentation quality. In addition, since there are multiple valid segmentations, we view the subword segmentation of an output sentence in machine translation as a latent variable that should be marginalized to obtain the probability of the output sentence given the input.

In this chapter, we propose a new algorithm called *Dynamic Programming Encoding (DPE)* to mitigate the aforementioned issues. We present a new family of mixed character-subword transformers, for which simple dynamic programming algorithms exist for exact marginalization and MAP inference of subword segmentations. The time complexity of the dynamic programming algorithms is $O(TV)$, where T is the length of the target sentence in characters, and V is the size of the subword vocabulary. By comparison, even computing the conditional probabilities of subword units in an autoregressive model requires $O(TV)$ to estimate the normalizing constant of the categorical distributions. Thus, our dynamic programming algorithm does not incur additional asymptotic costs. We use a lightweight mixed character-subword transformer as a means of pre-processing translation datasets to segment output sentences using DPE for MAP inference.

The performance of a standard subword transformer (Vaswani et al., 2017) trained on WMT datasets tokenized using DPE is compared against Byte Pair Encoding (BPE) (Sennrich et al., 2015) and BPE dropout (Provilkov et al., 2019). Empirical results on English $\leftrightarrow$ (German, Romanian, Estonian, Finnish, Hungarian) suggest that stochastic subword segmentation is effective for tokenizing source sentences, whereas deterministic DPE is superior for segmenting target sentences. DPE achieves an average improvement of 0.9 BLEU over greedy BPE (Sennrich et al., 2015) and an average improvement of 0.55 BLEU over stochastic BPE dropout (Provilkov et al., 2019)

3.2 Latent Subword Segmentation

Let $\boldsymbol{x}$ denote a source sentence and $\boldsymbol{y} = (y_1, \ldots, y_T)$ denote a target sentence comprising T characters. The goal of machine translation is to learn a conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ from a large corpus of source-target sentences. State-of-the-art neural machine

translation systems make use of a dictionary of subword units to tokenize the target sentences in a more succinct way as a sequence of $M \leq T$ subword units. Given a subword vocabulary, there are multiple ways to segment a rare word into a sequence of subwords (see Figure 3.1). The common practice in neural machine translation considers subword segmentation as a pre-process and uses greedy algorithms to segment each word across a translation corpus in a consistent way. This chapter aims to find optimal subword segmentations for the task of machine translation.

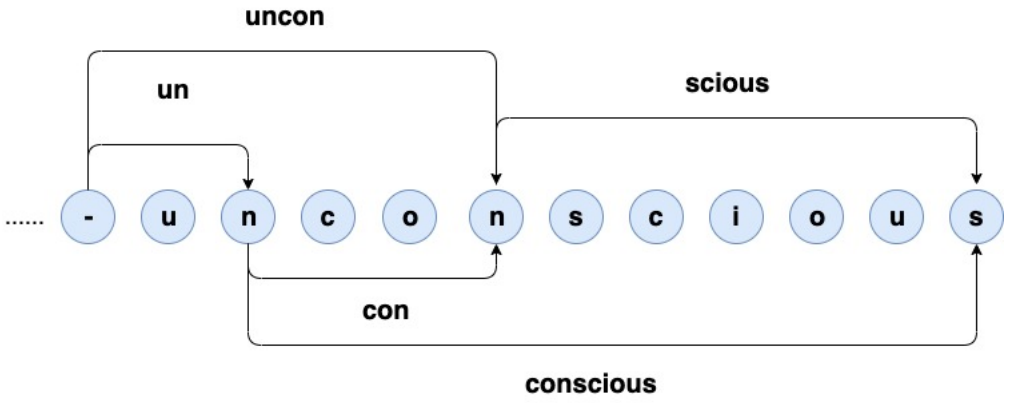


FIGURE 3.1: An illustration of marginalizing subword segmentations of the word 'unconscious'

Let $\boldsymbol{z} = (z_1, .., z_{M+1})$ denote a sequence of character indices $0 = z_1 < z_2 < \ldots < z_M < z_{M+1} = T$ in an ascending order, defining the boundary of M subword segments $\{\boldsymbol{y}_{z_i, z_{i+1}}\}_{i=1}^{M}$. Let $\boldsymbol{y}_{a,b} \equiv [y_{a+1}, \ldots, y_b]$ denote a subword that spans the segment between $(a+1)^{\text{th}}$ and b^{th} characters, including the boundary characters. For example, given a subword dictionary {'c', 'a', 't', 'at', 'ca'}, the word 'cat' may be segmented using $\boldsymbol{z} = (0, 1, 3)$ as ('c', 'at'), or using $\boldsymbol{z} = (0, 2, 3)$ as ('ca', 't'), or using $\boldsymbol{z} = (0, 1, 2, 3)$ as ('c', 'a', 't'). The segmentation $\boldsymbol{z} = (0, 3)$ is not valid since 'cat' does not appear in the subword vocabulary.

Autoregressive language models create a categorical distribution over the subword vocabulary at every subword position and represent the log-probability of a subword sequence using chain rule,

$$\log p(\boldsymbol{y}, \boldsymbol{z}) \;=\; \sum_{i=1}^{|\boldsymbol{z}|} \log p(\boldsymbol{y}_{z_i, z_{i+1}} \mid \boldsymbol{y}_{z_1, z_2}, \ldots, \boldsymbol{y}_{z_{i-1}, z_i}) \; . \tag{3.1}$$

Note that we suppress the dependence of p on $\boldsymbol{x}$ to reduce notational clutter. Most neural machine translation approaches assume that $\boldsymbol{z}$ is a deterministic function of $\boldsymbol{y}$

and implicitly assume that $\log p(\boldsymbol{y}, \boldsymbol{z}) \approx \log p(\boldsymbol{y})$.

We consider a subword segmentation $\boldsymbol{z}$ as a latent variable and let each value of $\boldsymbol{z} \in \mathcal{Z}_y$, in the set of segmentations compatible with $\boldsymbol{y}$, contribute its share to $p(\boldsymbol{y})$ according to $p(\boldsymbol{y}) = \sum_{\boldsymbol{z}} p(\boldsymbol{y}, \boldsymbol{z})$,

$$\log p(\boldsymbol{y}) \;=\; \log \sum_{\boldsymbol{z} \in \mathcal{Z}_y} \exp \sum_{i=1}^{|\boldsymbol{z}|} \log p(\boldsymbol{y}_{z_i, z_{i+1}} \mid \ldots, \boldsymbol{y}_{z_{i-1}, z_i}) \; . \tag{3.2}$$

Note that each particular subword segmentation $\boldsymbol{z} \in \mathcal{Z}_y$ provides a lower bound on the log marginal likelihood $\log p(\boldsymbol{y}) \geq \log p(\boldsymbol{y}, \boldsymbol{z})$. Hence, optimizing Equation 3.1 for a greedily selected segmentation can be justified as a lower bound on Equation 3.2. That said, optimizing Equation 3.2 directly is more desirable. Unfortunately, exact marginalization over all segmentations is computationally prohibitive in a combinatorially large space $\mathcal{Z}_y$, especially because the probability of each subword depends on the segmentation of its conditioning context. In the next section, we discuss a sequence model in which the segmentation of the conditioning context does not influence the probability of the next subword. We describe an efficient Dynamic Programming algorithm to exactly marginalize out all possible subword segmentations in this model.

3.3 A Mixed Character-Subword Transformer

We propose a mixed character-subword transformer architecture, which enables one to marginalize out latent subword segmentations exactly using dynamic programming (see Figure 3.2). Our key insight is to let the transformer architecture process the inputs and the conditioning context based on characters to remain oblivious to the specific choice of subword segmentation in the conditioning context and enable exact marginalization. That said, the output of the transformer is based on subword units and at every position it creates a categorical distribution over the subword vocabulary. More precisely, when generating a subword $\boldsymbol{y}_{z_i, z_{i+1}}$, the model processes the conditioning context $(y_{z_1}, \ldots, y_{z_i})$ based solely on characters using,

$$\log p(\boldsymbol{y}, \boldsymbol{z}) = \sum_{i=1}^{|\boldsymbol{z}|} \log p(\boldsymbol{y}_{z_i, z_{i+1}} \mid y_{z_1}, ..., y_{z_i}) \; , \tag{3.3}$$

where the dependence of p on $\boldsymbol{x}$ is suppressed to reduce notational clutter.

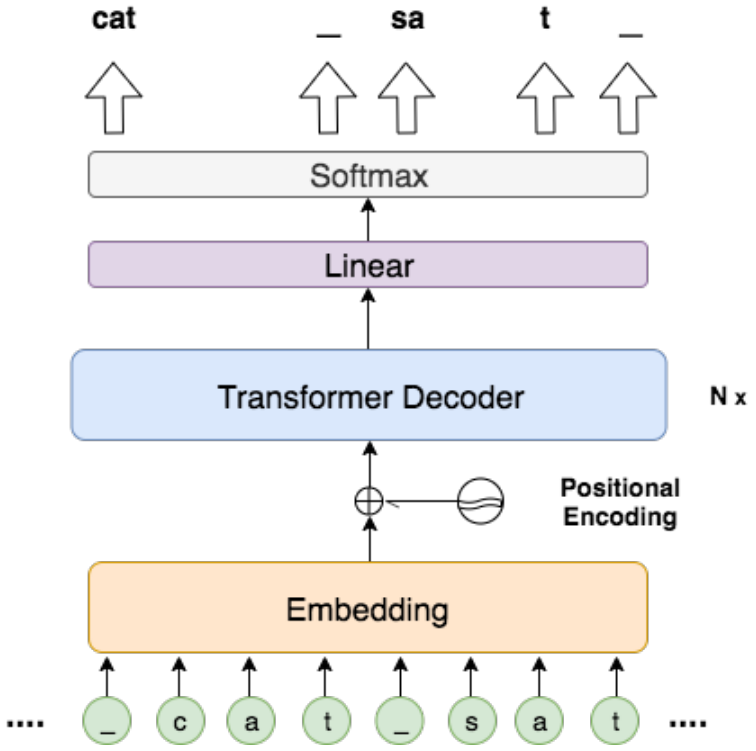


FIGURE 3.2: An illustration of the mixed character-subword Transformer. The input is a list of characters, whereas the output is a sequence of subwords.

Given a fixed subword vocabulary denoted V, at every character position t within $\boldsymbol{y}$, the mixed character-subword model induces a distribution over the next subword $w \in V$ based on,

$$p(w \,|\, y_1, .., y_t) = \frac{\exp(f(y_1, .., y_t)^\top e(w))}{\sum_{w' \in V} \exp(f(y_1, .., y_t)^\top e(w'))}$$

where $f(\cdot)$ processes the conditioning context using a Transformer, and $e(\cdot)$ represents the weights of the softmax layer.

As depicted in Figure 3.2, the mixed character-subword Transformer consumes characters as input generates subwords as output. This figure only shows the decoder architecture, as the encoder that processes $\boldsymbol{x}$ is a standard subword Transformer. Once a subword w is emitted at time step t, the characters of the subword w are fed into the decoder for time steps $t+1$ to $t+|w|$, and the next subword is generated at time step $t+|w|$, conditioned on all of the previously generated characters.

3.3.1 Optimization

We use $\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log P_\theta(\boldsymbol{y}|\boldsymbol{x})$ as the training objective for our latent segmentation translation model, where $\mathcal{D}$ is the training corpus consisting of parallel bilingual sentence pairs. Maximizing the training objective requires marginalization and the computation of the gradient of the log marginal likelihood.

Exact Marginalization. Under our model, the probability of a subword only depends on the character-based encoding of the conditioning context and not its segmentation, as in Equation 3.3. This means that we can compute the log marginal likelihood for a single example $\boldsymbol{y}$, exactly, using the Dynamic Programming algorithm shown in Algorithm 1. The core of the algorithm is line 3, where the probability of the prefix string $\boldsymbol{y}_{0,k}$ is computed by summing terms corresponding to different segmentations. Each term consists of the product of the probability of a subword $\boldsymbol{y}_{j,k}$ times the probability of its conditioning context $(y_1, \ldots, y_j)$. The running time of the algorithm is $\mathcal{O}(mT)$, where T is the length of the string, and m is the size of the longest subword unit in the vocabulary.

Algorithm 1 Dynamic Programming (DP) for Exact Marginalization

Input: $\boldsymbol{y}$ is a sequence of T characters, V is a subword vocabulary, m is the maximum subword length

Output: $\log p(\boldsymbol{y})$ marginalizing out different subword segmentations.

1: $\alpha_0 \leftarrow 0$
2: **for** $k = 1$ **to** T **do**
3: $\quad \alpha_k \leftarrow \log \sum_{j=k-m}^{k-1} \mathbb{1}[\boldsymbol{y}_{j,k} \in V] \exp\left(\alpha_j + \log P_\theta(\boldsymbol{y}_{j,k}|y_1, .., y_j)\right)$
4: **end for**
5: **return** α_T ▷ the marginal probability $\log p(\boldsymbol{y}) = \log \sum_{\boldsymbol{z}\in\mathcal{Z}_y} p(\boldsymbol{y}, \boldsymbol{z})$

Gradient Computation. We use automatic differentiation in PyTorch to backpropagate through the dynamic program in Algorithm 1 and compute its gradient. Compared to a standard Transformer decoder, our mixed character-subword Transformer is 8x slower with a larger memory footprint, due to computation involved in the DP algorithm and large sequence length in characters. To address these issues, we reduce the number of transformer layers from 6 to 4, and accumulate 16 consecutive gradients before one update.

3.3.2 Segmenting Target Sentences

Once the mixed character-subword transformer is trained, it is used to segment the target side of a bilingual corpus. We randomize the subword segmentation of source sentences using BPE dropout (Provilkov et al., 2019). Conditional on the source sentence, we use Algorithm 2, called Dynamic Programming Encoding (DPE) to find a segmentation of the target sentence with the highest posterior probability. This algorithm is similar to the marginalization algorithm, where we use a max operation instead of log-sum-exp. The mixed character-subword transformer is used only for tokenization, and a standard subword transformer is trained on the segmented sentences. For inference using beam search, the mixed character-subword transformer is not needed.

Algorithm 2 Dynamic Programming Encoding (DPE) for Subword Segmentation

Input: $\boldsymbol{y}$ is a sequence of T characters, V is a subword vocabulary, m is the maximum subword length

Output: Segmentation $\boldsymbol{z}$ with highest posterior probability.

for $k = 1$ **to** T **do**

$\beta_k \leftarrow \max_{\{j \in [k-m,k-1] \,|\, \boldsymbol{y}_{j,k} \in V\}} \beta_j + \log P_\theta(\boldsymbol{y}_{j,k} | y_1, .., y_j)$

$b_k \leftarrow \text{argmax}_{\{j \in [k-m,k-1] \,|\, \boldsymbol{y}_{j,k} \in V\}} \beta_j + \log P_\theta(\boldsymbol{y}_{j,k} | y_1, .., y_j)$

end for

$\boldsymbol{z} \leftarrow \text{backtrace}(b_1, .., b_T)$ ▷ backtrace the best segmentation using $\boldsymbol{b}$

3.4 Experiments

Dataset We use WMT09 for En-Hu, WMT14 for En-De, WMT15 for En-Fi, WMT16 for En-Ro and WMT18 for En-Et. We utilize Moses toolkit[1] to pre-process all corpora, and preserve the true case of the text. Unlike Lee et al. (2018), we retain diacritics for En-Ro to retain the morphological richness. We use all of the sentence pairs where the length of either side is less than 80 tokens for training. Byte pair encoding (BPE) (Sennrich et al., 2015) is applied to all language pairs to construct a subword vocabulary and provide a baseline segmentation algorithm. The statistics of all corpora are summarized in Table 3.1.

Training with BPE Dropout. We apply BPE dropout (Provilkov et al., 2019) to each mini-batch. For each complete word, during the BPE merge operation, we randomly

[1] https://github.com/moses-smt/mosesdecoder

TABLE 3.1: Statistics of the corpora.

	Number of sentences			Vocab
	train	dev	test	size
En-Hu WMT09	0.6M	2,051	2,525	32K
En-De WMT14	4.2M	3000	3003	32K
En-Fi WMT15	1.7M	1,500	1,370	32K
En-Ro WMT16	0.6M	1,999	1,999	32K
En-Et WMT18	1.9M	2,000	2,000	32K

drop a particular merge with a probability of 0.05. This value worked the best in our experiments. A word can be split into different segmentations at the training stage, which helps improve the BPE baseline.

DPE Segmentation. DPE can be used for target sentences, but its use for source sentences is not justified as source segmentations should not be marginalized out. Accordingly, we use BPE dropout for segmenting source sentences. That is, we train a mixed character-subword transformer to marginalize out the latent segmentations of a target sentence, given a randomized segmentation of the source sentence by BPE dropout. After the mixed character-subword transformer is trained, it is used to segment the target sentences as described in Section 3.3.2 for tokenization.

As summarized in Figure 3.3, we first train a mixed character-subword transformer with dynamic programming. Then, this model is frozen and used for DPE segmentation of target sentences. Finally, a standard subword transformer is trained on source sentences segmented by BPE dropout and target sentences segmented by DPE. The mixed character-subword transformer is not needed for translation inference.

Transformer Architectures. We use transformer models to train three translation models on BPE, BPE dropout, and DPE corpora. We make use of *transformer base* for all of the experiments.

3.4.1 Main Results

Table 3.2 shows the main results. First, we see that BPE dropout consistently outperforms BPE across language pairs. In Table 3.2, the column labeled to Δ_1 shows the

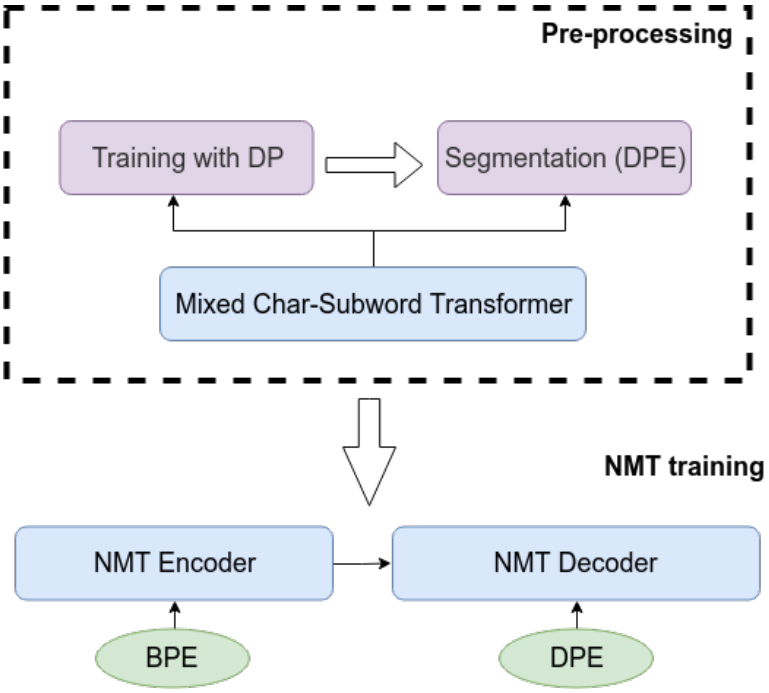


FIGURE 3.3: The workflow of the proposed DPE approach.

improvement of BPE dropout over BPE. This gain can be attributed to the robustness of the NMT model to the segmentation error on the source side, as our analysis in Section 3.4.3 will confirm. Second, we observe further gains resulted from DPE compared to BPE dropout. The column labeled Δ_2 shows the improvement of DPE over BPE dropout. DPE provides an average improvement of 0.55 BLEU over BPE dropout and BPE dropout provides an average improvement of 0.35 BLEU over BPE. As our proposal uses BPE dropout for segmenting the source, we attribute our BLEU score improvements to a better segmentation of the target language with DPE. Finally, compared to BPE for segmenting the source and target, our proposed segmentation method results in large improvements in the translation quality, up to 1.49 BLEU score improvements in Et→En.

3.4.2 Segmentation Examples

Table 3.3 shows examples of target sentences segmented using DPE and BPE and the corresponding source sentences. In addition, Table 3.4 presents the top 50 most common English words that result in a disagreement between BPE and DPE segmentations based on the Et→En corpus. For DPE, for each word, we consider all segmentations produced

TABLE 3.2: Average test BLEU scores (averaged over 3 independent runs) for 3 segmentation algorithms, namely BPE (Sennrich et al., 2015), BPE dropout (Provilkov et al., 2019), and our DPE algorithm on 10 different WMT datasets. Δ_1 shows the improvement of BPE dropout compared to BPE, and Δ_2 shows further improvement DPE compared to BPE dropout. All of the segmentation algorithms use the same subword dictionary with 32K tokens shared between source and target languages.

Method	BPE	BPE dropout	Δ_1	Ours	Δ_2
Source segmentation	BPE	BPE dropout		BPE dropout	
Target segmentation	BPE	BPE dropout		DPE	
En→De	27.11	27.27	+0.16	27.61	+0.34
En→Ro	27.90	28.07	+0.17	28.66	+0.59
En→Et	17.64	18.20	+0.56	18.80	+0.60
En→Fi	15.88	16.18	+0.30	16.89	+0.71
En→Hu	12.80	12.94	+0.14	13.36	+0.42
De→En	30.82	30.85	+0.03	31.21	+0.36
Ro→En	31.67	32.56	+0.89	32.99	+0.43
Et→En	23.13	23.65	+0.52	24.62	+0.97
Fi→En	19.10	19.34	+0.24	19.87	+0.53
Hu→En	16.14	16.61	+0.47	17.05	+0.44
Average	22.22	22.57	+0.35	23.12	+0.55

TABLE 3.3: Two examples of segmentation of English sentences given German inputs.

BPE source: Die G+le+is+anlage war so ausgestattet , dass dort elektr+isch betrie+bene Wagen eingesetzt werden konnten . DPE target: The railway system was equipped in such a way that electrical+ly powered cart+s could be used on it . BPE target: The railway system was equipped in such a way that elect+r+ically powered car+ts could be used on it .
BPE source: Normalerweise wird Kok+ain in kleineren Mengen und nicht durch Tunnel geschm+ug+gelt . DPE target: Normal+ly c+oca+ine is sm+ugg+led in smaller quantities and not through tunnel+s . BPE target: Norm+ally co+c+aine is sm+ugg+led in smaller quantities and not through tun+nels .

TABLE 3.4: Word fragments obtained by BPE vs. DPE. The most frequent words that resulted in a disagreement between BPE and DPE segmentations on $Et \rightarrow En$ are shown.

BPE	DPE (ours)
recognises	recognise + s
advocates	advocate + s
eurozone	euro + zone
underlines	underline + s
strengthens	strengthen + s
entrepreneurship	entrepreneur + ship
acknowledges	acknowledge + s
11.30	11 + .30
wines	wine + s
pres + ently	present + ly
f + illed	fill + ed
endors + ement	endorse + ment
blo + c	bl + oc
cru + cially	crucial + ly
eval + uations	evaluation + s
tre + es	tr + ees
tick + ets	tick + et + s
predic + table	predict + able
multilater + alism	multilateral + ism
rat + ings	rating + s
predic + ted	predict + ed
mo + tives	motiv + es
reinfor + ces	reinforce + s
pro + tocols	protocol + s
pro + gressively	progressive + ly
sk + ill	ski + ll
preva + ils	prevail + s
decent + ralisation	decent + ral + isation
sto + red	stor + ed
influ + enz + a	influen + za
margin + alised	marginal + ised
12.00	12 + .00
sta + ying	stay + ing
intens + ity	intensi + ty
rec + ast	re + cast
guid + eline	guide + line
emb + arked	embark + ed
out + lines	outline + s
scen + ari + os	scenario + s
n + ative	na + tive
preven + tative	prevent + ative
hom + eland	home + land
bat + hing	bath + ing
endang + ered	endanger + ed
cont + inen + tal	continent + al
t + enth	ten + th
vul + n + era + bility	vul + ner + ability
realis + ing	real + ising
t + ighter	tight + er

and show the segmentation that attains the highest frequency of usage in Table 3.4. As can be observed, DPE produces more linguistically plausible morpheme-based subwords compared to BPE. For instance, BPE segments *"carts"* into *"car"+"ts"*, as both *"car"* and *"ts"* are common subwords and listed in the BPE merge table. By contrast DPE segments *"carts"* into *"cart"+"s"*. We attribute the linguistic characteristics of the DPE segments to the fact that DPE conditions the segmentation of a target word on the source sentence and the previous tokens of the target sentence, as opposed to BPE, which mainly makes use of the frequency of subwords, without any context.

DPE generally identifies and leverages some linguistic properties, *e.g.,* plural, antonym, normalization, verb tenses, *etc.* However, BPE tends to deliver less linguistically plausible segmentations, due to its greedy nature and the lack of context.

3.4.3 Analysis

Conditional Subword Segmentation. One of our hypotheses for the effectiveness of subword segmentation with DPE is that it conditions the segmentation of the target on the source language. To verify this hypothesis, we train mixed character-subword Transformer solely on the target language sentences in the bilingual training corpus using the *language model* training objective. This is in contrast to the mixed character-subword model used in the DPE segmentation of the main results in Table 3.2, where the model is conditioned on the source language and trained on the sentence pairs using a *conditional language model* training objective. Once the mixed character-subword Transformer language model is trained, it is then used to segment the target sentence of the bilingual corpus in the pre-processing step before a translation model is trained.

Table 3.5 shows the results. It compares the unconditional language model (LM) DPE *v.s.* the conditional DPE for segmenting the target language, where we use BPE dropout for segmenting the source language. We observe that without the information from the source, LM DPE is on-par with BPE, and is significantly outperformed by conditional DPE. This observation confirms our hypothesis that segmentation in NMT should be source-dependent.

We are further interested in analyzing the differences of the target language segmentation depending on the source language. For this analysis, we filtered out a multilingual

TABLE 3.5: DPE-LM learns a segmentation of the target based on language modeling, which is *not* conditioned on the source language.

Source Target	BPE drop BPE drop	BPE drop LM DPE	BPE drop DPE
En→Ro	28.07	28.07	28.66
En→Hu	12.94	12.87	13.36
Ro→En	32.56	32.57	32.99
Hu→En	16.61	16.41	17.05

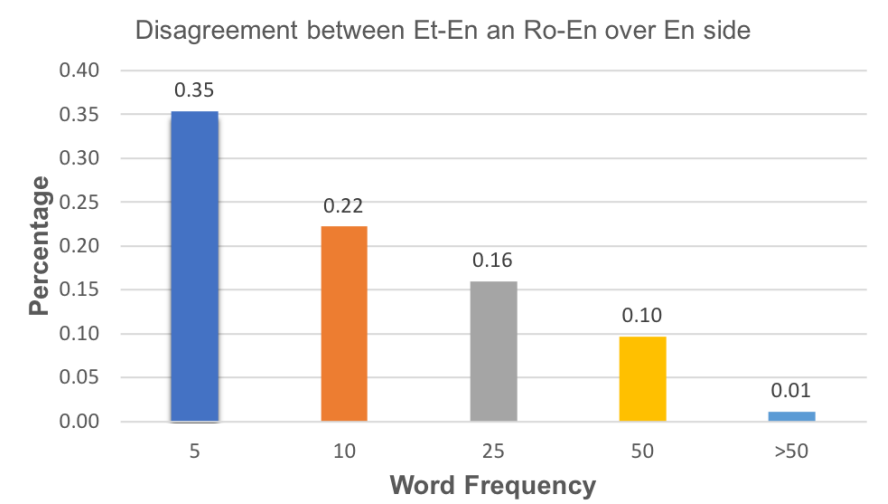


FIGURE 3.4: Disagreement of DPE segments between Et-En and Ro-En over English vocabulary

parallel corpus from WMT, which contains parallel sentences in three languages English, Estonian and Romanian. That is, for each English sentence we have the corresponding sentences in Et and Ro. We then trained two DPE segmentation models for the translation tasks of Et→En and Ro→En, where English is the target language. Figure 3.4 shows when conditioning the segmentation of the target on different source languages, DPE can lead to different segmentations even for an identical multi-parallel corpus. The differences are more significant for low-frequency words.

Another aspect of DPE segmentation method is its dependency on the *segmentation* of the source. As mentioned, we segment the target sentence *on the fly* using our mixed character-subword model *given* a randomized segmentation of the source produced by BPE dropout. That means during the training of the NMT model where we use BPE dropout for the source sentence, the corresponding target sentence may get a different DPE segmentation given the randomized segmentation of the source sentence. We are

TABLE 3.6: "DPE Fixed" obtains a fixed segmentation of the target sentence given the BPE-segmented source sentence, whereas "DPE On The Fly" obtain the best segmentation of the target sentence given a randomized segmentation of the source produced by BPE dropout.

Source Target	BPE drop DPE Fixed	BPE drop DPE On The Fly
En→Ro	28.58	28.66
En→Hu	13.14	13.36
En→Et	18.51	18.80
Ro→En	32.73	32.99
Hu→En	16.82	17.05
Et→En	24.37	24.62

interested in the effectiveness of the target segmentation if we commit to a fixed DPE segmentation conditioned on the BPE segmentation of the input. Table 3.6 shows the results. We observe a marginal drop when using the fixed DPE, which indicates that the encoder can benefit from a stochastic segmentation, while the decoder prefers a deterministic segmentation corresponding to the segmentation of the source.

DPE vs BPE. We are interested in comparing the effectiveness of DPE *v.s.* BPE for the target, given BPE dropout as the same segmentation method for the source. Table 3.7 shows the results. As observed, target segmentation with DPE consistently outperforms BPE, leading to up to 0.9 BLEU score improvements. We further note that using BPE dropout on the target has a similar performance to BPE, and it is consistently outperformed by DPE.

TABLE 3.7: BLEU score of different target segmentation methods.

Source Target	BPE drop BPE	BPE drop BPE drop	BPE drop DPE
En→Ro	28.04	28.07	28.66
En→Et	18.09	18.20	18.80
Ro→En	32.40	32.56	32.99
Et→En	23.52	23.65	24.62

We further analyze the segmentations produced by DPE *v.s.* BPE. Figure 3.5 shows the percentage of the target words which have different segmentation with BPE and DPE, for different word frequency bands in En→Et translation task. We observe that

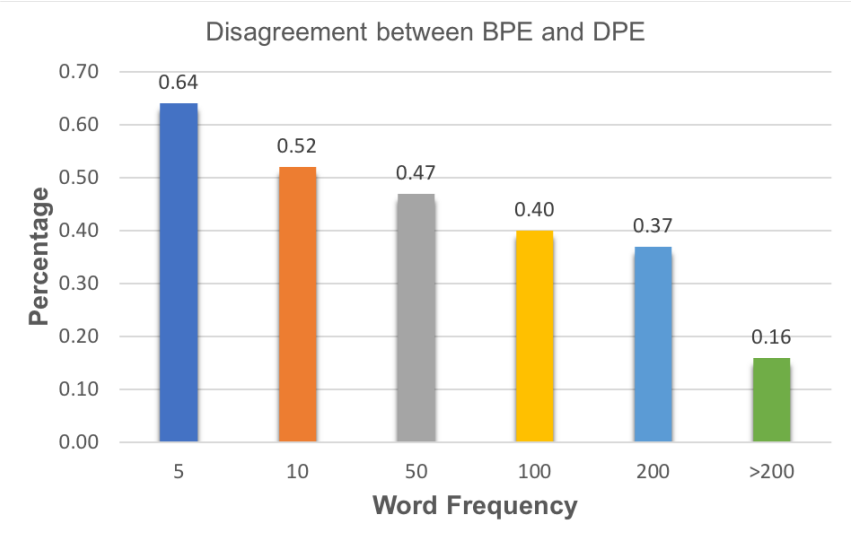


FIGURE 3.5: Disagreement of segments between BPE and DPE over Estonian vocabulary.

for Estonian words whose occurrence is up to 5 in the training set, the disagreement rate between DPE and BPE is 64%. The disagreement rate decreases as we go to words in higher frequency bands. This may imply that the main difference between the relatively large BLEU score difference between BPE and DPE is due to their different segmentation mainly for low-frequency words.

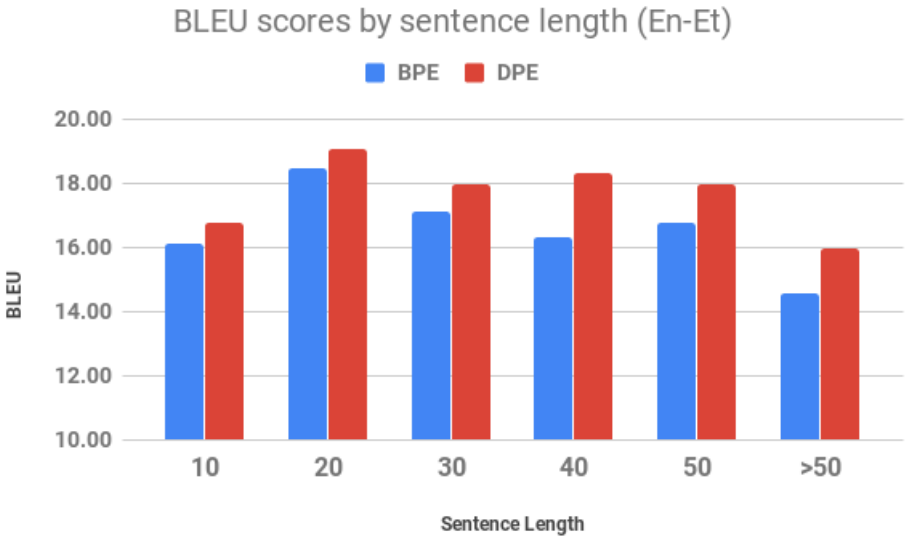


FIGURE 3.6: BLEU scores of BPE vs DPE by the lengths of sentences for En→Et.

We further plot the distribution of BLEU scores by the length of target sentences. As shown in Figure 3.6, DPE demonstrates much better gains on the longer sentences, compared with the BPE version.

3.5 Summary

This chapter introduces *Dynamic Programming Encoding* in order to incorporate the information of the source language into subword segmentation of the target language. Our approach utilizes dynamic programming for marginalizing the latent segmentations when training, and inferring the highest probability segmentation when tokenizing. Our comprehensive experiments show impressive improvements compared to state-of-the-art segmentation methods in NMT, *i.e.,* BPE and its stochastic variant BPE dropout.

Part II

Context-dependent Natural Language Understanding

Chapter 4

Context-conditional Scene Graph Modification

This chapter is based on:

Xuanli He, Quan Hung Tran, Gholamreza Haffari, Walter Chang, Zhe Lin, Trung Bui, Franck Dernoncourt, and Nhan Dam. 2020. Scene Graph Modification Based on Natural Language Commands. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 972–990, Online. Association for Computational Linguistics.

The primary goal of this thesis is to leverage context to advance natural language understanding and generation. In Part I, we have testified that incorporating short-range or within-sentence context into subword segmentation can produce morphologically plausible subword units, leading to significant improvements over previous approaches. However, as shown in Section 1, the context can span beyond a single sentence, especially in a dialogue system, where we have to capture a long-range or cross-sentence context to alleviate the uncertainty owing to back-and-forth conversations. In this chapter and the next one, we focus on two dialogue-driven natural language understanding tasks that demand the modeling of the long-range context.

This chapter targets the modification of scene graphs. As described in Section 2.2.2, scene graphs are machine-understandable structural representations of natural sentences.

Modern search engines have introduced a click-through feature to refine the original query. Inspired by this, we present a novel task to support the refinement of the initial scene graph via natural language commands. We first formulate the initial scene and a modification query as the context. Then we generate an updated scene graph based on the interaction within the context using a hierarchical SEQ2SEQ model. Since this scenario is under-explored, we unveil three datasets as the testbed. Our experiments show that the proposed architecture and its variants can fulfill the objective, *i.e.,* updating an initial scene graph from a modification query.

4.1 Introduction

Parsing text into structured semantics representation is one of the most long-standing and active research problems in NLP. Numerous parsing methods have been developed for many different semantic structure representations (Chen and Manning, 2014; Mrini et al., 2019; Zhou and Zhao, 2019; Clark et al., 2018; Wang et al., 2018). However, most of these previous works focus on parsing a *single* sentence, while a typical human-computer interaction session or conversation is *not single-turn*. A prominent example is image search. Users usually start with short phrases describing the main objects or topics they are looking for. Depending on the result, the users may then *modify* their query to add more constraints or give additional information. In this case, without the modification capability, a static representation is not suitable to track the changing intent of the user. We argue that the back-and-forth and multi-turn nature of human-computer interactions necessitate the need for updating the structured representation. Another advantage of modifying a structured representation in the interactive setting is that it makes it easier to check the consistency. For instance, it is much easier to check whether the user requests two contradicting attributes for the same object in a scene graph during the interactive search, which can be done automatically.

In this chapter, we propose the problem of *scene graph modification* for search. A scene graph (Johnson et al., 2015) is a semantic formalism that represents the desired image as a graph of objects with relations and attributes. This semantic representation has been shown to be very successful in retrieval systems (Johnson et al., 2015; Schuster et al., 2015; Vendrov et al., 2015). Inspired by the dialogue state tracking setting (Perez and Liu, 2017; Ren et al., 2018), we consider the scene graph modification problem as

follows. Given an initial scene graph and a new query issued by the user, the goal is to generate a new scene graph taking into account the original graph and the new query.

We formulate the problem as *conditional graph modification*, and create three datasets for this problem. We propose novel encoder-decoder architectures for conditional graph modification. More specifically, our graph encoder is built upon the self-attention architecture popular in state-of-the-art machine translation models (Vaswani et al., 2017; Edunov et al., 2018), which is superior to, according to our study, Graph Convolutional Networks (GCN) (Kipf and Welling, 2017). Unique to our problem, however, is the fact that we have an open set of relation types in the graphs. Thus, we propose a novel graph-conditioned sparse Transformer, in which the relation information is embedded directly into the self-attention grid. For the decoder, we treat the graph modification task as a sequence generation problem (Li et al., 2018; Simonovsky and Komodakis, 2018; You et al., 2018). Furthermore, to encourage the information sharing between the input graph and modification query, we introduce two techniques, *i.e.,* late feature fusion through gating and early feature fusion through cross-attention. We further create three datasets to evaluate our models. The first two datasets are derived from public sources: MSCOCO (Lin et al., 2014) and Google Conceptual Captioning (GCC) (Sharma et al., 2018) while the last is collected using Amazon Mechanical Turk (MTurk). Experiments show that our best model achieves up to 8.5% improvement over the strong baselines on both the synthetic and user-generated data in terms of F1 score.

Our contributions are three-fold. Firstly, we introduce the problem of scene graph modification – an important component in multi-modal search and dialogue. Secondly, we propose a novel encoder-decoder architecture relying on a graph-conditioned Transformer and cross-attention to tackle the problem, outperforming the strong baselines that we set up for the task. Thirdly, we introduce three datasets that can serve as evaluation benchmarks for future research[1].

4.2 Data Creation

In this section, we detail our data creation process. We start with information on scene graphs and a parser to generate them for the captions in two existing datasets, *i.e.,*

[1]The dataset is available at: https://github.com/xlhex/SceneGraphModification.git

MSCOCO (Lin et al., 2014) and GCC (Sharma et al., 2018). We then describe how to generate modified scene graphs and modification queries based on these scene graphs, and leverage human annotators to increase and analyze data quality.

4.2.1 Scene Graphs

Schuster et al. (2015) introduce scene graphs as semantic representations of images. As shown in Figure 2.16, a parser will parse a sentence into a list of objects. Although there are several scene graphs annotated datasets for images (Krishna et al., 2017), the alignments between graphs and text are unavailable. Moreover, image grounded scene graphs, *e.g.,* the Visual Genome dataset (Krishna et al., 2017), also contain lots of non-salient objects and relations, while search queries focus more on the main objects and their connections.

The lack of a large-scale and high-quality public dataset prompts us to create our own benchmark datasets. To do this, we start with the popular captioning datasets: MSCOCO (Lin et al., 2014) and GCC (Sharma et al., 2018). We use a scene graph parser developed by Adobe to parse a random subset of MSCOCO description data and GCC captions to construct scene graphs. The parser is built upon a dependency parser (Dozat and Manning, 2016), similar to the SPICE system (Anderson et al., 2016).

4.2.2 Modified MSCOCO and GCC for Graph Modification

Our first two datasets add annotations on top of the captions for MSCOCO and GCC. The parser described in Section 4.2.1 is used to create 200k scene graphs from MSCOCO and 420k scene graphs from GCC data. Comparing the two datasets, the graphs from MSCOCO are simpler, while the GCC graphs are much more complicated.

Given a scene graph $\mathcal{G}$, we construct a triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, where $\boldsymbol{x}$ is the source graph, $\boldsymbol{y}$ indicates the modification query, and $\boldsymbol{z}$ represents the target graph. More specifically, we uniformly select and apply an action $\boldsymbol{a}$ from the set of all possible graph modification operations $A = \{\textsc{Delete}, \textsc{Insert}, \textsc{Substitute}\}$. The actions are applied to the graph as follows:

TABLE 4.1: Simplified templates for synthetic data, with each operation has 10 templates.

Insertion: I want **xx**, I prefer **xx**, I like **xx** I would like to see **xx**, Show me **xx**, Give me **xx**, I'm interested in **xx** I need **xx**, Search for **xx**, Return **xx** (**xx** are nodes to be inserted)
Deletion: remove **xx**, I do not want **xx**, delete **xx** I do not like **xx**, omit **xx**, I do not need **xx** erase **xx**, ignore **xx**, discard **xx**, drop **xx** (**xx** denotes the node to be deleted)
Substitution: change **xx** to **yy**, update **xx** to **yy** replace **xx** with **yy**, substitute **yy** for **xx** I prefer **yy** to **xx**, modify **xx** to **yy** I want **yy** rather than **xx**, switch **xx** to **yy** convert **xx** to **yy**, give me **yy** instead of **xx** (**xx** and **yy** are old nodes and updated nodes)

- **Delete.** We randomly select a node from $\mathcal{G}$ (denoting the source graph $\boldsymbol{x}$), and then remove this node and its associated edges. The remaining nodes and edges are then the target graph $\boldsymbol{z}$. As for the modification query $\boldsymbol{y}$, it is generated from a randomly selected *deletion template* or by MTurk workers. These templates are based upon the Edit Me dataset (Manuvinakurike et al., 2018);

- **Insert.** We treat insertion as the inversion of deletion. Specifically, we produce the source graph $\boldsymbol{x}$ via a DELETE operation on $\mathcal{G}$, where the target graph $\boldsymbol{z}$ is set to $\mathcal{G}$. Like the deletion operator, the insertion query $\boldsymbol{y}$ is generated by either the MTurk workers, or by templates;

- **Substitute.** We replace a randomly selected node from the source graph $\mathcal{G}$ with a semantically similar node to get the target graph. To find the new node, we make use of the AllenNLP toolkit (Gardner et al., 2018) to get a list of candidate words based on their semantic similarity scores to the old node.

In Table 4.1, we summarize the templates used for our synthetic data.

4.2.3 Crowdsourcing User Data

As described above, apart from using templates, we crowdsource more diverse, and natural modification queries from MTurk. As depicted in Figure 4.1, we first show the workers an example that includes a source graph, a target graph, and three acceptable modification queries. Then the workers are asked to fill in their descriptions for the unannotated instances. We refer to the template-based version of the datasets as "synthetic" while the user-generated contents as "user-generated".

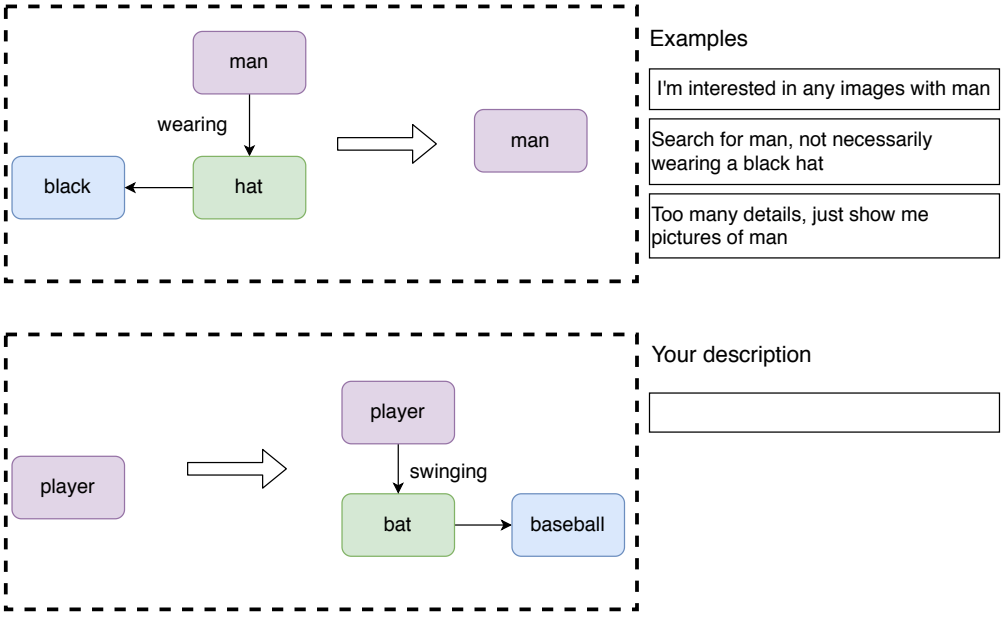


FIGURE 4.1: An interface of the crowd-sourcing stage. We first present some examples to workers. Then we ask the workers to write a description based on the new instance.

We notice several difficulties in our preliminary trials within the data collection process. Firstly, understanding the graphs requires some knowledge of NLP, thus not all MTurk workers can provide good modification queries. Secondly, due to deletion and parser errors, we encounter some graphs with disconnected components in the data. Thirdly, there are many overly complicated graphs that are not representative of search queries, as most of the search queries are relatively short, with just one or two objects. To mitigate these problems, we manually filter the data by removing graphs with disconnected components, low-quality instances, or excessively long descriptions (*i.e.,* more than 5 nodes). The final dataset contains 32k examples.

To test the quality of our crowd-sourced dataset, we performed a limited user study with 15 testers who were not aware of the nature of the work and how we collected

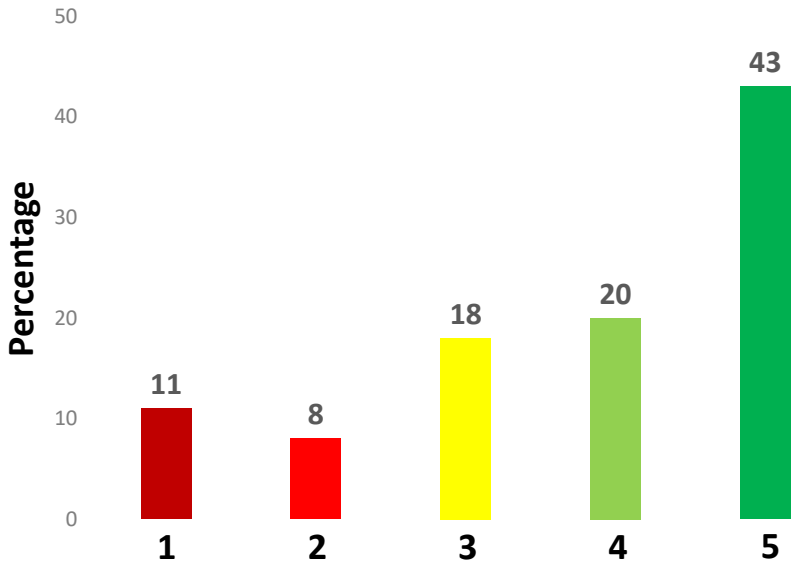


FIGURE 4.2: Quality score distribution on the crowdsourced dataset. On a scale of 1 to 5, 1 is the worst instance while 5 is the best example.

the dataset. We give them a random collection of instances, each of which is a triplet of source graph, modification query, and target graph. The tester would then give a score indicating the quality of each instance based on the following two criteria: (i) *how well the modification query is reflected in the target graph?* and (2) *how natural are the query and the graphs?* Regarding the second criterion, we instruct the scorer to assess whether the query and graph are human-like, grammatically and semantically. Furthermore, as most scorers are knowledgeable in image search, they are also required to evaluate whether they think the query is plausible in a search scenario.

Figure 4.2 shows the score distribution from 200 randomly chosen instances. We observe that most of the quality scores of 3 or 4 are due to the modification query or graphs to be unnatural. Testers tend to give the score of 1 for semantically wrong instances (e.g the modification query does not match the changes). Overall, the testers judge the data to be good with the average score of 3.76.

4.3 Methodology

In this section, we explore different methods to tackle our proposed problem. By analyzing the results and comparing different models, we establish baselines and set up the

research direction for future work. We start by formalizing the problem, and defining the input as well as the expected output along with the notations. We then define our encoder-decoder architecture with the focus on our novel modeling characteristics: (i) the graph encoder with graph-conditioned, sparsely connected Transformer and (ii) the early and late feature fusion models for combining information from the input text and graph.

Notations. A graph is represented by $\boldsymbol{x}^{\mathcal{G}} := (\boldsymbol{x}^{\mathcal{N}}, \boldsymbol{x}^{\mathcal{E}})$. The node set is denoted by $\boldsymbol{x}^{\mathcal{N}} := \{x_1, .., x_{|\boldsymbol{x}^{\mathcal{N}}|}\}$ where $|\boldsymbol{x}^{\mathcal{N}}|$ is the number of nodes, and $x_i \in V_{\mathcal{N}}$ where $V_{\mathcal{N}}$ is the node vocabulary. The edge set is denoted by $\boldsymbol{x}^{\mathcal{E}} := \{x_{i,j} | x_i, x_j \in \boldsymbol{x}^{\mathcal{N}}, x_{i,j} \in V_{\mathcal{E}}\}$ where $V_{\mathcal{E}}$ is the edge vocabulary.

4.3.1 Problem Formulation

We formulate the task as a conditional generation problem. Formally, given a source graph $\boldsymbol{x}^{\mathcal{G}}$ and a modification query $\boldsymbol{y}$, one can produce a target graph $\boldsymbol{z}^{\mathcal{G}}$ by maximizing the conditional probability $p(\boldsymbol{z}^{\mathcal{G}} \mid \boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y})$. As a graph consists of a list of *typed* nodes and edges, we further decompose the conditional probability (You et al., 2018) as,

$$p(\boldsymbol{z}^{\mathcal{G}} \mid \boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y}) = p(\boldsymbol{z}^{\mathcal{N}} \mid \boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y}) \times p(\boldsymbol{z}^{\mathcal{E}} \mid \boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y}, \boldsymbol{z}^{\mathcal{N}}), \tag{4.1}$$

where $\boldsymbol{z}^{\mathcal{N}}$ and $\boldsymbol{z}^{\mathcal{E}}$ respectively denote the nodes and edges of the graph $\boldsymbol{z}^{\mathcal{G}}$.

Given a training dataset of input-output pairs, denoted by $\mathcal{D} \equiv \{(\boldsymbol{x}_d^{\mathcal{G}}, \boldsymbol{y}_d, \boldsymbol{z}_d^{\mathcal{G}})\}_{d=1}^{\mathcal{D}}$, we train the model by maximizing the *conditional log-likelihood* $\ell_{\text{CLL}} = \ell_{\text{Node}} + \ell_{\text{Edge}}$ where,

$$\ell_{\text{Node}} = \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \in \mathcal{D}} \log p(\boldsymbol{z}^{\mathcal{N}} \mid \boldsymbol{x}, \boldsymbol{y}; \theta_{\mathcal{N}}) \tag{4.2}$$

$$\ell_{\text{Edge}} = \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \in \mathcal{D}} \log p(\boldsymbol{z}^{\mathcal{E}} \mid \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}^{\mathcal{N}}; \theta_{\mathcal{E}}). \tag{4.3}$$

During learning and decoding, we sort the nodes according to a topological order which exists for all the directed graphs in our user-generated and synthetic datasets.

4.3.2 Graph-based Encoder-Decoder Model

Inspired by the machine translation literature (Bahdanau et al., 2014; Vaswani et al., 2017), we build our model based on the encoder-decoder framework. Since our task takes a source graph and a modification query as inputs, we need two encoders to model the graph and text information separately. Thus, our model has four main components: the query encoder, the graph encoder, the edge decoder, and the node decoder. The information flow between the components is shown in Figure 4.3. In general, we encode the graph and text modification query into a joint representation, generating the target graph in two stages. Firstly, the target nodes are generated via node-level RNNs. Then we leverage another RNNs to produce the target edges over the nodes.

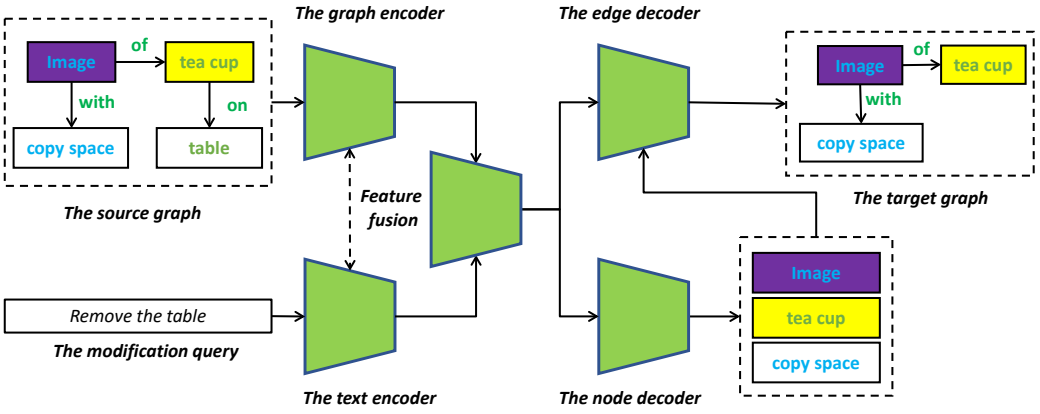


FIGURE 4.3: The information flow of our model. Green boxes denote the main computational units.

4.3.2.1 Graph Encoder: Sparsely Connected Transformer

The standard Transformer architecture (Vaswani et al., 2017; Yang et al., 2019a) relies on a grid of fully-connected self-attention to obtain the contextualized representations from a sequence of elements. In this work, we propose the graph-conditioned, sparsely connected Transformer to encode the information from a graph. Our idea is partially inspired by the sparse Transformer through factorization (Child et al., 2019). Despite the similar name, the two methods share very few similarities in both motivations and mechanisms. The architecture of our graph encoder with the sparely connected Transformer is detailed below.

Compared to natural language text, graphs are structured data comprised of two main components: nodes and edges. To efficiently encode a graph, we need to encode the information not only from these constituent components, but also from their interactions, namely the node-edge association and connectivity. Thus, we incorporate the information from all the edges to the nodes from which these edges are originated. More formally, our edge-aware node embedding $\boldsymbol{x}_i$ can be obtained from the list of source graph nodes and edges via,

$$\boldsymbol{x}_i = \boldsymbol{E}^{\mathcal{N}}[x_i] + \sum_{j \in J(i)} \boldsymbol{E}^{\mathcal{E}}[x_{ij}], \tag{4.4}$$

where $\boldsymbol{E}^{\mathcal{N}}$ and $\boldsymbol{E}^{\mathcal{E}}$ are the embedding tables for node and edge labels respectively, and $J(i)$ is the set of nodes connected (both inbound and outbound) to the ith node in the graph.

After getting the edge-aware node embeddings, we employ the sparsely connected Transformer to learn the contextualized embeddings of the whole graph. Unlike the conventional Transformer, we do not incorporate the positional encoding into our graph inputs because the nodes are not in a predetermined sequence. Given the edge information from $\boldsymbol{x}^{\mathcal{E}}$, we enforce the connectivity information by making nodes only visible to its *first order neighbor*. Let us denote the attention grid of the Transformer as $\mathbf{A}$. We then define $A[\boldsymbol{x}_i, \boldsymbol{x}_j] = f(\boldsymbol{x}_i, \boldsymbol{x}_j)$ if $x_{i,j} \in \boldsymbol{x}^{\mathcal{E}}$ or zero otherwise, where f denotes the normalized inner product function.

The sparsely connected Transformer, thus, provides the graph node representations conditioned on the graph structure, using the edge labels in the input embeddings and sparse layers in self-attention. We denote the node representations in the output of the sparsely connected Transformer by $[\boldsymbol{m}_{x_1}, .., \boldsymbol{m}_{x_{|\boldsymbol{x}^{\mathcal{N}}|}}]$.

4.3.2.2 Query Encoder

We use a standard Transformer encoder (Vaswani et al., 2017) to encode the modification query $\boldsymbol{y} = (y_1, .., y_{|\boldsymbol{y}|})$ into $[\boldsymbol{m}_{y_1}, ..., \boldsymbol{m}_{y_{|\boldsymbol{y}|}}]$. Crucially, in order to encourage semantic alignment, we share the parameters of the graph and query encoders.

4.3.2.3 Information Fusion of Encoders

In a conventional encoder-decoder model, usually there is only one encoder. In our scenario, there are two sources of information, which require separate encoders. The most straightforward way to incorporate the two information sources is through concatenation. Concretely, the combined representation would be,

$$\boldsymbol{m} = [\boldsymbol{m}_{x_1}, ..., \boldsymbol{m}_{x_{|\boldsymbol{x}^{\mathcal{V}}|}}, \boldsymbol{m}_{y_1}, ..., \boldsymbol{m}_{y_{|\boldsymbol{y}|}}]. \tag{4.5}$$

The decoder component will then be responsible for information communication between the two encoders through its connections to them. In the following, we propose more advanced methods to combine the two sources of information.

Late Fusion via Gating. To enhance the ability of the model to combine the encoders' information for better use of the decoder, we introduce a parametric approach with the gating mechanism. Through the gating mechanism, we aim to filter useful information from the graph based on the modification query, and vice versa.

More specifically, we add a special [CLS] token to the graph and in front of the query sentence. The representation of this token in the encoders will then capture the holistic understanding, which we denote by $\boldsymbol{m}_{\boldsymbol{x}^{\mathcal{G}}}$ and $\boldsymbol{m}_{\boldsymbol{y}}$ for the graph and modification query respectively.

We make use of these holistic meaning vectors to filter useful information from the representations of the graph nodes $\boldsymbol{m}_{x_i}$ and modification query tokens $\boldsymbol{m}_{y_j}$ as follows,

$$\boldsymbol{g}_{x_i} = \boldsymbol{\sigma}(\text{MLP}(\boldsymbol{m}_{x_i}, \boldsymbol{m}_{\boldsymbol{y}})) \tag{4.6}$$

$$\boldsymbol{m}'_{x_i} = \boldsymbol{g}_{x_i} \odot \boldsymbol{m}_{x_i} \tag{4.7}$$

$$\boldsymbol{g}_{y_j} = \boldsymbol{\sigma}(\text{MLP}(\boldsymbol{m}_{y_j}, \boldsymbol{m}_{\boldsymbol{x}^{\mathcal{G}}})) \tag{4.8}$$

$$\boldsymbol{m}'_{y_j} = \boldsymbol{g}_{y_j} \odot \boldsymbol{m}_{y_j}, \tag{4.9}$$

where MLP is a multi-layer perceptron, $\odot$ indicates an element-wise multiplication, and $\boldsymbol{\sigma}$ is the element-wise sigmoid function used to construct the gates $\boldsymbol{g}_{x_i}$ and $\boldsymbol{g}_{y_j}$. The updated node $\boldsymbol{m}'_{x_i}$ and token $\boldsymbol{m}'_{y_j}$ are then used in the joint encoders representation of Equation 4.5.

We refer to this gating mechanism as *late fusion* since it does not let the information from the graph and text interact in their respective lower level encoders. In other words, the fusion happens after the contextualized information has already been learned.

Early Fusion via Cross-Attention. To allow a deeper interaction between the graph and text encoders, we explore fusing features at the early stage before the contextualized node $\boldsymbol{m}_{x_i}$ and token $\boldsymbol{m}_{y_i}$ representations are learned. This is achieved via *cross-attention*, an early fusion technique.

Recall that the parameters of the graph and query encoders are shared to enable the encoding of the two sources in the same semantic space. That is, we use the same Transformer encoder for both sources. In cross-attention, we concatenate the $\boldsymbol{x}$ (from Equation 4.4) and $\boldsymbol{y}$ before rather than after the Transformer encoder. As such, the encoder's input is $[\boldsymbol{x}, \boldsymbol{y}]$. In the Transformer, the representation of each query token gets updated by self-attending to the representations of all the query tokens and graph nodes in the previous layer. However, the representation of each graph node gets updated by self-attending only to its graph neighbors according to the connections of the sparsely connected Transformer as well as all query tokens. The final representation $\boldsymbol{m}$ is taken from the output of Transformer. Figure 4.4 shows the information flow in the cross-attention mechanism.

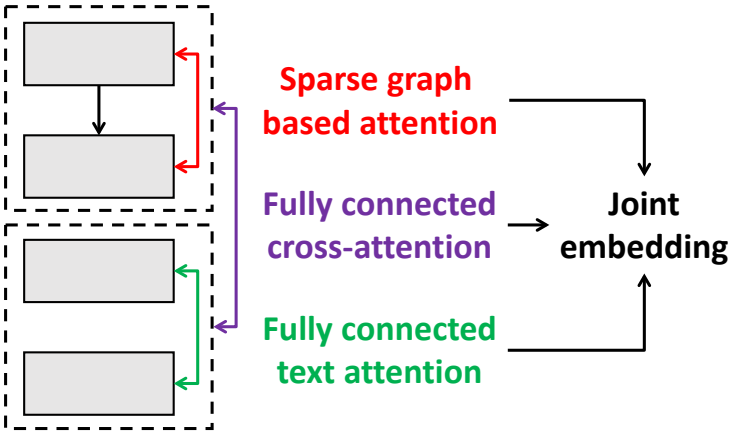


FIGURE 4.4: Cross-attention fusion. Graphical components can attend to the text input, and vice versa.

4.3.2.4 Node-level Decoder

We use GRU cells (Cho et al., 2014) for our RNN decoders. The node-level decoder is a vanilla auto-regressive model described as,

$$\boldsymbol{h}_t^{\mathcal{N}} = \text{GRU}^{\mathcal{N}}(z_{t-1}, \boldsymbol{h}_{t-1}^{\mathcal{N}}) \tag{4.10}$$

$$\boldsymbol{c}_t^{\mathcal{N}} = \text{ATTN}^{\mathcal{N}}(\boldsymbol{h}_t^{\mathcal{N}}, \boldsymbol{m}) \tag{4.11}$$

$$p(z_t \mid \boldsymbol{z}_{<t}, \boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y}) = \text{softmax}(\boldsymbol{W}[\boldsymbol{h}_t^{\mathcal{N}}, \boldsymbol{c}_t^{\mathcal{N}}] + \boldsymbol{b}) \tag{4.12}$$

where $\boldsymbol{z}_{<t}$ denotes the nodes generated before time step t, $\text{ATTN}^{\mathcal{N}}$ is a Luong-style attention (Luong et al., 2015), and $\boldsymbol{m}$ is the memory vectors from information fusion of the encoders (see Section 4.3.2.3).

4.3.2.5 Edge-level Decoder

For the edge decoder, we first use an adjacency-style generation (You et al., 2018). The rows/columns of the adjacency matrix are labeled by the nodes in the order that the node-level decoder has generated them. For each row, we have an auto-regressive decoder that emits each edge's label to other nodes from the edge vocabulary, including a special token [NULL] showing an edge does not exist. As shown in Figure 4.5, we are only interested in the lower triangle part of the matrix, as we assume that the node decoder has generated the nodes in a topologically sorted manner. The dashed upper-triangle part of the adjacency matrix is used only for parallel computation, and it will be discarded.

We use an attentional decoder using GRU units for generating edges. It operates similarly to the node-level decoder using Equation 4.10 and Equation 4.11. For more accurate typed edge generation, however, we incorporate the hidden states of the source and target nodes (from the node decoder) as inputs when updating the hidden state of the edge decoder:

$$\boldsymbol{h}_{i,j}^{\mathcal{E}} = \text{GRU}^{\mathcal{E}}(z_{i,j-1}, \boldsymbol{h}_i^{\mathcal{N}}, \boldsymbol{h}_j^{\mathcal{N}}, \boldsymbol{h}_{i,j-1}^{\mathcal{E}}), \tag{4.13}$$

where $\boldsymbol{h}_{i,j}^{\mathcal{E}}$ is the hidden state of the edge decoder for row i and column j, and $z_{i,j-1}$ is the label of the previously generated edge from node i to $j-1$.

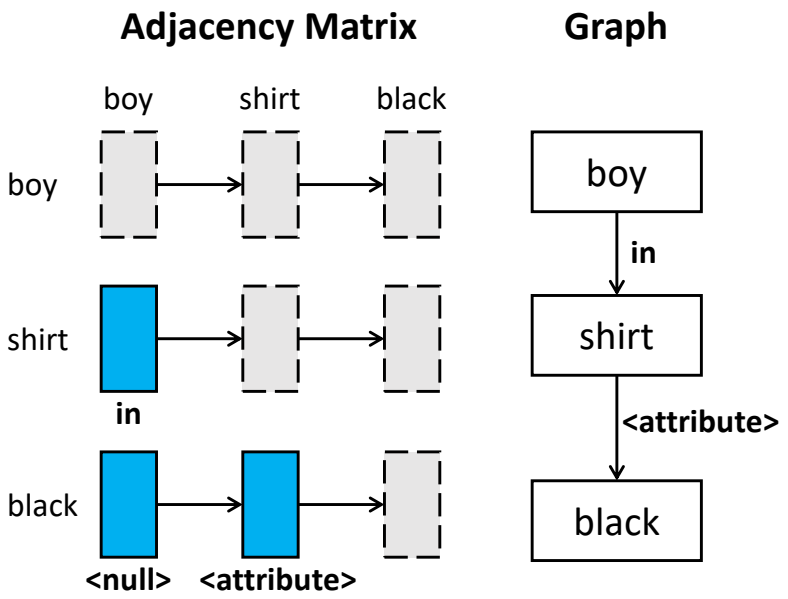


FIGURE 4.5: Adjacency matrix style decoder. We represent a graph via an adjacency matrix. Rows and columns mean the nodes, while the cells represent the edges.

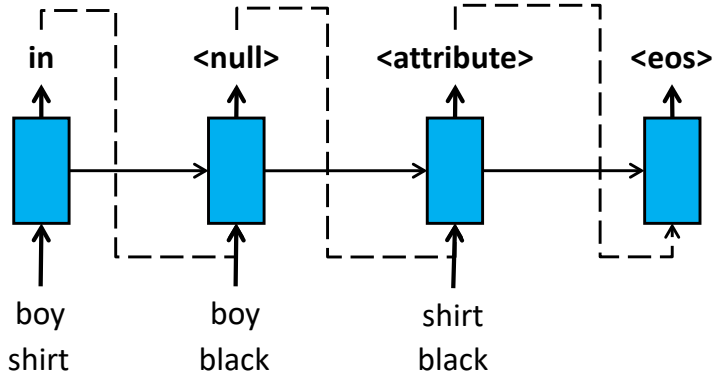


FIGURE 4.6: A flat edge-level decoder. We generate edges linearly. The inputs are node pairs, while the outputs are the associated edges between the nodes.

However, there are two drawbacks to this edge generation method. Firstly, the dummy edges in the adjacency matrix cause waste of computation. Secondly, the edges generated by the previous rows are not conditioned upon when the edges in the next row are generated. However, it may be beneficial to use the information about the outgoing edges of the previous nodes to enhance the generation accuracy of the outgoing edges of the next node. We will analyze this hypothesis in Section 4.4.1. Hence, we suggest flattening the lower triangle of the adjacency matrix. We remove the dummy edges and concatenate the rows of the lower triangular matrix to form a sequence of pairs of nodes for which we need to generate edges (see Figure 4.6). This strategy results in using

information about *all* previously generated edges when a new edge is generated.

4.4 Experiments

Dataset We report the statistics of the synthetic data and crowdsourced data in Table 4.2. The graph size distributions of source and target graphs are almost identical among the sets. With the increase in text description length, the source graphs become more complicated accordingly. In addition, according to the average number of involved nodes and edges when editing the source graphs, GCC data demands more editions than the other two, which suggests that GCC is much more complicated.

TABLE 4.2: Statistics of the created datasets. #editing nodes indicates the number of involved nodes during the graph editing, whereas #editing edges means the number of involved edges.

	Synthetic (Train/Dev/Test)		Crowsourced
	MSCOCO	GCC	Train/Dev/Test
size	196k / 2k / 2k	400k / 7k / 7k	30k / 1k / 1k
Ave. #tokens / text desc	5.2 / 5.2 / 5.2	10.1 / 10.1 / 10.2	4.8 / 4.8 / 4.8
Ave. #nodes / src graph	2.9 / 2.9 / 2.9	3.8 / 3.8 / 3.8	2.0 / 2.0 / 2.0
Ave. #edges / src graph	1.9 / 1.9 / 1.9	2.9 / 2.8 / 2.8	1.0 / 1.0 / 1.0
Ave. #nodes / tgt graph	2.9 / 2.9 / 2.8	3.8 / 3.8 / 3.8	2.0 / 2.0 / 2.0
Ave. #edges / tgt graph	1.9 / 1.9 / 1.8	2.9 / 2.8 / 2.8	1.0 / 1.0 / 1.0
Ave. #tokens / src query	4.7 / 4.8 / 4.7	4.9 / 4.8 / 4.9	10.1 / 10.2 / 10.0
Ave. #editing nodes	0.6 / 0.6 / 0.6	0.8 / 0.8 / 0.8	0.7 / 0.7 / 0.7
Ave. #editing edges	0.7 / 0.6 / 0.7	1.0 / 0.9 / 0.9	0.7 / 0.7 / 0.7

Baselines. We consider five baselines for comparison. In "Copy Source" baseline (i), the system copies the source graph to the target graph[2]. In the "Text2Text" baseline (ii), we linearize the graph and reconstruct the natural sentence similarly to the modification query. In the "Modified GraphRNN" baseline (iii), we use the breadth-first-search (BFS) based node order to flatten the graph[3], and use RNNs as the encoders (You et al., 2018) and a decoder similar to our systems. In the final two baselines, "Graph Transformer" (iv) and "Deep Convolutional Graph Networks" (DCGCN) (v), we use the Graph Transformers (Cai and Lam, 2019) and Deep Convolutional Graph Networks (Guo et al., 2019) to encode the source graph (the decoder is identical to ours).

[2]It is based on the observation that the user only modifies a small portion of the source graph.
[3]The topological ties are broken by order of the nodes appearing in the original query.

Our Model Configurations. We report the results of different configurations of our model. The "Fully Connected Transformer" uses dense connections for the graph encoder. This is in contrast to "Sparse Transformer", which uses the connectivity structure of the source graph in self attention (see Section 4.3.2.1). The information from the graph and query encoders can be combined by "Concatenation", late fused by "Gating", or early fused by "Cross Attention" (see Section 4.3.2.3). The "Adjacency Matrix" style for edge decoding can be replaced with "Flat-Edge" generation (see Section 4.3.2.5).

Training Details Our encoder is comprised of 3 stacked sparse transformers, with 4 heads at each layer. The embedding size is 256, and the inner-layer of feed-forward networks has a dimension of 512. Both node-level and edge-level decoders are one-layer GRU-RNN with a hidden size of 256, and the size of embeddings is 256 as well. We train 30 epochs and 300 epochs for synthetic and user-generated data respectively, with a batch size of 256. We evaluate the model over the dev set every epoch, and choose the checkpoint with the best graph accuracy for the inference.

Evaluation Metrics. We use two automatic metrics for the evaluation. Firstly, we calculate the precision/recall/F1-score of the generated nodes and edges. Secondly, we use the strict-match accuracy, which requires the generated graph to be identical to the target graph for a correct prediction.

4.4.1 Experimental Results

Table 4.3 reports the results of our model and the baselines on the synthetic and user-generated datasets. From the experimental results, various configurations of our model are superior to the baselines by a significant margin. Noticeably, DCGCN and graph transformer are strong baselines, delivering SOTA performance across tasks such as AMR-to-text generation and syntax-based neural machine translation (Guo et al., 2019; Cai and Lam, 2019). We believe the larger number of edge types in our task impairs their capability.

We ablate the different components of the proposed methods to appraise their effectiveness (*c.f.,* the bottom pane of Table 4.3). First, our hypothesis about the preference

TABLE 4.3: Node-level, edge-level and graph-level matching score (%) over two datasets (modified from MSCOCO). "*" indicates statistically significant difference (p <0.0001) from the best baseline.

	Synthetic Data			User-Generated Data		
	Edge F1	Node F1	Graph Acc	Edge F1	Node F1	Graph Acc
Baselines						
Copy Source	64.62	78.41	-	31.42	66.17	-
Text2Text	72.74	91.47	64.42	52.68	78.59	52.15
Modified GraphRNN (You et al., 2018)	55.76	80.64	50.72	57.17	80.68	56.75
Graph Transformer (Cai and Lam, 2019)	75.68	91.21	71.38	59.43	81.47	58.23
DCGCN (Guo et al., 2019)	72.47	89.08	68.89	54.23	79.05	52.67
Our Models						
Fully Conn Trans + Adj Matrix + Concat	76.49*	91.54	72.13*	57.47	81.29	56.91
Sparse Trans + Adj Matrix + Concat	77.94*	91.94	74.68*	57.78	81.36	56.98
Sparse Trans + Flat-Edge + Concat	79.13*	92.11	76.13*	57.92	81.74	57.03
Sparse Trans + Flat-Edge + Gating	80.13*	92.54*	77.04*	59.58*	82.39*	59.63*
Sparse Trans + Flat-Edge + Cross-Attn	**86.52***	**95.40***	**82.97***	**62.10***	**83.69***	**60.90***

for flat-edge generation over adjacency matrix-style edge generation is confirmed. Furthermore, the two-way communication between the graph and query encoders through the gating mechanism consistently outperforms a simple concatenation in terms of both edge-level and node-level generation. Eventually, the cross-attention – the early fusion mechanism, leads to substantial improvement in all metrics.

We also observe that generating the graphs for the crowdsourced data is much harder than the synthetic data, which we believe is caused by the annotators' diversity in semantics and expressions. Consequently, all models suffer from performance degradation. Nevertheless, the performance trends of different configurations of our model are almost identical on the user-generated and synthetic data.

In addition, Figure 4.7 presents the breakdown performance of our best model for node-level F1 scores based on the frequency of nodes. Similar to other long-tail problems (Zhang et al., 2021), the performance gradually increases with the increase in node frequency.

Finally, Table 4.4 indicates that with the increase of the complexity of graphs, the models have difficulty in inferring the relations among nodes for GCC data, which causes a dramatic drop in terms of the edge F1 score and graph accuracy.

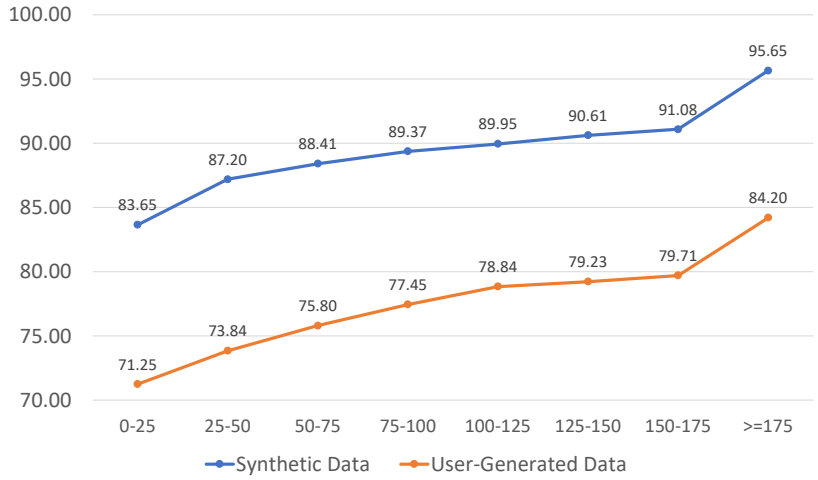


FIGURE 4.7: Node-level F1 scores of different node frequency groups on synthetic data and user-generated data. X axis indicates the frequency group, where Y axis is F1 score.

TABLE 4.4: Node/Edge/Graph level matching scores comparing the best baseline - Graph Transformer to our model variants on synthetic MSCOCO and GCC.

	MSCOCO			GCC		
	Edge F1	Node F1	Graph Acc	Edge F1	Node F1	Graph Acc
Graph Trans.	75.68	91.21	71.38	42.76	82.38	34.31
Concat	79.13	92.11	76.13	45.09	86.93	37.53
Gating	80.13	92.54	77.04	52.85	91.60	45.79
Cross-Attn	86.52	95.40	82.97	**57.68**	**93.84**	**52.50**

4.4.2 Quantitative Analysis

The best configuration of our model is based on cross-attention, with a flat-edge decoder, and sparse Transformer. We investigate which cases in this configuration outperforms the baselines. As seen in Figure 4.8, cross-attention is able to understand the *pronoun* and correctly removes the connected object and its associated relation as evidenced by the first example **A**. In addition, example **B** demonstrates that when the graph transformer observes a longer description, it lacks the capability of fusing the semantics between the source graph and the modification query; then, certain nodes from the source graph are not preserved. We believe that the proposed approach can reduce the noise in graph generation, and retain fine-grained details better than the baselines.

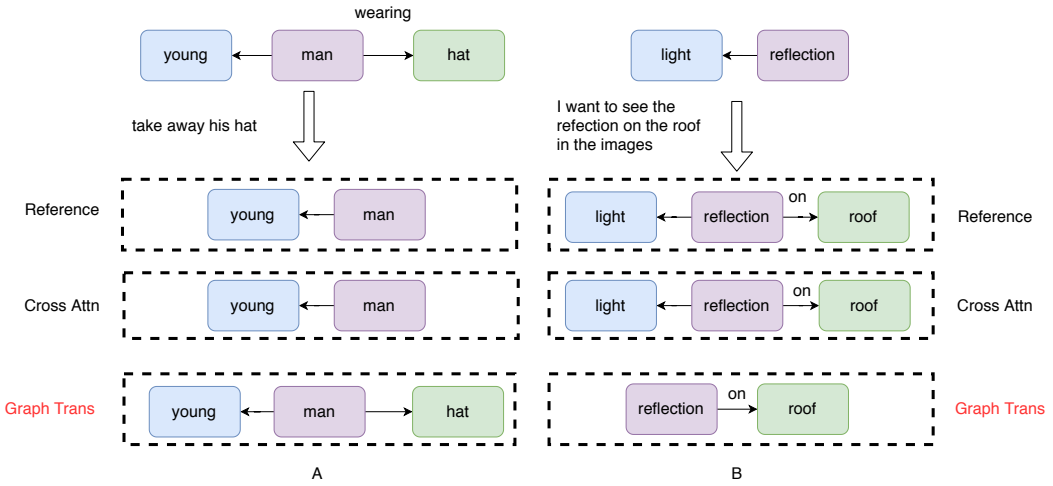


FIGURE 4.8: Our best model *v.s.* the Graph Transformer on two modification examples.

4.5 Summary

In this chapter, we explore a novel problem of conditional graph modification, in which a system needs to understand the context, *i.e.,* a source graph and a modification command, to modify the source graph. Since there is no available dataset, we craft two synthetic datasets and one crowdsourced data for this novel task. Then we devise an innovative architecture to conduct this task. Finally, our best system, based on graph-conditioned transformers and cross-attention information fusion, outperforms strong baselines adapted from machine translations and graph generations.

Chapter 5

Context-dependent Semantic Parsing

This chapter is based on:

Xuanli He, Quan Tran, and Gholamreza Haffari. 2019. A Pointer Network Architecture for Context-Dependent Semantic Parsing. In Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association, pages 94–99, Sydney, Australia. Australasian Language Technology Association.

In the previous chapter, we initiated our exploration of long-range context-dependent NLU through the lens of scene graphs. We have illustrated that one can utilize the preceding context to adjust the components of a scene graph such that the changing intent of users can be dynamically modeled via an interaction.

The accomplishment achieved in the previous chapter corroborates the necessity of modeling a long-range context in a dialogue-driven NLU. However, since a dialogue system involves multi-turn instead of single-turn conversation, one has to capture a much broader context; otherwise, a misunderstanding can be incurred, as shown in Table 5.1.

To explore the significance of the longer context in NLU, this chapter focus on a dialogue-based semantic parsing task. In this task, one should consider the multi-sentence context dependency when parsing an utterance into the corresponding logical form. Moreover, since the arguments of the logical form are substrings of the utterance, the parser has

TABLE 5.1: An error incurred by the dialogue-driven parsing system without considering the long-range context.

dialog history ... *user*: compose a new email. the recipient is mom. *user*: the subject is hello *user*: update mom to "mom and dad" ...
current utterance: update mom to "mom and dad"
logical form reference: (setFieldFromString (getProbMutableFieldByFieldName recipient _ list) (stringValue " mom and dad ")) system: (unknown)

to identify and place them in the argument slots. A parser without such capability will fail to produce a proper logical form. To address these issues, we propose a novel architecture to capture the contextual information and alleviate the argument fill-in errors via a copying mechanism.

5.1 Introduction

Recently, due to the breakthrough of deep learning, numerous and various tasks within the field of natural language processing (NLP) have made impressive achievements (Vaswani et al., 2017; Devlin et al., 2018; Edunov et al., 2018). However, most of these achievements are assessed by automatic metrics, which are relatively superficial and brittle, and can be easily tricked (Paulus et al., 2017; Jia and Liang, 2017; Läubli et al., 2018). Hence, understanding the underlying meaning of natural language sentences is crucial to NLP tasks.

As an appealing direction in natural language understanding, semantic parsing has been widely studied in the NLP community (Ling et al., 2016; Dong and Lapata, 2016; Jia and Liang, 2017). Semantic parsing aims at converting human utterances to machine-executable representations. Most existing work focuses on parsing individual utterances independently, and even they have access to contextual information. In spite of several pioneering efforts (Zettlemoyer and Collins, 2009; Srivastava et al., 2017), these pre-neural models suffer from complicated hand-crafted feature engineering, compared to their neural counterparts (Rabinovich et al., 2017; Dong and Lapata, 2018). One notable

exception is the work of Suhr et al. (2018), who incorporates context into ATIS data with a neural approach.

In this work, we propose a neural semantic parser for the email assistant task. As shown in Table 5.2, in order to resolve this task, we have to incorporates the conversation context and a copying mechanism to fill in the arguments of the logical forms from the input sentence. Our model achieves state-of-the-art (SOTA) performance. We further provide a details analysis of where these improvements come from.

TABLE 5.2: Example of a real-world interaction between a human (User) and an automated email assistant (Agent)

User:	Create contact for mom
Agent:	(createInstanceByFullNames contact (stringNoun " mom "))
User:	mom 's email is momthebest7 @ bestforyou.com
Agent:	(setFieldFromString (getProbMutableFieldByInstanceNameAndFieldName mom email) (stringValue " momthebest7 @ bestforyou.com "))
User:	create new outgoing mail
Agent:	(createInstanceByConceptName outgoing _ email)
User:	set the recipient to mom 's email
Agent:	(setFieldFromFieldVal (getProbMutableFieldByFieldName recipient _ list) (evalField (getProbFieldByInstanceNameAndFieldName mom email)))
User:	send email to mom
Agent:	(send email)

In summary, the contributions of this chapter are as follows:

- We propose a novel and efficient approach to modeling the historical context of a semantic parsing task. This avenue significantly boosts the performance by 1.8 scores using contextual information.
- We incorporate a copying mechanism to alleviate the argument fill-in errors incurred by misalignment between source sentences and logical forms, leading to a gain of 2.3 scores in the best case.
- We show that one can superimpose the copying mechanism on the context-dependent design to advance the performance by 1.0 scores on RNNs and Transformer.
- We quantitatively study the error reduction brought by the copying mechanism and context-dependent modeling, respectively.

5.2 Models

To build our models, we follow a process of error-driven design. We first start with a simple SEQ2SEQ model, then we closely examine the errors, group them, and then propose a solution to each of these error groups. Our examination identifies two primary sources of errors in a SEQ2SEQ model: i) the overly strong influence of the language model component and ii) the lack of contextual information. Thus we design our model to incorporate the *Pointer Mechanism* and *Context-dependent Mechanism* to solve these problems. From this point, we refer to the errors caused by the first source (language model) as *Copy-related errors*, and the ones caused by the second source (lack of context) as *Context-related errors*.

5.2.1 Word Copy using the Pointer Mechanism

With the basic SEQ2SEQ architecture, the model's generation is heavily influenced by the language model aspect. Thus, it tends to use the strings it has seen in the training dataset (see Table 5.3).

TABLE 5.3: An error made by the base SEQ2SEQ model. Copy mechanism can fix it.

current utterance:
set body to blue
logical form
reference: (setFieldFromString (getProbMutableFieldByFieldName body) (stringValue " blue "))
seq2seq: (setFieldFromString (getProbMutableFieldByFieldName body) (stringValue " charlie is on his way "))

From this analysis, we realize that it would be crucial for the model to learn when to copy from the source sentence, and when to generate a new token. Thus, we incorporate the pointer mechanism into our base SEQ2SEQ approach.

As shown in Figure 5.1, for an email assistant system, users inputs are usually comprised of a functional part and a content part. A semantic parser should be able to distinguish and handle them differently. Specifically, the parser must generate a series of lambda-like functions for the functional part, while the content part should be copied to the argument slot.

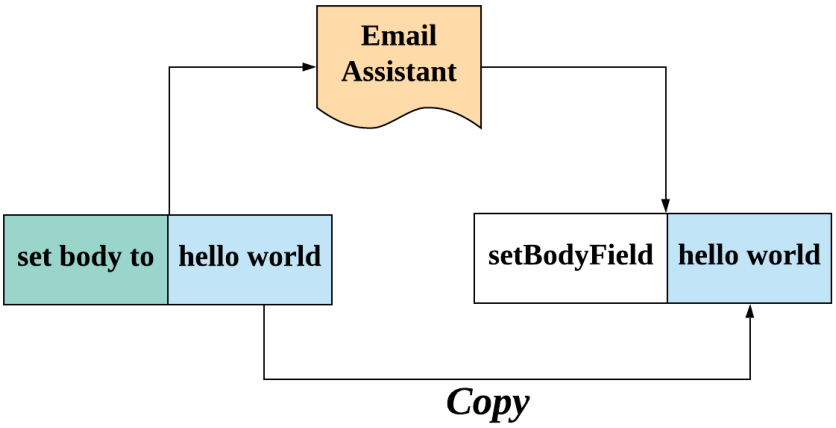


FIGURE 5.1: A example of semantic parsing on the email assistant system.

Our pointer network is inspired by that of See et al. (2017) designed for the summarization task. Given an utterance $\boldsymbol{x}$ and a logical form $\boldsymbol{y}$, at each time step t, we have a *soft* switch which determines the contributions of the token generator and the copier which uses a pointer over the words of the input utterance:

$$P(\boldsymbol{y}_t) = p_{gen} P_{vocab}(\boldsymbol{y}_t) + (1 - p_{gen}) \sum_{i:\boldsymbol{x}_i = \boldsymbol{y}_t} \alpha_i^t$$

where α_i^t is the attention score over the position i in the t-th generation step, and P_{vocab} is a probability distribution over the vocabulary. $p_{gen} \in [0, 1]$ is the *generation probability*, modeled as:

$$p_{gen} = \sigma(\boldsymbol{w}_c^T \boldsymbol{c}_t + \boldsymbol{w}_s^T \boldsymbol{s}_t + \boldsymbol{w}_x^T \boldsymbol{x}_t + b)$$

where $\boldsymbol{c}_t$ and $\boldsymbol{s}_t$ are the context vector and the decoder state respectively (*c.f.,* Equation 2.3 and 2.4), while $\boldsymbol{w}_c^T$, $\boldsymbol{w}_s^T$, $\boldsymbol{w}_x^T$ and b are learnable parameters.

5.2.2 Conditioning on Conversation Context

Understanding conversations between a user and the system requires the comprehension of the flow of the discourse among a sequence of utterances. Processing utterances independently within a conversation leads to misinterpreting users inputs, which will result in incorrect logical form generation (see Table 5.4). Therefore, we incorporate the context when processing the current utterance for a better generation.

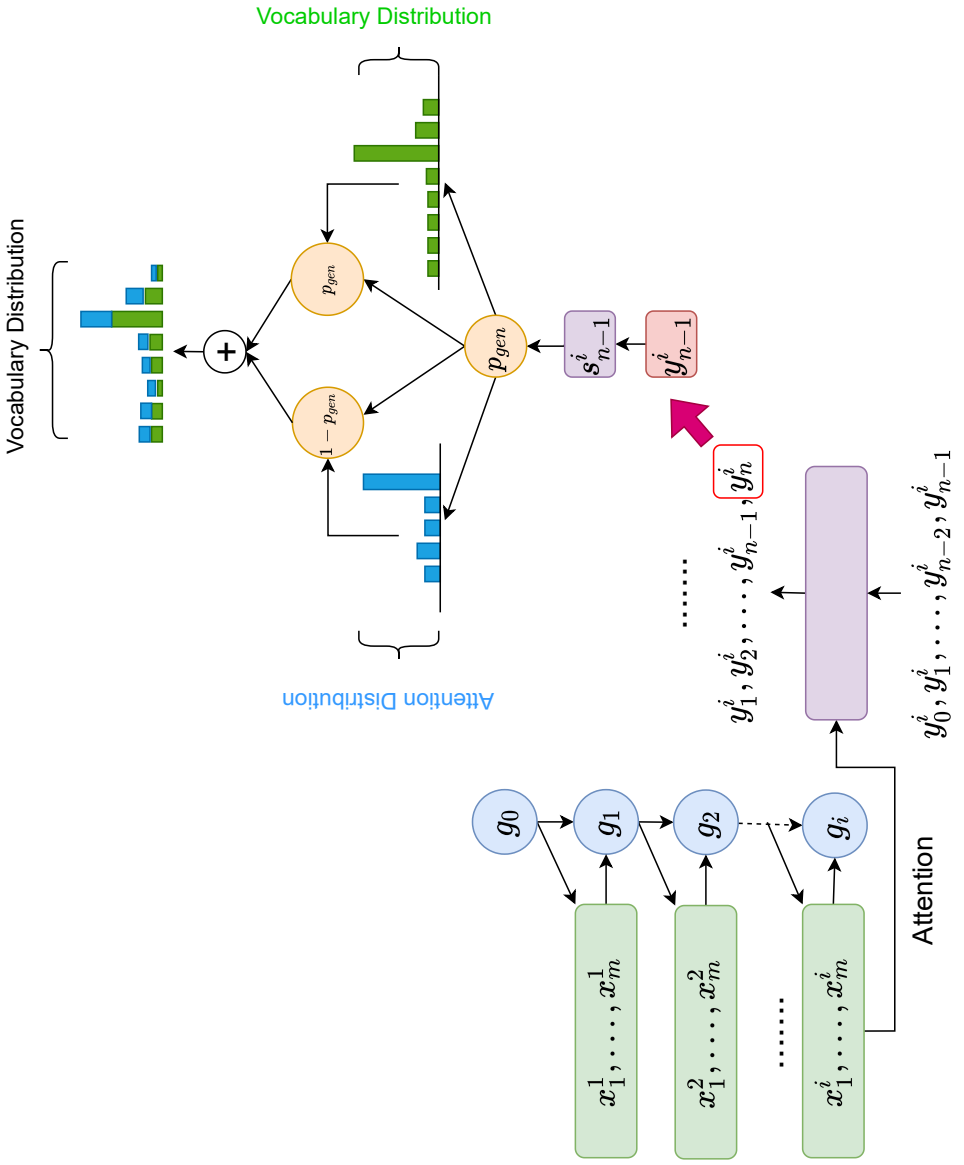


FIGURE 5.2: Overall architecture of our semantic parser with the copying mechanism.

TABLE 5.4: An error made by the base SEQ2SEQ model. It is clear that without the context information, the model cannot infer the correct logical form.

dialog history
...
user: compose a new email. the recipient is mom. the subject is hello
user: cancel
...
current utterance:
cancel
logical form
reference: (undo)
seq2seq: (cancel)

Basically, a conversation consists of a sequence of user utterances:$\{\boldsymbol{x}^1, ..., \boldsymbol{x}^T\}$ paired with a list of logical forms: $\{\boldsymbol{y}^1, ..., \boldsymbol{y}^T\}$. For a given utterance sequence $\boldsymbol{x}^i = \{\boldsymbol{x}^i_1, ..., \boldsymbol{x}^i_m\}$, a semantic parser should predict its associated logical form $\boldsymbol{y}^i = \{\boldsymbol{y}^i_1, ..., \boldsymbol{y}^i_n\}$. Inspired by Suhr et al. (2018), we introduce a hierarchical architecture to model both utterance-level and conversation-level information (see Figure 5.2). At the utterance level, we use an attentional SEQ2SEQ model to establish the mapping from an utterance $\boldsymbol{x}^i$ to its corresponding logical form $\boldsymbol{y}^i$:

$$\boldsymbol{h}^i_{1:m} = \text{Encoder}(\boldsymbol{x}^i_1, ..., \boldsymbol{x}^i_m), \tag{5.1}$$

$$\boldsymbol{c}^i_t = \text{Attention}(\boldsymbol{h}^i_{1:m}, \boldsymbol{s}^i_{t-1}), \tag{5.2}$$

$$\boldsymbol{y}^i_t, \boldsymbol{s}^i_t = \text{Decoder}(\boldsymbol{y}^i_{t-1}, \boldsymbol{s}^i_{t-1}, \boldsymbol{c}^i_t) \tag{5.3}$$

As the SEQ2SEQ model, we investigate the use of RNN-based and Transformer-based architectures. Furthermore, we make use of a conversation-level RNN to capture the wider conversational context:

$$\boldsymbol{g}_i = \text{RNN}(\boldsymbol{h}^i_m, \boldsymbol{g}_{i-1}) \tag{5.4}$$

where $\boldsymbol{h}^i_m$ is the last hidden state of the ith utterance, and $\boldsymbol{g}$ is the conversational hidden state. In order to incorporate the conversational information into our model, we modify the Equation 5.1 by injecting $\boldsymbol{g}_{i-1}$:

$$\boldsymbol{h}^i_{1:m} = \text{Encoder}([\boldsymbol{x}^i_1 : \boldsymbol{g}_{i-1}], ..., [\boldsymbol{x}^i_m : \boldsymbol{g}_{i-1}])$$

where [:] denotes a concatenation operation.

Similar to memory networks (Sukhbaatar et al., 2015), it is essential to give the decoder a direct access to the last k utterances, if we want to leverage the discourse information effectively. Hence, we concatenate the previous k utterance $\{\boldsymbol{x}_{i-k}, .., \boldsymbol{x}_{i-1}\}$ with the current utterance. Now Equation 5.2 is rewritten as:

$$\boldsymbol{c}_t^i = \text{Attention}(\boldsymbol{h}_{1:m}^{i-k}, .., \boldsymbol{h}_{1:m}^{i-1}, \boldsymbol{h}_{1:m}^{i}, \boldsymbol{s}_{t-1}^{i})$$

In addition, since the importance of the concatenated utterances is different, it is significant to differentiate these utterances to reduce confusion. Therefore, as suggested by Suhr et al. (2018), we add relative position embeddings $\text{E}_{pos}[\cdot]$ to the utterances when we compute attention scores. Depending on their distances from the current utterance, we append $\text{E}_{pos}[0], .., \text{E}_{pos}[k]$ to the previous utterances respectively.

5.3 Experiments

Dataset Semantic paring is crucial to dialogue systems, especially for multi-turn conversations. Additionally, understanding users' intentions and extracting salient requirements play an essential role in dialogue-related semantic parsing. We use a dataset created by Srivastava et al. (2017) as a case study to explore the performance of semantic parsing in dialogue systems. This dataset is collected from an email assistant, which can help users manage their emails. As shown in Table 5.5, users can type some human sentences from the interface. Then the email assistant can automatically convert the natural sentences to machine-understandable logical forms.

TABLE 5.5: A partial conversation from the data.

dialog history
...
user: Define the concept " contact "
user: add field " email " to concept " contact "
user: create contact " Mom "
...
logical form
...
(defineConcept (stringNoun " contact "))
(addFieldToConcept contact (stringNoun " email "))
(createInstanceByFullNames contact (stringNoun " mom "))
...

Following Srivastava et al. (2017), we partition the dataset into a training fold (93 conversations) and a test fold (20 conversations) as well. However, this partition might be different from Srivastava et al. (2017), as they only release the raw Email Assistant dataset. The total number of user utterances is 4759, the number of sessions is 113, and the mean/max of the number of utterances per interactive session is 42/273.

5.3.1 Main Results

Prior to this work, Srivastava et al. (2017) also incorporate the conversational context into a CCG parser (Zettlemoyer and Collins, 2007). CCG requires extensive hand-feature engineering to construct text-based features. However, neural semantic parsers have been demonstrating impressive improvement over various and numerous dataset (Suhr et al., 2018; Dong and Lapata, 2018). Hence, we explore both RNN-based (Bahdanau et al., 2014) and Transformer-based (Vaswani et al., 2017) architectures for our attentional SEQ2SEQ model, denoted as *RNNS2S* and *Transformer* respectively.

In the RNNS2S model, at the utterance level, a one-layer bidirectional RNNs are for the encoder, while the decoder is a two-layer RNNs. We use a one-layer RNNs to represent the conversational information flow. All RNNs use LSTM cells, with a hidden size of 128. The dimensions of word embeddings and position embeddings are 128 and 50 respectively. We train our models for 10 epochs by Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001. The batch size of non-context training is 16, while the context variant is 1.

For the Transformer model, we use 3 identical transformer blocks for both encoder and decoder. Within each block, the size of the embeddings is 256, while the feed-forward network has 512 neurons. We set the size of heads to 4. The conversational encoder is a one-layer RNNs with a size of 256. The optimizer and training schedule is same as Vaswani et al. (2017), except $warmup\_steps = 500$. Due to the warmup steps, We train this model for 14 epochs. The batch size is the same as that of RNNS2S.

Unless otherwise mentioned, we use 3 previous utterances as the history. Since there is no validation set, we use 10-fold cross-validation over the training set to find the best parameters.

TABLE 5.6: Test accuracy on Email Assistant dataset. **Bold** indicates the best result. SPCon is the best CCG parser with contextual information in Srivastava et al. (2017)

	Accuracy
Previous methods	
Seq2seq Srivastava et al. (2017)	52.3
SPCon Srivastava et al. (2017)	62.3
Our models	
RNNS2S	68.0
RNNS2S + pointer	69.3
RNNS2S + context	69.8
RNNS2S + context + pointer	70.5
Transformer	69.3
Transformer + pointer	72.2
Transformer + context	71.0
Transformer + context + pointer	**73.4**

Table 5.6 demonstrates the accuracy of different models. Our RNNS2S baseline already surpasses the previous SOTA result with a large margin. However, since we use our partition, this comparison should not be a reference. Both pointer network and conversational architecture dramatically advance the accuracy. Besides, the RNNS2S baseline lags behind our approach by 2.5 scores when combining these two techniques. Finally, the Transformer model also benefits from this synergy and obtains an accuracy of 73.4, marking a new SOTA result on this dataset.

5.3.2 Analysis

In this section, we provide some deep analysis on our models, including the utility of the copying mechanism and the context-dependent mechanism.

(A) RNNS2S (B) Transformer

FIGURE 5.3: Number of copy-related incorrect instances that can be corrected by a pointer network on RNNS2S (left) and Transformer (right).

The effects of the copying mechanism We analyze the test data, and count the number of errors that can be rectified by introducing the pointer network for both vanilla and context-dependent SEQ2SEQ models. According to Figure 5.3, our pointer network fixes at least half of the incorrect instances. Clearly, the pointer mechanism cannot solve all copy-related errors. After scrutinizing the system-generated results, we realize that the pointer network tends to retain the copy mode once it is triggered. This phenomenon is consistent with the observations by See et al. (2017). Consequently, the extra copies impinge on the accuracy of the system.

TABLE 5.7: An example of complex and compositional commands.

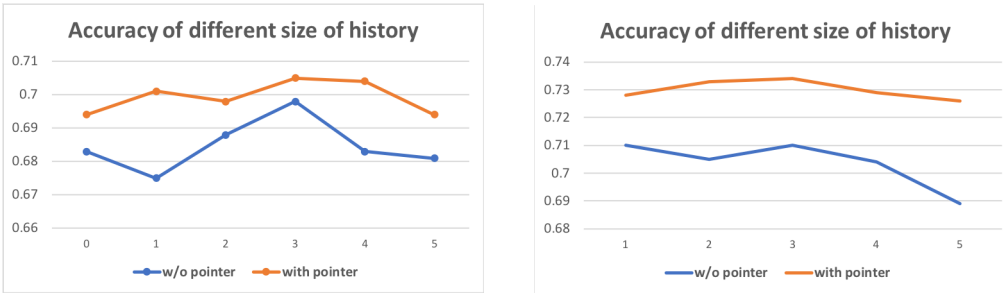
utterance: Set recipient to Mom's email . Set subject to hello and send the email
logical form: (doSeq (setFieldFromFieldVal (getProbMutableFieldByFieldName body) (evalField (getProbFieldByInstanceNameAndFieldName inbox body))) (doSeq (setFieldFromFieldVal (getProbMutableFieldByFieldName recipient_list) (evalField (getProbFieldByInstanceNameAndFieldName inbox sender))) (send email)))

The effects of the context-dependent mechanism. In the experiments, our context-dependent mechanism is shown to be able to address context-related errors, especially when the user's input implies a complex and compositional command. These complex commands usually involve a series of complicated actions, as shown in Table 5.7. According to Table 5.8, our context-dependent model rectifies half of the context-related errors.

TABLE 5.8: Incorrect instances of RNNS2S, context-dependent RNNS2S, Transformer and context-dependent Transformer models in terms of complex commands and context dependency.

	#incorrect		#incorrect
complex command			
RNNS2S	39	Transformer	35
RNNS2S + context	20	Transformer + context	24
context dependency			
RNNS2S	29	Transformer	25
RNNS2S + context	21	Transformer + context	16

Since we notice that previous utterances can also obfuscate the model, we conduct an ablation study over the size of history. As shown in Figure 5.4, overall, incorporating 3 previous utterances reach the best performance. According to Figure 5.5, we believe that



(A) RNNS2S (B) Transformer

FIGURE 5.4: Accuracy of different size of historical utterances for RNNS2S (left) and Transformer (right). X axis is the accuracy, and Y axis is the number of the historical utterances.

incorporating 3 previous utterances covers sufficient contextual information fo RNNS2S. Less than this number, the system cannot better utilize context, while the salient information is contaminated by the extra history. The same behavior is observed in the Transformer model. We argue that the effective history size would depend on different datasets, but they will demonstrate the same trend.

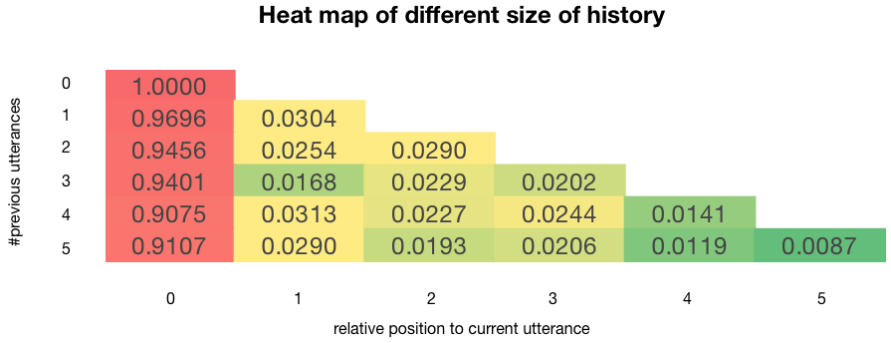


FIGURE 5.5: Attention scores of different size of history on RNNS2S. X axis indicates a relative position of the studied utterance to the current one. Y axis refer to the number of historical utterances.

5.4 Summary

In this chapter, we explore a neural semantic parser architecture that incorporates conversational context and copying mechanisms. These modeling improvements are solidly grounded by our analysis, and they significantly boost the performance of the base model. As a result, our best architecture establishes a new state-of-the-art performance on the Email Assistant dataset.

Part III

Modeling Task-specific Context

Chapter 6

Advancing Text Classification by Modeling Task-specific Context

This chapter is based on:

He, Xuanli, Gholamreza Haffari, and Mohammad Norouzi. "Generate, Annotate, and Learn: NLP with Synthetic Text", Accepted to Transactions of the Association for Computational Linguistics, 2022

In Part I and II, we have highlighted the significance of context from the perspective of short- and long-range context, which explicitly embeds the contextual information in the surrounding words or sentences. However, the definition of context can be implicit or abstract. For instance, Michael Jordan tends to be a reference to the computer scientist when the text topic relates to machine learning. In contrast, basketball player "Michael Jordan" is likely to be more relevant in the sports domain. In addition, advanced words should be used in formal writing, while simple words frequently appear in daily conversation. To conclude, the structure, genre, and meaning of sentences are all affected by an abstract context, denoted by the meta (or task-specific) context. Thus, this Part is geared toward the exploration and application of the task-specific context.

In addition to the methodology, we also attribute the breakthrough of deep learning to the availability of plentiful data. Indeed, we can rival human parity to some extent in tasks where the in-domain data is abundant (Hassan et al., 2018; Liu et al., 2019;

Lan et al., 2020). Nevertheless, we still face data scarcity for many tasks, which severely hinders the progress of these tasks. As GPT-family models are able to synthesize human-like data, we advocate for the use of these models to synthesize task-specific data.

In relation to the capability of data synthesization, since GPT models are trained on diverse data, the generation space is more comprehensive than we demand. Therefore, we should steer the vanilla models toward generating task-specific data. To fulfill this requirement, we first tailor the generation space by fine-tuning or conditioning GPT models on in-distribution examples. Then we leverage these tailored generative models to synthesize unlabeled task-specific text. Next, we use state-of-the-art classifiers to annotate the generated data with pseudo labels. Finally, we combine the labeled data with the pseudo-labeled data to train more effective classifiers or for the purpose of knowledge distillation and few-shot learning.

6.1 Introduction

There is an abundance of unlabeled data in the real world, but task-specific unlabeled data within the scope of a given machine learning problem can be challenging to find. For instance, one cannot easily find in-domain unlabeled text conforming to the input distribution of a specific Natural Language Processing (NLP) task from the GLUE benchmark (Wang et al., 2019b). Some NLP tasks require an input comprising a pair of sentences with a particular relationship between them. Moreover, classification datasets typically represent a tailored distribution of data and only include a limited number of class labels. We denote this tailored distribution as a task-specific context. If task-specific unlabeled data were available, one could adopt self-training (Yarowsky, 1995) to automatically annotate unlabeled data with pseudo labels to improve the accuracy and robustness of classifiers (Carmon et al., 2019; Xie et al., 2020; Bari et al., 2021). In addition, one can use knowledge distillation (Hinton et al., 2015) on fresh task-specific unlabeled data to more effectively compress deep neural networks and ensembles (Buciluǎ et al., 2006; Chen et al., 2020).

In the absence of task-specific unlabeled data, one could *retrieve* unlabeled examples from a large and diverse open-domain dataset (Du et al., 2021). However, such a retrieval-based approach may not scale to problems with complex input schemes,

e.g., sentence pairs with certain relations. Recent work (Yang et al., 2020; Kumar et al., 2020) has considered the use of Language Models (LMs) like GPT-2 (Radford et al., 2019) as a means of data augmentation, showing the effectiveness of this approach for commonsense reasoning and classification tasks. Existing approaches often consider *class-conditional* generation, where the synthetic data is produced by conditioning on a specified class label. However, it is unclear whether the class-conditional generation is best suited for NLP tasks. Furthermore, existing pipelines often make synthetic data generation complicated as one needs to detect and discard low-quality synthetic *labeled* data or optionally re-label data (Yang et al., 2020; Vu et al., 2021). For instance, Kumar et al. (2020) observes that it is difficult for sentences generated by label-conditioned GPT-2 to retain the semantics/pragmatics of the conditioning label, leading to poor performance on downstream tasks.

We unify and simplify existing work on LMs as a data source for NLP and develop a general framework called "generate, annotate, and learn (GAL)". The generality of GAL allows us to use LM-generated synthetic data within novel applications such as Knowledge Distillation (KD) and few-shot learning. GAL builds on recent advances in text generation (Radford et al., 2019; Gao et al., 2021) and uses powerful LMs to synthesize task-specific unlabeled text by fine-tuning or conditioning a large LM on in-distribution examples. We use state-of-the-art classifiers to annotate generated text with soft pseudo labels when possible. We then combine labeled data and pseudo-labeled data to train more effective supervised models, resulting in significant gains on a range of NLP tasks like KD and few-shot learning.

We present a justification for GAL based on the empirical and vicinal risk minimization frameworks (Vapnik, 1992; Chapelle et al., 2001). We also investigate key components of GAL. We find that even if class-conditional LMs are available for text generation, it is more effective to discard the conditioning labels and let the teacher models produce pseudo labels. This observation is supported by our theoretical and empirical results. Accordingly, in contrast to prior work (Yang et al., 2020; Vu et al., 2021), we advocate for the use of simple unconditional LMs for text synthesis. Further, we avoid any form of data filtering. Not surprisingly, we find that the diversity of synthetic text matters. That said, simple unconditional generation given random seeds provides sufficient diversity, and crafting diverse LM prompts is not needed.

In summary, Our contributions are as follows:

- We develop GAL, a simple and effective approach to the use of LMs for task-specific unlabeled text generation. We show that GAL can be used effectively for KD, self-training, and few-shot learning in NLP.
- We present theoretical and empirical investigations for GAL, explaining why it works and why using class-conditional LMs to generate synthetic labeled data is not as effective.
- GAL advances KD for NLP and establishes a new SoTA result for a single 6-layer transformer on the GLUE test set. It further improves prompt-based few-shot learning, providing an average improvement of 1.3% on four 4-shot learning NLP tasks, outperforming GPT-3-6B.

6.2 Preliminaries

Self-training As one of the earliest and most successful approaches in semi-supervised learning (Fralick, 1967), recently, there has been a resurgence of interest in self-training (Hendrycks et al., 2020; Sohn et al., 2020; Xie et al., 2020; Du et al., 2021). The core of self-training is to pseudo-label the unlabeled data via a base model. Then one can advance the model from the human- and pseudo-labeled data.

Given a labeled dataset $L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and an unlabeled dataset $U = \{\boldsymbol{x}_j\}_{j=1}^M$, we summarize the procedure of self-training as:

1. First, an initial model denoted f_1 is trained using supervised learning on the labeled dataset L.
2. Then, at iteration t, one adopts f_t as the teacher model to annotate the unlabeled dataset U using *pseudo labels*. Optionally, one uses a selection method to pick a subset $S_t \subseteq \{(\boldsymbol{x}_j, f_t(\boldsymbol{x}_j))\}_{j=1}^M$ of pseudo labeled examples.
3. A student model f_{t+1} is trained to optimize a classification loss on the combination of L and S_t:

$$\ell_{t+1} = \mathbb{E}_{(\boldsymbol{x},y)\sim(L\cup S_t)} H(y, f_{t+1}(\boldsymbol{x})) \ , \tag{6.1}$$

where $H(q, p) = q^\top \log p$ is the softmax cross entropy loss, and y is assumed to be a one-hot vector (original labels) or a vector of class probabilities (*soft* pseudo labels).

4. Self-training iterations are repeated T times or until performance plateaus.

Many different variants of the basic self-training algorithm discussed above exist in the literature. These variants differ in the type of pseudo labels used, the selection strategy to filter pseudo labeled examples, the speed at which f_t is replaced with f_{t+1}, the choice of data augmentation strategy in the teacher and student models, and the weighting of the two datasets in the objective (Berthelot et al., 2019; Xie et al., 2020; Sohn et al., 2020; Du et al., 2021).

An important design choice is the type of pseudo labels used. One can simply use soft class probabilities predicted by a teacher f_t (Du et al., 2021), sharpened class probabilities (Berthelot et al., 2019), or hard labels (a one-hot vector that is zero except at $\text{argmax} f_t(\boldsymbol{x})$) (Lee et al., 2013). Another important consideration is the selection strategy to retain a subset of pseudo-labeled examples. FixMatch (Sohn et al., 2020) uses a hyper-parameter τ to select examples on which the teacher model has a certain level of confidence, *i.e.,*

$$S_t = \{(\boldsymbol{x}, f_t(\boldsymbol{x})) \mid \boldsymbol{x} \in U \;\&\; \max(f_t(\boldsymbol{x})) \geq \tau\} \; . \tag{6.2}$$

NoisyStudent (Xie et al., 2020) also uses a form of confidence filtering but ensures that the class labels in the selected subset are balanced. In principle, any method for out-of-distribution detection (Hendrycks and Gimpel, 2016) can be adopted for filtering pseudo-labeled examples. We adopt the simplest variant of self-training and limit hyper-parameter tuning to a bare minimum.

Knowledge Distillation Like self-training, knowledge distillation (Buciluǎ et al., 2006; Hinton et al., 2015) has been experiencing a renaissance due in part to the emergence of large pre-trained language models (Sanh et al., 2019; Sun et al., 2019a; Xu et al., 2020).

Knowledge distillation aims to compress a large model or an ensemble of models into a compact model without a significant performance degradation. Analogous to self-training, this goal can be achieved by mimicking the predictions of the teacher model

via pseudo-labeling. However, there is no iterative process since the student model is usually inferior to the teacher model.

6.3 Generate, Annotate, and Learn (GAL)

Given a labeled dataset $L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we first train an unconditional domain-specific generative model $g(\boldsymbol{x})$ on $L_{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=1}^N$, and then use it to synthesize unlabeled data. Such synthetic unlabeled data is used within self-training and KD even in the absence of in-domain unlabeled data. We restrict our attention to basic KD and self-training methods, even though GAL can be combined with more sophisticated semi-supervised techniques too.

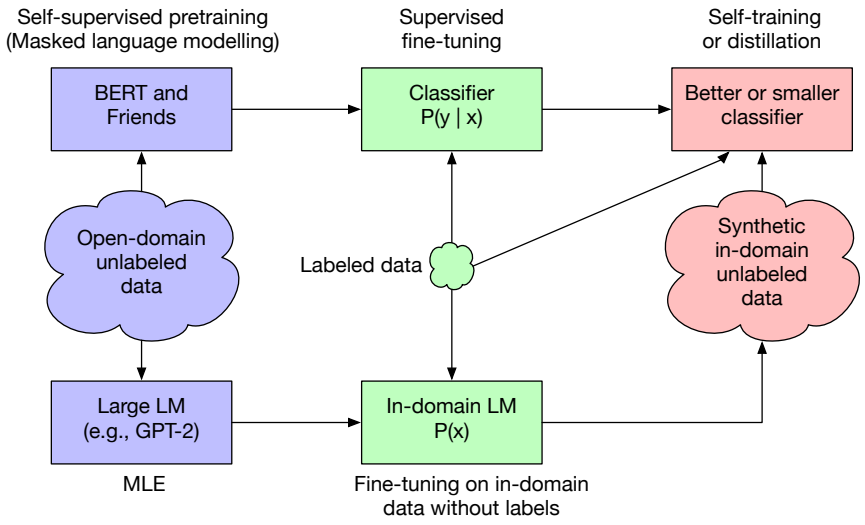


FIGURE 6.1: An illustration of GAL for NLP. We use open-domain data once for self-supervised pretraining (*e.g.,* BERT) and once for training a large LM (*e.g.,* GPT-2). BERT is fine-tuned on labeled data to yield a classifier for the task of interest. GPT-2 is fine-tuned on the same data without labels to obtain an unconditional task-specific LM, which is used to generate lots of synthetic in-domain unlabeled data for self-training and KD.

The effectiveness of GAL depends on the fidelity and diversity of synthetic examples. If we had access to the oracle generative process, we were able to obtain the best KD and self-training results, as if we had access to real task-specific unlabeled data. Our preliminary experiments suggest that large language models are particularly effective within the GAL framework. Hence, as shown in Figure 6.1, to build the best domain-specific language model, we adopt a large language model pre-trained on lots of open-domain text, and fine-tune it on a given dataset's inputs, *i.e.,* $L_{\boldsymbol{x}}$, *ignoring class labels*. Both our theory and ablations confirm that ignoring class labels is a good idea (Section 6.4

and 6.5). Transferring the knowledge of large language models is particularly beneficial a small input dataset $L_{\boldsymbol{x}}$ of text is available (Hernandez et al., 2021).

To improve the computational efficiency of GAL, we do not generate unlabeled data on the fly, but generate as many unconditional samples as possible and store them in a synthetic unlabeled dataset U. We use soft-pseudo labels within self-training and KD, as we empirically found it is more effective than using hard labels on synthetic data.

Algorithm 3 GAL-KD(L, g_0, f_0, h, k)

Input: Labeled dataset $L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
Initial parameters of a generative model g_0
Initial parameters of a classifier f_0
A teacher model h
Output: A well-trained student classifier f_s after KD
▷ unlabeled data generation
1: train a generative model g by fine-tuning g_0 on $L_{\boldsymbol{x}}$ where $L_{\boldsymbol{x}} = \{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in L\}$
2: generate $U = \{\widetilde{\boldsymbol{x}}_j\}_{j=1}^{kN}$ by drawing kN random samples *i.i.d.* from $g(\boldsymbol{x})$
▷ knowledge distillation
3: apply h to unlabeled instances of U to get U'
4: train f_s by fine-tuning f_0 on $L \cup U'$
5: **return** f_s

6.3.1 Knowledge Distillation with GAL

KD distills knowledge of an expressive teacher model into a smaller student model (Hinton et al., 2015). We pose the following objective function for KD with labeled and synthetic unlabeled data,

$$\ell_{kd} = \lambda\, \mathbb{E}_{(\boldsymbol{x},y)\sim L} H(y, f_s(\boldsymbol{x})) + (1-\lambda)\, \mathbb{E}_{\widetilde{\boldsymbol{x}}\sim g(\boldsymbol{x})} H(h(\widetilde{\boldsymbol{x}}), f_s(\widetilde{\boldsymbol{x}})) \tag{6.3}$$

where h is the *teacher* model, f_s is the *student* model, g is the large pre-trained language model (*e.g.,* GPT2) fine-tuned on the text in the training data $L_{\boldsymbol{x}}$. $H(q, p) = q^\top \log p$ is the softmax cross entropy loss. Note the use of $g(\boldsymbol{x})$, approximating the unknown real data distribution $P(\boldsymbol{x})$ in Equation 6.3. Algorithm 3 summarizes the GAL-KD process.

6.3.2 Self-Training with GAL

Self-training encourages knowledge transfer between a *teacher* and a *student* model in such a way that the student can outperform the teacher. Algorithm 4 summarizes

the GAL-self-training process. Given the labeled dataset L and the synthetic unlabeled dataset U, an initial model denoted f_1 is trained using supervised learning on the labeled dataset L. Then, at iteration t, one adopts f_t as the teacher model to annotate the unlabeled dataset U using *pseudo labels*. In self-training GAL, the student model f_{t+1} is trained to optimize a classification loss on the combination of L and U:

$$\ell_{t+1} \quad = \quad \lambda \, \mathbb{E}_{(\boldsymbol{x},y)\sim L} H(y, f_{t+1}(\boldsymbol{x})) + (1-\lambda) \, \mathbb{E}_{\widetilde{\boldsymbol{x}}\sim g(\boldsymbol{x})} H(f_t(\widetilde{\boldsymbol{x}}), f_{t+1}(\widetilde{\boldsymbol{x}})) \; . \tag{6.4}$$

where $\lambda = 0.5$ unless stated otherwise.

Algorithm 4 GAL-self-training(L, g_0, f_0, k, T)

Input: Labeled dataset $L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
Initial parameters of a generative model g_0
Initial parameters of a classifier f_0
Output: A better self-training classifier f_{T+1} after T steps
▷ unlabeled data generation
train a generative model g by fine-tuning g_0 on $L_{\boldsymbol{x}}$ where $L_{\boldsymbol{x}} = \{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in L\}$
generate $U = \{\widetilde{\boldsymbol{x}}_j\}_{j=1}^{kN}$ by drawing kN random samples *i.i.d.* from $g(\boldsymbol{x})$
▷ self-training
train a base model f_1 by fine-tuning f_0 on L
for $t = 1$ to T do:
  apply f_t to unlabeled instances of U to get U'
  train f_{t+1} by fine-tuning f_0 on $L \cup U'$
return f_{T+1}

6.3.3 Domain-Specific Text Generation

We take a pre-trained GPT-2 language model (Radford et al., 2019) and fine-tune it separately on each dataset of interest after removing class labels. We find that training from scratch on these datasets is hopeless, but the larger the pre-trained GPT-2 variant, the better the validation perplexity scores are. For tasks modeling a relationship between multiple sentences, we concatenate a separator token "[SEP]" between consecutive sentences. Once a fine-tuned GPT-2 model is obtained, we generate task-specific synthetic data up to 40× larger than the original training sets. For some samples of generated text for GLUE see Appendix 10. We believe using bigger LMs and larger synthetic datasets will improve our results, but we are constrained by compute resources.

6.4 An Empirical Risk Minimization Perspective

In supervised learning, one seeks to learn a mapping f that given an input $\boldsymbol{x}$, predicts a reasonable output y. To define the supervised learning problem formally, one assumes that input-output pairs are drawn from a joint distribution P, *i.e.,* $(\boldsymbol{x}, y) \sim P(\boldsymbol{x}, y)$, and a loss function $H(y, f(\boldsymbol{x}))$ is used to assess the quality of a mapping f. This loss is used to define a notion of *expected risk*:

$$R(f) = \mathbb{E}_{P(\boldsymbol{x},y)} H(y, f(\boldsymbol{x})) \ . \tag{6.5}$$

In almost all practical applications $P(\boldsymbol{x}, y)$ is unknown. Hence, a labeled dataset of examples $L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is used to approximate $R(f)$ as

$$\widehat{R}(f) = \frac{1}{N} \sum\nolimits_{i=1}^{N} H(y_i, f(\boldsymbol{x}_i)) \ . \tag{6.6}$$

This objective function is known as *empirical risk*, and learning f through minimizing $\widehat{R}(f)$ is known as the *empirical risk minimization* principle (Vapnik, 1992). To compensate for the finite sample size in Equation 6.6, one typically combines $\widehat{R}(f)$ with a regularizer to improve generalization.

Beyond empirical risk minimization. Empirical risk minimization Equation 6.6 is motivated as a way to approximate $P(\boldsymbol{x}, y)$ through a set of Dirac delta functions on labeled examples: $P_\delta(\boldsymbol{x}, y) = \sum_i \delta(\boldsymbol{x} = \boldsymbol{x}_i, y = y_i)/N$. However, this approximation is far from perfect, hence one uses a held-out validation set for early stopping and hyper parameter tuning.

Chapelle et al. (2001) approximates expected risk as $\mathbb{E}_{P_\nu(\boldsymbol{x},y)} H(y, f(\boldsymbol{x}))$, denoted as vicinal risk minimization, by using a *vicinity distribution*, *e.g.,* $\nu(\tilde{\boldsymbol{x}}, \tilde{y} \mid \boldsymbol{x}, y) = \mathcal{N}(\tilde{\boldsymbol{x}} - \boldsymbol{x}, \sigma^2)\delta(\tilde{y} = y)$ to approximate $P(\boldsymbol{x}, y)$ as

$$P_\nu(\boldsymbol{x}, y) = \frac{1}{N} \sum\nolimits_{i=1}^{N} \nu(\tilde{\boldsymbol{x}} = \boldsymbol{x}, \tilde{y} = y \mid \boldsymbol{x}_i, y_i) \ . \tag{6.7}$$

The goal is to increase the support of each labeled data point and improve the quality and robustness of the risk function.

Recent work on mixup regularization (Zhang et al., 2018) proposes an effective way to construct another vicinity distribution by interpolating between two data points and their labels. Albeit their simplicity, these smoothing techniques tend to improve matters.

Generative models for risk minimization. One can factorize the joint distribution of input-output pairs as $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y \mid \boldsymbol{x})$. Accordingly, if one is able to learn a reasonable unconditional generative model of $\boldsymbol{x}$ denoted $g(\boldsymbol{x})$, then one can draw a pair $(\boldsymbol{x}, y)$ by first drawing $\boldsymbol{x} \sim g(\boldsymbol{x})$ and then using the current instance of f_t to draw $y \sim f_t(\boldsymbol{x})$. Then, one can use f_t and g to approximate expected risk as

$$R_t(f_{t+1}) = \mathbb{E}_{\boldsymbol{x} \sim g(\boldsymbol{x})} \mathbb{E}_{y \sim f_t(\boldsymbol{x})} H(y, f_{t+1}(\boldsymbol{x})) \ . \tag{6.8}$$

The quality of this approximation highly depends on the quality of f_t and g. If f_t is far from an optimal classifier f^* or $g(\boldsymbol{x})$ is far from $P(\boldsymbol{x})$, Equation 6.8 yields a poor approximation.

The expected risk in Equation 6.8 smoothens the risk landscape in complex ways beyond simple Gaussian smoothing and interpolation. This smoothing is applicable to any continuous, discrete, or structured domain as long as expressive generative models of $P(\boldsymbol{x})$ are available. That said, for almost all reasonable loss functions H (*e.g.,* softmax cross entropy and squared error), Equation 6.8 is minimized when $f_{t+1} = f_t$, which is not ideal, especially when f_t is far from f^*. On the other hand, empirical risk Equation 6.6 anchors the problem in real labeled examples that are provided as ground truth.

GAL-self-training aims to combine the benefits of Equation 6.6 and Equation 6.8 via:

$$R_t(f_{t+1}) \ = \ \frac{\lambda}{N} \sum\nolimits_{i=1}^{N} H(y_i, f_{t+1}(\boldsymbol{x}_i)) + (1-\lambda) \mathbb{E}_{\boldsymbol{x} \sim g(\boldsymbol{x})} \mathbb{E}_{y \sim f_t(\boldsymbol{x})} H(y, f_{t+1}(\boldsymbol{x})) \tag{6.9}$$

In this formulation, if f_t represents the minimizer of empirical risk Equation 6.6, then $f_{t+1} = f_t$ is the minimizer of Equation 6.9 too. However, one does not seek the global minimizer of empirical risk, but rather the best performance on held-out data. If f_t is obtained by stochastic gradient descent on any risk function, but early stopped according to empirical risk on a held-out set, then using such f_t in Equation 6.9 to define $R_t(f_{t+1})$ promotes the selection of a mapping f_{t+1} that minimizes empirical risk while staying close to the best performing mapping so far (*i.e.,* f_t). This formulation motivates self-training and GAL as regularizers in the functional space and explains why they can

conceivably work. Although the arguments are provided here for GAL-self-training, extending them to GAL-KD is straightforward.

How about class-conditional generative models? One can also factorize the joint distribution $P(\boldsymbol{x}, y)$ as $P(y)P(\boldsymbol{x} \mid y)$ and accordingly utilize a class-conditional generative model $g(\boldsymbol{x} \mid y)$ to derive the following expected risk formulation:

$$R(f) = \mathbb{E}_{y \sim P(y)} \mathbb{E}_{\boldsymbol{x} \sim g(\boldsymbol{x}|y)} H(y, f_{t+1}(\boldsymbol{x})) \ . \tag{6.10}$$

In this setting, pseudo labeling is not needed as synthetic data is already labeled. One can show that the optimal classifier f_g^* that minimizes Equation 6.10 for the cross-entropy loss is given by,

$$f_g^*(y \mid \boldsymbol{x}) = g(\boldsymbol{x}|y)P(y) \Big/ \sum\nolimits_{y'} g(\boldsymbol{x}|y')P(y') \ , \tag{6.11}$$

that is turning the class-conditional generative model into a classifier by using the Bayes rule yields the optimal solution.

Provided that the accuracy of generative classifiers on text classification is behind their discriminate counterparts (*e.g.,* Ravuri and Vinyals, 2019), we think substituting Equation 6.10 into Equation 6.9 is not a good idea. Essentially, by substituting Equation 6.10 into the classification objective, one is regularizing f to remain close to f_g^*, which is not an effective strategy if f_g^* is not competitive. This argument corroborates the evidence from our ablation studies and recent work showing that using class-conditional generative models to augment supervised learning does not provide big gains (Ravuri and Vinyals, 2019).

That said, one can still use class-conditional generative models to synthesize high-fidelity samples. As long as these samples are treated as unlabeled examples and annotated using a classifier, *e.g.,* f_t, we believe this is a reasonable approach falling under GAL. Note that our argument above only applies to the scenario that class-conditional generative models are used to synthesize labeled examples. In other words, GAL emphasizes the prediction of the labels in the course of the algorithm, rather than having the labels predefined. If one uses the unlabeled synthetic examples from class-conditional generative models, it still aligns to Equation 6.9, which will be verified in Section 6.5.5.

TABLE 6.1: Summary of the GLUE benchmark used for evaluation of GAL. STS-B is a regression task, so #classes is not applicable.

Dataset	task	domain	#train	#dev	#test	#classes
SST-2	sentiment analysis	movie reviews	67k	872	1.8k	2
QQP	paraphrase	social QA questions	364k	40k	391k	2
QNLI	QA/natural language inference	Wikipedia	105k	5k	5.4k	2
RTE	natural language inference	news, Wikipedia	2.5k	277	3k	2
MNLI	natural language inference	misc.	393k	20k	20k	3
MRPC	paraphrase	news	3.7k	408	1.7k	2
CoLA	acceptability	misc.	8.5k	1043	1k	2
STS-B	sentence similarity	misc.	5.8k	15k	1.4k	–

6.5 Experiments

In this section, we asses the effectiveness of GAL on KD, self-training and few-shot learning.

6.5.1 Data

We use the GLUE benchmark (Wang et al., 2019b) for our KD and self-training experiments. The statistics of GLUE are reported in Table 6.1. Regarding few-shot learning, Brown et al. (2020) studied a total of 51 few-shot learning tasks. Studying all of these tasks is prohibitively expensive. Thus, we filter tasks by following these two steps. First, since generating m synthetic examples for each test instance is computationally expensive, we exclude tasks that have more than 5k test examples. Second, we filter tasks on which GPT-3-6B achieves a score lower than 65% (please refer to Table H.1 in Brown et al. (2020) for more details). After applying the filtering steps, we use four datasets: SST-2 (Wang et al., 2019b), PIQA (Bisk et al., 2020), COPA and BoolQ (Wang et al., 2019a) as the testbed.

Generating Synthetic Text for GLUE To generate domain-specific synthetic data, we fine-tune GPT-2-large on the training set of each downstream task, excluding labels. For tasks with multiple input sentences, we concatenate input sentences into a long sequences and separate sentences by special [SEP] tokens. We generate new domain-specific data by using top-k random sampling similar to Radford et al. (2019). We do not feed any prompt to the LM, but a special [BOS] token to initiate the generation chain. A generation episode is terminated when a special [EOS] token is produced. We generate diverse sentences by varying the random seed. After collecting enough synthetic

data, we only retain unique sentences. For tasks with α input sentences, we discard generated samples that violate this constraint (approximately 10% of samples were rejected). Finally, our synthetic unlabeled dataset U includes 40× as many examples as the original dataset for each task in GLUE.

6.5.2 State-of-the-art Results of Knowledge Distillation with GAL on GLUE

It is known that KD on fresh data, unseen during training, performs better (Buciluǎ et al., 2006; Chen et al., 2020) than KD on original training data. Hence, we investigate the effectiveness of KD using generated unlabeled data through GAL.

We use the HuggingFace implementation (Wolf et al., 2020) for KD experiments and adopt a standard experimental setup consistent with previous work (Sun et al., 2019a; Xu et al., 2020). Following Rashid et al. (2021), fine-tuned RoBERTa-large (24-layer transformer) represents the teacher and a DistilRoBERTa (6-layer transformer) (Sanh et al., 2019) is used as the student. We train the student model on U and L, where U is annotated by an ensemble of 10 models, achieving an average score of 87.9. We then mix L and U with a ratio of 1:4, which is equivalent to $\lambda = 0.2$. This ratio works best on the dev set.

Table 6.2 shows the results of individual 6-layer transformers on the GLUE test set. All of the baselines use an identical student architecture. GAL achieves the best entry on the GLUE leaderboard, marking a new state-of-the-art for KD on NLP. It outperforms strong KD baselines such as DistilRoBERTa (Sanh et al., 2019), BERT-PKD (Sun et al., 2019a), BERT-Theseus (Xu et al., 2020), tinyBERT (Jiao et al., 2020) and MATE-KD (Rashid et al., 2021). It also outperforms our own DistilRoBERTa+KD baseline, which learns from soft labels produced by an identical RoBERTa-large ensemble on the original labeled dataset. While the use of soft labels outperform the vanilla fine-tuned DistilRoBERTa model, it significantly underperforms our KD+GAL baseline. We also compare with round-trip translation (RT), a strong data-augmentation baseline (*e.g.,* Yu et al., 2018; Shleifer, 2019). We mirror the experimental setup of GAL and generate 40× unlabeled data using German as the bridge language (English →German→English). The translations are generated via the best model in WMT19 (Ng et al., 2019). Although

TABLE 6.2: GLUE test results for a 6-layer transformer. GAL establishes a new state of the art on KD for NLP. Baselines: BERT-Theseus (Xu et al., 2020), BERT-PKD (Sun et al., 2019a), tinyBERT (Jiao et al., 2020) MATE-KD (Rashid et al., 2021), DistilRoBERTa (Sanh et al., 2019), and DistilRoBERTa + KD (standard KD) and DistilRoBERTa + RT (round-trip translation). MNLI-m and MNLI-mm indicate matched and mismatched respectively.

Model	**MNLI(m/mm)**	**CoLA**	**SST-2**	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**	**Avg**
Previous work:									
BERT-Theseus	82.4/82.1	47.8	92.2	87.6/83.2	85.6/84.1	71.6/89.3	89.6	66.2	78.6
BERT-PKD	81.5/81.0	-	92.0	85.0/79.9	-	70.7/88.9	89.0	65.5	-
tinyBERT	84.6/83.2	51.1	93.1	87.3/82.6	85.0/83.7	71.6/89.1	90.4	70.0	79.8
MATE-KD	86.2/85.6	58.6	95.1	91.2/88.1	88.5/88.4	73.0/89.7	92.4	76.6	83.5
Our results:									
DistilRoBERTa	83.8/83.4	55.9	93.2	87.4/83.1	87.5/87.5	71.7/89.1	90.6	73.3	81.2
DistilRoBERTa + KD	84.7/84.5	54.9	94.1	88.0/84.4	87.4/86.6	72.1/89.2	91.6	73.8	81.6
DistilRoBERTa + RT	86.1/86.1	53.0	94.6	91.0/87.8	89.2/88.8	73.1/89.9	92.4	76.9	82.7
DistilRoBERTa + GAL	**87.4/86.5**	**60.0**	**95.3**	**91.9/89.2**	**90.0/89.6**	**73.3/90.0**	**92.7**	**81.8**	**84.8**

TABLE 6.3: RoBERTa base and GAL self-training results on GLUE dev sets, averaged across 5 independent runs (numbers in the subscript indicate the error bar, *i.e.*, standard deviation divided by $\sqrt{5}$.).

Model	MNLI	CoLA	SST-2	MRPC	STS-B	QQP	QNLI	RTE	Avg
RoBERTa base	$87.7_{0.1}$	$63.6_{0.4}$	$94.8_{0.1}$	$90.1_{0.4}$	$90.8_{0.1}$	$91.5_{0.1}$	$92.6_{0.1}$	$78.8_{0.4}$	86.2
+ GAL (iter 1)	$87.9_{0.1}$	$65.1_{0.5}$	$95.3_{0.1}$	$91.7_{0.5}$	$91.4_{0.1}$	$91.8_{0.1}$	$93.1_{0.1}$	$81.4_{0.4}$	87.2
+ GAL (iter 2)	$88.0_{0.1}$	$65.2_{0.5}$	$95.3_{0.1}$	$92.2_{0.4}$	$91.5_{0.1}$	$91.7_{0.1}$	$93.2_{0.1}$	$82.4_{0.5}$	**87.4**
+ GAL (iter 3)	$87.9_{0.1}$	$65.5_{0.5}$	$95.3_{0.1}$	$92.2_{0.5}$	$91.7_{0.2}$	$91.7_{0.1}$	$93.2_{0.1}$	$82.0_{0.5}$	**87.4**
RoBERTa base + self-distillation	$88.1_{0.1}$	$63.7_{0.5}$	$95.2_{0.1}$	$90.3_{0.4}$	$90.4_{0.1}$	$91.5_{0.1}$	$93.1_{0.1}$	$79.7_{0.5}$	86.5

DistilRoBERTa+RT is better than vanilla DistilRoBERTa and KD variants, it still drastically underperforms our approach.

6.5.3 Self-Training with GAL on GLUE

We fine-tune pretrained RoBERTa model provided by fairseq (Ott et al., 2019) on each GLUE task. Fine-tuned RoBERTa serves as the first teacher model for self-training. Each student model is initialized with the original pretrained RoBERTa and fine-tuned with exactly the same hyper-parameters as suggested by fairseq (Ott et al., 2019). We combine the labeled dataset L and the synthetic dataset U with a ratio of 1:1, by oversampling labeled data. This corresponds to $\lambda = 0.5$ in Equation 6.9.

Table 6.3 shows that GAL provides an average improvement of +1.3% over RoBERTa-base. We see consistent improvements with more GAL iterations, but performance saturates after three iterations. We further compare our approach with a self-distillation baseline (Furlanello et al., 2018), in which the teacher and student models use the same architecture and transfer knowledge via the original labeled training set. Although self-distillation provides a slight improvement, the gains from GAL are more significant.

We delve deeper and combine GAL self-training with RoBERTa-large and report test results for both single model and ensemble model in Table 6.4. We observe consistent gains coming from GAL on RoBERTa-large. Our results underperform the latest and biggest LMs from the GLUE leaderboard, but we are optimistic that GAL can be effectively combined with enormous LMs to provide additional gains.

TABLE 6.4: RoBERTa-large with GAL self-training and SoTA methods evaluated on GLUE test sets. The benefit of GAL on single models is larger than ensembles. It appears that self-training reduce the variance of models. Baselines including much larger models: RoBERTa-large (Liu et al., 2019), ELECTRA (Clark et al., 2020), T5 (Raffel et al., 2020), ERNIE (Sun et al., 2019b), and DeBERTa (He et al., 2020). MNLI-m and MNLI-mm indicate matched and mismatched respectively.

Model	**MNLI(m/mm)**	**CoLA**	**SST-2**	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**	**Avg**
Individual Models (our implementation):									
RoBERTa-large	90.1/89.7	63.8	96.1	91.2/88.3	90.9/90.7	72.5/89.6	94.5	85.9	86.5
RoBERTa-large + GAL	90.2/89.8	66.2	96.4	92.0/89.2	90.7/90.5	73.6/89.9	95.0	86.3	87.1
Ensemble Models (our implementation):									
RoBERTa-large	91.2/90.5	66.8	96.9	92.8/90.3	91.9/91.6	74.5/90.4	95.5	87.7	87.9
RoBERTa-large + GAL	91.0/90.7	67.9	97.1	93.1/90.8	91.6/91.4	74.5/90.4	95.8	88.2	88.2
State-of-the-art:									
RoBERTa-large	90.8/90.2	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.3	95.4	88.2	88.0
ELECTRA	91.3/90.8	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	95.8	89.8	89.2
T5	92.2/91.9	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	96.9	92.8	89.8
ERNIE	91.9/91.4	74.4	97.8	93.9/91.8	93.0/92.6	75.2/90.9	97.3	92.0	90.2
DeBERTa	91.9/91.6	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	99.2	93.2	90.3

6.5.4 Prompt-based Few-shot Experiments

GPT3 (Brown et al., 2020) has introduced an optimization-free paradigm for few-shot learning for NLP. Without updating the parameters, large LMs can correctly predict the labels of the inputs by conditioning on a prompt, which consists of an instruction, a few labeled instances and a new unlabeled input. We apply GAL to prompt-based few-shot learning. Specifically, we present k labeled examples as a prompt to GPT-J (Wang and Komatsuzaki, 2021), an open-sourced re-implementation of GPT-3-6B, and generate m synthetic examples, followed by the corresponding labels. Note that to mitigate noisy outputs, the generation of each synthetic example only conditions on the original k labeled examples. Finally, we concatenate the original k examples and m synthetic examples, and conduct a $(k+m)$-shot learning experiment with GPT-J.

TABLE 6.5: Few-shot learning results for GPT-J (6B) (Wang and Komatsuzaki, 2021) on four NLP datasets. Accuracy is reported for these datasets.

Model	**SST-2**	**PIQA**	**COPA**	**BoolQ**	**Avg**
4-shot	89.8 $_{0.8}$	76.0 $_{1.4}$	79.0 $_{1.5}$	64.3 $_{0.8}$	77.3
8-shot	91.3 $_{0.8}$	76.2 $_{1.2}$	79.0 $_{1.5}$	66.2 $_{0.8}$	78.2
16-shot	92.7 $_{0.6}$	77.0 $_{0.9}$	81.0 $_{1.1}$	66.8 $_{0.8}$	79.4
4-shot + synthetic 12-shot (GAL)	91.5 $_{0.7}$	76.7 $_{1.0}$	80.0 $_{1.2}$	65.9 $_{0.8}$	78.5

We notice that in order to generate valid synthetic data, GPT-J requires to see at least 4 labeled examples. In addition, at most 16 examples of BoolQ can be fed into GPT-J without truncation. Thus, we set k and m to 4 and 12 respectively. As seen in Table 6.5, GAL leads to an average improvement of 1.2% over 4-shot learning, and reduces the gap between 4-shot and 16-shot learning. We noticed that the quality of some generated examples is low. We believe the performance of few-shot learning can be further improved with high-quality instances. One solution is to generate many synthetic examples, and select a high-quality subset. Since each test instance conditions on distinct labeled instances, one has to generate different synthetic instances for each test example from GPT-J, which causes expensive computation. Due to such computational constraints, we leave the investigation of data selection strategies to the future work.

6.5.5 Ablating Components of GAL on GLUE

We conduct an in-depth study of different components of GAL on GLUE datasets. Unless stated otherwise, we use a RoBERTa-base model with a combination of the original training data and 40× synthetic data for each self-training experiment.

GPT-2 model size. Radford et al. (2019) present a few variants of the GPT-2 model including *GPT-2*, *GPT-2-medium*, *GPT-2-large*, and *GPT-2-XL*. Larger GPT-2 models yield better perplexity scores and higher generation quality. We utilize these models except GPT-2-XL within the GAL framework to study the impact of the generative model's quality on downstream task's performance. Table 6.6 shows that regardless of the GPT-2 model sizes, GAL consistently surpasses the vanilla RoBERTa base. Moreover, SST-2 and RTE datasets are not sensitive to the capacity of GPT-2, but higher quality synthetic text improves the results on MRPC and CoLA datasets. We leave investigation of GPT-2-XL and even larger LMs such as GPT-3 (Brown et al., 2020) to future work.

TABLE 6.6: GAL with various GPT-2 model sizes on GLUE dev sets. NA indicates a RoBERTa base model.

GPT-2	SST-2	RTE	MRPC	CoLA
NA	94.8	78.8	90.1	63.6
small	95.5	81.3	90.9	63.9
medium	95.3	81.3	91.3	63.7
large	95.3	81.4	91.7	65.1

Soft v.s. hard pseudo label. We investigate the use of soft and hard pseudo labels within the GAL framework. The results in Table 6.7 suggest that GAL using soft pseudo labels is more effective than hard labels on the GLUE benchmark. This finding is compatible with the intuition that soft labels enable measuring the functional similarity of neural networks better (Hinton et al., 2015).

TABLE 6.7: GAL with soft *v.s.* hard pseudo labels on GLUE dev sets.

Pseudo label	SST-2	RTE	MRPC	CoLA
hard	95.0	80.7	90.8	63.0
soft	95.3	81.4	91.7	65.1

Class-conditional synthetic data generation. Although it has shown that incorporating labeled synthetic data can significantly boost the performance of low-resource tagging tasks (Ding et al., 2020; Liu et al., 2021), previous work (Kumar et al., 2020; Ravuri and Vinyals, 2019) suggests that it is challenging to utilize labeled synthetic data from class-conditional generative models to boost the accuracy of text and image classifiers. Our theory in Section 6.4 points to the potential drawback of class-conditional synthetic data. We empirically study this phenomenon, by fine-tuning GPT-2 in a class-conditional manner. Then we utilize its synthetic examples in two different cases: 1) labeled synthetic examples and 2) unlabeled synthetic examples. Table 6.8 shows that not only class-conditional LMs underperform unconditional LMs in our GAL framework, but also they are much worse than the baseline, when using the pre-defined labels. Nevertheless, if we apply GAL to these examples, the class-conditional LM is on-par with the unconditional one, which corroborates the importance of the annotation step in GAL.

TABLE 6.8: Synthetic data from class-conditional LMs underperforms GAL and RoBERTa on GLUE dev sets.

Generative model	**Labeled synthetic data**	**SST-2**	**RTE**	**MRPC**	**CoLA**
None (baseline)	-	94.8	78.8	90.1	63.6
Class-conditional LM	✓	92.9	74.4	86.0	58.4
Unconditional LM (GAL)	✗	95.3	81.4	91.7	65.1
Class-conditional LM (GAL)	✗	95.4	81.0	91.4	65.2

Besides, to further verify this argument, we sample 100 instances from the synthetic RTE dataset generated by the label-prompted GPT2, as the class-conditional LM. Then we annotate these examples using a human annotator, GPT2 classifier and RoBERTa classifier. Finally, we compute the Accuracy, F1, Precision and Recall scores between human labels and GPT2 labels, between human labels and RoBERTa labels, and between human labels and conditioned labels used by GPT2 when data was generated. Table 6.9 shows that class-conditional LM has difficulty generating sentences retaining the semantics or pragmatics of a specified category, which also corroborates our theoretical analysis in Section 6.3. On the other hand, discriminative models, such as GPT2 classifier and RoBERTa classifier, are able to produce higher quality labels that correlate better with human annotations.

Quality of synthetic dataset. An effective generative model of text should learn the word preferences and genre associated with a given corpus, but still produce novel

TABLE 6.9: Performance of GPT2 annotation, RoBERTa annotation and conditioning labels on 100 random examples from the synthetic RTE dataset generated by a class-conditional LM.

Label type	Accuracy	F1	Precision	Recall
GPT2	86.0	87.0	88.7	85.5
RoBERTa	90.0	91.4	100.0	84.1
conditioning label	72.0	71.4	66.0	77.8

sentences. In order to study the characteristics of our synthetic datasets, Table 6.10 reports the number of unique n-grams in the training and synthetic datasets, as well as the number of unique n-grams shared between them. The high degree of overlap on uni-grams suggests that the fine-tuned GPT-2-large is somewhat domain-specific. Meanwhile, the large number of unique n-grams in the synthetic dataset suggests that many novel word combinations are generated, which is helpful for GAL.

TABLE 6.10: For each dataset we report the number of unique n-grams in (the original dataset, the synthetic dataset, shared between the two).

	SST-2	QNLI	RTE	MRPC	CoLA
1-gram	(15k, 33k, 11k)	(89k, 231k, 55k)	(18k, 34k, 13k)	(15k, 27k, 10k)	(6k, 6k, 4k)
3-gram	(107k, 2M, 38k)	(2M, 10M, 513k)	(120k, 750k, 30k)	(105k, 562k, 27k)	(39k, 198k, 14k)
5-gram	(109k, 4M, 9k)	(2M, 25M, 146k)	(130k, 1M, 4k)	(120k, 1M, 7k)	(35k, 389k, 5k)

6.6 Summary

We present Generate, Annotate, and Learn (GAL): a framework for self-training and knowledge distillation with generated unlabeled data. We motivate GAL from an expected risk minimization perspective and demonstrate both theoretically and empirically that the use of unconditional generative models for synthetic data generation is more effective than class-conditional generative models, previously used in the literature. GAL leverages advances in large pretrained language models to help supervised learning and can have implications for learning from limited labeled data. GAL significantly helps improve knowledge distillation and prompt-based few-shot learning. Finally, We hope that GAL will stimulate new research on the evaluation and development of large language models.

Chapter 7

Conclusion

7.1 Summary of the Thesis

The primary contribution of this thesis is the leveraging of various contexts to enhance the performance of natural language understanding and generation by mitigating uncertainty caused by the lack of context. We have further crystallized this high-level goal into three main directions: (1) encouraging morphologically plausible subword segmentation in NMT via source- and target-side contexts (Chapter 3); (2) enhancing the utility of context-dependent language understanding tasks through the preceding context (Chapter 4 and 5); and (3) advancing text classification tasks by modeling the task-specific context (Chapter 6).

We presented the first work incorporating source- and target-side sentences as context into subword word segmentation for NMT in Chapter 3. We found that as greedy and stochastic solutions disregard the context, the resultant subword segmentations are suboptimal. Inspired by this finding, we devised a mixed character-subword Transformer as a means of context modeling. The Transformer allowed us to access the source- and target-side information to capture the contextual features. Since there are multiple valid segments and the optimal one is latent, we utilized dynamic programming to marginalize possible segmentations during the training stage. We replaced the sum operation with a max one at the inference stage to produce the best segments. To assess the effectiveness of the proposed approach, we applied it to ten translation tasks involving five language

pairs. According to the linguistic analysis, our approach can deliver more morpheme-aware subwords than the context-agnostic segmentation baselines. As a reward, our solution outperformed the greedy one by 0.9 BLEU on average and achieved an average improvement of 0.55 BLEU over the stochastic approach.

We introduced a novel task in Chapter 4, aiming to model the changing intent of end-users through a conditional context lens which could be linked to an interactive click-through search. In this task, due to the lack of available data, we first created three new datasets supporting an update of the original scene graph via a textual description. Then we devised several task-oriented models to examine the feasibility of the proposed task. The empirical studies showed that our models and baselines could understand the demand of the modification command and update the original graph accordingly. Our best model, leveraging cross-attention to capture the association and difference between the original graph and the modification query, managed to generate high-fidelity and high-quality target graphs. Thus, it surpassed the strong baselines by up to 8.5% on the datasets when evaluated on F1 and accuracy. We released these datasets and the best models to promote further investigation along this line.

In Chapter 5, we looked into the problem of long-range context-dependent NLU, namely, semantic parsing. We first showed that an utterance-level parser could not handle the context-related parsing in dialogue-based semantic parsing. In addition, we also noted that the standard *end-to-end* parser could neglect the input arguments and produce erroneous ones. Hence, we integrated context-aware and copy mechanisms into the neural parser. We quantitatively analyzed the error reduction and found that our approach could rectify half of the errors spotted in the baseline. Therefore, our approach outperformed the baseline parser by more than 2.5% accuracy across multiple neural architectures.

We leveraged unlabeled synthetic data to advance self-training, knowledge distillation, and few-shot learning in Chapter 6. We asserted that context is not bounded by neighboring words and sentences. Instead, since different tasks conform to differing properties and formats, one can consider a task as having a meta context. It is known that one can leverage in-domain unlabeled data to advance the performance of a downstream task. However, it is challenging to find task-specific unlabeled data due to its unique characteristics. As a remedy, we tailored generic generative models by fine-tuning or

conditioning them on in-distribution examples. Then we could synthesize many unlabeled in-domain data points from the task-specific generative model. We assessed the efficacy of the proposed approaches on self-training, knowledge distillation, and few-shot learning. For self-training, with pseudo-labeled synthetic in-domain data, we obtained average improvements of 1.2 and 0.6 points on the GLUE benchmark dev and test set over the baseline, respectively. Our approach significantly advanced the knowledge distillation and marked new state-of-the-art results on the GLUE benchmark leaderboard, surpassing the previous best results by 1.3 points on average. The prompt-based few-shot learning also benefited from our approach, leading to an average improvement of 1.2% accuracy over 4-shot learning. In addition to the empirical studies, we theoretically dissected the key to the success of our approach.

In conclusion, since human languages are context-dependent, this thesis has dedicated efforts to improving natural language understanding and generation by incorporating various contextual information. However, compared to previous works, we have expanded the definition of context and explored the impact of this in relation to different tasks and scopes. Our studies consolidate the claim that context is crucial to human text. Thus we hope this work can invigorate research in this domain and encourage the community to expand the view of context and uncover its capabilities more thoroughly.

7.2 Future Directions

This section briefly discusses some potential extensions to the research field based on the findings of this thesis study.

Fast Context-dependent Subword Segmentation We have shown that context-dependent subword segmentation can produce morphologically plausible subwords, improving NMT across multiple language pairs. However, because of the use of dynamic programming, this approach is computationally expensive. Moreover, our approach can be merely used for target-side segmentation. Lee et al. (2017) and Cherry et al. (2018) suggested that NMT can be done at a character level. As character-level NMT models leverage an attention module to align the source characters and the target ones softly, it is natural to ask whether subword segmentation can benefit from such alignments,

which may boost the segmentation speed and enable two-sided segmentation. Moreover, to encourage more morphologically plausible segmentation, one can incorporate several types of auxiliary information into the model, such as morphological tags and lexical semantic tags.

Multi-turn Scene Graph Modifications This thesis has unveiled a novel task in relation to scene graph modification. As a starting point, we focused our attention on one-step modification. However, in reality the number of modifications is supposed to be varying on demand. Thus, a possible direction would be investigating multi-turn scene graph modification where the number of turns is variable.

Improving Robustness Using Synthetic Data We have demonstrated that one can advance the downstream tasks by generating fresh unlabeled data in terms of accuracy. In addition, a concurrent work (Gowal et al., 2021) has shown that using generated images can enhance the robustness of image classifiers and mitigate the vulnerability caused by adversarial attacks. We believe this finding applies to text classification problems as well because of the claim about the smoothness of the landscape of the data space (see Section 6.4).

Bibliography

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer, 2016.

Martin Andrews, Yew Ken Chia, and Sam Witteveen. Scene graph parsing by attention graph. *arXiv preprint arXiv:1909.06273*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

M Saiful Bari, Tasnim Mohiuddin, and Shafiq Joty. UXLA: A robust unsupervised data augmentation framework for zero-resource cross-lingual NLP. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1978–1992, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.154. URL https://aclanthology.org/2021.acl-long.154.

Anja Belz and Eric Kow. System building cost vs. output quality in data-to-text generation. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 16–24, Athens, Greece, March 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-0603.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1025. URL https://aclanthology.org/D16-1025.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

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

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. *arXiv preprint arXiv:1911.07470*, 2019.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *arXiv:1905.13736*, 2019.

Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2028–2038, 2014.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. *Advances in neural information processing systems*, 2001.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *NeurIPS*, 2020.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, 2018.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*, 2018.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *International Conference on Learning Representations*, 2020.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.

Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. In *Proceedings of the IEEE conference on computer vision and Pattern recognition*, pages 3076–3086, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

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, 2019.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.488. URL https://aclanthology.org/2020.emnlp-main.488.

Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, 2018.

Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

Jingfei Du, Édouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

S Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *International Conference on Machine Learning*, pages 1607–1616, 2018.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. September 2021. doi: 10.5281/zenodo.5371628. URL https://doi.org/10.5281/zenodo.5371628.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2501. URL https://aclanthology.org/W18-2501.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

E. Goldberg. Fog: Synthesizing forecast text directly from weather maps. In *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, pages 156–162, 1993. doi: 10.1109/CAIA.1993.366647.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312, 2019.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv:2006.03654*, 2020.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. Sequence to sequence mixture model for diverse machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 583–592, 2018.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*, 2016.

Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gmrxHFvB.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

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.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. 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 1–10, 2015.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, 2020.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

Tim Johnson. Natural language computing: the commercial applications. *The Knowledge Engineering Review*, 1(3):11–23, 1984.

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

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, 2017.

Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, 2020.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Brian Langner and Alan Black. Mountain: A translation-based approach to natural language generation for dialog systems. *Proceedings of IWSDS. Irsee, Germany*, 2009.

Samuel Läubli, Rico Sennrich, and Martin Volk. Has machine translation achieved human parity? a case for document-level evaluation. *arXiv preprint arXiv:1808.07048*, 2018.

Yann Lecun. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, pages 21–28. Morgan Kaufmann, 1988.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on challenges in representation learning, ICML*, 2013.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, 2018.

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

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, 2016a.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs, 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luis Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, 2015.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*, 2016.

Linlin Liu, Bosheng Ding, Lidong Bing, Shafiq Joty, Luo Si, and Chunyan Miao. MulDA: A multilingual data augmentation framework for low-resource cross-lingual

NER. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5834–5846, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.453. URL https://aclanthology.org/2021.acl-long.453.

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.

Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, 2016.

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, 2015.

Ramesh Manuvinakurike, Jacqueline Brixey, Trung Bui, Walter Chang, Doo Soon Kim, Ron Artstein, and Kallirroi Georgila. Edit me: A corpus and a framework for understanding natural language image editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

Kathleen McKeown, Karen Kukich, and James Shaw. Practical issues in automatic documentation generation. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, ANLC '94, page 7–14, USA, 1994. Association for Computational Linguistics. doi: 10.3115/974358.974361. URL https://doi.org/10.3115/974358.974361.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, et al. Recurrent neural network based language model. In *In INTERSPEECH 2010,*. Citeseer, 2010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Khalil Mrini, Franck Dernoncourt, Trung Bui, Walter Chang, and Ndapa Nakashole. Rethinking self-attention: An interpretable self-attentive encoder-decoder parser. *arXiv preprint arXiv:1911.03875*, 2019.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, 2019.

Alice H. Oh and Alexander I. Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*, 2000. URL https://aclanthology.org/W00-0306.

Christopher Olah. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Online; accessed 12 Feb 2022.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, 2019.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

C. Paris and K. Vander Linden. An interactive support tool for writing multilingual manuals. *Computer*, 29(7):49–56, 1996. doi: 10.1109/2.511968.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Julien Perez and Fei Liu. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 305–314, 2017.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 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 2227–2237, 2018.

Shrimai Prabhumoye, Yulia Tsvetkov, Alan W Black, and Ruslan Salakhutdinov. Style transfer through multilingual and feedback-based back-translation. *arXiv preprint arXiv:1809.06284*, 2018.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. *arXiv:1910.13267*, 2019.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21: 1–67, 2020.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. Mate-kd: Masked adversarial text, a companion to knowledge distillation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1062–1071, 2021.

Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models. *Advances in Neural Information Processing Systems*, pages 12268–12279, 2019.

Ehud Reiter and Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2786, 2018.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet

large scale visual recognition challenge. *International journal of computer vision*, 115 (3):211–252, 2015.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80, 2015.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1238. URL https://www.aclweb.org/anthology/P18-1238.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR, 2019.

Sam Shleifer. Low resource text classification with ulmfit and backtranslation. *arXiv preprint arXiv:1903.09244*, 2019.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608, 2020.

Shashank Srivastava, Amos Azaria, and Tom M Mitchell. Parsing natural language conversations using contextual cues. In *IJCAI*, pages 4089–4095, 2017.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to map context-dependent sentences to executable formal queries. 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 2238–2249, 2018.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *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 4314–4323, 2019a.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019b.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

Damien Teney, Lingqiao Liu, and Anton van Den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2017.

Antonio Toral and Víctor M. Sánchez-Cartagena. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1063–1073, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1100.

Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 1992.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.

Stephan Vogel, Franz Josef Och, Christof Tillmann, Sonja Nießen, Hassan Sawaf, and Hermann Ney. Statistical methods for machine translation. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 377–393. Springer, 2000.

Tu Vu, Minh-Thang Luong, Quoc Le, Grady Simon, and Mohit Iyyer. Strata: Self-training with task augmentation for better few-shot learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5715–5731, 2021.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. Word representation models for morphologically rich languages in neural machine translation. *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 2017. doi: 10.18653/v1/w17-4115. URL http://dx.doi.org/10.18653/v1/w17-4115.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv:1905.00537*, 2019a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *International Conference on Learning Representations*, 2019b.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Yu-Siang Wang, Chenxi Liu, Xiaohui Zeng, and Alan Yuille. Scene graph parsing as dependency parsing. 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 397–407, 2018.

Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, October 2020. doi: 10.18653/v1/2020.emnlp-demos.6.

Yuk Wah Wong and Raymond Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, 2006.

William A Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450, 1973.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2020.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. Bert-of-theseus: Compressing bert by progressive module replacing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, 2020.

Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019a.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, 2020.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019b.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. QANet: Combining local convolution with global self-attention for reading comprehension. *ICLR*, 2018.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

Luke Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, 2007.

Luke S Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984, 2009.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2018.

Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021.

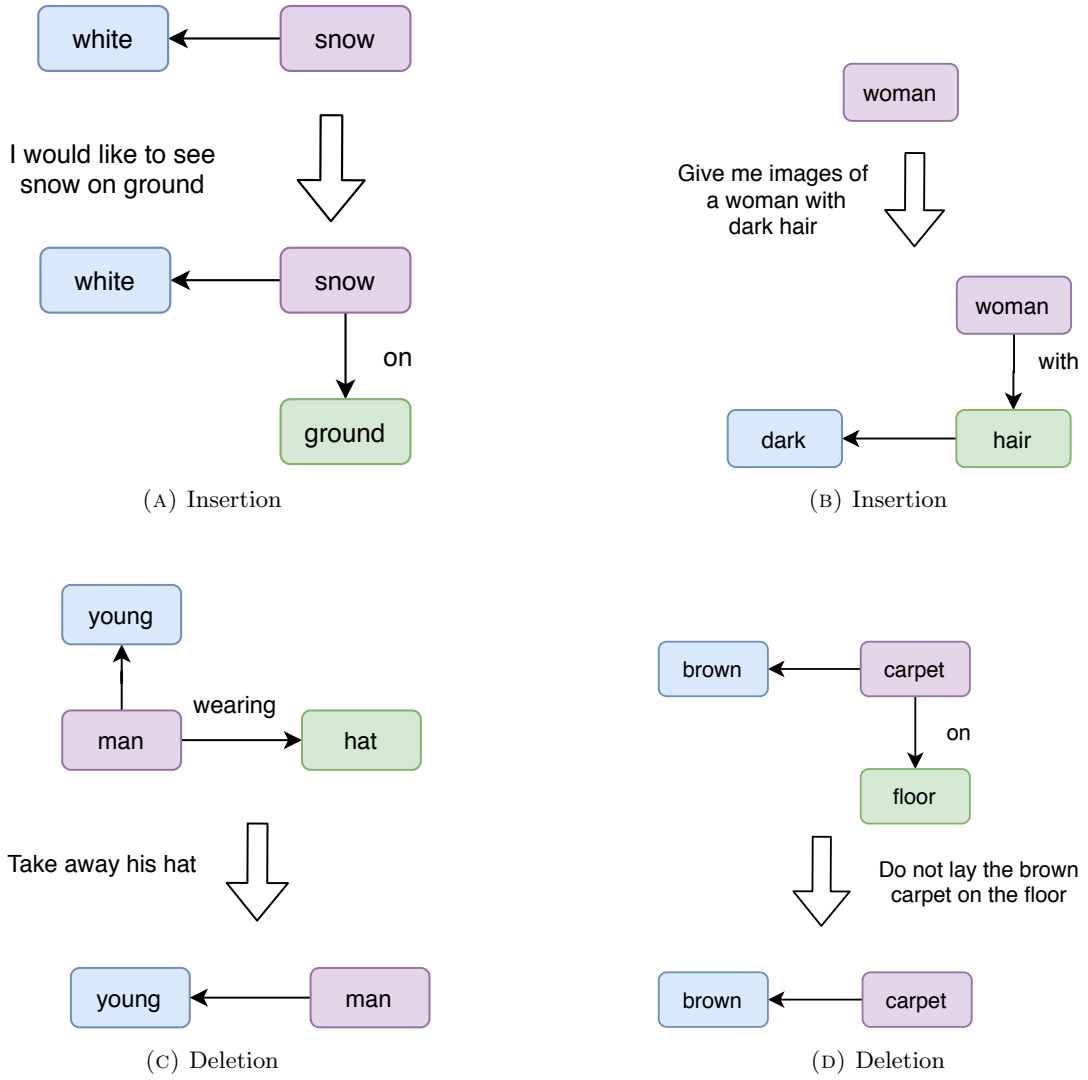
Junru Zhou and Hai Zhao. Head-driven phrase structure grammar parsing on penn treebank. *arXiv preprint arXiv:1907.02684*, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual

explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

User-generated Dataset for Scene Graph Modification

We provides some examples of our user-generated dataset for scene graph modification in Figure 1 and 2.



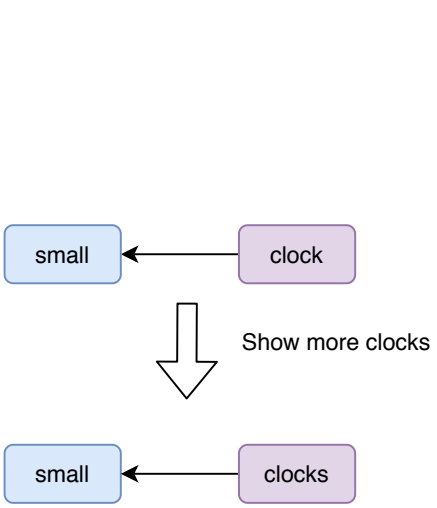
(A) Insertion

(B) Insertion

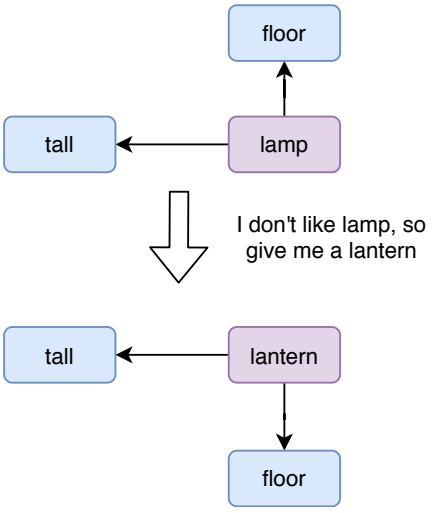
(C) Deletion

(D) Deletion

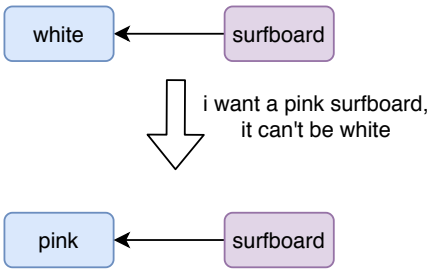
FIGURE 1: Examples from the user-generated dataset for scene graph modification.



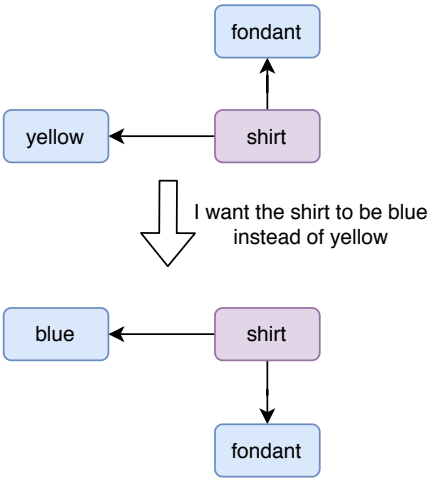
(A) Object substitution



(B) Object substitution



(C) Attribute substitution



(D) Attribute substitution

FIGURE 2: Examples from the user-generated dataset for scene graph modification.

Generated Unlabeled Examples Annotated with Pseudo Labels

We provide some synthetic sentences generated by GAL in Table 1, 2, 3, 4, 5, and 6.

TABLE 1: **QNLI**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

When did the third Digimon series begin? [SEP] Unlike the two seasons before it and most of the seasons that followed, Digimon Tamers takes a darker and more realistic approach to its story featuring Digimon who do not reincarnate after their deaths and more complex character development in the original Japanese. (**not entailment**)
KNN: 1: What is the name of the third season? [SEP] In addition to the first two seasons, the third season is the season that introduced new characters such as Captain Malice, a supervillain who became the antagonist in season two; and the villains known as the Heartbreakers, who introduced a group of crime fighters. (**not entailment**) 2: When did the "Walking Dead" series end? [SEP] In 2013, AMC announced that it would develop a "superhero series", which would follow the storylines and characters from the "Walking Dead" series in order to bring the popular AMC original series to a new and younger audience. (**not entailment**) 3: What is the main objective of the first season of the X-Files? [SEP] The first season was notable in that the characters were introduced and developed within the space of a single season, as was the format of the show itself. (**not entailment**)
What did Arsenal consider the yellow and blue colors to be after losing a FA Cup final wearing red and white? [SEP] Arsenal then competed in three consecutive FA Cup finals between 1978 and 1980 wearing their "lucky" yellow and blue strip, which remained the club's away strip until the release of a green and navy away kit in 1982–83. (**entailment**)
KNN: 1: Who was the most important player for Arsenal Football Club in the 1950s? [SEP] Wenger continued to use Arsenal's famous red shirts and red kits throughout the 1950s and 1960s, and the red strip became the club's most recognised and recognizable symbol. (**not entailment**) 2: When were the first two teams to play for the trophy in the Premier League? [SEP] The trophy was awarded to Manchester United in 1990-91 and was named after Sir Bobby Charlton, the club's manager until 1990, and later Sir Stanley Matthews, the club's most successful manager. (**not entailment**) 3: What were the last four players to wear the yellow in the final? [SEP] With Arsenal having won all four major trophies in the period, they became the only club to have won five in a row. (**not entailment**)

TABLE 2: **QQP**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

How is the life of a math student? Could you describe your own experiences? [SEP] Which level of prepration is enough for the exam jlpt5? (**not duplicated**)
KNN: 1: What are the best courses for a mechanical engineering student? [SEP] What is the best course to do after completing a B.Tech in mechanical engineering? (**not duplicated**) 2: How much marks are needed to get through the GATE with electronics? [SEP] What is the average score of the Gate EE exam? What are the cut-offs? (**not duplicated**) 3: What is the best time table for students to prepare for IAS? [SEP] How can one study for IAS in a best time? (**not duplicated**)
How does an IQ test work and what is determined from an IQ test? [SEP] How does IQ test works? (**duplicated**)
KNN: 1: What is the average IQ of the U.S. population? [SEP] How does an IQ test work? (**not duplicated**) 2: Is the Iq test an effective way to measure intelligence? [SEP] How do IQ tests work? (**duplicated**) 3: How is an IQ test on a scale from 1 to 100 scored? [SEP] How do you get your IQ tested? (**not duplicated**)

TABLE 3: **SST-2**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

are more deeply thought through than in most ‘ right-thinking ’ films (**positive**)
KNN: 1: is far more sophisticated , insightful and thought-provoking than his previous films . (**positive**) 2: is more sophisticated than its more obvious and less-than-dazzling counterparts (**positive**) 3: is about as well-thought as the idea of a bad hair day , (**negative**)
contains no wit , only labored gags (**negative**)
KNN: 1: lacks insight , and lacks empathy (**negative**) 2: has little humor or intelligence (**negative**) 3: lacks all wit and humanity (**negative**)

TABLE 4: **RTE**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

Like the United States, U.N. officials are also dismayed that Aristide killed a conference called by Prime Minister Robert Malval in Port-au-Prince in hopes of bringing all the feuding parties together. [SEP] Aristide had Prime Minister Robert Malval murdered in Port-au-Prince. (**not entailment**)
KNN: 1: The government has been criticized for failing to prevent the mass protests that led to the ouster of President Nicolas Sarkozy earlier this month, which led to his second election defeat since assuming office two years ago. [SEP] Prime Minister Jean-Marc Ayrault is a former president of France. (**not entailment**) 2: The French president, Jacques Chirac, has been urged by both the Vatican and the U.N. Security Council to step up efforts to prevent the return of former dictator Nicolas Sarkozy. [SEP] Nicolas Sarkozy left France. (**not entailment**) 3: The French newspaper Le Monde says the French President Nicolas Sarkozy was advised by U.S. President George W. Bush about a possible trip to Iraq on Thursday. [SEP] Nicolas Sarkozy is a member of the United States. (**not entailment**)
Only a week after it had no comment on upping the storage capacity of its Hotmail e-mail service, Microsoft early Thursday announced it was boosting the allowance to 250MB to follow similar moves by rivals such as Google, Yahoo, and Lycos. [SEP] Microsoft's Hotmail has raised its storage capacity to 250MB. (**entailment**)
KNN: 1: The company, known as Microsoft Office, said it plans to sell all of the copies of its popular Office suite at a loss in the wake of the launch of Microsoft Windows 7, saying it will also make $25 million in advertising costs, a move likely to hurt its long-standing position among consumers and business leaders. [SEP] Microsoft Office is a popular office suite. (**entailment**) 2: The company's shares shot up more than 35% after the company said it has sold all of its remaining inventory of the new Kindle e-readers at $70 each. The shares rose to $65.20 on Wednesday, their highest since March 6, 2011. "The Kindle is our best selling product," said Jeff Bezos, founder and CEO of Amazon.com. [SEP] Amazon.com is based in Seattle. (**not entailment**) 3: In response to concerns expressed by some investors, Microsoft last week said it would reduce the amount of shares that will be available to the public by 10 percent in the first quarter, with a further reduction to 3 percent in the second quarter. The stock price has plunged from $24 to $17, and Microsoft is currently offering $17 to $19 a share to its most senior employees. Some investors had criticized Microsoft's response to concerns about the price of its stock and about the perception that the company is in trouble. [SEP] Microsoft is struggling to sell its stock. (**not entailment**)

TABLE 5: **MRPC**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

A BMI of 25 or above is considered overweight ; 30 or above is considered obese . [SEP] A BMI between 18.5 and 24.9 is considered normal , over 25 is considered overweight and 30 or greater is defined as obese . (**paraphrase**)
KNN: 1: The report said that the average woman in her twenties who takes oral contraceptives daily can expect a loss of around 40 per cent of her bone density between the ages of 20 and 45 . [SEP] The study said the average woman in her twenties who used the pill every day , or every day for up to five years , can expect a loss of about 40 per cent of her bone density between the ages of 20 and 45 . (**paraphrase**) 2: The report found that 17 percent of U.S. adults between ages 18 and 64 have a body mass index at or above the " normal " 20 . [SEP] For people of that age , 17.1 percent of adults have a body mass index at or above the " normal " 20 , while 12.6 percent have a body mass index of 30 or above . (**not paraphrase**) 3: The survey shows the proportion of women between 20 and 44 who were obese was 6.3 percent , up from 5.7 percent in 2001 . [SEP] The proportion of women between 20 and 44 who were obese increased to 6.3 percent from 5.7 percent in 2001 . (**paraphrase**)
Shares of Genentech , a much larger company with several products on the market , rose more than 2 percent . [SEP] Shares of Xoma fell 16 percent in early trade , while shares of Genentech , a much larger company with several products on the market , were up 2 percent .(**not paraphrase**)
KNN: 1: Shares in Aventura fell as much as 5 percent , while shares in Medi-Cal climbed 2.5 percent . [SEP] Shares in Aventura were up 2.5 percent , while shares in Medi-Cal rose 2.5 percent . (**paraphrase**) 2: Shares of Amgen rose $ 2.29 , or 2.2 percent , to $ 41.10 in after-hours trading . [SEP] Shares of Amgen , a division of Sanofi-Aventis , rose $ 1.62 , or 1.6 percent , to $ 41.06 in after-hours trading .(**paraphrase**) 3: Shares of General Electric Co . GE.N rose more than 6 percent on the New York Stock Exchange , while shares of PepsiCo Inc . PEP.N rose 4.7 percent . [SPE] General Electric 's shares jumped almost 6 percent on the New York Stock Exchange , while PepsiCo 's climbed 4.7 percent . (**paraphrase**)

TABLE 6: **MNLI**: Two labeled examples, along with 3 nearest neighbors (based on RoBERTa representations) from our synthetic dataset. We include **labels** for original examples and **pseudo-labels** for synthetic examples in parenthesis.

One of our number will carry out your instructions minutely. [SEP] A member of my team will execute your orders with immense precision. (**entailment**)
KNN: 1: We are at your disposal to help you with your investigation and provide a full range of pro bono services. [SEP] We are the only ones who can help you with your investigation. (**neutral**) 2: I will speak with the chief officer of the contractor, who will be informed about the results of this effort. [SEP] The contractor is being informed about the results of the effort. (**entailment**) 3: We have an office here to assist you. [SEP] An office is where we will assist you, said the manager. (**neutral**)
Conceptually cream skimming has two basic dimensions - product and geography. [SEP] Product and geography are what make cream skimming work. (**neutral**)
KNN: 1: There are two main types of analysis and they are the case study and the case report. [SEP] The case study is the most popular method used to analyze a subject. (**neutral**) 2: A third approach to capturing and using this type of experience is to engage the program management and finance systems of the organization. [SEP] There are two strategies to capturing and using experience. (**contradiction**) 3: The first is to see the basic elements of a business model in action. [SEP] Basic elements of business models are the most important for the success of any company. (**neutral**)
I don't mean to be glib about your concerns, but if I were you, I might be more concerned about the near-term rate implications of this $1. [SEP] I am concerned more about your issues than the near-term rate implications. (**contradiction**)
KNN: 1: I'm not here to tell you of my own experiences, but they are important to others who might have similar concerns. [SEP] If you were to have similar concerns, I'd like to encourage you to tell them to me. (**neutral**) 2: I don't mean to sound judgmental, but as a person, I think that's an issue you're probably pretty much on your own if you think about it. [SEP] You're probably right if you think about it. (**neutral**) 3: But I don't mean to take your word for it. [SEP] I know you are correct, but I want to make sure it's clear that I do not agree. (**contradiction**)