

©Copyright 2025

Yuanhe Tian

Knowledge-driven Natural Language Understanding

Yuanhe Tian

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Fei Xia, Chair

Yan Song

Shane Steinert-Threlkeld

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Knowledge-driven Natural Language Understanding

Yuanhe Tian

Chair of the Supervisory Committee:

Fei Xia

Department of Linguistics

Recent advances in natural language processing (NLP) mainly relied on pre-trained language models (PLMs) that are trained on vast amounts of data. Although these PLMs achieve remarkable success in improving NLP performance over conventional approaches, they still struggle to accurately understand the semantics of the text. Therefore, extra knowledge, especially the dynamically extracted one, is expected to be leveraged to improve the understanding of the models in the text. This thesis proposes a knowledge-driven approach for NLP that improves PLMs. By dynamically integrating external knowledge from multiple sources, the proposed approach enhances model generalization in different scenarios. Specifically, the thesis leverages three types of knowledge, namely, the lexicon knowledge (e.g., n-grams) extracted directly from raw data, syntax knowledge (e.g., dependency parse trees) obtained through existing toolkits, and the pattern knowledge (e.g., vectors) captured during the training process. Several novel architectures are proposed to leverage the knowledge, such as key-value memory networks for incorporating wordhood information, span attention mechanisms with categorical grouping for improved syntactic parsing, and graph convolutional networks to further enrich contextual representations. Extensive experiments on different NLP tasks at various levels demonstrate the effectiveness of the proposed approaches, which outperform strong baselines and existing studies. Overall, this dissertation not only broadens the definition and utilization of knowledge in natural language processing but also lays a solid foundation for future research in multi-modal, cross-domain, and low-resource environments.

Table of Contents

	Page
List of Figures	iii
Glossary	vii
Chapter 1: Introduction	1
1.1 Background	2
1.2 Research Questions and Goals	6
1.3 Contributions	8
1.4 Outline	9
Chapter 2: Literature Review	10
2.1 Knowledge Usage in NLP	10
2.2 Knowledge Sources for NLP	11
2.3 Knowledge Integration Techniques	13
Chapter 3: Nature Language Processing with Different Types of Knowledge	18
3.1 Knowledge Types	18
3.2 Nature Language Processing Tasks	24
3.3 Preliminary Approaches	27
3.4 Combining Knowledge, Tasks, and Approaches	30
Chapter 4: Leveraging Lexicon Knowledge	33
4.1 Chinese Word Segmentation	33
4.2 Constituency Parsing	46
4.3 CCG Supertagging	60

4.4	Summary	68
Chapter 5:	Incorporating Syntax Knowledge	70
5.1	Named Entity Recognition	70
5.2	Relation Extraction	82
5.3	Aspect-based Sentiment Analysis	96
5.4	Summary	109
Chapter 6:	Using Pattern Knowledge	110
6.1	Multimodal Sentiment Analysis	110
6.2	Medical Large Language Modeling	123
6.3	Summary	139
Chapter 7:	Conclusion	140
Bibliography	142

List of Figures

Figure Number	Page
1.1 The development of the natural language process (NLP) approaches, where the focus of this thesis is highlighted in orange color. “PLM” is the abbreviation for pre-trained language models.	2
4.1 The architecture of WMSeg. “ \mathcal{N} ” denotes a lexicon constructed by wordhood measures. N-grams (keys) appearing in the input sentence “部分居民生活水平” (<i>some residents’ living standard</i>) and the wordhood information (values) of those n-grams are extracted from the lexicon. Then, together with the output from the text encoder, n-grams (keys) and their wordhood information (values) are fed into the memory module, whose output passes through a decoder to get final predictions of segmentation labels for every character in the input sentence.	35
4.2 The F-scores of WMSeg (BERT) using three different wordhood measures, namely AV (red) and PMI (green), on five benchmark datasets.	45
4.3 Heatmaps of weights learned for (a) keys and (b) values in the memory, and (c) the tags from the decoder, with respect to each character in an input sentence. Higher weights are visualized with darker colors.	46
4.4 The architecture of the chart-based constituency parser with span attention, with an example partial input sentence and its output. The right part of the figure shows the categorical span attention, where extracted n-grams in span (n', n'') are categorized by their length so that n-grams in different categories are weighted separately (different colors refer to different n-gram categories). Note that for normal span attention, all n-grams are weighted together, where attention $w_{n', n''}$ directly corresponds to $\alpha_{n', n''}$, in the figure.	47
4.5 The F1 curves with respect to the minimal test sentence length (the horizontal axis) of different models performed on ATB (a), CTB (b), and PTB (c).	57
4.6 The F1 curves with respect to the max length (the horizontal axis) of n-grams used in different models performed on ATB (a), CTB (b), and PTB (c).	58

4.7	The histograms of average weights assigned to n-gram categories in different lengths, with weights from SA and CatSA show different patterns.	59
4.8	An example sentence with its parsing results from the best-performing baseline and our model. The correct and wrong parsing results are highlighted on the span labels by green and red, respectively. The superscripts on the span labels illustrate their heights. “V” is a POS tag so there is no height for it.	60
4.9	The architecture of our CCG supertagger with A-GCN and an example input sentence with its supertagging and parsing output. The supertagging process for “buy” is highlighted in green. The adjacency matrix illustrates the edges of the graph that is built upon the chunks (n-grams) extracted from the lexicon \mathcal{N} , with the chunks illustrated in the red boxes.	61
4.10	Examples of the two types of edges for building the graph in an input sentence, in which chunks (n-grams) extracted from the lexicon \mathcal{V} are highlighted in green; example in-chunk and cross-chunk edges are marked in blue and red color, respectively.	62
5.1	The overall architecture of BioKMNER. The top part of the figure shows the syntactic information extraction process: for the input word sequence, we firstly use off-the-shelf NLP toolkits to obtain its syntactic information (e.g., syntax tree), then map the context features and the syntactic information into keys and values, and finally convert them into embeddings. The bottom part is our sequence labeling based BioNER tagger, which uses BioBERT as the encoder and a softmax layer as the decoder. Between the encoder and decoder are the key-value memory networks (KVMN) which weighs syntactic information (values) according to the importance of the context features (keys). The output of KVMN is fed into the decoder to predict output labels.	72
5.2	Three types of syntactic information extracted for an example “ <i>Dihydropyrimidine dehydrogenase deficiency is an autosomal recessive disease</i> ” in the biomedical domain. The context features and their corresponding POS labels, syntactic constituents, and dependency relations for x_5 =“ <i>deficiency</i> ” are highlighted in part (a), (b), and (c) respectively.	73
5.3	The figure presents an example of syntactic information and the context features for “ <i>dystrophy</i> ”. The weights for syntactic information learned from the memories are highlighted with the darker color referring to greater value.	81
5.4	An illustration of noises in the dependency tree that may hurt relation extraction, where the word dependency connected in between “ <i>pumpkin mixture</i> ” and “ <i>bowl</i> ” (whose relation is <i>content-container</i>) may introduce confusion to the system when the object is to predict the relation between “ <i>milk</i> ” and “ <i>pumpkin mixture</i> ” (whose relation is <i>entity-destination</i>).	83

5.5	The overall architecture of the proposed T-GCN for RE illustrated with an example input sentence (the two entities “ <i>defamation</i> ” and “ <i>bishop</i> ” are highlighted in blue and red colors, respectively) and its dependency tree. The left part shows our T-GCN model, where the attention weights are applied to different connections to model the dependency type-aware contextual information. The right part illustrates the adjacency matrix \mathbf{A} for the dependency graph and the process to compute the attention weights (i.e., $p_{i,j}^{(l)}$) for different connections.	84
5.6	An illustration on the two (i.e., local and global) groups of dependency connections for an example sentence (entities are highlighted in red color) with an adjacency matrix (on the right) built upon all connections from the two groups. Local and global connections are represented in orange and blue colors, respectively,	87
5.7	Performance (F1 scores) of different models (i.e., BERT-only, two layers of standard GCN, and two layers of T-GCN) with the BERT-large encoder on three groups of test instances from SemEval, where each group is generated based on the distance (i.e., number of words) between two entities in an instance.	93
5.8	Visualizations of weights assigned to different dependency connections of T-GCN (Full) and T-GCN (L + G) for an example input, where darker and thicker lines refer to connections with higher weights.	96
5.9	An example sentence (including the aspect term “ <i>drink menu</i> ”) with its dependency parsing result.	97
5.10	The overall architecture of our approach with an example sentence-aspect pair input (the aspect words “ <i>dink menu</i> ” are in boldface) from a sentence. Our T-GCN and ALE are marked on top of the figure.	98
5.11	An illustration of how we build the type-aware graph from dependency parsing results and the detail of a T-GCN layer that consumes the graph. Edges and their dependency types are illustrated in the adjacency matrix and the relation matrix, respectively.	99
5.12	The illustration of how we compute $\mathbf{h}_{n'}^{(l)}$ for $x_3 = \text{“menu”}$ through a T-GCN layer. All words $x_{n''}$ connected to “ <i>menu</i> ” with their dependency types (in embeddings $\mathbf{r}_{n',n''}$) are shown at the bottom part.	100
5.13	The histograms of weights assigned to different T-GCN layers (blue, green, and orange bars refer to the weights for the 1st, 2nd, and 3rd layer, respectively) in ALE with respect to each dataset.	107
5.14	Visualization of the weights assigned to different edges and dependency types in each T-GCN layer for an example sentence with two aspects (in red) in conflict sentiment polarities. The edge and type weights (in blue) for “ <i>OK</i> ” in the first and second layer are illustrated on the left, while such weights (in green) for “ <i>food</i> ” and ALE weights (in yellow) for each layer are illustrated on the right. Deeper color refers to the higher weight.	108

6.1	Three memes: (a) is hateful; (b) and (c) are not. Here, (a) and (b) share the same text, and (a) and (c) share the same image. This example shows that a meme can be hateful even when neither its image nor its text is.	111
6.2	The overall architecture of our approach using HMD as a demonstration example. The top part illustrates our backbone model with cross-modal memory, and the bottom part shows the RCR process. The workflow of collecting data and training the reward model is illustrated in orange arrows in the RCR part. Then the trained reward model is used in the self-rejection training to optimize the model, illustrated at the bottom right part. Visual encoding and cross-modal memory are shared by the backbone model in the main process and the reward model training process, which is marked by the boxes with dashed black lines.	112
6.3	Curves of model performance on HMC and Memeplate with respect to different numbers of memory vectors used in the memory module.	122
6.4	Demonstration of three memes for multimodal sentiment analysis with the polarities predicted by different models. The gold standard sentiment labels of all memes are also presented. The model produced labels that match the gold standard and the labels that do not match are highlighted in green and red colors, respectively. . . .	123
6.5	An illustration of the overall training process of the ChiMed-GPT, which consists of three stages, including pre-training, supervised fine-tuning, and reinforcement learning from human feedback (RLHF).	125
6.6	Average bias scores of different LLMs on CAMI and MICA scales, where higher scores indicate more severe bias. The ranges for scale scores are also illustrated below the scale name for better illustration.	138

Glossary

ABSA: Aspect-based Sentiment Analysis

AGI: Artificial General Intelligence

AI: Artificial Intelligence

BIONER: Biomedical Named Entity Recognition

CCG: Combinatory Categorical Grammar

COT: Chain-of-Thought

CWS: Chinese Word Segmentation

CNN: Convolutional Neural Networks

GCN: Graph Convolutional Networks

KVMN: Key-value Memory Networks

HMM: Hidden Markov Models

LLM: Large Language Model

LSTM: Long Short-term Memory

MLP: Multi-Layer Perceptrons

NER: Named Entity Recognition

NLP: Natural Language Processing

NLU: Natural Language Understanding

OOV: Out-of-vocabulary

PLM: Pre-Trained Language Model

POS: Part-of-speech

QA: Question-answering

RAG: Retrieval-Augmented Generation

RE: Relation Extraction

RLHF: Reinforcement Learning From Human Feedback

RNN: Recurrent Neural Networks

SEQ2SEQ: Sequence-to-sequence

SFT: Supervised Fine-tuning

SOTA: State-of-the-art

Acknowledgments

I express my deepest gratitude to my committee members, Professors Fei Xia, Yan Song, and Shane Steinert-Threlkeld, for their insightful suggestions and encouragement that enriched this research, especially my advisor, Professor Fei Xia, and co-advisor, Professor Yan Song, who provide unwavering guidance, patient support, and constructive feedback throughout my entire Ph.D. journey. Much of the content presented in this thesis was derived from the papers published during my PhD studies, and I am particularly grateful to my co-authors for their collaboration, valuable insights, and unwavering support throughout our joint work. Special thanks are due to the faculty and staff at the University of Washington's Department of Linguistics, as well as to my colleagues and friends for their constant support and stimulating discussions. Finally, I extend my heartfelt appreciation to my family, whose love, patience, and unwavering belief in me have been the foundation of my academic pursuits.

Dedication

This dissertation is dedicated to my family and friends, who have been a constant source of strength and inspiration throughout my academic journey. Their endless encouragement and support have made this work possible, and I am forever grateful for their love and guidance.

Chapter 1

Introduction

Modern natural language processing (NLP) models, such as Transformer-based architectures (Vaswani et al., 2017) (e.g., BERT (Devlin et al., 2019) and GPT (Radford et al., 2018)), have demonstrated remarkable capabilities in learning patterns from vast amounts of data. However, these models still face challenges in understanding the deep semantics of texts and performing accurate reasoning processing, and thus require additional knowledge sources (e.g., knowledge bases) to obtain better performance (Laskar et al., 2024; Li et al., 2024; Matarazzo and Torlone, 2025). Typically, knowledge sources refer to the existing knowledge bases or knowledge graphs (Miller, 1995; Baker et al., 1998; Mendes et al., 2012), which are fixed and unable to be modified based on the input. In practice, there are other types of information, such as the raw text from online resources (Gabrilovich and Markovitch, 2009; Zhu et al., 2022), the parse tree of the input sentence (Schuster and Manning, 2016; Huang and Carley, 2019), and feature vectors of memory module (Hatalis et al., 2023; Wang et al., 2023b), that are useful for text understanding. Therefore, one is able to extend the concept of “*knowledge*” to cover a broader type of sources, referring to any useful information that dynamically aids in decision-making and task execution rather than that confined to the conventional notion of static information. This research explores how to utilize knowledge that is obtained automatically from toolkits or learning processes to improve NLP tasks, where three different types of knowledge in various forms, namely, ngrams, parse trees, and vectors, are studied.

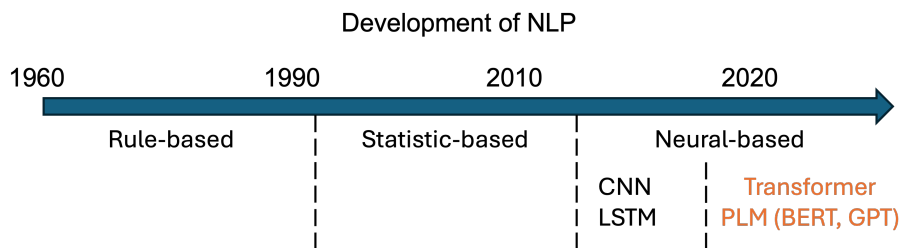


Figure 1.1: The development of the natural language process (NLP) approaches, where the focus of this thesis is highlighted in orange color. “PLM” is the abbreviation for pre-trained language models.

1.1 Background

NLP has undergone several paradigm shifts over the past decades. It develops from rule-based models (Sumita and Iida, 1991; Somers, 1999; Polanyi et al., 2004) through statistical learning approaches (Klein and Manning, 2003; Forman, 2003; Koehn, 2009) to the latest neural network-based models (Devlin et al., 2019; Tian et al., 2020e; Touvron et al., 2023a), which are summarized in Figure 1.1. The NLP approaches across different periods generally include a foundational paradigm along with additional enhancements built upon it. Among all these enhancements, the use of knowledge is particularly representative and has been a consistent theme throughout the development of NLP. In this section, we first present the changes in the NLP paradigm, followed by the background of knowledge utilization within it.

1.1.1 NLP Paradigms

Early NLP models are mainly rule-based (Sumita and Iida, 1991; Somers, 1999; Polanyi et al., 2004), relying on handcrafted linguistic rules and symbolic representations of knowledge. These rule-driven approaches are often precise in narrow domains but require extensive manual effort by experts to enumerate linguistic and world knowledge, making them brittle and not easily scalable to the diversity of natural language. Subsequently, as a series of natural language corpora (e.g., PTB (Marcus et al., 1993; Xue et al., 2005)) are constructed to provide a data foundation for NLP development, statistical approaches gradually become mainstream (Klein and Manning, 2003; Forman, 2003; Koehn, 2009). These approaches avoid manual feature engineering and expert knowledge; they instead learn patterns

from large-scale data for NLP tasks and outperform the rule-based approaches.

In order to improve the capability of models to capture deep contextual features and to handle unseen cases, approaches based on neural networks are proposed (Wang et al., 2016a; Liu et al., 2016a; Lyu et al., 2017), where better performance is obtained on many NLP tasks. Earlier neural network approaches employ multi-layer perceptrons (MLPs), but they face challenges such as limited feature extraction capabilities and scalability issues. Subsequently, more complicated neural network architectures are proposed to address these issues. For example, convolutional neural networks (CNNs) (LeCun et al., 1998) primarily extract local features through a layered structure, yet they are not effective in handling sequential data that requires global understanding. To overcome this, approaches based on recurrent neural networks (RNNs) (Rumelhart et al., 1986) are proposed, utilizing recurrent connections to process sequences and capture temporal dependencies between tokens. In particular, long short-term memory (LSTM) (Hochreiter, 1997) is a type of RNNs that employ gated mechanisms to regulate information flow to overcome the vanishing gradient problem in the standard RNN. It shows promising capability to capture contextual information of the running text and thus achieves state-of-the-art performance on many NLP tasks such as named entity recognition (NER), parsing, and machine translation (Limsopatham and Collier, 2016; Kiperwasser and Goldberg, 2016; Zhou et al., 2016a). Despite these improvements, LSTM still faces challenges such as low computational efficiency and difficulties in capturing long-distance¹ relations between tokens.

To address these challenges, the Transformer (Vaswani et al., 2017) model is proposed and becomes the dominant approach in the field. The Transformer architecture utilizes a self-attention mechanism that enables the parallel processing of input tokens, thereby enhancing computational efficiency and facilitating model scalability. Moreover, the attention mechanism effectively identifies important tokens, allowing the model to focus on critical information and improve contextual encoding. Meanwhile, the rapid expansion of internet data has spurred the development of Transformer-based **pre-trained language models (PLM)**, which are initially trained on vast unlabeled data and then fine-tuned for specific tasks. Two representative PLMs are BERT (Devlin et al., 2019) and GPT series (Brown et al., 2020), which achieve remarkable performance on many NLP tasks. For BERT, it employs a bidirectional encoding mechanism to capture context from both directions, and is able

¹The “*distance*” means the number of tokens between two tokens in a sentence.

to be adapted to a variety of NLP tasks by firstly being pre-trained on large corpora via masked language modeling and next sentence prediction and then fine-tuned on labeled task data through supervised learning. For the GPT series, they adopt an auto-regressive, unidirectional approach, and are pre-trained on large text corpora to predict the next token and are subsequently fine-tuned on NLP tasks, especially the generative ones, to enhance downstream performance.

Particularly, with the evolution of GPT series from relatively small PLMs (e.g., GPT (Radford et al., 2018)) with millions of parameters to larger models (e.g., GPT-3 (Brown et al., 2020)) with billions of parameters, researchers find the zero-shot and few-shot learning capabilities of these models with more training data and larger model size. However, these models still primarily focus on text completion and lack the capability to understand and follow human instructions. To enhance their instruction-following abilities and align with human preferences, a three-stage training paradigm, which includes pre-training, supervised fine-tuning (SFT), and reinforcement learning from human feedback (RLHF), for LLMs is proposed (Ouyang et al., 2022). Being trained through the paradigm, LLMs (e.g., GPT-4 (Achiam et al., 2023)) emerge intelligence by presenting the capability of understanding, memorizing, generating, and reasoning, although challenges such as hallucinations still remain (Touvron et al., 2023b; Taori et al., 2023b; Wang et al., 2024b). Therefore, LLMs become the main focus of NLP research, where techniques such as chain-of-thought (CoT) prompting (Wei et al., 2022) and retrieval-augmented generation (RAG) (Lewis et al., 2020b) are proposed to enhance their capabilities. Nowadays, PLMs (particularly the LLMs) become the foundational backbone of NLP, with emerging trends of applying them to NLP tasks from the general domain to a specific domain, employing them for unimodal tasks to multimodal ones, and even utilizing them for artificial general intelligence (AGI), such as agents or integrating them with robotics (Zeng et al., 2023; Gao et al., 2024).

1.1.2 Enhancing NLP with Knowledge

Despite remarkable progress in the paradigm of NLP, many challenges remain to be addressed, one of which is the difficulty of having an accurate understanding of the text, which helps the model to produce reliable output. To improve text understanding, current deep learning models typically rely on training data, especially the annotated data, so as to learn the semantics of the texts as well as

the correlation between the text features and the desired output. Therefore, the neural models may not perform well in scenarios where a large amount of training data is not available. Furthermore, it is generally hard for these models to apply to domains that differ from their training data since the models lack the domain knowledge to address the domain-specific tasks, which restricts the applicability of models across diverse fields. Even with enough training data, current models still show shortcomings when processing complex contexts. Particularly when handling long texts, models often struggle to capture the key semantic information that provides hints for performing the task. For example, in tasks that need to analyze the fine-grained sentiment of a particular term, LLMs may fail to capture the local semantic relation between the modifier and the term, and predict the sentiment of the entire text; in tasks that require reasoning, LLMs may produce fluent text, but the reasoning process is problematic. This limitation causes models to produce errors or unreasonable responses when they are applied to challenging tasks. Therefore, further enhancement is needed to improve the models' understanding of the text.

Among different types of enhancements, knowledge that provides helpful hints for the task is able to improve the model performance on NLP tasks. For example, in low-resource settings where there is insufficient training data, conventional deep learning approaches struggle to effectively learn the way to process and understand the semantics and context information of the input. When external knowledge, such as domain-related texts and reference materials, is leveraged, it is able to improve the generalization capability of existing models without requiring intensive training (Hedderich et al., 2021). For instance, in the machine translation task for low-resource languages, a list of parallel word pairs between the languages is able to improve the translation quality (Jean et al., 2015; Gu et al., 2018; Ranathunga et al., 2023). Meanwhile, in cross-domain settings where a model trained for a particular domain is applied to another domain, the model generally obtains inferior performance since it lacks the domain knowledge to perform the task. For example, a model trained on general domain data may not be able to perform well on medical data since the language used in the medical domain differs from that in the general domain, which makes it hard for the model to capture the task-related features and leads to inferior performance. Knowledge is able to bridge the gap and help the model to achieve better performance, which has been demonstrated by many existing researchers (Roy and Pan, 2021; Gilson et al., 2022).

To leverage knowledge, many existing studies focus on existing knowledge bases or knowledge

graphs (e.g., WordNet (Miller, 1995), FrameNet (Baker et al., 1998), and ConceptNet (Speer and Havasi, 2013)). In this thesis, we name knowledge from these resources as **static knowledge** since these knowledge bases are pre-constructed manually or semi-automatically and remain relatively unchanged once established. These static knowledge resources face some problems. First, they are hard to scale to address new information. Second, they have long update cycles, which prevent them from applying to domains where the topics and word usage change rapidly (e.g., the social media domain). Therefore, new types of knowledge are highly needed. In contrast to static knowledge, there is another type of knowledge that is automatically obtained from data, such as the parse tree of the sentence. We name this type of automatically acquired knowledge as **dynamic knowledge**, and it is valuable to explore the approaches to leverage dynamic knowledge.

There are various types of dynamic knowledge utilized by language models. Conventional forms include n-grams and syntactic parse trees, which can be explicitly extracted from text and incorporated as additional inputs to enhance model performance. With the advent of LLMs, however, the concept of knowledge representation has expanded from explicit, tangible forms to implicit, abstract representations encoded directly within the model’s parameters. This evolution is driven by the substantial parameter size of LLMs, which allows them to learn and memorize deeper and more nuanced forms of knowledge, such as common-sense reasoning and world knowledge, that smaller models are typically incapable of capturing effectively. As a result, LLMs have demonstrated emergent capabilities indicative of enhanced intelligence (OpenAI, 2023; Taori et al., 2023a; Touvron et al., 2023b). Within this context, researchers increasingly regard the training of LLMs as a form of knowledge acquisition. This perspective becomes particularly prominent during domain-specific or task-specific fine-tuning processes, where LLMs learn specialized knowledge from domain-specific datasets (Xu, 2023; Han et al., 2023). The outcome of this learning is effectively captured and represented through the model’s parameters, demonstrating the shift from externally provided explicit knowledge to internally encoded implicit knowledge representations.

1.2 Research Questions and Goals

Based on the research background and motivations, this thesis primarily addresses the following overarching research question:

Given a specific NLP task, how to determine what types of knowledge are relevant, how to effectively acquire such knowledge, and how to integrate it efficiently into the models to enhance performance?

This question comprises three interrelated components. First, identifying the **relevant knowledge** requires examining the characteristics of tasks, such as their granularity, domain specificity, and underlying linguistic phenomena, to determine suitable knowledge forms and sources. Second, addressing the **acquisition of knowledge** needs to effectively leverage automated knowledge extraction and learning approaches for dynamically obtaining task-specific knowledge. Lastly, the question of **knowledge integration** requires developing model architectures to encode, filter, and dynamically update knowledge representations, enabling the model to leverage knowledge during inference.

Therefore, the overall objective of this thesis is to develop approaches that integrate diverse types of knowledge into NLP models, and thus improve the performance on various tasks across different levels. In other words, this thesis identifies which types of information are appropriate for a given task and to design model architectures that are able to effectively incorporate external knowledge during training and inference. In this thesis, we focus on three categories of knowledge: information extracted from raw data through unsupervised approaches, knowledge obtained via external toolkits, and features learned from the training process. Each category provides unique advantages in terms of the level of context information it contributes to the task. For instance, information from raw data, such as n-grams extracted using unsupervised approaches, is used to offer local contextual clues for tasks at the lexical level. Outputs from external toolkits, such as dependency parse trees, provide structural details that support semantic and pragmatic understanding. Furthermore, knowledge derived from the learning process appears as task-specific vectors or soft prompts that enhance the model's performance on complex application-level tasks.

Meanwhile, this thesis also aims to establish a framework that determines the appropriate type of information for a specific task. Tasks at the level of word segmentation or syntactic parsing require simpler forms of contextual clues, whereas tasks related to information extraction, sentiment analysis, or dialogue generation need higher-level representations and reasoning abilities. By carefully matching the form of external information to the needs of each task, our approaches are expected to perform well even in different scenarios.

A further goal is to design architectures that effectively incorporate external information into

deep learning models. Specifically, we propose specialized mechanisms such as key-value memory networks (KVMN), attentive graph convolutional networks (A-GCN), and reinforced optimization approaches. These components enable the model to selectively leverage information that is most relevant to the task while reducing noise from less useful features. Overall, this thesis establishes a unified approach for leveraging external information to enhance model generalization and adaptability, particularly in challenging or dynamic application scenarios.

1.3 Contributions

This thesis makes several contributions by proposing frameworks for dynamic external knowledge integration and designing model architectures that incorporate diverse knowledge sources into PLMs. The details are illustrated as follows.

First, this thesis utilizes various types of knowledge in different forms, namely, ngrams, parse trees, and vectors that are obtained from raw data, existing toolkits, and the learning process, respectively, for diverse NLP tasks. Specifically, extracting n-grams from raw text provides local cues that support tasks that require shallow understanding of the text; dependency structures from external toolkits offer structured guidance for tasks that require a deep processing of the text; knowledge vectors that are learned during the training process are able to provide hints for complicated tasks that need to understand the core semantics of the data. Second, this thesis proposes novel model architectures that fuse diverse external information with PLMs. We develop specialized modules, including KVMN, A-GCN, and attentive layer ensembles, which allow different forms of knowledge, such as sequential, graph-structured, or vectorized knowledge, to be incorporated into task-specific representations. Finally, empirical evaluations show that our integration mechanisms consistently enhance performance across a range of NLP tasks, such as parsing, named entity recognition, relation extraction, and aspect-based sentiment analysis. Moreover, our approach effectively addresses the noise in the knowledge while preserving essential context, enabling robust and adaptive performance in both standard and challenging scenarios.

1.4 Outline

The thesis is organized as follows. In Chapter 2, we provide a literature review on knowledge usage in NLP. In Chapter 3, we provide an overview of the knowledge sources, approaches, and tasks that are going to be discussed in the thesis. In Chapter 4, Chapter 5, and Chapter 6, we present our study on leveraging knowledge from raw data, toolkits, and learning processes, respectively. In Chapter 7, we draw conclusions to summarize the paper.

Chapter 2

Literature Review

In the literature review, we illustrate existing studies for knowledge usage in NLP, knowledge sources, and knowledge integration techniques.

2.1 Knowledge Usage in NLP

With the development in NLP, the use of knowledge undergoes remarkable changes. Early NLP approaches primarily rely on manually crafted rules and knowledge (Sumita and Iida, 1991; Somers, 1999; Polanyi et al., 2004). At this stage, the most notable characteristic is that linguistic rules or predicate logic are applied directly to process natural language, leading to the emergence of rule-based NLP systems, such as ELIZA (Weizenbaum, 1966), SHRDLU (Winograd, 1971), and LUNAR (Woods, 1973). ELIZA uses a pattern-matching approach to simulate a psychotherapist engaging in dialogue. SHRDLU employs a rule-based and dialogue management approach to handle tasks involving robotic manipulation of objects. LUNAR applies a fact-querying approach to interpret and answer questions regarding lunar rocks. As computational power increases and corpora accumulate, statistical approaches such as n-gram language models, hidden Markov models, and maximum entropy models, increasingly dominate NLP. Although the approach shifts from rule-driven to data-driven, knowledge continues to play a crucial role. For instance, the PCFG (Charniak et al., 2006) syntactic parsing approach integrates rules and statistics by assigning statistical probabilities to grammatical rules to aid in resolving ambiguities. Similarly, linguistic knowledge is harnessed to

build feature templates, which facilitates the more effective extraction of task-relevant features.

With the advent and evolution of PLMs (including the LLMs), NLP gives rise to paradigms such as pre-training and fine-tuning, zero-shot, few-shot, chain-of-thought, and retrieval-augmented generation, thereby transforming the approach to knowledge utilization (Touvron et al., 2023a,b; Taori et al., 2023b; Chiang et al., 2023). During this period, PLMs and LLMs are characterized by having more parameters and by training on massive datasets, which allows them to embed the knowledge present in the training data within their parameters. Sometimes, this embedded knowledge is sufficient to address the task at hand, shifting the emphasis to activating the relevant information; as a result, numerous prompt engineering approaches emerge. For instances where LLMs do not learn certain knowledge, additional information, such as the knowledge acquired via external toolkits, is provided to help the model learn or utilize this knowledge, thus accomplishing the specific task.

In summary, the application of knowledge evolves from explicit rule-based systems to implicit learning, and from static knowledge bases to dynamic knowledge acquisition. Each stage presents its unique form of knowledge representation and utilization, reflecting the evolution of technology and task requirements.

2.2 Knowledge Sources for NLP

Knowledge plays a vital role in the NLP field, and commonly used forms of knowledge include knowledge bases, data features, toolkit-extracted features, and knowledge vectors (Shrestha, 2014; Huang et al., 2019b; Mandya et al., 2020; Liu et al., 2021). A knowledge base, such as a dictionary or knowledge graph, is generally built through manual or automated approaches and stores carefully curated, structured information with clearly defined relationships. For example, WordNet (Miller, 1992) is a lexical database that groups English words into sets of synonyms and maps their semantic relations; ConceptNet (Liu and Singh, 2004) is a more recent commonsense knowledge base that connects everyday concepts. Structured knowledge bases, enriched by human expertise, present high accuracy and quality, making them suitable for scenarios with little or no data and widely applicable in various NLP research, especially during its early stages. However, these knowledge bases require frequent maintenance, incur high costs, and struggle to adapt quickly to changes, resulting in suboptimal computational efficiency. With the development of data resources, such as

encyclopedic sources like Wikipedia and specialized document repositories, it becomes possible to extract knowledge from the raw data of these datasets. These forms include statistical features (e.g., term frequency–inverse document frequency (TF-IDF)), contextual features (e.g., N-grams), and semantic features (e.g., distributed word embeddings). Such knowledge presents latent patterns in language use, enabling dynamic extraction when needed and supporting NLP systems, yet in a relatively superficial manner.

With the advancement of model training techniques, various knowledge extraction toolkits are used (Manning et al., 2014b; Neumann et al., 2019; Qi et al., 2020). These toolkits are typically trained on manually annotated data or leverage human-designed rules, allowing them to extract deeper insights from data more efficiently and accurately. A typical example is the Stanford CoreNLP toolkit (Manning et al., 2014b), which provides a pipeline for performing a wide range of NLP tasks, such as tokenization, sentence splitting, lemmatization, part-of-speech tagging, named entity recognition, constituency and dependency parsing, coreference resolution, and sentiment analysis. Although these toolkits dynamically extract more in-depth knowledge compared to data-derived features, they sometimes suffer from limited expressiveness and insufficient coverage.

Moreover, to enhance the computability of knowledge and its integration with neural network models, particularly LLMs, vectorized knowledge is developed (Liu et al., 2021). This type of knowledge is obtained either by vectorizing existing knowledge through text representation models or directly from the learning process. In terms of vectorizing existing knowledge, text representation models are employed to encode knowledge bases into vector repositories; for example, the Universal Sentence Encoder (Cer et al., 2018) converts sentences into semantic embeddings, and Sentence-BERT (Reimers and Gurevych, 2019) produces semantically meaningful sentence vectors. From the learning perspective, knowledge is acquired by establishing knowledge-related supervisory signals and updating the parameters of knowledge vectors via backpropagation (Kim et al., 2020). For instance, some studies fine-tune LLMs on domain-specific data to capture specialized knowledge (Xu, 2023), others employ reinforcement learning to capture human preferences (Ouyang et al., 2022), and some utilize soft prompting to learn domain and task-specific knowledge (Lester et al., 2021). Knowledge acquired through the learning process presents high expressiveness and flexibility, yet it suffers from poor interpretability.

In summary, the sources of knowledge are gradually shifting from singular static knowledge

bases to multi-sourced, multi-form approaches; particularly in a data-rich environment, dynamically extracted and learned knowledge attracts significant attention due to its lower acquisition costs and enhanced computational efficiency.

2.3 Knowledge Integration Techniques

In different situations and tasks, the approaches for knowledge fusion vary. In different situations and tasks, the approaches for knowledge fusion vary. In recent years, researchers have developed a variety of techniques that combine multiple sources of information into natural language processing systems. Early work in this area focused on using statistical patterns derived from large-scale unannotated texts. For example, in Chinese word segmentation, methods that extract n-gram patterns using measures such as pointwise mutual information have been incorporated into character-level models (Xue and Shen, 2003; Peng et al., 2004; Levow, 2006; Zhao et al., 2006; Zhao and Kit, 2008; Li and Sun, 2009; Song et al., 2009a; Li, 2011; Sun and Xu, 2011; Mansur et al., 2013; Zhang et al., 2013; Pei et al., 2014; Chen et al., 2015; Ma and Hinrichs, 2015; Liu et al., 2016b; Zhang et al., 2016; Wang and Xu, 2017; Zhou et al., 2017; Chen et al., 2017c; Ma et al., 2018b; Higashiyama et al., 2019; Gong et al., 2019; Qiu et al., 2019). These techniques extract patterns directly from raw data and fuse them with neural representations, thus providing additional local context that improves segmentation performance. Empirical studies reveal that such integration improves evaluation scores in segmentation tasks compared to methods relying solely on character-based features.

Neural models have emerged as powerful frameworks for fusing external information with learned representations. In the domain of parsing, researchers initially developed transition-based approaches (Sagae and Lavie, 2005) and chart-based models (Collins, 1997; Glaysheer and Moldovan, 2006) that incorporate handcrafted features. With the advent of deep learning, recurrent neural networks and later Transformer-based models provided enhanced encoders that capture contextual dependencies over long distances (Dyer et al., 2016; ?; Liu and Zhang, 2017; Stern et al., 2017; Gaddy et al., 2018; Kitaev and Klein, 2018; Kitaev et al., 2019; Fried et al., 2019). In this context, additional knowledge is often integrated by incorporating extra resources. For instance, some studies reformulate parsing as a sequence-to-sequence generation problem (Vinyals et al., 2015; Suzuki et al., 2018) or as a sequence labeling task (Gómez-Rodríguez and Vilares, 2018). Other research

incorporates information from pre-trained embeddings or employs extra grammatical representations such as head phrase structure grammar (HPSG) (Zhou and Zhao, 2019; Mrini et al., 2019) to refine parsing outputs. Such techniques use knowledge extracted from external corpora or computed from automatically parsed text, enabling models to capture both local and global dependencies. Experimental evaluations have shown that these methods yield higher parsing accuracies and better tree structures in comparison with approaches that rely solely on raw sequential representations.

The integration of external information is also widely explored in the field of biomedical named entity recognition (BioNER). Researchers in this area leverage both domain-specific text encoders and external knowledge sources to capture medical entities with higher precision. Early feature-based methods relied on manually designed features and domain-specific lexicons (Song et al., 2005; Rebholz-Schuhmann et al., 2007; Hettne et al., 2009; Liao and Wu, 2012; Leaman et al., 2015). With the emergence of neural models such as biLSTM and later BioBERT (Habibi et al., 2017; Luo et al., 2018; Dang et al., 2018; Lee et al., 2019), the focus shifted toward integrating automatically derived syntactic information with learned representations. Many studies use off-the-shelf tools to obtain syntactic structures—for instance, dependency parse trees and constituent labels—which are then transformed into embedding vectors. These vectors are concatenated with word embeddings or fused using element-wise operations. For example, one approach uses part-of-speech and constituency label embeddings, merging them with word-level features to enhance a biLSTM-CRF model for chemical entity recognition (Luo et al., 2018). Other models such as D3NER (Dang et al., 2018) utilize similar integration strategies, although they generally treat all syntactic signals uniformly. Empirical findings indicate that while these methods often improve BioNER performance over baseline models, they are susceptible to errors introduced by noise in the automatically extracted syntactic information.

In relation extraction, external knowledge is pivotal for capturing long-distance associations between entities. Dependency structures, in particular, offer valuable cues for linking entities that do not appear in close proximity within the text. Several studies adopt graph convolutional networks (GCN) to encode dependency trees and extract features relevant to semantic relations (Zhang et al., 2018; Guo et al., 2019). Some approaches prune dependency graphs using heuristics such as retaining only the shortest dependency path between two entities (Xu et al., 2015b; Miwa and Bansal, 2016), but such strategies are prone to discarding useful contextual signals. Subsequent work uses a trade-off pruning strategy that balances the need to filter out noise with the preservation of essential relational

information (Zhang et al., 2018). Additional strategies include dynamic pruning (Yu et al., 2020) and the application of extra attention layers (Mandya et al., 2020), which provide the model with the capacity to assign varying importance to different dependency edges. These approaches collectively contribute to enhanced precision and recall in relation extraction tasks, as evidenced by improved F1 scores over models that omit dependency structure integration.

Aspect-based sentiment analysis (ABSA) represents another domain where the fusion of external knowledge plays a crucial role. ABSA requires the model to determine sentiment polarity for specific aspects within a sentence. Early models address the task by combining aspect terms with overall sentence representations using recurrent networks or convolutional networks (Dong et al., 2014; Wang et al., 2016b; Tang et al., 2016a; Ma et al., 2017; Chen et al., 2017a; Xue and Li, 2018; Li et al., 2018; Xu et al., 2019). More recent work incorporates dependency parses to build graph-based representations that capture the syntactic relationships between aspect terms and sentiment-bearing words (Huang and Carley, 2019; Sun et al., 2019; Zhang et al., 2019a; Wang et al., 2020; Tang et al., 2020). However, many of these models build graphs solely from the dependency structure without differentiating among dependency types, and they often use only the final layer’s outputs for prediction. As a result, some approaches may neglect important details in the linguistic signals or fail to distinguish between informative and spurious connections. Experiment results have demonstrated that models employing refined graph convolutional networks with attention mechanisms, which incorporate both dependency structure and type information, achieve higher accuracy and more robust sentiment predictions on various benchmark datasets.

In the area of multimodal sentiment analysis, techniques for integrating external knowledge have taken a different form. Recent studies utilize pre-trained visual encoders and language models such as ViLBERT (Lu et al., 2019), VisualBERT (Li et al., 2019), and BLIP2 (Li et al., 2023b) to align image and text features. Most early approaches focus on the matching and fusion of features using simple operations such as concatenation or basic attention mechanisms. However, tasks such as hateful meme detection or multimodal sentiment analysis depend on understanding the contrast between modalities rather than their shared content. In these cases, researchers adopt methods that not only align features but also explicitly model the disparities between image and text. Some studies rely on contrastive learning to separate and highlight distinctive features (Radford et al., 2021; Lee et al., 2022; Wang et al., 2024a). More advanced techniques, such as cross-modal memory modules

and reinforced contrast recognition, have been developed to dynamically extract and emphasize the differences between modalities. These approaches provide additional signals that improve the accuracy of downstream classification tasks. Empirical results across several benchmark datasets show that models incorporating such dynamic contrast mechanisms achieve higher accuracy and better area-under-curve metrics compared with systems that focus solely on feature alignment.

Recent advancements in LLMs have led to a transformation in how knowledge is integrated into natural language processing systems. Earlier approaches for learning text representations were based on statistical models and traditional deep learning methods, but the emergence of models such as BERT (Devlin et al., 2019) and GPT (Brown et al., 2020) shifted the focus toward pre-trained language models that are fine-tuned for various tasks. These models are trained on vast amounts of text data, thereby acquiring latent representations that capture both syntax and semantics. Researchers have extended these techniques to incorporate external knowledge by adjusting model architectures and training procedures. For example, some methods incorporate additional modules, such as key-value memory networks, to merge external domain-specific knowledge with the representations learned during pre-training. Other studies propose RLHF (Ouyang et al., 2022) to adjust model outputs in line with human preferences, thus refining the effective use of dynamic knowledge acquired during training. Empirical findings indicate that these approaches yield models with improved generalization and reasoning capabilities, which are especially important in specialized domains such as medicine and finance.

Across various research domains, the integration techniques for external knowledge show common themes. One key idea is the transformation of external information into a format that is compatible with neural network representations, such as converting syntactic trees or dependency graphs into vectorized embeddings. Another recurring concept is the selective weighting of external signals to reduce the influence of noisy or irrelevant information. Techniques that apply attention mechanisms or reinforcement-based filtering strategies are frequently used to achieve this goal. In tasks such as relation extraction and aspect-based sentiment analysis, models that dynamically weigh the contribution of dependency relations or contrast information yield improved evaluation scores. Additionally, the design of multi-layer aggregation methods, which combine outputs from different stages of processing, contributes to a more robust representation that leverages both local and global contextual cues.

In summary, existing research on knowledge fusion in NLP spans a broad spectrum of approaches. From the extraction of statistical patterns in raw data to the utilization of automatically generated syntactic structures and knowledge acquired through model training, each method offers distinct advantages for different tasks. Empirical studies across Chinese word segmentation, parsing, biomedical named entity recognition, relation extraction, ABSA, multimodal sentiment analysis, and large language modeling consistently demonstrate that incorporating external information leads to improved performance. These results emphasize the importance of designing models that are capable of both integrating diverse knowledge sources and filtering out noise, thereby supporting the development of more adaptive and effective NLP systems.

Chapter 3

Nature Language Processing with Different Types of Knowledge

In this chapter, we present an overview of the thesis, which includes the types of knowledge, the tasks in which the knowledge is applied, the model architectures, and the ways in which we combine them.

3.1 Knowledge Types

In general, knowledge is categorized into knowledge bases that are constructed using manual or automatic approaches and knowledge that is dynamically obtained or learned by automatic approaches in real-time. Because the latter is able to dynamically extract knowledge from input and present superior flexibility, it attracts much attention and is widely used in many existing studies. However, existing research does not effectively address how to utilize this type of knowledge, so our study focuses on it. Among different types of knowledge, in this thesis, we mainly focus on three types based on their form, namely, lexicon knowledge in the form of n-grams, syntax knowledge in the form of parse trees, and pattern knowledge in the form of vectors¹, which are summarized in Table 3.1. In this thesis, we mainly focus on the n-gram, parse tree, and vector knowledge, which are illustrated as follows.

¹The vectors here also include matrices. For simplicity, we use the term *vectors* to refer to both vectors and matrices.

Type	Form	Descriptions	Source
Lexicon	N-grams	Knowledge extracted from data	Raw Data
Syntax	Parse trees	Knowledge extracted from existing toolkits that are trained on labeled data	Toolkits
Pattern	Vectors	Knowledge obtained by learning from one or more supervisions	Learning Process

Table 3.1: The three forms of knowledge studied in the thesis.

3.1.1 N-grams

In this thesis, n-grams are obtained directly from the data through unsupervised approaches². N-grams are the combination of multiple continuous tokens and generally reflect the latent features within the data, such as how different words co-occur in a particular context. N-grams are widely used in existing studies as contextual knowledge, as they contain large-grained contextual information that is helpful for downstream tasks (Song et al., 2009b; Song and Xia, 2012; Ishiwatari et al., 2017; Yoon et al., 2018; Zhang et al., 2019b; Tian et al., 2020b). Specifically, current text encoders treat tokens as the basic units that carry a particular meaning during the encoding process. Because tokens often generate new meanings when they are combined, the semantics of multiple tokens (i.e., an n-gram) is not directly the sum of the meaning of its individual tokens. Therefore, these n-grams provide unique information to assist the encoder in processing text. Therefore, in the thesis, we choose n-grams extracted by unsupervised approaches to explore the way to use it in NLP tasks.

Knowledge Extraction

Although n-grams provide large-grained information, not all arbitrary combinations of tokens are meaningful. Therefore, it is necessary to carefully choose n-grams that carry meaningful information. In doing so, we adopt pointwise mutual information (PMI) (Sun et al., 1998) to extract n-grams.

²There are other approaches to obtain the n-grams, which are not covered by the thesis

Specifically, PMI measures pointwise mutual information between two tokens, namely, x' and x'' , via

$$PMI(x', x'') = \log \frac{p(x'x'')}{p(x')p(x'')} \quad (3.1)$$

where p computes the probability of an n-gram (i.e., x' , x'' , and $x'x''$) in a dataset. A high PMI score indicates that the two adjacent tokens co-occur a lot in the dataset and are likely to form a meaningful n-gram. For each pair of adjacent tokens x_{n-1}, x_n in a sentence $\mathcal{X} = x_1x_2 \cdots x_{n-1}x_n \cdots x_N$, we compute the PMI score of the two tokens and use a threshold to determine whether a delimiter should be inserted between them. PMI score lower than the threshold results in a segmentation, which segments the sentence \mathcal{X} into pieces of n-grams.³ Using PMI, one collects the resulting n-grams from the segmented sentence and constructs an n-gram lexicon \mathcal{V} , accordingly.

Knowledge Properties

N-gram knowledge has the following properties. First, in this thesis, the knowledge is obtained through an unsupervised approach (i.e., PMI) that highlights the co-occurrence statistics of different tokens within the data, without resorting to any manually annotated sources. This makes such knowledge particularly focused on providing shallow contextual information. Second, the knowledge extracted by this unsupervised approach is not entirely accurate and may contain noise and thus needs to be carefully modeled when leveraging it for NLP tasks. Finally, this knowledge is employed in various ways. For example, n-grams themselves are used as basic units to provide contextual information. Moreover, because the n-gram extraction approach (i.e., PMI) is based on the co-occurrence of tokens, there are relations between the tokens within the same n-grams and across different n-grams in a sentence. That is, the tokens within an n-gram are more tightly connected, whereas the relationships between distinct n-grams are relatively looser, which is similar to the relations between morphemes within a word versus the syntactic relationships between words. These properties provide the foundation for designing and selecting appropriate model architectures and NLP tasks.

³For example, a sentence is segmented into $\mathcal{X} = x_1/x_2x_3x_4/x_5$ (“/” refers to a delimiter) if the PMI of x_1, x_2 and x_4, x_5 are lower than the threshold and the PMI of x_2, x_3 and x_3, x_4 are greater than the threshold; we thus obtain three n-grams, i.e., $x_1, x_2x_3x_4$, and x_5 from this sentence.

3.1.2 Parse Tree

Generally, parse trees are extracted using existing toolkits, such as parsers. Because these toolkits are typically trained on manually annotated datasets or leverage knowledge bases constructed by humans, the knowledge they extract indirectly incorporates human expertise. One representative type of parse tree is the dependency tree, where each node in the tree represents a token, and every token is connected to a unique head token with a directed dependency relation (such as a nominal subject). The head token and the connected token are named the governor and the dependent, respectively. In a dependency tree, there is exactly one unique dependency path (if the direction of the connection is ignored) connecting any two tokens. The length of this path (i.e., the number of edges it contains) naturally reflects the syntactic distance between these tokens. To characterize such syntactic distances explicitly, the concept of *order* in dependency parsing is introduced. Specifically, a *first-order* dependency considers only individual dependency relations between a governor and its dependent. A *second-order* dependency further includes interactions between two adjacent dependencies, such as sibling or grandparent relations. Extending this, a *third-order* dependency captures even richer contexts by modeling interactions among three related dependencies. Such higher-order dependency representations encode more comprehensive syntactic structures, effectively leveraging long-distance⁴ contextual information beneficial to NLP tasks (Sun and Xu, 2011; Song and Xia, 2012; Gong et al., 2012; Song et al., 2012b; Xu et al., 2015b; Chen et al., 2017a; Zhang et al., 2019b; Tang et al., 2020). Given the potential of the dependency knowledge to be beneficial for NLP tasks, we focus on it in this thesis.

Knowledge Extraction

In general, parsers are trained on human-annotated datasets, which grants them a certain level of accuracy and enables them to capture the syntactic structure of sentences. We utilize off-the-shelf toolkits to extract the dependency parse tree, which is represented by a list of dependency tuples $(x_{n'}, x_{n''}, r_{n',n''})$ with $r_{n',n''}$ denoting the dependency type between $x_{n'}$ (the governor) and $x_{n''}$ (the dependent). Since the dependency parse tree is a graph over the tokens, we use an adjacency matrix $\mathbf{A} = \{a_{n',n''}\}_{N \times N}$ to present the graph by recording token relations in all tuples and a relation type

⁴“Long-distance” here refers to tokens that are far apart in the original text.

matrix $\mathbf{R} = \{r_{n',n''}\}_{N \times N}$ to represent the edges with their dependency types. Therefore, \mathbf{A} is a 0-1 matrix where $a_{n',n''} = 1$ if there is a dependency tuple $(x_{n'}, x_{n''}, r_{n',n''})$, and $a_{n',n''} = 0$, otherwise. For \mathbf{R} , each element $r_{n',n''}$ in it uses a mark to denote the dependency type between $x_{n'}$ and $x_{n''}$. The tuples, adjacency matrix, and relation type matrix are used in our approach for enhancing NLP tasks.

Knowledge Properties

The dependency parse tree obtained from existing parsers has some properties. First, the toolkits are generally trained on human-labeled data (e.g., the dependency parsers are trained on human-annotated treebanks), which transfers human knowledge in the labeled data into the toolkits. The knowledge extracted from toolkits indirectly contains human knowledge, as the toolkits serve as automatic human annotators in the knowledge extraction process. Therefore, compared with the knowledge extracted from the raw data, the knowledge from toolkits provides deeper contextual information and thus is expected to be helpful for tasks that require a deep understanding of the text. Second, the knowledge from toolkits contains rich information. For example, dependency knowledge includes not only the directed relationships between tokens but also the relation types, which require approaches to selectively use the elements based on the task requirements to facilitate computation. Third, since the toolkits have the risk of making mistakes, the knowledge from toolkits is not always correct and thus needs to be leveraged carefully.

3.1.3 Vectors

With the advent of LLMs, vector-based knowledge representations have gained increasing recognition as a crucial form of dynamic knowledge. Different from conventional explicit knowledge, such as n-grams or parse trees, vector-based knowledge is implicitly encoded within the model parameters themselves. The rationale behind treating vectors as knowledge representations stems from the fact that the vast number of parameters in LLMs allows them to learn and store nuanced knowledge, including common sense, world knowledge, and other forms of implicit reasoning, that are beyond the reach of smaller models. If generalized further, the model’s parameters themselves are considered a comprehensive knowledge representation since they encapsulate learned information acquired

through extensive training on various datasets. This implicitly learned knowledge avoids reliance on external extraction tools or algorithms, providing enhanced flexibility and adaptability. Consequently, vector-based knowledge has garnered attention from recent studies (Ouyang et al., 2022; Liu et al., 2021; Vu et al., 2021), demonstrating impressive performance across numerous tasks and domains. In this thesis, we focus specifically on methodologies for effectively learning and leveraging this internally encoded knowledge for NLP tasks.

Knowledge Learning

Different from the aforementioned two types of knowledge that are extracted from text using unsupervised approaches or off-the-shelf toolkits, the knowledge from the learning process needs to be learned with some supervision. In general, models are trained for one or multiple target tasks, which allows the model to learn the information that is required to perform the tasks. Existing studies have shown that adding additional optimization objectives is able to improve the model performance on the target tasks (Chen et al., 2024) since the knowledge vectors and the model are able to learn extra knowledge through backpropagation based on the supervision of these additional objectives. Typically, there are two types of approaches to learn from the additional supervision. The first is to perform multi-task learning, where the knowledge vectors and the model are updated simultaneously based on the combination of the target objectives and the additional optimization objectives. The second is to perform multi-stage learning, where the knowledge vectors and the model are trained on a particular task or objective at a stage. Please note that it is possible to combine the two approaches, where each training stage may contain multiple target tasks and additional supervision (overlapping is allowed across stages). One representative example is the three-stage training of LLMs that includes pre-training, supervised fine-tuning (SFT), and reinforcement learning from human feedback (RLHF), where SFT is regarded as multi-task learning since the training data generally contain different types of instructions and expected output. The knowledge vectors trained through the process store the information in the data and use them to instruct the LLM for better performance on the tasks.

Knowledge Properties

Knowledge from the learning process has some properties. First, it differs from the aforementioned two in that it does not have an explicit form⁵; instead, it only has an implicit form, e.g., vectors, which enables it to express more advanced, deep, and abstract knowledge, which is used to capture complex patterns and relationships within the data (for example, the shared and diverse information among various contents). Therefore, it is expected to be leveraged for tasks in complicated settings. Second, the quality of the knowledge highly depends on the learning process, which emphasizes a careful design of the learning algorithm. Third, for a particular input, not all knowledge vectors are equally helpful for processing it for the task. Therefore, it is expected to design a mechanism to dynamically distinguish the relevant vectors from the irrelevant ones and leverage them accordingly.

3.2 Nature Language Processing Tasks

In this thesis, we focus on eight tasks, namely, Chinese word segmentation (CWS), constituency parsing, combinatory categorial grammar (CCG) supertagging, named entity recognition, relation extraction, aspect-based sentiment analysis, multimodal sentiment analysis, and medical large language modeling. These tasks progressively increase in complexity and difficulty, requiring varied levels of contextual information and specialized knowledge.

3.2.1 Chinese Word Segmentation

Chinese word segmentation (CWS) aims to segment Chinese text into tokens, which is a fundamental task at the lexical level. Different from most written languages in the world, the Chinese writing system does not use explicit delimiters (e.g., white space) to separate tokens in written text. Therefore, CWS conventionally serves as the first step in Chinese language processing, especially for many downstream tasks such as text classification (Zeng et al., 2018), question answering (Liu et al., 2018), machine translation (Yang et al., 2018), etc.

⁵For example, n-grams are represented by a string of one or multiple tokens; dependency tree is represented by a graph.

3.2.2 Constituency Parsing

Constituency parsing aims to produce a structured syntactic parse tree for a given sentence, which requires a deeper understanding of the text compared to CWS. It plays an important role in many downstream tasks such as relation extraction (Jiang and Diesner, 2019), natural language inference (Chen et al., 2017b), and machine translation (Ma et al., 2018a).

3.2.3 Combinatory Categorical Grammar (CCG) Supertagging

Combinatory categorical grammar (CCG) is a lexicalized grammatical formalism, where the lexical categories (also known as supertags) of the tokens in a sentence provide informative syntactic and semantic knowledge for text understanding. Therefore, CCG parse often provides useful information for many downstream NLP tasks, such as logical reasoning (Yoshikawa et al., 2018) and semantic parsing (Beschke, 2019). To perform CCG parsing in different languages, most studies conducted a supertagging-parsing pipeline (Clark and Curran, 2007; Kummerfeld et al., 2010; Song et al., 2012a; Lewis and Steedman, 2014b; Huang and Song, 2015; Xu et al., 2015a; Lewis et al., 2016; Vaswani et al., 2016; Yoshikawa et al., 2017), in which their main focus is the first step, and they generated the CCG parse trees directly from supertags with a few rules afterwards. Therefore, CCG supertagging is of great importance and is selected as the task to be tested in this thesis.

3.2.4 Named Entity Recognition

Named entity recognition (NER) aims to extract the named entities (NEs) in sentences, which requires a deep semantic understanding of the text. In this thesis, we focus on NER in the medical domain, named biomedical NER (BioNER), which extracts NEs, such as diseases, genes, species, etc., in biomedical texts and plays an important role in many downstream NLP tasks, such as drug-drug interaction task (Segura Bedmar et al., 2013; Lim et al., 2018) and knowledge base completion (Xie et al., 2013; Szklarczyk et al., 2016). Compared to the general domain NER, BioNER is considered to be more difficult due to the lack of large-scale labeled training data and domain knowledge.

3.2.5 Relation Extraction

Relation extraction (RE) aims to detect the relationship between NEs from raw text. It is one of the most important tasks in information extraction and retrieval and plays a crucial role in supporting many downstream NLP applications such as text mining (Distiawan et al., 2019), sentiment analysis (Sun et al., 2019), question answering (Xu et al., 2016), and summarization (Wang and Cardie, 2012).

3.2.6 Aspect-based Sentiment Analysis (ABSA)

Aspect-based sentiment analysis (ABSA) processes fine-grained sentiment polarities toward specific aspects. In many cases, it is required to identify different sentiments for multiple aspects in the same context. For example, in the sentence “*The drink menu is limited but the wines are excellent.*”, the sentiment polarity towards “*drink menu*” is negative while that towards “*wines*” is positive.

3.2.7 Multimodal Sentiment Analysis

The goal of multimodal sentiment analysis is to identify and interpret emotion-related information across modalities from various sources of data (e.g., visual and textual content from news, social media, business conversations, etc.). This task is essential for various real-world applications, ranging from improving user experiences and moderating harmful or hateful content on social media to developing empathetic human–computer interaction systems (Wang et al., 2023a; Zhong and Shao, 2024; Sun and Zhu, 2025).

3.2.8 Medical Large Language Modeling

Conventionally, language modeling is regarded as a fundamental NLP task that mainly focuses on predicting tokens in a sentence with the given contexts (Brown et al., 1992; Mikolov et al., 2013; Devlin et al., 2019; Brown et al., 2020). With the development of large language models (LLMs), which emerge intelligence in completing complex tasks such as dialogue systems and question answering (Touvron et al., 2023a; Taori et al., 2023a; OpenAI, 2023), there is an increasing demand for extending the capabilities of LLMs to produce fluent and accurate texts following human instructions. Therefore, large language modeling is more complicated than conventional language modeling tasks and requires a deep understanding of the text. Particularly, in this thesis, we focus on

large language modeling in the medical domain, which serves as an essential application of LLMs in the medical domain (Venigalla et al., 2022; Han et al., 2023; Singhal et al., 2023). Medical LLM plays an essential role in advancing healthcare services, a cornerstone of societal welfare that drives social development and elevates public well-being.

3.3 Preliminary Approaches

To leverage the knowledge, one generally utilizes a backbone model (which is a PLM in the thesis) and the knowledge encoding and integration module. In this section, we briefly introduce the standard text modeling paradigm of the PLM and three existing architectures, namely, attention mechanism, key-value memory networks (KVMN), and graph convolutional networks (GCN), that are able to be used for modeling different types of knowledge.

3.3.1 Standard Text Modeling Paradigm

In general, there are three types of PLMs based on the architecture of the Transformer model they use. The first is the encoder-based model that utilizes the Transformer encoder, such as BERT; the second is the decoder-based model that uses the Transformer decoder, such as GPT; and the third is the combination of encoder and decoder, such as BART. In this thesis, we mainly focus on the encoder-based PLMs and the decoder-based PLMs since the former and the latter are widely used in understanding and generation tasks, respectively.

For the encoder-based PLM (denoted as $f_{encoder}$), given an input sentence \mathcal{X} with N tokens (the n -th token is denoted as x_n), the PLM $f_{encoder}$ processes it and produce a list of hidden vectors $\mathbf{h}_1, \dots, \mathbf{h}_n, \dots, \mathbf{h}_N$ by

$$\mathbf{h}_1, \dots, \mathbf{h}_n, \dots, \mathbf{h}_N = f_{encoder}(\mathcal{X}) \quad (3.2)$$

where the n -th hidden vector \mathbf{h}_n corresponds to the representation of the n -th token x_n . Then, the hidden vectors are fed into a task-specific decoder $f_{decoder}$ to produce the output. For sequence labeling tasks, the decoder $f_{decoder}$ predicts a list of label $\hat{y}_1 \cdots \hat{y}_n \cdots \hat{y}_N$ by

$$\hat{y}_1 \cdots \hat{y}_n \cdots \hat{y}_N = f_{decoder}(\mathbf{h}_1, \dots, \mathbf{h}_n, \dots, \mathbf{h}_N) \quad (3.3)$$

where the n -th predicted label \hat{y}_n is associated with the n -th input token (e.g., the corresponding POS tags in POS tagging task). Some representative decoders for sequence-labeling tasks are softmax decoders and conditional random field (CRF) (Tseng et al., 2005b). For text classification tasks, the decoder $f_{decoder}$ predicts a label \hat{y} (e.g., the sentiment of a particular sentence-aspect pair) by

$$\hat{y} = f_{decoder}(\mathbf{h}_1, \dots, \mathbf{h}_n, \dots, \mathbf{h}_N) \quad (3.4)$$

One representative decoder for text classification is the softmax decoder.

For the decoder-based PLMs $g_{decoder}$, they directly take the input text \mathcal{X} and generate the content $\hat{\mathcal{Y}}$ by

$$\hat{\mathcal{Y}} = g_{decoder}(\mathcal{X}) \quad (3.5)$$

Herein, the PLMs also take an optional prompt as the additional input, which is either hard-prompt \mathcal{P} in string or soft-prompt \mathbf{P} in a matrix (or stacked vectors). In these cases, Equation (3.5) is rewritten as

$$\hat{\mathcal{Y}} = g_{decoder}(\mathcal{P}, \mathcal{X}) \quad (3.6)$$

or

$$\hat{\mathcal{Y}} = g_{decoder}(\mathbf{P}, \mathcal{X}) \quad (3.7)$$

for hard- and soft-prompts, respectively.

In training, for both types of PLMs, the model predictions are compared with the human-annotated labels to compute the cross-entropy loss. Then, the loss is used to compute the gradient with respect to the trainable parameters in the model, which are updated accordingly through backpropagation.

3.3.2 Attention Mechanism

Many existing studies find that various elements contribute differently to NLP tasks (Raffel and Ellis, 2015; Vaswani et al., 2017). To address these differences, the attention mechanism is proposed to allocate attention to different elements, where attention value determines the extent to which an element contributes to the model’s final output. Over the past several years, the attention mechanism has developed many variants (Veličković et al., 2017; Fu et al., 2019; Gheini et al., 2021). The following illustrates the basic ideas of the attention mechanism. Assume there is a list of hidden

vectors $\alpha_1 \cdots \alpha_t \cdots \alpha_T$ to be weighted, the attention mechanism utilizes a vector ω to compute the attention weight w_t for the t -th hidden vector through

$$w_t = \frac{\exp(\omega \cdot \alpha_t)}{\sum_{t=1}^T \exp(\omega \cdot \alpha_t)} \quad (3.8)$$

The weight w_t is then applied to the corresponding hidden vector to compute the weighted sum of $\alpha_1 \cdots \alpha_t \cdots \alpha_T$ by

$$\alpha = \sum_{t=1}^T w_t \cdot \alpha_t \quad (3.9)$$

where α is the output representation used for further processing. Therefore, attention is able to be used to selectively leverage different knowledge with necessary adaptations.

3.3.3 Key-value Memory Networks

Originally, key-value memory networks (KVMN) (Miller et al., 2016b) is proposed to incorporate the information in a list of memory slots $(k_1, v_1), \cdots, (k_m, v_m), \cdots, (k_M, v_M)$ (where k_m and v_m refer to the key and value for the m -th memory slot, respectively) into a model for question answering tasks. In KVMN, it addresses the keys by assigning a probability weight to the value in each memory slot by comparing the question (which is denoted as x) to each key:

$$p_m = \text{softmax}(\mathbf{W}^Q \mathbf{x} \cdot \mathbf{W}^K \mathbf{k}_m) \quad (3.10)$$

where \mathbf{W}^Q and \mathbf{W}^K are trainable question matrix and key matrix, respectively, and \mathbf{x} and \mathbf{k}_m are the vector representation of x and k_m , respectively. Then, KVMN reads the values by computing the weighted sum using the resulting probability weights:

$$\mathbf{o} = \sum_{m=1}^M p_m \cdot \mathbf{W}^V \mathbf{v}_m \quad (3.11)$$

where \mathbf{W}^V is a trainable value matrix and \mathbf{v}_m is the vector representation of v_m . Afterwards, \mathbf{o} is incorporated into the question representation by an element-wise addition: $\mathbf{o}' = \mathbf{W}^Q \mathbf{x} + \mathbf{o}$ and the resulting \mathbf{o}' is used to predict the answers of the question. Therefore, in KVMN, the keys are used to compute the weights, which are used to address the values with respect to the input; the values are used to incorporate useful information into the input presentation and thus improve model performance. Considering that knowledge base entries have been used as a possible type of resource

for the memory slots to incorporate extra knowledge into the input representation by transforms between the keys and values (Miller et al., 2016b), we are able to use such transforms between context features and syntactic information instances to incorporate the syntactic information into our backbone model.

3.3.4 Graph Convolutional Networks

The standard graph convolutional network (GCN) model is generally used to encode graphs, such as the dependency tree of the sentence. In this thesis, since the GCN model and its variants are used to encode the dependency knowledge of the sentence, we illustrate how standard GCN is used for encoding the knowledge in the following text. Specifically, the standard GCN contains L layers to encode the word pairs suggested by the dependency parsing results of the input sentence \mathcal{X} , where the connections between all pairs of words $x_{n'}$ and $x_{n''}$ are represented by an adjacency matrix $\tilde{\mathbf{A}} = \{\tilde{a}_{n',n''}\}_{N \times N}$. Herein, $\tilde{\mathbf{A}}$ is a variant of the adjacency matrix \mathbf{A} introduced in Section 3.1.2, where $\tilde{a}_{n',n''} = 1$ if there is a dependency edge between $x_{n'}$ and $x_{n''}$ or $n' = n''$ (the direction of the edge is ignored), and $\tilde{a}_{n',n''} = 0$ otherwise. Based on the adjacency matrix $\tilde{\mathbf{A}}$, for each $x_{n'}$, the l -th GCN layer finds all $x_{n''}$ associated with $x_{n'}$ (where $\tilde{a}_{n',n''} = 1$), takes their hidden vectors $\mathbf{h}_{n''}^{(l-1)}$ from the $(l-1)$ -th layer, and computes the output for $x_{n'}$ by

$$\mathbf{h}_{n'}^{(l)} = \sigma(LN(\sum_{n''=1}^N \tilde{a}_{n',n''}(\mathbf{W}^{(l)} \cdot \mathbf{h}_{n''}^{(l-1)} + \mathbf{b}^{(l)}))) \quad (3.12)$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are trainable matrix and bias for the l -th GCN layer, LN refers to layer normalization and σ the *ReLU* activation function. Therefore, the standard GCN is able to encode the dependency graph of the sentence, where the contextual information carried by different tokens is equally transferred to the connected tokens through the forward pass of GCN layers.

3.4 Combining Knowledge, Tasks, and Approaches

Based on our previous analysis of the properties of different knowledge types, we notice that different types of knowledge provide varying levels of contextual information and expressive power. At the same time, NLP tasks at different levels require varying degrees of knowledge expressiveness. More superficial knowledge (such as n-grams) is suitable for relatively simple tasks, including

CWS, constituency parsing, and CCG supertagging, while more advanced tasks require deeper and richer knowledge for adequate support. For instance, NER, relation extraction, and ABSA typically require structured, explicit, and relatively rich information provided by knowledge from toolkits. For more complex tasks, including multimodal sentiment analysis and medical large language modeling, explicit knowledge alone may be insufficient for achieving deeper understanding; thus, abstract and implicit vectorized knowledge from the learning process is necessary.

Meanwhile, the choice of model architecture and approaches should also meet the requirements of the tasks and the types of knowledge used. Specifically, for CWS, constituency parsing, CCG supertagging, NER, relation extraction, and ABSA tasks that emphasize understanding, we choose Transformer encoder-based PLM, namely, BERT, as the backbone model, as numerous studies have demonstrated their strong performance in text understanding tasks (Devlin et al., 2019; Diao et al., 2020; Dai et al., 2019; Joshi et al., 2020). For higher-level, generation- or reasoning-intensive tasks (i.e., multimodal sentiment analysis and medical large language modeling), we use Transformer decoder-based LLMs, e.g., LLaMA (Touvron et al., 2023a), as the backbone model, as they have achieved widespread success in generation-oriented tasks (Ouyang et al., 2022; Zhang et al., 2022a; Taori et al., 2023a; Chiang et al., 2023). Meanwhile, to effectively leverage different knowledge types, we design new task-specific approaches based on the architectures introduced in Section 3.3 to efficiently integrate and utilize knowledge.

The specific combinations of the aforementioned knowledge types, corresponding tasks, and approaches are summarized in Table 3.2, where each of them is associated with a published paper. These studies have been conducted collaboratively with my advisers and other co-authors, to whom I extend my sincere gratitude for their valuable contributions. We elaborate on these contributions and the detailed approaches in subsequent chapters.

Knowledge	Tasks	Approaches	Chapter	Paper
Lexicon (n-grams)	CWS	KVMN	4.1	Tian et al. (2020e)
	Constituency Parsing	Attention	4.2	Tian et al. (2020d)
	CCG supertagging	Attention	4.3	Tian et al. (2020c)
Syntax (Parse Tree)	NER	KVMN	5.1	Tian et al. (2020a)
	Relation Extraction	GCN	5.2	Tian et al. (2021b)
	ABSA	GCN	5.3	Tian et al. (2021a)
Pattern (Vectors)	Multimodal Sentiment Analysis	Memory	6.1	Tian et al. (2024b)
	Medical Large Language Modeling	Language Modeling	6.2	Tian et al. (2024a)

Table 3.2: The combination of knowledge types, corresponding NLP tasks, utilized approaches, their chapters in the thesis, and the associated paper.

Chapter 4

Leveraging Lexicon Knowledge

In this chapter, we present our research on applying lexicon knowledge (i.e., n-grams) to CWS, constituency parsing, and CCG supertagging, where the knowledge is expected to provide models with additional contextual information that enhances their ability to handle token-level and structural ambiguities.

4.1 Chinese Word Segmentation

Lexical-level tasks are performed at the token level. They focus on breaking down text into its basic units (tokens) and performing operations such as normalization, stemming, lemmatization, and morphological analysis. As a representative lexical-level task, CWS attracts much attention from existing studies. Therefore, we utilize CWS as the lexical-level task to explore the effective approach to leverage the knowledge obtained from data. The following are the details.

In the past two decades, mainstream approaches for CWS generally formulate it as a character-based sequence labeling task (Tseng et al., 2005a; Sun and Xu, 2011; Chen et al., 2015; Ma et al., 2018b; Higashiyama et al., 2019). Many existing approaches, including both conventional and neural approaches, focus on effectively extracting contextual features to facilitate accurate segmentation label predictions (Zhou et al., 2017; Zhang et al., 2013). Among these contextual features, wordhood measures for n-grams have been demonstrated to be effective for CWS (Sun et al., 1998; Xue and Shen, 2003; Feng et al., 2004; Song and Xia, 2012). Although some neural studies attempted to

leverage contextual n-grams (Pei et al., 2014; Zhou et al., 2017) or word-level attention mechanisms (Higashiyama et al., 2019), their approaches either rely on simple concatenation of embeddings or require a predefined lexicon. Consequently, effective and flexible incorporation of explicit wordhood information into neural CWS models remains inadequately explored, particularly regarding generalizing the integration of various wordhood measures without dependency on handcrafted lexical resources.

To leverage wordhood n-gram information for CWS, we propose **WMSeg**, a neural framework with a memory mechanism. Specifically, WMSeg employs key-value memory networks (Miller et al., 2016a), mapping character n-grams and their corresponding wordhood measures into memory keys and values. For each input character, WMSeg retrieves relevant contextual n-grams from memory, enriching the representation and assisting the decoder in assigning accurate segmentation labels. Extensive experiments on five benchmark datasets demonstrate the effectiveness of WMSeg to leverage n-gram information for CWS.

4.1.1 The Approach

Following previous studies, we regard CWS as a character-based sequence labeling task. The architecture of WMSeg is illustrated in Figure 4.1, where the general sequence labeling paradigm is the top part with a memory module inserted between the encoder and the decoder. The model predicts a tag (e.g., tag *B* for the 1st character in a word) for each character, and the predicted tag sequence is then converted to word boundary in the system output. The bottom part of the figure starts with a lexicon \mathcal{N} , which is simply a list of n-grams and can be built by various approaches. Given an input sentence $\mathcal{X} = x_1 \cdots x_n \cdots x_N$, for each character x_n in \mathcal{X} , our approach uses the lexicon \mathcal{V} to generate (keys, values) for x_n and send it to the memory module f_M . In all, the process of WMSeg f_{WMSeg} to perform CWS can be formalized as

$$\hat{\mathcal{Y}} = f_{WMSeg}(\mathcal{X}, f_M(\mathcal{X}, \mathcal{V})) \quad (4.1)$$

where $\hat{\mathcal{Y}}$ represent the label sequence for the input. In the rest of this section, we describe the construction of the n-gram lexicon, the proposed wordhood memory networks, and how it is integrated with different encoders and decoders, respectively.

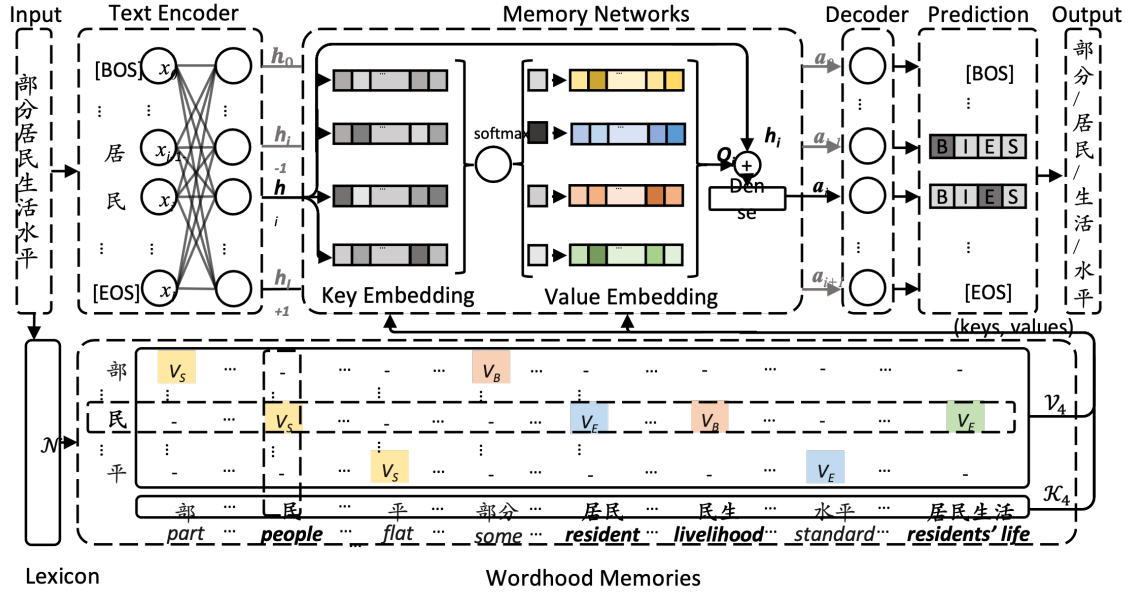


Figure 4.1: The architecture of WMSeg. “ \mathcal{N} ” denotes a lexicon constructed by wordhood measures. N-grams (keys) appearing in the input sentence “部分居民生活水平” (*some residents’ living standard*) and the wordhood information (values) of those n-grams are extracted from the lexicon. Then, together with the output from the text encoder, n-grams (keys) and their wordhood information (values) are fed into the memory module, whose output passes through a decoder to get final predictions of segmentation labels for every character in the input sentence.

Lexicon Construction

To build the wordhood memory networks, the first step is to construct the lexicon \mathcal{V} because the keys in the memory module are built upon \mathcal{V} , where each n-gram in \mathcal{V} is stored as a key in it.¹ In this study, \mathcal{V} is simply a list of n-grams, and technically, it can be constructed through many existing resources or automatic methods. We utilize unsupervised approaches to construct the lexicon. Specifically, in addition to the PMI approach illustrated in Section 3.1.1 to construct the lexicon \mathcal{V} , we also utilize accessor variety (AV) (Feng et al., 2004) to extract the n-grams. For AV, it measures the wordhood of

¹N-gram and key are equivalent in the memory.

an n-gram k by

$$AV(k) = \min(L_{av}(k), R_{av}(k)) \quad (4.2)$$

where $L_{av}(k)$ and $R_{av}(k)$ denote the number of different character types that can precede (left access number) or follow (right access number) the n-gram k . Normally, the higher the AV score is, the more likely the n-gram forms a word. We compute the AV scores of all n-grams in the raw data and use a threshold to keep n-grams whose AV scores are higher than the threshold, which leads to the lexicon \mathcal{V} .

Wordhood Memory Networks

To encode both n-grams and the wordhood information they carry, one requires an appropriate framework to do so for CWS. Compared with other network structures that can exploit n-grams such as the attention mechanism, key-value memory networks are more appropriate to model such pairwise knowledge via transforms between keys and values. In the memory, we map n-grams and their wordhood information to keys and values, respectively. Following the KVMN in Section 3.3.3, we illustrate how to adapt the memory module for CWS.

For each x_n in a training/test instance, normally there are many n-grams in \mathcal{V} that contain x_n . Therefore, the first step is to generate all n-grams from x_n 's context (including x_n) and keep only the ones that appear in \mathcal{V} , resulting $\mathcal{K}_n = [k_{n,1}, k_{n,2}, \dots, k_{n,m}, \dots, k_{n,M_n}]$ that x_n is a part of $k_{n,m}$. For example, in the input sentence shown in Figure 4.4, the n-grams that contain the character $x_4 = \text{“民”}$ (*people*) form the list $\mathcal{K}_4 = [\text{“民” (people)}, \text{“居民” (resident)}, \text{“民生” (livelihood)}, \text{“居民生活” (residents' life)}]$, which are highlighted in the dashed boxes illustrated at the bottom part of the figure. Then, the memory module activates the corresponding keys in it, addresses their embeddings (which are denoted as $\mathbf{k}_{n,m}$ for each $k_{n,m}$), and computes the probability distribution for them with

$$p_{n,m} = \frac{\exp(\mathbf{h}_n \cdot \mathbf{k}_{n,m})}{\sum_{m=1}^{M_n} \exp(\mathbf{h}_n \cdot \mathbf{k}_{n,m})} \quad (4.3)$$

for each key, where \mathbf{h}_n is the vector for x_n which can be generated from any text encoder.

Values in the memory represent the wordhood information for a given x_n and $k_{n,m}$ pair, which is not a straightforward mapping because x_n may have different roles in each $k_{n,m}$. For example, $k_{n,m}$ delivers different wordhood information when x_n appears at the beginning or the ending of

Rule	$v_{n,m}$
x_n is the beginning of the key $k_{n,m}$	V_B
x_n is inside the key $k_{n,m}$	V_I
x_n is the ending of the key $k_{n,m}$	V_E
x_n is the single-character key $k_{n,m}$	V_S

Table 4.1: The rules for assigning different values to x_i according to its position in a key $k_{n,m}$.

$k_{n,m}$. Therefore, we set rules in Table 4.1 to read a value for a key according to different situations of x_n in $k_{n,m}$, where we use a set of values $\{V_B, V_I, V_E, V_S\}$ with embeddings $\{\mathbf{e}_{V_B}, \mathbf{e}_{V_I}, \mathbf{e}_{V_E}, \mathbf{e}_{V_S}\}$ (illustrated in different colors in Figure 4.4) so that all n-grams should map to one of the values based on x_n 's position in $k_{n,m}$. To illustrate that, in the aforementioned example, n-grams in \mathcal{K}_4 for $x_4 = \text{“民”}$ (*people*) are mapped to a value list $\mathcal{V}_4 = [V_S, V_E, V_B, V_I]$ (see Figure 4.4). As a result, each K_n for x_n has a list of values denoted by $\mathcal{V}_n = [v_{n,1}, v_{n,2} \cdots, v_{n,m} \cdots v_{n,M_n}]$. Then the total wordhood memory for x_n is computed from the weighted sum of all keys and values by

$$\mathbf{o}_n = \sum_{m=1}^{M_n} p_{n,m} \mathbf{v}_{n,m} \quad (4.4)$$

where $\mathbf{v}_{n,m}$ is the embedding for $v_{n,m}$.

CWS with Wordhood Memories

We perform CWS following the standard encoding and decoding process with encoder-based PLM. Specifically, we utilize Equation (3.2) in Section 3.3.1 to compute the $\mathbf{h}_1, \cdots, \mathbf{h}_n, \cdots, \mathbf{h}_N$ for the memory module. Once all \mathbf{o}_n are generated from the memory for each x_n , we merge it with the corresponding \mathbf{h}_n by

$$\mathbf{o}'_n = \mathbf{W}_o \cdot (\mathbf{h}_n + \mathbf{o}_n) \quad (4.5)$$

where \mathbf{W}_o is a trainable matrix and \mathbf{o}'_n is the resulting vector, which is fed into the decoder to obtain the output sequence through Equation (3.3)². Herein, we try two different types of decoders: one is

²The \mathbf{h}_n in Equation (3.3) is replaced by \mathbf{o}'_n .

	MSR		PKU		AS		CityU		CTB6		
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Dev	Test
Char #	4,050K	184K	1,826K	173K	8,368K	198K	2,403K	68K	1,056K	100K	134K
Word #	2,368K	107K	1,110K	104K	5,500K	123K	1,456K	41K	641K	60K	82K
Char Type #	5K	3K	5K	3K	6K	4K	5K	3K	4K	3K	3K
Word Type #	88K	13K	55K	13K	141K	19K	69K	9K	42K	10K	12K
OOV Rate	-	2.7	-	5.8	-	4.3	-	7.2	-	5.4	5.6

Table 4.2: Statistics of the five benchmark datasets, in terms of the number of character and word tokens and types in each training and test set. Out-of-vocabulary (OOV) rate is the percentage of unseen word tokens in the test set.

the softmax decoder, and the other is the conditional random field (CRF) decoder.

4.1.2 Experiment Settings

Datasets

We employ five benchmark datasets in our experiments: four of them, namely, MSR, PKU, AS, and CityU, are from SIGHAN 2005 Bakeoff Emerson (2005) and the fifth one is CTB6 (Xue et al., 2005). AS and CityU are in traditional Chinese characters whereas the other three use simplified ones. Following previous studies (Chen et al., 2015, 2017c; Qiu et al., 2019), we convert traditional Chinese characters in AS and CityU into simplified ones.³ For MSR, AS, PKU, and CityU, we follow their official training/test data split. For CTB6, we use the same split as that stated in Yang and Xue (2012); Chen et al. (2015); Higashiyama et al. (2019), and only use its test set for the final experiment. Table 4.2 shows the statistics of all datasets in terms of the number of characters and words and the percentage of out-of-vocabulary (OOV) words in the dev/test sets with respect to the training set.

In addition, we also use CTB7 (LDC2010T07) to perform our cross-domain experiments. There are five genres in CTB7, including broadcast conversation (BC), broadcast news (BN), magazine (MZ), newswire (NW), and weblog (Web). The statistics of all the genres are reported in Table

³The conversion scripts are from <https://github.com/skydark/nstools/tree/master/zhtools>

	BC	BN	MZ	NW	Web
Char #	275K	483K	403K	443K	342K
Word #	184K	287K	258K	260K	210K
Char Type #	3K	3K	4K	3K	4K
Word Type #	12K	23K	26K	21K	21K
OOV Rate	3.4	6.0	8.9	5.9	7.1

Table 4.3: Statistics of CTB7 with respect to five different genres. The OOV rate for each genre is computed based on the vocabulary from all the other four genres.

4.3, where the OOV rate for each genre is computed according to the union of all other genres. For example, the OOV rate for BC is computed with respect to the union of BN, MZ, NW, and Web.

Implementation Details

Following previous studies (Sun and Xu, 2011; Chen et al., 2015, 2017c; Ma et al., 2018b; Qiu et al., 2019), we use four segmentation labels in our experiments, i.e., $\mathcal{T} = \{B, I, E, S\}$. Among them, B , I , and E indicate a character is the beginning, inside, and the ending of a word and S denotes that the character is a single-character word.

Since text representation plays an important role to facilitate many tasks (Conneau et al., 2017; Song et al., 2017, 2018c; Sileo et al., 2019), we try two effective and well-known encoders, i.e., BERT⁴ and ZEN⁵ (Diao et al., 2020). For the decoders, we use softmax and CRF, and set their loss functions as cross-entropy and negative log-likelihood, respectively. The memory module can be initialized by random or pre-trained word embeddings for keys and values. In our experiments, we use random initialization for them.⁶

⁴We use the Chinese base model from <https://s3.amazonaws.com/models.huggingface.co/>.

⁵<https://github.com/sinovation/ZEN>.

⁶We tried different initialization methods, and they did not show a significant difference in CWS performance.

Config		MSR		PKU		AS		CityU		CTB6	
En-Dn	WM	F	R _{OOV}	F	R _{OOV}	F	R _{OOV}	F	R _{OOV}	F	R _{OOV}
BT-SM	×	97.84	86.32	96.20	84.43	96.33	77.86	97.51	86.69	96.90	88.46
	✓	98.16	86.50	96.47	86.34	96.52	78.67	97.77	86.62	97.13	88.30
BT-CRF	×	97.98	85.52	96.32	85.04	96.34	77.75	97.63	86.66	96.98	87.43
	✓	98.28	86.67	96.51	86.76	96.58	78.48	97.80	87.57	97.16	88.00
ZEN-SM	×	98.35	85.78	96.27	84.50	96.38	77.62	97.78	90.69	97.08	86.20
	✓	98.36	85.30	96.49	84.95	96.55	78.02	97.86	90.89	97.22	86.83
ZEN-CRF	×	98.36	86.82	96.36	84.81	96.39	77.81	97.81	91.78	97.13	87.08
	✓	98.40	84.87	96.53	85.36	96.62	79.64	97.93	90.15	97.25	88.46

Table 4.4: Experimental results of WMSeg on SIGHAN2005 and CTB6 datasets with different configurations. “En-Dn” stands for the text encoders (“BT” for BERT) and decoders (“SM” for softmax and “CRF” for CRF). The “WM” column indicates whether the wordhood memories are used (✓) or not (×).

4.1.3 Results and Analysis

In this section, we firstly report the results of WMSeg with different configurations on five benchmark datasets and its comparison with existing models. Then we explore the effect of using different lexicon \mathcal{N} and different wordhood measures in WMSeg. We also use a cross-domain experiment to illustrate the effectiveness of WMSeg when more OOVs are in the test set. Lastly, a case study is performed to visualize how the wordhood information used in WMSeg helps CWS.

Overall Results

In the main experiment, we illustrate the validity of the proposed memory module by comparing WMSeg in different configurations, i.e., with and without the memory in integrating with two encoders, i.e., BERT, and ZEN, and two decoders, i.e., softmax and CRF. The experimental results on the aforementioned five benchmark datasets are shown in Table 4.4, where the overall F-score and the recall of OOV are reported. With five datasets and six encoder-decoder configurations, the table

	MSR		PKU		AS		CityU		CTB6	
	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}
Zhang et al. (2013)	97.5	-	96.1	73.1	-	-	-	-	-	-
Pei et al. (2014)	97.2	-	95.2	-	-	-	-	-	-	-
Ma and Hinrichs (2015)	96.6	87.2	95.1	76.0	-	-	-	-	-	-
Chen et al. (2015)	97.4	-	96.5	-	-	-	-	-	96.0	-
Xu and Sun (2016)	96.3	-	96.1	-	-	-	-	-	95.8	-
Zhang et al. (2016)	97.7	-	95.7	-	-	-	-	-	95.95	-
Chen et al. (2017c)	96.04	71.60	94.32	72.64	94.75	75.34	95.55	81.40	-	-
Wang and Xu (2017)	98.0	-	96.5	-	-	-	-	-	-	-
Zhou et al. (2017)	97.8	-	96.0	-	-	-	-	-	96.2	-
Ma et al. (2018b)	98.1	80.0	96.1	78.8	96.2	70.7	97.2	87.5	96.7	85.4
Gong et al. (2019)	97.78	64.20	96.15	69.88	95.22	77.33	96.22	73.58	-	-
Higashiyama et al. (2019)	97.8	-	-	-	-	-	-	-	96.4	-
Qiu et al. (2019)	98.05	78.92	96.41	78.91	96.44	76.39	96.91	86.91	-	-
WMSeg (BERT-CRF)	98.28	86.67	96.51	86.76	96.58	78.48	97.80	87.57	97.16	88.00
WMSeg (ZEN-CRF)	98.40	84.87	96.53	85.36	96.62	79.64	97.93	90.15	97.25	88.46

Table 4.5: Performance (F-score) comparison between WMSeg (BT-CRF and ZEN-CRF with wordhood memory networks) and previous state-of-the-art models on the test set of five benchmark datasets.

includes results from 30 pairs of experiments, each pair with or without using the memories.

There are several observations drawn from the results. First, the overall comparison clearly indicates that, WMSeg (i.e., the model with wordhood memories) outperforms the baseline (i.e., the model without wordhood memories) for all settings in terms of F-scores and for most settings in terms of R_{OOV} . Second, the proposed memory module works smoothly with different encoders and decoders, where some improvement is pretty significant; for instance, when using Bi-LSTM as the encoder and CRF as the decoder, WMSeg improves the F-score on the AS dataset from 94.39 to 95.07 and R_{OOV} from 61.59 to 68.17. With BERT or ZEN as the encoder, even when the baseline system performs very well, the improvement of WMSeg on F-scores is still decent. Third, among the models

with ZEN, the ones with the memory module further improve their baselines, although the context information carried by n-grams is already learned in pre-training ZEN. This indicates that wordhood information provides additional cues (besides the contextual features) that can benefit CWS, and our proposed memory module is able to provide further task-specific guidance to an n-gram integrated encoder. To summarize, the results in this experiment not only confirm that wordhood information is a simple yet effective source of knowledge to help CWS without requiring external support such as a well-defined dictionary or manually crafted heuristics, but also fully illustrate that the design of our model can effectively integrate this type of knowledge.

To further illustrate the validity and the effectiveness of WMSeg, we compare our best-performing model with the ones in previous studies on the same benchmark datasets. The comparison is presented in Table 4.5, where WMSeg (both the one with BERT and ZEN) outperforms all existing models with respect to the F-scores and achieves new state-of-the-art performance on all datasets.

Cross-Domain Performance

As domain variance is always an important factor affecting the performance of NLP systems, especially word segmenters (Song et al., 2012b; Song and Xia, 2013), in addition to the experiments on benchmark datasets, we also run WMSeg on CTB7 across domains (genres in this case) with and without the memory module. To test on each genre, we use the union of the data from the other four genres to train our segmenter and use AV to extract n-grams from the entire raw text from CTB7 in this experiment. Table 4.6 reports the results in F-score and OOV recall, which show a similar trend as that in Table 4.4, where WMSeg outperforms baselines for all five genres. Particularly, for genres with large domain variance (e.g., the ones with high OOV rates such as MZ and Web), CWS is difficult, and its relatively low F-scores in Table 4.6 from baseline models confirm that. Yet WMSeg offers a decent way to improve cross-domain CWS performance without any help from external knowledge or complicated model design, which further illustrates the effectiveness of the memory module. The reason could be that many n-grams are shared in both training and test data; these n-grams with their wordhood information present a strong indication to the model on what combinations of characters can be treated as words, even though some of them never appear in the training data.

Config		BC		BN		MZ		NW		Web	
En-Dn	WM	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}	F	R_{OOV}
BT-SM	×	96.27	80.76	96.88	87.90	94.97	84.45	97.08	89.78	94.82	74.00
	✓	96.41	81.15	97.00	89.47	95.10	85.48	97.24	91.96	95.00	75.51
BT-CRF	×	96.25	79.04	96.87	89.15	94.94	85.27	96.99	91.34	94.79	75.58
	✓	96.43	81.29	97.09	90.29	95.11	85.32	97.21	92.48	95.03	76.30
ZEN-SM	×	96.39	79.97	96.95	88.93	95.05	85.14	97.17	91.33	94.03	75.33
	✓	96.45	81.34	97.03	89.78	95.06	85.60	97.21	91.73	95.08	75.60
ZEN-CRF	×	96.30	80.05	96.97	90.38	94.93	85.64	97.10	91.03	94.90	74.98
	✓	96.50	80.44	97.11	90.29	95.13	85.96	97.24	91.68	95.04	75.74

Table 4.6: Experimental results on five genres of CTB7. Abbreviations follow the same notation in Table 4.4.

Effect of Using Different \mathcal{N}

To analyze the robustness of WMSeg with respect to the lexicon, we compare four ways (ID: 2-5 in Table 4.7) of constructing the lexicon (\mathcal{N}): the first one simply uses the vocabulary from the training data (marked as Gold Label in Table 4.7; ID: 2); the other three ways use AV to extract n-grams from the unsegmented training data only (ID: 3), the test data only (ID: 4), and training + test set (ID: 5), respectively.⁷ Table 4.7 shows the results of running BERT-CRF on the Web genre of CTB7 without the wordhood memories (ID: 1) and with the memories (ID: 2-5), following the cross-domain setting in §4.1.3. While the four methods with memories achieve similar results on the F score, indicating the robustness of our proposed framework, the one that builds \mathcal{N} using the raw texts from both training and test sets through unsupervised method (ID: 5) achieves the biggest improvement on R_{OOV} , demonstrating the advantage of including the unlabeled test set by incorporating the results from unsupervised wordhood measures into the models.

⁷One could also use an external corpus to build \mathcal{N} , which is not considered in this experiment.

ID	Train	Test	Gold Label	F	R _{OOV}
1	-	-	-	94.79	75.58
2	×	×	✓	+0.22	+0.21
3	✓	×	×	+0.21	+0.20
4	×	✓	×	+0.23	+0.33
5	✓	✓	×	+0.24	+0.72

Table 4.7: Comparisons of performance gain on the Web genre of CTB7 with respect to the baseline BERT-CRF model when the n-gram lexicon \mathcal{N} for WMSeg is built upon different sources. ✓ and × refer to whether a corresponding data source is used or not, respectively.

Effect of Different Wordhood Measures

WMSeg provides a general way of integrating wordhood information for CWS, we expect other wordhood measures to play the same role in it. Therefore, we test PMI in our model and compare it with the previous results from AV (see Table 4.4). Specifically, we use our best-performing BERT-based model, i.e., BERT-CRF, with the n-gram lexicons constructed by the aforementioned three measures and run it on all benchmark datasets. We draw the histograms of the F-scores obtained from WMSeg with each measure (red and green bars for AV and PMI, respectively) in Figure 4.2, where the F-scores of the baseline model are also presented in orange bars.

As shown in the figure, the performances of using the three measures are very similar, which indicates that WMSeg is able to robustly incorporate the wordhood information from various measures, despite that those measures focus on different aspects of n-grams when determining whether the n-grams should be treated as words. This observation also reveals the possibility that many n-grams may be redundant for our model, and WMSeg is thus able to identify the most useful ones from them, which is analyzed in the case study.

Case Study

To investigate how the memory learns from the wordhood information carried by n-grams, we conduct a case study with an example input sentence “他/从小/学/电脑/技术” (*He learned computer*

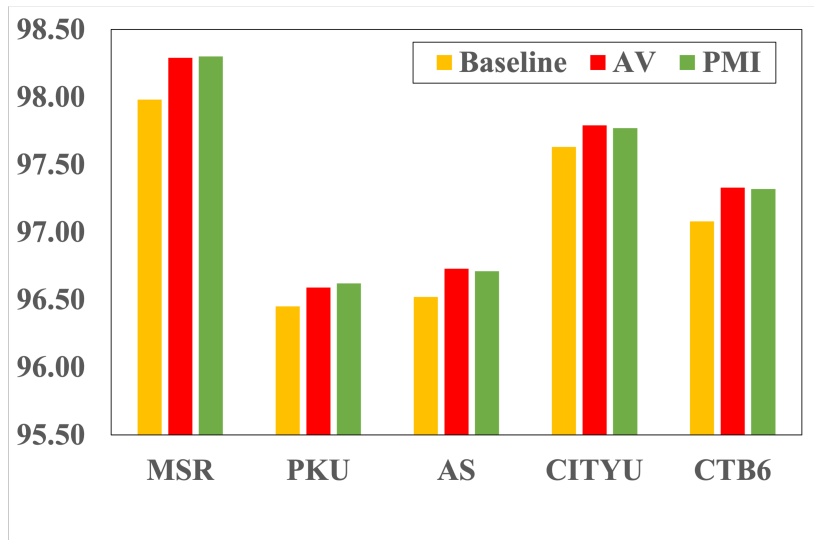


Figure 4.2: The F-scores of WMSeg (BERT) using three different wordhood measures, namely AV (red) and PMI (green), on five benchmark datasets.

techniques since childhood). In this sentence, the n-gram “从小学” is ambiguous with two possible interpretations: “从小/学” (*learn since childhood*) and “从/小学” (*from primary school*). Native Chinese speakers can easily choose the first one with the given context but a word segmenter might incorrectly choose the second segmentation.

We feed this case into our BERT-CRF model with the memory module. In Figure 4.3, we visualize the resulting weights that are learned from keys (a) and values (b) of the memory, as well as from the final tagger (c). The heatmaps of all keys and values in the memory with respect to each corresponding input character clearly illustrate that the appropriate n-grams, e.g., “他” (*he*), “学” (*learn*), “从小” (*from childhood*), etc., receive higher weights than others and the corresponding values for them are also emphasized, which further affects final CWS tagging so that the weight distributions from (b) and (c) look alike to each other. Therefore, this visualization explains, to some extent, that the proposed memory mechanism can identify and distinguish important n-grams within a certain context and thus improve CWS performance accordingly.

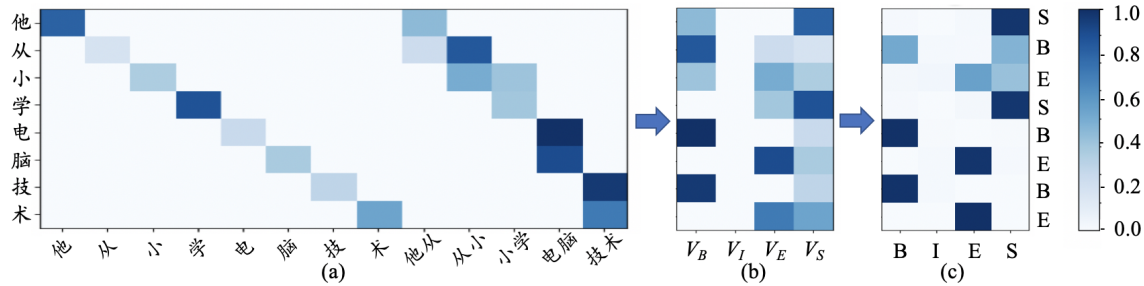


Figure 4.3: Heatmaps of weights learned for (a) keys and (b) values in the memory, and (c) the tags from the decoder, with respect to each character in an input sentence. Higher weights are visualized with darker colors.

4.2 Constituency Parsing

Recently, neural parsers without explicit grammar rules have significantly surpassed traditional statistical grammar-based approaches, due to the powerful ability of neural networks, especially recurrent models, to capture long-range contextual dependencies essential for parsing (Dyer et al., 2016; Stern et al., 2017; Kitaev et al., 2019). Recently, advanced neural text encoders like Transformer and BERT enable the high performance of chart-based parsers by providing richer contextual representations (Kitaev and Klein, 2018; Zhou and Zhao, 2019). Beyond advanced encoders, additional sources of contextual information, such as n-grams, demonstrate effectiveness in various NLP tasks (Song et al., 2018a; Zhang et al., 2019b; Yoon et al., 2018). Although some studies try to incorporate n-gram features (Sagae and Lavie, 2005; Pitler et al., 2010), they treat all n-grams equally without distinguishing their relative importance, potentially introducing misleading signals.

We propose a span attention module to enhance chart-based neural constituency parsing by effectively incorporating meaningful n-grams into span representations. Specifically, our approach firstly extracts all n-grams from a given span that appear in an n-gram lexicon, then employs an attention mechanism to weight these n-grams according to their relevance for predicting the constituency label of that span. Considering shorter n-grams typically dominate attention weights due to higher frequency, we further introduce a categorical mechanism that groups n-grams by length and separately computes attention within each group, ensuring that longer, context-rich n-

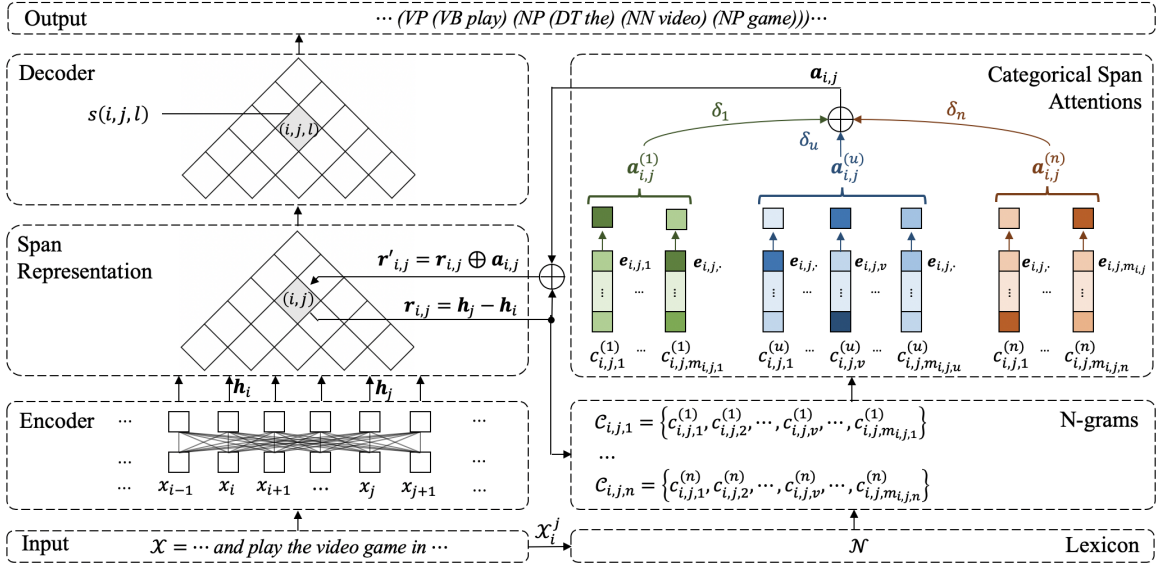


Figure 4.4: The architecture of the chart-based constituency parser with span attention, with an example partial input sentence and its output. The right part of the figure shows the categorical span attention, where extracted n-grams in span (n', n'') are categorized by their length so that n-grams in different categories are weighted separately (different colors refer to different n-gram categories). Note that for normal span attention, all n-grams are weighted together, where attention $w_{n', n''}$ directly corresponds to $\alpha_{n', n''}$ in the figure.

grams are adequately utilized. Experiments on three benchmark datasets demonstrate state-of-the-art performance, confirming our approach’s effectiveness.

4.2.1 The Approach

Our approach follows the chart-based paradigm for constituency parsing, where the parse tree \mathcal{Y} of an input sentence $\mathcal{X} = x_1 x_2 \cdots x_{n'} \cdots x_{n''} \cdots x_N$ is represented as a set of labeled spans. A span is denoted by a triplet (n', n'', y) with n' and n'' referring to the beginning and ending positions of a span with a label y (e.g., NP, VP, PP, etc.). The architecture of our approach is shown in Figure 4.4. The left side is the backbone chart-based parser. It assigns real value scores $s(n', n'', y)$ to the labeled spans, then computes the score of a candidate tree by summing up the scores of all its spans,

and finally chooses a valid tree $\hat{\mathcal{Y}}$ with the highest score s by

$$\hat{\mathcal{Y}} = \arg \max_{\mathcal{Y}} \sum_{(n', n'', y) \in \mathcal{Y}} s(n', n'', y), \quad 0 < n' < n'' \leq N \quad (4.6)$$

In this section, we start with a brief introduction of neural chart-based parsing, then describe our span attention, and end with an illustration of incorporating span attention into the parsing process.

Neural Chart-based Parsing

Recent neural chart-based parsers (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019; Zhou and Zhao, 2019) follow the encoder-decoder way, where the encoder receives \mathcal{X} and generates a sequence of context-sensitive hidden vectors (denoted as $\mathbf{h}_{n'}$ and $\mathbf{h}_{n''}$ for $x_{n'}$ and $x_{n''}$, respectively), which are used to compute the span representation $\mathbf{r}_{n', n''}$ for (n', n'') by subtraction: $\mathbf{r}_{n', n''} = \mathbf{h}_{n''} - \mathbf{h}_{n'}$. This span representation assumes that, for a recurrent model, e.g., LSTM, its hidden vector at each time step relies on the previous ones so that such subtraction could, to some extent, capture the contextual information of all the words in that span.⁸

For decoders, most recent neural chart-based parsers follow the strategy proposed by Stern et al. (2017), where all span representations $\mathbf{r}_{n', n''}$ are fed into a variant of Cocke–Younger–Kasami (CYK) algorithm to generate a globally optimized tree for each sentence. Normally, $\mathbf{r}_{n', n''}$ is fed into multi-layer perceptrons (MLP) to compute its scores $s(n', n'', \cdot)$ over the label set. Afterwards, a recursion function is applied to find the highest score $s^*(n', n'')$ of span (n', n'') , which is computed by searching the best constituency label and the corresponding boundary n''' ($n' < n''' < n''$) by

$$\begin{aligned} s^*(n', n'') &= \max_y s(n', n'', y) \\ &+ \max_{n' < n''' < n''} [s^*(n', n''') + s^*(n''', n'')] \end{aligned} \quad (4.7)$$

Note that in the special case where $n'' = n' + 1$, the best score only relies on the candidate label:

$$s^*(n', n'') = \max_y s(n', n'', y) \quad (4.8)$$

Therefore, to parse the entire sentence, one computes $s^*(1, N)$ through the above steps and uses a back pointer to recover the full tree structure.

⁸Note that the focus of the section is to improve the widely used span representation (i.e., by hidden vector subtraction) proposed by Stern et al. (2017) so as to make a fair comparison, although there are other possible approaches to representing a span (e.g., max pooling).

Span Attention

Although the encoding from subtraction of hidden states is demonstrated to be effective (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019), the subtraction might not represent all the crucial information in the text span. Especially, for Transformer-based encoders, unlike recurrent models, their $\mathbf{h}_{n'}$ and $\mathbf{h}_{n''}$ have no strong dependency on each other so that subtraction may fail to fully capture the contextual information in the span, especially when the span is long. Since n-grams are a good source of the information in the text span, we propose span attention to incorporate weighted n-gram information into span representations to help score the spans (n', n'', y) .

In detail, for each span (n', n'') in \mathcal{X} , we extract all n-grams in that span that appear in Lexicon \mathcal{V} to form a set $\mathcal{C}_{n',n''} = \{c_{n',n'',1}, c_{n',n'',2}, \dots, c_{n',n'',t}, \dots, c_{n',n'',T}\}$ ⁹ and use the set in span attention. The attention of each n-gram $c_{n',n'',t}$ for (n', n'') is activated by

$$w_{n',n'',t} = \frac{\exp(\mathbf{r}_{n',n''}^\top \cdot \alpha_{n',n'',t})}{\sum_{t=1}^T \exp(\mathbf{r}_{n',n''}^\top \cdot \alpha_{n',n'',t})} \quad (4.9)$$

where $\alpha_{n',n'',t}$ is the embedding of $c_{n',n'',t}$ whose dimension is identical to that of $\mathbf{r}_{n',n''}$. The resulting attention vector $\alpha_{n',n''}$ is thus computed by the weighted average of n-gram embeddings by

$$\alpha_{n',n''} = \sum_{t=1}^T a_{n',n'',t} \alpha_{n',n'',t} \quad (4.10)$$

and it is used to enhance the span representation.

In normal attention, all n-grams are weighted globally and short n-grams may dominate the attention because they occur much more frequently than long ones and are intensively updated. However, there are cases that long n-grams can play an important role in parsing when they carry useful context and boundary information. Therefore, we extend the span attention with a category mechanism (namely, categorical span attention) by grouping n-grams based on their lengths and weighting them within each category.¹⁰ In doing so, all n-grams in \mathcal{V} are categorized into U groups according to their lengths, i.e., $\mathcal{C}_{n',n''} = \{\mathcal{C}_{n',n'',1}, \mathcal{C}_{n',n'',2}, \dots, \mathcal{C}_{n',n'',u}, \dots, \mathcal{C}_{n',n'',U}\}$, with $u \in [1, U]$ denoting the n-gram length. Then, for each category with n-grams in length u , we follow the same

⁹Actually, the value of T is relevant to the span (n', n'') . We use T here for better presentation.

¹⁰We use length as the categorization criterion for the following reasons. First, n-gram frequencies vary in different datasets and it is hard to find an appropriate scheme to divide them; second, n-grams with the same length may have similar ability to deliver contextual information, so they are suitable to be grouped by such ability.

process in Eq. (4.9) and (4.10) to compute $a_{n',n'',t}^{(u)}$ and $\mathbf{a}_{n',n''}^{(u)}$. The final attention is obtained from the concatenation of all categorical attentions by

$$\alpha_{n',n''} = \bigoplus_{1 \leq u \leq U} \delta_u \alpha_{n',n''}^{(u)} \quad (4.11)$$

with a positive trainable parameter δ_u to balance the contribution of attentions from different categories.

Parsing with Span Attention

The backbone parser follows Kitaev et al. (2019) to use BERT as the encoder, where $\mathbf{r}_{n',n''} = \mathbf{h}_{n''} - \mathbf{h}_{n'}$ is applied to represent the span (n', n'') . Once $\alpha_{n',n''}$ is obtained from the span attention for (n', n'') , we incorporate it into the backbone parsing process by directly concatenating it with $\mathbf{r}_{n',n''}$: $\mathbf{r}'_{n',n''} = \mathbf{r}_{n',n''} \oplus \alpha_{n',n''}$. Then, we apply two fully connected layers with *ReLU* activation function to $\mathbf{r}'_{n',n''}$ and compute the span scores $s(n', n'', \cdot)$ over the label set, which can be formalized by:

$$\mathbf{o}_{n',n''} = \text{ReLU}(\text{LN}(\mathbf{W}_1 \cdot \mathbf{r}'_{n',n''} + \mathbf{b}_1)) \quad (4.12)$$

and

$$s(n', n'', \cdot) = \mathbf{W}_2 \cdot \mathbf{o}_{n',n''} + \mathbf{b}_2 \quad (4.13)$$

Herein, *LN* denotes the layer normalization operation; \mathbf{W}_1 , \mathbf{W}_2 and \mathbf{b}_1 , \mathbf{b}_2 are trainable parameters in the fully connected layers. Afterwards, we use Eq. (4.7) and (4.8) to recursively find the highest score $s_{best}(1, N)$, and use a back pointer to recover the globally optimized parse tree.

4.2.2 Experiment Settings

Datasets

We test our approach on Arabic, Chinese and English benchmark datasets, namely part 1-3 of the Arabic Penn Treebank 2.0 (ATB) (Maamouri et al., 2004), the Chinese Penn Treebank 5 (CTB5) (Xue et al., 2005), and Penn Treebank 3 (PTB) (Marcus et al., 1993).¹¹ For ATB, we follow Chiang et al.

¹¹All the datasets are obtained from the official release of Linguistic Data Consortium. The catalog numbers for ATB part 1-3 are LDC2003T06, LDC2004T02, LDC2005T20, for CTB5 is LDC2005T01, and for PTB is LDC99T42.

Datasets		Sent	Token	ASL
ATB	Train	16K	596K	31.4
	Dev	2K	70K	30.5
	Test	2K	70K	29.9
CTB5	Train	17K	478K	27.4
	Dev	350	7K	19.5
	Test	348	8K	23.0
PTB	Train	40K	950K	23.9
	Dev	2K	40K	23.6
	Test	2K	57K	23.5
Brown (Full)		24K	458K	19.0
Genia (Full)		17K	446K	26.2

Table 4.8: The statistics of all experimental datasets (with splits) in terms of sentence and token numbers, and average sentence length (ASL).

(2006) and Green and Manning (2010) to use their split¹² to get the training/dev/test sets and convert the texts in the dataset from Buckwalter transliteration¹³ to modern standard Arabic. For CTB5 and PTB, we follow Shen et al. (2018) and Kamigaito et al. (2017) to split the datasets. Moreover, we use the Brown Corpus (Marcus et al., 1993) and Genia (Tateisi et al., 2005) for cross-domain experiments.¹⁴ For all datasets, we follow Suzuki et al. (2018) to clean up the raw data¹⁵ and report the statistics of each resulted dataset in Table 4.8.

¹²Such split uses the “Johns Hopkins 2005 Workshop” standard, for which we follow the detailed split guideline offered by <https://nlp.stanford.edu/software/parser-arabic-data-splits.shtml>.

¹³<http://languagelog.ldc.upenn.edu/myl/ldc/morph/buckwalter.html>

¹⁴The Brown Corpus is obtained together with PTB (LDC99T42), and the Genia corpus is obtained by its official PTB format from <https://nlp.stanford.edu/~mcclosky/biomedical.html>.

¹⁵We use the clean-up code from <https://github.com/nikitakit/parser-data-gen>.

Data	Models	- POS					+ POS				
		Parm	P	R	F1	M	Parm	P	R	F1	M
ATB	BERT	188M	82.99	82.99	82.99	18.87	188M	82.96	83.17	83.07	19.09
	+ SA	191M	83.36	83.05	83.21	19.13	191M	83.37	83.12	83.24	19.43
	+ CatSA	192M	83.33	83.20	83.27	20.04	192M	83.41	83.20	83.30	19.65
CTB5	BERT	113M	93.95	93.35	93.65	47.71	113M	94.30	93.88	94.09	48.86
	+ SA	116M	94.07	93.39	93.73	49.43	116M	94.80	93.73	94.26	49.14
	+ CatSA	117M	94.02	93.65	93.83	50.00	117M	94.70	94.00	94.35	50.00
	ZEN	235M	93.82	93.65	93.73	50.29	235M	94.37	93.69	94.03	48.87
	+ SA	238M	94.08	93.53	93.80	51.14	238M	94.68	93.81	94.24	51.43
	+ CatSA	239M	94.23	93.66	93.94	51.41	239M	94.69	93.91	94.30	52.00
PTB	BERT-LC	344M	95.71	95.53	95.62	54.06	344M	95.71	95.61	95.66	53.35
	+ SA	349M	95.80	95.55	95.68	53.94	349M	95.71	95.70	95.70	54.29
	+ CatSA	350M	96.02	95.51	95.77	54.64	350M	95.79	95.85	95.82	55.79
	BERT-LU	345M	95.61	95.59	95.60	54.29	345M	95.59	95.76	95.67	54.24
	+ SA	350M	95.61	95.71	95.66	54.24	350M	95.69	95.75	95.72	54.53
	+ CatSA	351M	95.76	95.74	95.75	55.29	351M	95.77	95.84	95.80	54.71
	XLNet-LC	371M	95.78	95.79	95.78	54.81	371M	95.97	95.60	95.79	54.70
	+ SA	375M	95.83	95.95	95.89	54.94	375M	95.92	95.95	95.93	55.71
	+ CatSA	376M	96.02	95.84	95.93	55.88	376M	95.97	96.02	95.99	56.06

Table 4.9: Experiment results in terms of precision (P), recall (R), F-score (F1) and complete match score (M) of our models on the development set of ATB, CTB5 and PTB with different configurations, i.e., with and without POS, span attention (SA), and categorical span attention (CatSA). PARM reports the number of trainable parameters in each model.

Implementation Details

In our experiments, we use BERT (Devlin et al., 2019) as the basic encoder for all three languages and use ZEN (Diao et al., 2020) and XLNet-large (Yang et al., 2019) for Chinese and English, respectively.¹⁶ For BERT, ZEN, and XLNet, we use the default hyper-parameter settings. (e.g., 24 layers with 1024 dimensional hidden vector for the large models). In addition, following Kitaev et al. (2019), Zhou and Zhao (2019) and Mrini et al. (2019), we add three additional token-level self-attention layers to the top of BERT, ZEN, and XLNet.

For n-gram extraction, we employ the pointwise mutual information (PMI) to collect all n-grams

¹⁶We download BERT models for Arabic and English from <https://github.com/google-research/bert>, and for Chinese from <https://s3.amazonaws.com/models.huggingface.co/>. We download ZEN and XLNet at <https://github.com/sinovation/ZEN> and <https://github.com/zihangdai/xlnet>.

Models	ATB			CTB5			PTB		
	P	R	F1	P	R	F1	P	R	F1
Green and Manning (2010)	78.92	77.72	78.32	-	-	-	-	-	-
Kitaev and Klein (2018)	-	-	-	-	-	-	95.40	94.85	95.13
Kitaev et al. (2019) (BERT)	-	-	-	91.96	91.55	91.75	95.73	95.46	95.59
Fried et al. (2019) (BERT)	-	-	-	-	-	92.14	-	-	95.71
Zhou and Zhao (2019) (BERT)	-	-	-	92.03	92.33	92.18	95.70	95.98	95.84
Zhou and Zhao (2019) (XLNet)	-	-	-	-	-	-	96.21	96.46	96.33
*Mrini et al. (2019) (BERT/XLNet + POS)	-	-	-	91.85	93.45	92.64	96.24	96.53	96.38
SCT (Manning et al., 2014a)	68.33	71.78	70.02	†	†	†	86.21	86.73	86.47
BNP (Kitaev and Klein, 2018)	72.84	76.59	74.67	91.83	91.53	91.68	95.46	94.89	95.17
BERT	83.06	82.87	82.96	92.16	91.98	92.07	95.91	95.17	95.54
+ SA	83.25	82.85	83.05	92.31	92.03	92.17	96.04	95.40	95.72
+ CatSA	83.40	83.11	83.26	92.25	92.14	92.20	96.11	95.58	95.85
ZEN/XLNet	-	-	-	92.20	92.05	92.13	96.52	95.70	96.11
+ SA	-	-	-	92.34	92.02	92.18	96.58	96.03	96.31
+ CatSA	-	-	-	92.50	91.98	92.24	96.64	96.07	96.36
*BERT + POS	82.98	82.97	82.97	92.52	92.06	92.29	95.92	95.27	95.60
+ SA	83.36	82.80	83.08	92.61	92.20	92.40	95.96	95.51	95.73
+ CatSA	83.48	83.07	83.27	92.83	92.50	92.66	96.09	95.62	95.86
*ZEN/XLNet + POS	-	-	-	92.37	92.16	92.26	96.42	95.86	96.14
+ SA	-	-	-	92.40	92.32	92.36	96.56	96.10	96.33
+ CatSA	-	-	-	92.61	92.42	92.52	96.61	96.19	96.40

Table 4.10: Comparing our best performing models with previous studies and prevailing toolkits (i.e., SCT and BNP). The results for SCT are not comparable to other systems including ours (as indicated by †) because SCT is trained on a different dataset. Models marked by * use predicted POS tags as additional input.

($n \leq 5$)¹⁷ appearing at least twice in the training and development sets combined. We randomly initialize all n-gram embeddings used in our attention module¹⁸ with their dimension matching that of the hidden vectors obtained from the encoder (e.g., 1024 for BERT-large). Besides, we run our experiments with and without predicted part-of-speech (POS) tags. Following previous studies, for the experiments without POS tags, we take sentences as the only input; for the experiments with POS

¹⁷We empirically set the max n-gram length to 5 as a unified threshold for all three languages.

¹⁸We also try initializing the n-grams with pre-trained embeddings (Pennington et al., 2014; Song et al., 2018c; Yamada et al., 2020), where the results show small differences.

tags, we obtain the POS tags from Stanford POS Tagger (Toutanova et al., 2003) and incorporate the POS tags by directly concatenating their embeddings with the output of the BERT/ZEN/XLNet encoder.

Following previous studies (Suzuki et al., 2018; Kitaev et al., 2019), we use hinge loss during the training process and evaluate different models by by precision, recall, F1 score, and complete match score via the standard evaluation toolkit Evalb¹⁹. During the training process, we try three learning rates, i.e., 5e-5, 1e-5, 5e-6, with a fixed random seed, pick the model with the best F1 score on the development set, and evaluate it on the test set.

4.2.3 Results and Analyses

Overall Performance

In the main experiment, we compare the proposed models with and without the span attention to explore the effect of the span attention on chart-based constituency parsing. For models with the span attention, we also run the settings with and without the categorical mechanism. The results (i.e., precision, recall, F1 score, and complete match scores of all models, as well as their number of trainable parameters) with different configurations (including whether to use the predicted POS tags) on the development sets of ATB, CTB5, and PTB are reported in Table 4.9.

There are several observations. First, the span attention over n-grams shows its generalization ability, where consistent improvements of F1 over the baseline models are observed on all languages under different settings (i.e., with and without using predicted POS tags; using BERT or XLNet encoders). Second, compared with span attention without the category mechanism, in which n-grams are weighted together, models with categorical span attention perform better on both F1 and complete match scores with a relatively small increase of parameter numbers (around $1M$). Particularly, for the complete match scores, the span attention with normal attentions does not outperform the baseline models in some cases, whereas the categorical span attention mechanism does in all cases. These results could be explained by that frequent short n-grams dominate the general attentions so that the long ones containing more contextual information fail to function well in filling the missing information in the span representation, and thus harm the understanding of long spans, which results

¹⁹<https://nlp.cs.nyu.edu/evalb/>

Models	Brown	Genia
BERT (Fried et al., 2019)	93.10	87.54
BERT	93.13	87.58
+ SA	93.24	87.50
+ CatSA	93.29	87.53

Table 4.11: Cross-domain experiment results (F1 scores) from previous studies and our models (based on BERT-LC), on the entire Brown and Genia corpora when trained from the training set of PTB.

in inferior results in complete match score. In contrast, the categorical span attention is able to weight n-grams in different length separately, so that the attentions are not dominated by high-frequency short n-grams and thus reasonable weights can be assigned to long n-grams. Therefore, our model can learn from the important long n-grams and have a good performance on the long spans, which results in consistent improvements over baseline models in complete match scores. Third, on CTB5, models with ZEN encoder consistently outperform the ones with BERT without using POS tags, while they fail to do so with the POS tags as the additional input, which suggests that the predicted POS tags may have more conflict with ZEN compared with BERT.

Moreover, we run our models on the test set of each dataset and compare the results with previous studies, as well as the ones from prevailing parsers, i.e., Stanford CoreNLP Toolkits (SCT)²⁰ (Manning et al., 2014a) and Berkeley Neural Parser (BNP)²¹ (Kitaev and Klein, 2018). The results are reported in Table 4.10, where the models using predicted POS tags are marked with “*”.²² Our models with CatSA outperform previous best performing models from Zhou and Zhao (2019) and Mrini et al. (2019) under different settings (i.e., whether to use the predicted POS tags), and achieve state-of-the-art performance on all datasets. Compared with Zhou and Zhao (2019) and Mrini et al. (2019) which improve constituency parsing by leveraging the dependency information when training

²⁰We use the version of 3.9.2 obtained from <https://stanfordnlp.github.io/CoreNLP/>.

²¹We obtain their models from <https://github.com/nikitakit/self-attentive-parser>.

²²For our models with BERT encoder, we only report the results of the ones using the cased version of BERT-large.

their head phrase structure grammar (HPSG) parser, our approach enhances the task from another direction by incorporating n-gram information through the span attentions as a way to address the limitation of using hidden vector subtraction to represent spans.

Cross-domain Experiments

To further explore whether our approach can be generalized across domains, we follow the setting of Fried et al. (2019) to conduct cross-domain experiments on the Brown and Genia corpus using the models with SA and CatSA, as well as their corresponding baseline. Note that, for fair comparison, we use BERT-large cased as the encoder without using the predicted POS tags. We follow Fried et al. (2019) to train models on the training set of PTB and evaluate them on the entire Brown corpus and the entire Genia corpus. To construct \mathcal{N} in this experiment, we extract n-grams by PMI from the training set of PTB. The results (F1 scores) are reported in Table 4.11. From the table, we find that our model with categorical span attentions (+ CatSA) outperforms the BERT baseline (Fried et al., 2019) on the Brown corpus while fails to do so on the Genia corpus. The explanation could be that the distance between Genia (medical domain) and PTB (news wire domain) is much larger than that between Brown and PTB, so that the n-gram overlap in two domains are limited and thus has little influence to the target domain.

Effect of CatSA on Long Sentences

To explore the effect of our approach, we investigate our best performing models (where predicted POS tags are used) with the span attention module and the corresponding baselines on different length of sentences in the test sets. The curves of F1 scores with respect to the minimal test sentence length (the horizontal axis) from different models on ATB, CTB5, and PTB are illustrated in Figure 4.5(a), 4.5(b), and 4.5(c), respectively.²³

In general, long sentences are harder to parse and thus all models' performance degrades when sentence length increases. Yet, our models with CatSA outperform the baseline for all sentence groups and the gap is bigger for long sentences, which indicates our approach can handle long sentences better than the baselines. One possible explanation for this is that long sentences will

²³Given the variance of average sentence length in different datasets (see Table 4.8), we set the minimal length from 5 to 50 on CTB5 and PTB, and 15 to 60 on ATB, with a step of 5.

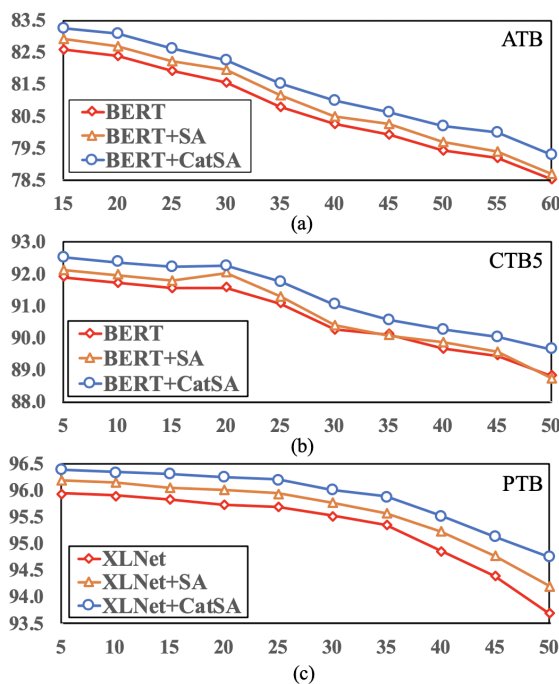


Figure 4.5: The F1 curves with respect to the minimal test sentence length (the horizontal axis) of different models performed on ATB (a), CTB (b), and PTB (c).

have larger text spans and may require more long-distance contextual information. Our approach incorporates n-gram information into the span representation and thus can appropriately leverage the infrequent long n-grams by separately weighting them in different categories.

Analysis on Different N-gram Lengths

To test using n-grams in different length, we conduct an ablation study on the n-grams with respect to their length. In doing so, we conduct experiments on the best performing models (where predicted POS tags are used) with the span attention module, by restricting that n-grams whose length are larger than a threshold is excluded from the lexicon \mathcal{N} . We try the threshold from 1 to 5 and demonstrate the curves (F-scores) on the test set of ATB, CTB5, and PTB in Figure 4.6(a), (b), and (c), respectively. The results of their corresponding baselines are also represented in red curves for reference. It is found from the curves that our models with span attentions consistently outperform the baseline models, which indicates the robustness of our approach with respect to different n-grams used in the

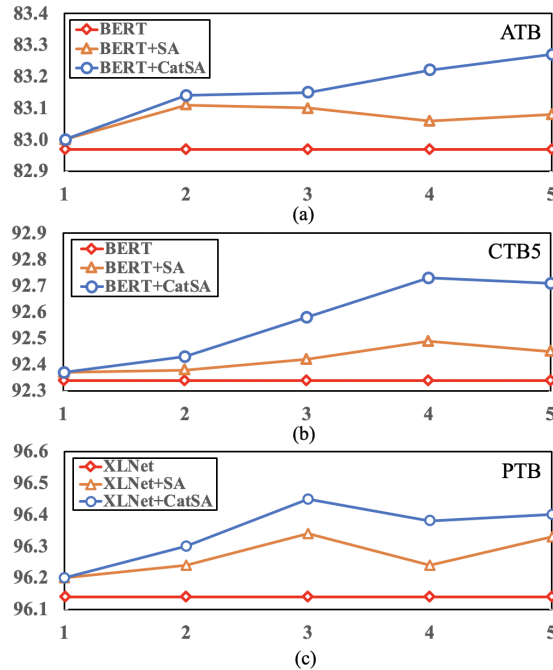


Figure 4.6: The F1 curves with respect to the max length (the horizontal axis) of n-grams used in different models performed on ATB (a), CTB (b), and PTB (c).

model. In addition, for different languages, the n-gram threshold varies when the best performance is obtained. For example, the best performing model on English is with three words as the maximum length of n-grams, while that is five for Arabic and four for Chinese.

Moreover, to investigate how categorical span attention addresses the problem that high-frequency short n-grams can dominate the general attention, we run the best-performing models with span attention on the whole ATB, CTB5, and PTB datasets, obtain the total weight assigned to each n-gram and compute the average weight for the n-grams in each n-gram length category. Figure 4.7 shows the histograms of the average weights from models with SA and CatSA.

The histograms show that the models with SA (the orange bars) tend to assign short n-grams relatively high weights, especially the uni-grams. This is not surprising because short n-grams occur more frequently and are thus updated more times than long ones. In contrast, the models with CatSA show a different weight distribution (the blue bars) among n-grams with different lengths, which indicates that the CatSA module could balance the weight distribution and thus enable the model to learn from infrequent long n-grams.

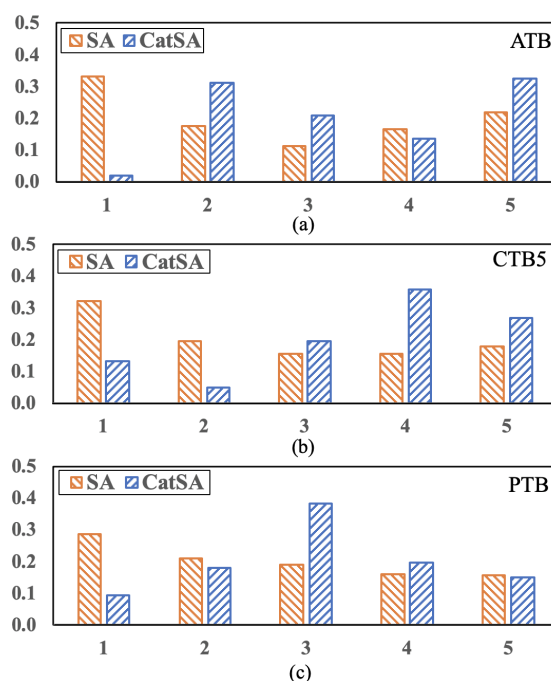


Figure 4.7: The histograms of average weights assigned to n-gram categories in different lengths, with weights from SA and CatSA show different patterns.

Case Study

To illustrate how our model improves baselines with the span attention, especially for long sentences, we show the parse trees produced by the two models for an example sentence in Figure 4.8, where the superscript for the internal node is the height of the subtree rooted at that node. In this case, our model correctly attaches the “*PP*” (“*with two ... utilities*”) containing 24 words to the verb “*compete*”, while the baseline attaches it to the noun “*customers*”. Since the distances between the boundary positions of the wrongly predicted spans (highlighted in red) are relatively long, the baseline system, which simply represents the span as subtraction of the hidden vectors at the boundary positions, may fail to capture the important context information within the text span. In contrast, the span representations used in our model are enhanced by weighted n-gram information and thus contain more context information. Therefore, in deciding which component (i.e., “*compete*” or “*customer*”) the *with-PP* should attach to, n-grams (e.g., the uni-gram “*companies*”) may provide useful cues, since “customers with companies” is less likely than “compete with companies”.

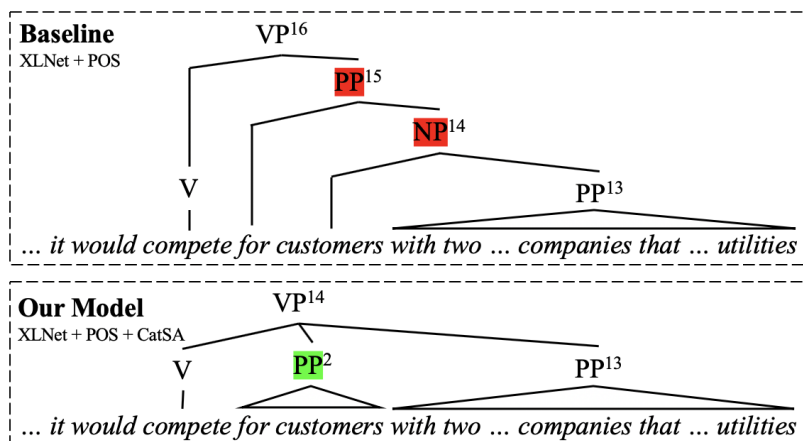


Figure 4.8: An example sentence with its parsing results from the best-performing baseline and our model. The correct and wrong parsing results are highlighted on the span labels by green and red, respectively. The superscripts on the span labels illustrate their heights. “V” is a POS tag so there is no height for it.

4.3 CCG Supertagging

Recent neural approaches for CCG supertagging typically formulate it as a sequence labeling task, leveraging powerful encoders such as recurrent neural networks to capture contextual information (Lewis et al., 2016; Vaswani et al., 2016; Clark et al., 2018). However, these approaches pay limited attention to explicitly modeling additional contextual features, particularly strong lexical relationships among words. Graph convolutional networks (GCN) is demonstrated as an effective architecture for modeling token relationships in NLP tasks (Marcheggiani and Titov, 2017; De Cao et al., 2019; Huang et al., 2019a); yet, directly applying conventional GCN to supertagging is challenging, as they typically rely on dependency structures, whose quality and availability cannot be guaranteed. Although frequent lexical chunks (n-grams) offer an alternative source of contextual information, existing studies treat all token-token connections equally, potentially obscuring the importance of meaningful interactions and limiting model effectiveness.

In this thesis, we propose an attentive GCN (A-GCN) specifically designed for CCG supertagging, which incorporates meaningful lexical chunks extracted through unsupervised approaches. Specifically, our approach constructs an input graph with two distinct edge types, representing relations

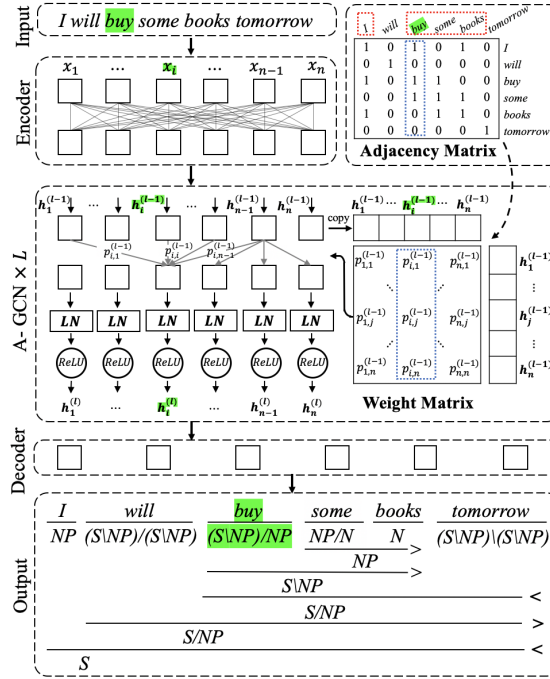


Figure 4.9: The architecture of our CCG supertagger with A-GCN and an example input sentence with its supertagging and parsing output. The supertagging process for “buy” is highlighted in green. The adjacency matrix illustrates the edges of the graph that is built upon the chunks (n-grams) extracted from the lexicon \mathcal{N} , with the chunks illustrated in the red boxes.

within and across extracted chunks. We further equip the GCN with an attention mechanism to dynamically weigh these edges, enabling discriminative modeling of relevant contextual information without relying on external resources or dependency parsers. Experiment evaluations on the standard CCGbank benchmark (Hockenmaier and Steedman, 2007) demonstrate the effectiveness of our approach to leveraging the chunk information.

4.3.1 The Approach

We treat CCG supertagging as a sequence labeling task, where the input is a sentence with n words $\mathcal{X} = x_1 x_2 \dots x_n \dots x_N$, and the output is a sequence of supertags $\hat{\mathcal{Y}} = \hat{y}_1 \hat{y}_2 \dots \hat{y}_n \dots \hat{y}_N$. Our approach uses A-GCN to incorporate information of token pairs through a graph; the graph is built based on n-grams in the input sentence that appear in a lexicon \mathcal{V} . This lexicon consists of n-grams

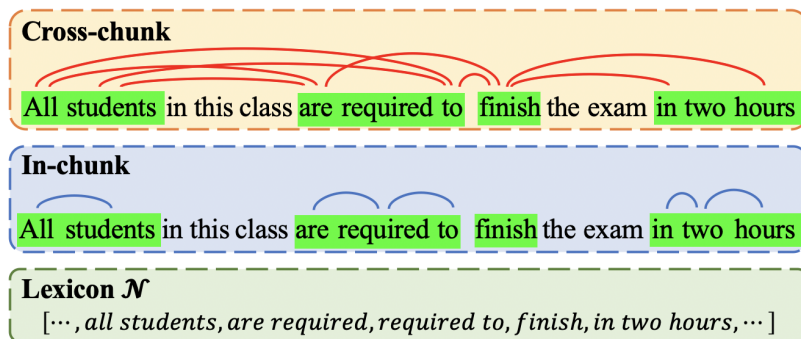


Figure 4.10: Examples of the two types of edges for building the graph in an input sentence, in which chunks (n-grams) extracted from the lexicon \mathcal{V} are highlighted in green; example in-chunk and cross-chunk edges are marked in blue and red color, respectively.

automatically extracted from raw corpora by unsupervised approaches. The overall architecture of our tagger is illustrated in Figure 4.9, with an input sentence and corresponding supertagging and parsing output. The details of the main components of the architecture are provided below.

Graph Construction based on Chunks

Since CCG supertagging is also a parsing task, we do not want our approach to rely on the existence of a dependency parser. Without such a parser, we need an alternative for finding good token pairs to build the graph in A-GCN (which is equivalent to building the adjacency matrix \mathbf{A}). Inspired by the studies that leverage chunks (n-grams) as effective features to carry contextual information and enhance model performance (Song et al., 2009b; Song and Xia, 2012; Ishiwatari et al., 2017; Yoon et al., 2018; Zhang et al., 2019b; Tian et al., 2020b), we propose to construct the graph based on the chunks (n-grams) extracted from an n-gram lexicon \mathcal{V} constructed by PMI.

Then for graph building, given an input sentence \mathcal{X} , we find all the n-grams in \mathcal{X} that appear in \mathcal{V} . In our approach for CCG supertagging, a *chunk* is either a n-gram that does not overlap with other n-grams or a text span that covers multiple overlapping n-grams. For example, in Figure 4.10, we find four chunks (i.e., “*all students*”, “*are required to*”, “*finish*”, and “*in two hours*”) in the example sentence according to the lexicon \mathcal{N} (the chunks are highlighted in green). In these chunks, “*all students*”, “*finish*”, and “*in two hours*” are non-overlapping n-grams included in the lexicon and “*are required to*” is a text span that covers the overlapping n-grams “*are required*” and “*required to*”. In

most cases, the adjacent words within the same chunk tend to have a strong word-word relation in terms of co-occurrence, and thus we can build the graph and its adjacency matrix accordingly.

Based on the chunks, we construct the graph by two types of edges, i.e., the in-chunk and cross-chunk ones: the first type is to model local word pairs, and the graph includes edges between any two adjacent words within the same chunk. For example, as shown in Figure 4.10, the in-chunk edges (blue lines) for the chunk “*in two hours*” are “(*in, two*)” and “(*two hours*)”. The second type is to model cross chunk word pairs, which are built from any two adjacent chunks with the starting and ending words in the two chunks connected. The motivation for using the starting and ending words is that English phrases tend to be head-initial (e.g., verb phrase such as “*buy some books*”) or head-final (e.g., adjective phrase such as “*red apples*”) in many cases. E.g., for the two chunks “*all students*” and “*are required to*” in Figure 4.10, the corresponding cross-chunk edges (red lines) are “(*all, are*)”, “(*all, to*)”, “(*students, are*)”, and “(*students, to*)”. The graph is equivalent to the adjacency matrix \mathcal{A} , where $a_{n',n''} = 1$ if there is an edge between $x_{n'}$ and $x_{n''}$ in the graph or $n' = n''$, and $a_{n',n''} = 0$ otherwise.²⁴

The Attentive GCN

When learning from a graph, conventional GCN models treat all word pairs from the graph equally, and thus are unable to account for the possibility that the contribution of different $x_{n''}$ on $x_{n'}$ may vary. Particularly for our graph built from chunks, it is important to be able to distinguish different word pairs because all the chunks and the graph are constructed automatically without a dependency parser. Therefore, we apply an attention mechanism to the adjacency matrix and adapt Eq. (3.12) used in the normal GCN for our A-GCN by replacing the $a_{n',n''} \in \{0, 1\}$ by a weight $w_{n',n''}^{(l)} \in (0, 1)$. For each $x_{n'}$ and all its associated $x_{n''}$, the weight $w_{n',n''}^{(l)}$ for this word pair is computed by

$$w_{n',n''}^{(l)} = \frac{a_{n',n''} \cdot \exp(\mathbf{h}_{n'}^{(l-1)} \cdot \mathbf{W}_{pos}^{(l)} \cdot \mathbf{h}_{n''}^{(l-1)})}{\sum_{n''=1}^N a_{n',n''} \cdot \exp(\mathbf{h}_{n'}^{(l-1)} \cdot \mathbf{W}_{pos}^{(l)} \cdot \mathbf{h}_{n''}^{(l-1)})} \quad (4.14)$$

where $\mathbf{W}_{pos}^{(l)}$ models the positional relation (i.e., *left*, *right*, or *self*) between x_i and x_j and it has three choices, i.e., $\mathbf{W}_{left}^{(l)}$, $\mathbf{W}_{right}^{(l)}$, and $\mathbf{W}_{self}^{(l)}$ for different n' and n'' ,²⁵ with each of them a trainable

²⁴We do not distinguish the two types of edges in \mathbf{A} .

²⁵For example, $\mathbf{W}_{pos}^{(l)} = \mathbf{W}_{left}^{(l)}$, if $n'' < n'$.

	Train	Dev	Test
Section No.	2-21	0	23
Sentence #	39,604	1,913	2,407
Token #	929,552	45,422	55,371

Table 4.12: The train/dev/test splits of English CCGBank and the statistics of sentences and tokens in them.

square matrix in the l -th layer of A-GCN.

Supertagging with A-GCN

To conduct supertagging with A-GCN, we firstly obtain the hidden vector $\mathbf{h}_{n'}^{(0)}$ for $x_{n'}$ from BERT (Devlin et al., 2019) to feed into the first GCN layer. Upon receiving the encoding results from A-GCN, the following supertagging process is straightforward: each $\mathbf{h}_{n'}^{(L)}$ is obtained from the last A-GCN layer and aligned with the output by $\mathbf{o}_{n'} = \mathbf{W}_d \cdot \mathbf{h}_{n'}^{(L)}$, where \mathbf{W}_d is a trainable matrix for the alignment. Then, a *softmax* decoder is used to predict the supertag for each token.

4.3.2 Experiment Settings

Datasets

We run experiments on the English CCGbank (Hockenmaier and Steedman, 2007)²⁶ and follow Clark and Curran (2007) to split it into train/dev/test sets, whose statistics (sentence and word numbers) are reported in Table 4.12.

Implementation Details

To construct n-gram lexicon \mathcal{N} for building the edges in our graph, we perform PMI on the training set of CCGbank to extract n-grams whose length is between $[1, 5]$, with the threshold of the PMI

²⁶The official dataset is obtained from <https://catalog.ldc.upenn.edu/LDC2005T13>.

Hyper-parameters	Values
Batch Size	16, 32
Drop-out Rate	0.2
Learning Rate	3e-5, 2e-5, 1e-5, 5e-6
Max Sentence Length	300
Random Seed	42
Training Epoch	50
Warm-up Rate	0.1, 0.2

Table 4.13: The list of hyper-parameters tested in our experiments. We run all models with the combination of those hyper-parameters and use the one achieving the highest supertagging results in our final experiments.

score set to 0. For the encoder, we try both cased and uncased BERT-Large (Devlin et al., 2019) with their default settings²⁷ and used two layers for A-GCN. To obtain CCG parse from the generated supertags, we adopt the parsing algorithm used in EasyCCG (Lewis and Steedman, 2014a). We follow previous studies (Lewis and Steedman, 2014a; Lewis et al., 2016; Yoshikawa et al., 2017) to use the most frequent 425 supertags as the tag set and evaluate our model on both the tagging accuracy and the labeled F-scores (LF) of the dependencies converted from CCG parse²⁸.

For other hyper-parameter settings, we test their values as shown in Table 4.13 when training our models. We try all combinations of them for each model and use the one achieving the highest supertagging results in our final experiments. Note that, with the best hyper-parameters, the best performance is achieved with a warm-up rate of 0.1, batch size of 16, and learning rate of 1e-5.

Models	PARM	TAG	LF
BERT-Cased	335M	96.04	90.31
+ A-GCN (Full)	343M	95.93	90.13
+ A-GCN (Chunk)	343M	96.11	90.49
BERT-Uncased	337M	96.06	90.37
+ A-GCN (Full)	345M	95.99	90.21
+ A-GCN (Chunk)	345M	96.17	90.60

Table 4.14: Results (supertagging accuracy and labeled F -scores) of different models with BERT-Large encoder on the development set of CCGbank. “PARM” is the number of trainable parameters in the models; “Full” uses the fully connected graph and “Chunk” uses the graph built based on chunks.

4.3.3 Results and Analysis

Overall Results

To explore the effectiveness of our approach, we run CCG taggers with and without A-GCN, and try two ways to construct the graph: one is a fully connected GCN where edges are built between every two words; the other is our proposed approach with the chunk-based graph. Experimental results on supertagging accuracy (Tag) and labeled F-scores (LF) for parsing on the development set of CCGbank are reported in Table 4.14, with the number of trainable parameters of all models also presented.

The experiments show that, for both cased and uncased BERT encoders, the proposed chunk A-GCN works the best in terms of both supertagging accuracy and parsing results. In contrast, Full A-GCN has inferior performance to the BERT baselines. This contrast shows the importance of appropriate construction of the graphs fed into A-GCN, since the fully connected graph with all words associated with one another may introduce noise word relations and thus yield bad performance.

²⁷We download the pre-trained BERT models from <https://github.com/google-research/bert>.

²⁸We use the “generate” script from C&C parser (Clark and Curran, 2007) to convert CCG parse results to their corresponding dependencies.

Models	TAG	LF
Lewis and Steedman (2014b)	91.3	86.11
Xu et al. (2015a)	93.00	87.07
Lewis et al. (2016)	94.7	88.1
Vaswani et al. (2016)	94.24	88.32
Yoshikawa et al. (2017)	-	90.4
Clark et al. (2018)	96.1	-
Stanojević and Steedman (2019)	95.4	90.5
EasyCCG†	-	86.14
BERT†	96.06	90.34
BERT + A-GCN (Full)†	95.91	90.20
BERT + A-GCN (Chunk)†	96.25	90.58

Table 4.15: Comparison of our models with uncased BERT encoder and previous studies on the test set of CCGbank. Models with “†” use the EasyCCG parser to generate CCG parse trees from the predicted supertags.

Furthermore, we run our models with uncased BERT encoder on the test set and compare the performance with previous studies on both supertagging and parsing. Table 4.15 shows the results, where the studies marked by † use the same parser (i.e., the EasyCCG parser) to generate CCG trees from supertags. Among the previous studies, Stanojević and Steedman (2019) performed CCG parsing directly without the supertagging step, whereas the rest all did supertagging first. Regardless of this difference, our approach performs the best on CCGbank in both supertagging accuracy and parsing LF.

Ablation Study

We conduct an ablation study to explore the effect of the two types of edges and the attention mechanism on our best model. The supertagging and parsing results of models with different configurations are reported in Table 4.16, where the results are categorized into four groups. The first group (ID 1) is the results of the best performing model where all settings are activated; the second

(ID 2-3) is the ablation of either in-chunk or cross-chunk edges with attention; the third (ID 4-6) is the result of using normal GCN without the attention mechanism; and the last group (ID 7) is the baseline model where none of the three settings is activated.

The results show that the model performance drops when either part is ablated (ID 1 vs. ID 2-6). Specifically, removing attention significantly hurts the performance, where all configurations without attention (ID 4-6) shows worse-than-baseline (ID 7) results; this observation confirms the importance of applying attention on GCN. One possible explanation to this phenomenon could be that considerable noises are introduced to the graph because the edges in our graph are derived from chunks and they do not follow syntax in most cases; thus, it is crucial to assign weights to the edges and not treat them with equally. Interestingly, comparing the two types of edges, models with cross-chunk edges yield much higher results than the ones with in-chunk edges only when the attention is not used (ID 5 vs. ID 6), while it is slightly better when attention is applied (ID 2 vs. ID 3). This comparison suggests that in-chunk edges could introduce more noise than cross-chunk edges. So that when the attention is not used (ID 6), the model fails to weight the edges and results in a significant drop on its performance; On the contrary, when the attention is applied (ID 3), our model is able to even the performance of models with in-chunk and cross-chunk edges, which confirms that weighting is essential in selecting useful information for CCG supertagging.

4.4 Summary

In this chapter, we leverage knowledge from raw data to three tasks, namely, CWS, constituency parsing, and CCG supertagging, at the lexical and syntactic levels. We propose different approaches, namely, wordhood memory networks, span attentions, and A-GCN for leveraging the knowledge with PLM. The results demonstrate that the knowledge from the raw data is able to enhance shallow tasks, where improvements are observed over the baselines without using the knowledge. Meanwhile, it is essential to distinguish the knowledge instances that provide hints for the task from the ones that contribute less to the task so as to alleviate the influence of noise on the model performance.

ID	Settings			Tag	LF
	In-chunk	Cross-chunk	Attention		
1	✓	✓	✓	96.25	90.58
2	×	✓	✓	96.18	90.49
3	✓	×	✓	96.11	90.41
4	✓	✓	×	87.26	81.96
5	×	✓	×	94.92	89.75
6	✓	×	×	89.67	84.39
7	×	×	×	96.06	90.34

Table 4.16: Experimental results of models with uncased BERT-Large encoder on the test set of CCGbank, where the in-chunk, cross-chunk edges or the attention mechanism in our A-GCN module is ablated.

Chapter 5

Incorporating Syntax Knowledge

In this chapter, we present our research on incorporating syntax knowledge (i.e., the dependency parse tree) obtained from off-the-shelf toolkits into NER, relation extraction, and ABSA. By leveraging this structured linguistic knowledge, we aim to enhance the models' abilities to capture semantic relations and contextual information, thereby improving their overall performance on these deeper linguistic tasks.

5.1 Named Entity Recognition

Over the past decades, the approach for biomedical named entity recognition (BioNER) transits from the conventional feature-based approaches (Song et al., 2005; Leaman et al., 2015; Leaman and Lu, 2016) to neural approaches leveraging powerful encoders such as BiLSTM (Habibi et al., 2017; Dang et al., 2018). Recently, PLMs (such as BERT) further improve the performance of NLP models for many tasks in the general domain (Peters et al., 2018; Devlin et al., 2019); however, their effectiveness in biomedical domains remains limited without integrating domain-specific knowledge. To overcome this, BioBERT (Lee et al., 2019), which is trained specifically on biomedical corpora, achieves state-of-the-art results for BioNER. However, BioBERT does not explicitly incorporate syntactic structures, which have demonstrated benefits in previous studies for improving BioNER (Liao and Wu, 2012; Chang et al., 2015; Luo et al., 2018). Biomedical sentences frequently present formal and complex structures, making syntactic cues potentially valuable for recognizing named entities. Previous

studies to leverage syntactic information typically concatenate syntactic embeddings directly with word embeddings (Luo et al., 2018; Dang et al., 2018), without paying much attention to the variable importance of such information across different contexts and potentially introducing noise due to the parsing errors. Therefore, an approach that effectively selects and weighs syntactic information, minimizing the negative impact of noisy parsing results, remains underexplored.

In this thesis, we propose **BioKMNER**, a novel approach leveraging key-value memory networks (KVMN) to selectively integrate syntactic information for improved BioNER. Specifically, BioKMNER extracts dependency relations from off-the-shelf parsers. It then uses KVMN to dynamically weigh syntactic information instances based on context similarity, effectively filtering noisy parsing outputs. These weighted syntactic features are concatenated with encoder outputs, enhancing the representation used by the sequence labeling decoder. Experiments on six benchmark BioNER datasets demonstrate that BioKMNER consistently outperforms strong baselines on the datasets, confirming its effectiveness in selectively incorporating syntactic information.

5.1.1 The Approach

The overall architecture of our approach, namely, BioKMNER is shown in Figure 5.1. Following the common approaches in BioNER, we treat it as a sequence labeling task, where the input token sequence $\mathcal{X} = \{x_1, x_2, \dots, x_n, \dots, x_N\}$ is tagged with a sequence of NE labels $\hat{\mathcal{Y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n, \dots, \hat{y}_N\}$. In our approach, we leverage KVMN to incorporate syntactic information. Specifically, context features and their carrying syntactic information instances are mapped to keys and values in KVMN, where the values are weighted according to the comparison between the keys and the input words. In this section, we firstly introduce the syntactic information extraction process, then elaborate the KVMN module used to incorporate the syntactic information, and finally explain the tagging process with the KVMN module.

Syntactic Information Extraction

In our study, we focused on dependency relations as they are demonstrated to be effective for providing structure information of the running text. To obtain such information, we first run an off-the-shelf NLP toolkit on the input sentence \mathcal{X} . Then for each token x_n in \mathcal{X} , we extract the

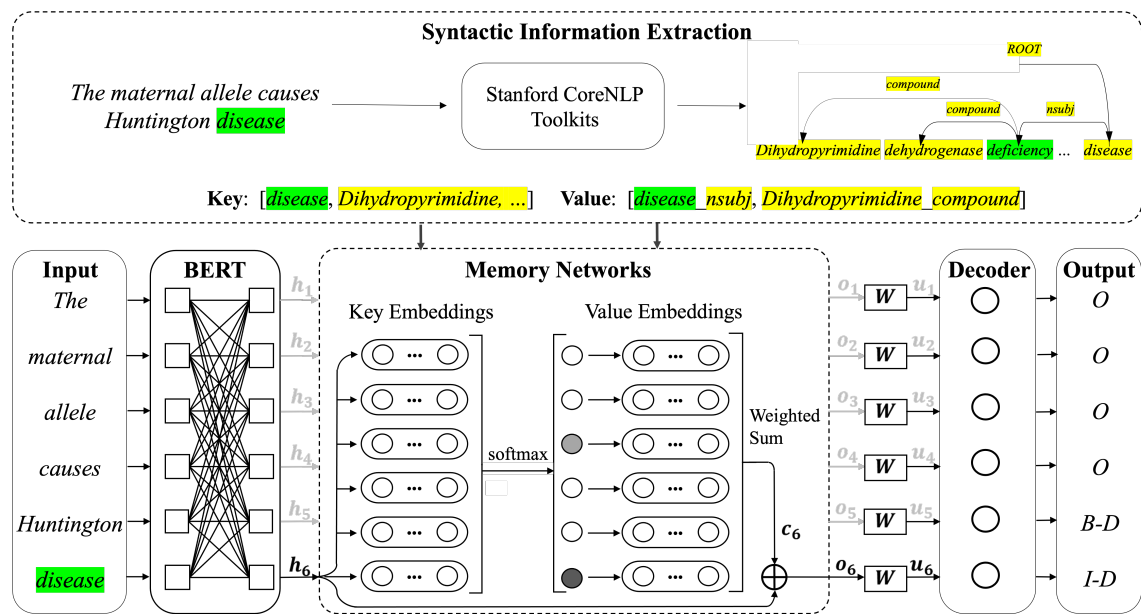


Figure 5.1: The overall architecture of BioKMNER. The top part of the figure shows the syntactic information extraction process: for the input word sequence, we firstly use off-the-shelf NLP toolkits to obtain its syntactic information (e.g., syntax tree), then map the context features and the syntactic information into keys and values, and finally convert them into embeddings. The bottom part is our sequence labeling based BioNER tagger, which uses BioBERT as the encoder and a softmax layer as the decoder. Between the encoder and decoder are the key-value memory networks (KVMN) which weighs syntactic information (values) according to the importance of the context features (keys). The output of KVMN is fed into the decoder to predict output labels.

context features associated with x_n and their corresponding syntactic knowledge instances. Figure 5.2 shows the dependency features and their corresponding syntactic knowledge instances¹ for the sentence “*Dihydropyrimidine dehydrogenase deficiency is an autosomal recessive disease*”.² This figure focuses on the word “*deficiency*” (in boldface) with its highlighted context features and their corresponding syntactic information.

¹We combine the word and its syntactic information to increase the diversity of the syntactic information instance.

²The parse tree is obtained from running the online demo of Stanford CoreNLP Toolkits (Manning et al., 2014a) at <https://corenlp.run/>.

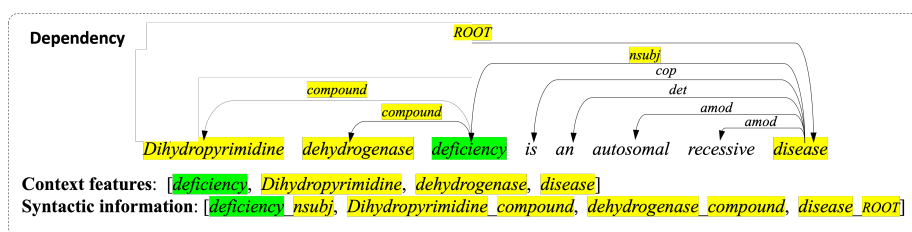


Figure 5.2: Three types of syntactic information extracted for an example “*Dihydropyrimidine dehydrogenase deficiency is an autosomal recessive disease*” in the biomedical domain. The context features and their corresponding POS labels, syntactic constituents, and dependency relations for x_5 = “*deficiency*” are highlighted in part (a), (b), and (c) respectively.

Specifically, we first collect the dependency knowledge through the approach in Section 3.1.2. Then, for each token, we regard its dependents, its governor, and the token itself, as the context features and regard the combination of these tokens and their dependency types as the syntactic instances. In figure 5.2, for the given word “*deficiency*”, it has two dependents (i.e., “*dihydropyrimidine*” and “*dehydrogenase*”) and one governor (i.e., “*disease*”, which is the root of the sentence). According to these dependency relations, the context features of “*deficiency*” are [*deficiency, dihydropyrimidine, dehydrogenase, a, metabolic*] and the syntactic information instances are [*deficiency_nsubj, dihydropyrimidine_compound, dehydrogenase_compound, disease_ROOT*].

Through the process, we build the context feature list \mathcal{K} and the syntactic instance list \mathcal{V} upon the extraction results. For each token x_n in the word sequence \mathcal{X} , in both training and predicting process, associated context features and syntactic information instances in \mathcal{K} and \mathcal{V} are activated and computed. We denote the context features and the syntactic information instances for x_n as $\mathcal{K}_n = [k_{n,1}, \dots, k_{n,m}, \dots, k_{n,M}]^3$ and $\mathcal{V}_n = [v_{n,1}, \dots, v_{n,m}, \dots, v_{n,M}]$, respectively. In the following sections, we illustrate our method to leverage the keys and values through KVMN.

The Memory Module

In our approach to bioNER, we adapt the KVMN to a sequence labeling paradigm by applying it to each word x_n in the input. Therefore, for x_n , its hidden vector \mathbf{h}_n obtained from an encoder serves

³Herein, the value of M is relevant to n . We use M for better presentation.

as the counterpart of input representation \mathbf{x} in Equation (3.10); its associated context features and the corresponding syntactic information instances stand for the keys k_m and values v_m , respectively. In details, the memory module takes \mathbf{h}_n for each x_n , activates the keys to address their embeddings and computes the probability weights for them by

$$p_{n,m} = \frac{\exp(\mathbf{h}_n \cdot \mathbf{k}_{n,m})}{\sum_{m=1}^M \exp(\mathbf{h}_n \cdot \mathbf{k}_{n,m})} \quad (5.1)$$

where $\mathbf{k}_{n,m}$ is the embedding vector of $k_{n,m}$. Afterwards, we use the resulting probabilities on syntactic information instances in \mathcal{V}_n to get the weighted value embedding \mathbf{o}_n :

$$\mathbf{o}_n = \sum_{m=1}^M p_{n,m} \mathbf{v}_{n,m} \quad (5.2)$$

where $\mathbf{v}_{n,m}$ is the embedding vector of the value $v_{n,m}$.

Tagging with KVMN

To facilitate the process of leveraging syntactic information through KVMN, we firstly use an encoder to obtain the hidden vector \mathbf{h}_n for each x_n and feed the hidden vectors for all tokens to the KVMN module. Once \mathbf{o}_n is obtained for each x_n from the KVMN module, we concatenate⁴ it with \mathbf{h}_n and apply a trainable matrix \mathbf{W} to the resulting vector to align its dimension to the output space, which is formalized by

$$\mathbf{u}_n = \mathbf{W} \cdot (\mathbf{h}_n \oplus \mathbf{o}_n) \quad (5.3)$$

where the softmax decoder is applied to the resulting \mathbf{u}_n to predict the BioNER label for x_n .

5.1.2 Experiment Settings

Datasets

We evaluate our methods on six English benchmark datasets that are widely used in previous studies (Leaman and Lu, 2016; Habibi et al., 2017; Giorgi and Bader, 2018; Yoon et al., 2019; Lee et al., 2019). These datasets focus on four different biomedical entity types: BC2GM dataset (Smith et al., 2008) and JNLPBA dataset (Kim et al., 2004) for gene/protein NER, BC5CDR-chemical dataset

⁴We use concatenation instead of element-wise summation in KVMN, which does not change the nature of using KVMN.

Entity Type	Datasets		Token #	Sent. #	Entity #
Gene/protein	BC2GM	Train	355.4k	12.5k	15.1k
		Dev	71.0k	2.5k	3.0k
		Test	143.4k	5.0k	6.3k
	JNLPBA	Train	443.6k	14.6k	32.1k
		Dev	117.2k	3.8k	8.5k
		Test	114.7k	3.8k	6.2k
Chemical	BC5CDR-chemical	Train	118.1K	4.5K	5.2K
		Dev	117.4K	4.5K	5.3K
		Test	124.7K	4.7K	5.3K
Disease	NCBI-disease	Train	135.7K	5.4K	5.1K
		Dev	23.9K	923	787
		Test	24.4K	940	960
Species	LINNAEUS	Train	281.2k	11.9k	2.1k
		Dev	93.8k	4.0k	711
		Test	165k	7.1k	1.4k
	Species-800	Train	147.2K	5.7K	2.5K
		Dev	22.2K	830	384
		Test	42.2K	1.6K	767

Table 5.1: The statistics of the four benchmark datasets. “Token #”, “Sent. #” and “Entity #” represent the number of tokens, sentences, and entities.

(Wei et al., 2015) for chemical NER, NCBI-disease dataset (Doğan et al., 2014) for disease NER, and LINNAEUS dataset (Gerner et al., 2010) and Species-800 dataset (Pafilis et al., 2013) for species NER.

BC2GM is a dataset used for the BioCreative II gene mention tagging task⁵. It contains 20,000 sentences from abstracts of biomedical publications and is annotated with 24,583 mentions of the names of genes, proteins and related entities. It has become the major benchmark for the NER of gene/proteins entity type (Habibi et al., 2017; Wang et al., 2018; Sachan et al., 2018; Giorgi and Bader, 2018; Yoon et al., 2019; Lee et al., 2019).

JNPBA is the dataset for the Joint Workshop on NLP in Biomedicine and its Application Shared task⁶. It was organized by the GENIA Project based on the annotations of the GENIA Term corpus and consists of 2,404 publication abstracts. It is widely used for evaluating multi-class biomedical entity taggers.

BC5CDR is a dataset used for the BioCreative V Chemical Disease Relation (CDR) Task⁷. It contains 1,500 titles and abstracts from PubMed⁸, where chemical and disease mentions are annotated by human annotators. Following previous studies (Luo et al., 2018), we only use the subset with chemical entities and denote it as BC5CDR-chemical.

NCBI-disease contains 793 PubMed abstracts that are annotated with disease mentions and their corresponding concepts. There are 6,892 disease mentions from 790 unique disease concepts in this dataset and 91% of the mentions are mapped to a single disease concept. It has been a widely used research resource for the disease NER.

The LINNAEUS dataset was created specifically for species named entity recognition and consists of 100 full-text documents. In the LINNAEUS dataset, all mentions of species terms were manually annotated and normalized to the NCBI taxonomy IDs of the intended species.

Species-800 is a novel benchmark corpus for species entities, which is based on manually annotated abstracts. It comprises 800 PubMed abstracts that contain identified organism mentions. Because the abstracts are select from journals on 8 different categories, the diversity of Species-800 is high and thus it is more challenging for NER systems.

We follow the study of Lee et al. (2019) to pre-process all datasets. In details, BC2GM, BC5CDR-

⁵<https://biocreative.bioinformatics.udel.edu/tasks/biocreative-ii/task-1a-gene-mention-tagging/>

⁶<http://www.geniaproject.org/shared-tasks/bionlp-jnlpba-shared-task-2004>

⁷<https://biocreative.bioinformatics.udel.edu/tasks/biocreative-v/track-3-cdr/>

⁸<https://pubmed.ncbi.nlm.nih.gov/>

chemical, LINNAUES, and NCBI-disease datasets are pre-processed based on the schema of Wang et al. (2018); JNPBA is pre-processed by CoNLL format⁹; and Species-800 is pre-processed by Pyysalo¹⁰. The statistics of all datasets in terms of the number of tokens, sentences, and entities are reported in Table 5.1.

Implementation Details

In the experiments, we use off-the-shelf NLP toolkits to generate syntactic information, following the common practice in previous studies such as Mohit and Hwa (2005), Tkachenko and Simanovsky (2012), and Luo et al. (2018). In our study, we focus on three types of syntactic information: POS labels, syntactic constituents, and dependency relations. We use Stanford CoreNLP Toolkits (SCT)¹¹ (Manning et al., 2014a), which is a well-known NLP toolkit used in many studies (Scozzafava et al., 2020; Tian et al., 2020b), to obtain the POS tagging, constituency, and dependency parsing results of a given input sentence.

For the encoder, considering that BERT (Devlin et al., 2019) and its variants (Beltagy et al., 2019; Lee et al., 2019; Diao et al., 2020; Antoun et al., 2020) achieve state-of-the-art performance on many NLP tasks, we use the variant for the medical domain, i.e., BioBERT (Lee et al., 2019), in our method. Specifically, we use both the base and large version of BioBERT¹² and follow the hyper-parameters used by Lee et al. (2019) (i.e., for BioBERT-Base, there are 12 self-attention heads with 768-dimensional hidden vectors; for BioBERT-Large, the number of heads is 24 with 1024-dimensional hidden vectors). All parameters in the encoder are fine-tuned in training. For the KVMN module, the embeddings of all keys and values are randomly initialized, with their dimension matching the dimension of hidden vectors in the BioBERT encoder. Besides, we follow the setting of Lee et al. (2019) to run the training process, where we do not use the “alternate” annotations for the BC2GM dataset. Moreover, for each method, we train five models with different random seeds to initialize the model parameters and use the average of their micro F1 scores for evaluation.¹³ In the

⁹<https://github.com/spyysalo/standoff2conll>

¹⁰<https://github.com/spyysalo/s800>

¹¹We use v3.9.2, downloaded from <https://stanfordnlp.github.io/CoreNLP/>.

¹²We obtain the pre-trained models v1.1 from <https://github.com/naver/biobert-pretrained>

¹³We evaluate all models by the widely used *seqeval* framework at <https://github.com/chakki-works/>

experiments, we train each model for 150 epochs for the BC2GM dataset and for 60 epochs for all other datasets.¹⁴ In each run, we evaluate the model on the development set after each epoch to find its best performing result.

5.1.3 Results and Analysis

Overall Results

We run the baseline methods without using syntactic information and our methods with KVMN (\mathcal{M}) to incorporate three types of syntactic information obtained from SCT on six aforementioned datasets, where two different encoders, i.e., BioBERT-Base and BioBERT-Large, are used. For reference, we also run baseline methods that use direct concatenation (DC) to incorporate such syntactic information, where the embeddings of context features and syntactic information are directly concatenated with the output of the BioBERT encoder. We report the experimental results (the average F1 scores of the five runs for each method as well as the standard deviations σ) in Table 5.2. There are some observations.

First, comparing with the baseline methods without using any syntactic information, our method with KVMN can work well with both BioBERT-Base and BioBERT-Large encoder, where decent improvements over the baseline methods are observed among all datasets. Second, compared with DC, our methods with KVMN to incorporate syntactic information achieve better results in most cases. Besides, in some cases where DC is applied, the syntactic information causes inferior results than baselines. One possible explanation could be: there are some noisy syntactic results extracted by off-the-shelf toolkits, which may influence the performance of the model and lead to worse results compared to the baselines only using BioBERT. Under this condition, methods with DC fail to distinguish the salient syntactic information that contributes more to the bioNER task in a specific context. On the contrary, KVMN can weigh such syntactic information according to the importance of the context features and thus, to some extent, avoid the errors caused by incorporating auto-processed syntactic information.

We compare the results of our best performing model with previous studies on all aforementioned

seqeval.

¹⁴All experiments are run on a single Nvidia Tesla V100 GPU with 16G RAM.

Methods	BC2GM		JNLPBA		BC5CDR-chemical		NCBI-disease		LINNAEUS		Species-800	
	F1	σ	F1	σ	F1	σ	F1	σ	F1	σ	F1	σ
Base	84.61	0.21	76.85	0.31	93.50	0.10	88.63	0.71	88.27	0.32	74.97	0.46
+ DR (DC)	84.33	0.30	77.01	0.28	93.66	0.15	89.05	0.23	88.43	0.19	75.12	0.52
+ DR (\mathcal{M})	84.65	0.27	77.32	0.35	93.78	0.18	89.24	0.60	88.57	0.15	75.81	0.71
Large	84.89	0.17	77.29	0.19	93.90	0.31	88.65	0.59	88.87	0.65	74.98	0.59
+ DR (DC)	85.01	0.12	77.58	0.10	93.97	0.17	89.37	0.30	88.99	0.22	75.01	0.83
+ DR (\mathcal{M})	85.17	0.10	77.73	0.11	94.05	0.10	88.81	0.51	89.04	0.27	75.17	0.91

Table 5.2: The experimental results are reported in terms of average F1 scores (F1) and the standard deviation σ . The methods in the group “Base” and “Large” refer to baselines with BioBERT-Base and BioBERT-Large encoder and our methods with KVMN (\mathcal{M}). “DC” refers to the baseline method using direct concatenation to incorporate syntactic information. “DR” stands for the dependency relations, respectively.

datasets. The results (F1 scores) are summarized in Table 5.3, where our method outperforms the previous study (i.e., Lee et al. (2019)) using the base and large version of BioBERT on all datasets. Specifically, for the BioBERT-Base, the improvement of F1 scores on BC2GM, JNLPBA, BC5CDR-chemical, NCBI-disease, LINNAEUS, and Species-800 are 0.20%, 0.23%, 0.53%, 0.37%, 0.55%, and 0.90% respectively; for the BioBERT-Large, the improvement on BC2GM and NCBI-disease are 0.28% and 0.84%, respectively. These results demonstrate the effectiveness of our method to leverage auto-processed syntactic information in recognizing different types of named entities in the biomedical domain. In addition, our method achieves state-of-the-art performance on four datasets, i.e., BC2GM, BC5CDR-chemical, NCBI-disease, and Species-800. Compared with (Yoon et al., 2019) and (Habibi et al., 2017), we do not outperform their results on JNLPBA and LINNAEUS, because the gaps between their results and our baseline method, i.e., BioBERT from Lee et al. (2019), are big on these datasets, which could be hard to compensate for by using syntactic information.

Methods	BC2GM	JNLPBA	BC5CDR- chemical	NCBI- disease	LINNAEUS	Species- 800
biLSTM + pre-trained embeddings (Habibi et al., 2017)	78.57	77.25	91.05	84.64	94.13	73.11
biLSTM + attentions (Luo et al., 2018)	-	-	92.57	-	-	-
biLSTM + multi-task learning (Wang et al., 2018)	80.74	73.52	-	86.14	-	-
biLSTM + pre-training (Sachan et al., 2018)	81.69	75.03	-	87.34	-	-
biLSTM + transfer learning (Giorgi and Bader, 2018)	78.66	-	91.64	84.72	93.54	74.98
biLSTM + model ensemble (Yoon et al., 2019)	79.73	78.58	93.31	86.36	-	-
SciBERT (Beltagy et al., 2019)	-	77.28	-	88.57	-	-
BERT (Lee et al., 2019)	81.79	74.94	91.16	85.63	87.60	71.63
BioBERT (Base) (Lee et al., 2019)	84.72	77.49	93.47	89.71	88.24	75.31
BioBERT (Large) (Lee et al., 2019)	85.01	-	-	88.79	-	-
BioBERT (Base) + DR (\mathcal{M})	84.92	77.72	94.00	90.08	88.79	76.21
BioBERT (Large) + DR (\mathcal{M})	85.29	77.83	94.22	89.63	89.24	76.33

Table 5.3: The result (F1 scores) of our method on each dataset comes from the best performing model. The results for the base and large version of BioBERT are from their paper and GitHub repository (We report the results of their version 1.1, which is identical to the BioBERT version used in our experiments).

Except for the two datasets, our method outperforms their methods on all other datasets.

Effect of Different Toolkits

To explore the effect of using syntactic information from different NLP toolkits, in addition to SCT, we try another toolkit, i.e., spaCy¹⁵, to obtain the auto-processed syntactic information. We report the results (the average F1 scores and the standard deviation σ) of our methods with KVMN on the BC5CDR-chemical dataset in Table 5.4. For reference, the results of our method using SCT as well as the baseline results are also reported. From the results, we can find that, for both base and large BioBERT encoders, our method can leverage the syntactic information from different NLP toolkits and thus achieves better performance comparing with the baseline methods.

¹⁵<https://spacy.io/>

	BioBERT-Base		BioBERT-Large	
	F1	σ	F1	σ
Baseline	93.50	0.10	93.90	0.31
Stanford CoreNLP Toolkits	93.78	0.18	94.05	0.10
spaCy	93.71	0.12	93.97	0.13

Table 5.4: Results of using different NLP toolkits on the BC5CDR-chemical dataset. The experimental results (the average F1 scores and the standard deviation (σ)) of our method with KVMN (\mathcal{M}) using different NLP toolkits (i.e., Stanford CoreNLP Toolkits and spaCy) to obtain dependency relations (DR). The results of baseline models without using any syntactic information are also reported for reference.

Case Study

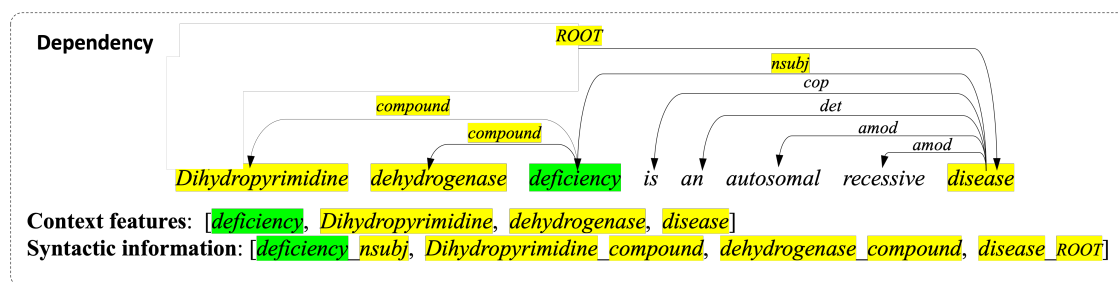


Figure 5.3: The figure presents an example of syntactic information and the context features for “*dystrophy*”. The weights for syntactic information learned from the memories are highlighted with the darker color referring to greater value.

To better understand how our method improves BioNER, in Figure 5.3, we show a sentence and illustrate the way of dependency relations to improve bioNER. We visualize the weights assigned to dependency relations for the word “*dystrophy*” in another example sentence. In this case, dependency information successfully finds the dependents, i.e., “*Myotonic*”, “*DM*”, and “*disorder*”, of “*dystro-*

phy”, which could suggest useful cues to predict the NE labels. Among those dependents, KVMN distinguishes that the dependent “*discorder*” with an “*appos*” dependency relation (appositional modifier) strongly suggests “*dystrophy*” is a disease entity. Therefore, KVMN assigns the highest weight to the dependency relation offered by “*disorder*”. Similarly, another modifier (i.e., “*Myotonic*”) of “*dystrophy*” is also distinguished and weighed by the KVMN, and the second-highest weight is assigned to it accordingly. It is worth noting that the dependency information that contributes most to recognizing “*dystrophy*” as a part of an NE comes from a word (“*disorder*”) in the long-distance; dependency information is able to capture that information and helps our method predict the NE tag for the word “*dystrophy*”.

5.2 Relation Extraction

Recent advances in relation extraction (RE) primarily rely on neural approaches utilizing powerful encoders, such as CNNs, RNNs, and Transformers, which effectively model contextual information without manual feature engineering (Zeng et al., 2014; Zhang and Wang, 2015; Zhou et al., 2016b). To further enhance performance, many studies incorporate syntactic knowledge, particularly dependency trees, which provide meaningful long-distance connections between entities and relevant context words (Miwa and Bansal, 2016; Zhang et al., 2018; Chen et al., 2021). However, automatically generated dependency trees often introduce noise, potentially confusing the model and harming performance. Conventional approaches alleviate this issue by employing pruning strategies to filter noisy dependencies before encoding structures with neural architectures such as graph convolutional networks (GCN) (Xu et al., 2015b; Zhang et al., 2018). However, fixed pruning approaches cannot ensure the retention of all critical contextual information nor completely eliminate noise. Additionally, current graph-based approaches typically assign equal importance to all dependency edges, neglecting valuable dependency labels (e.g., nominal subjects, objects) that explicitly convey syntactic relations and potentially important semantic cues. Thus, an effective approach that dynamically identifies and leverages essential contextual information within dependency structures without relying on heuristic pruning and fully exploiting informative dependency labels is required.

In this thesis, we propose a type-aware graph convolutional network (T-GCN) with an attention mechanism for RE, improving the model performance by dynamically modeling dependency trees

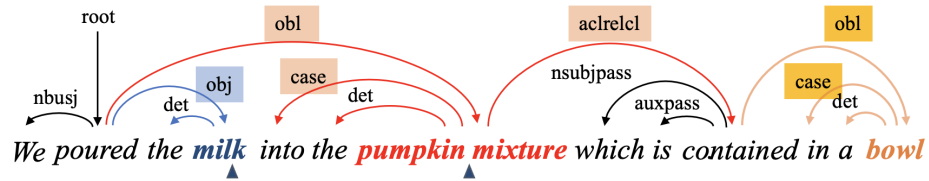


Figure 5.4: An illustration of noises in the dependency tree that may hurt relation extraction, where the word dependency connected in between “*pumpkin mixture*” and “*bowl*” (whose relation is *content-container*) may introduce confusion to the system when the object is to predict the relation between “*milk*” and “*pumpkin mixture*” (whose relation is *entity-destination*).

and dependency types. Specifically, our T-GCN assigns attention weights to dependency connections, differentiating important contextual information from noisy edges within dependency trees generated by off-the-shelf parsers. Furthermore, dependency label types are integrated into the attention computation, explicitly utilizing syntactic relations between connected words to better inform relation extraction decisions. Experiment results on benchmark datasets demonstrate the effectiveness of the approach to incorporate the dependency information for RE, which outperforms strong baselines and existing studies.

5.2.1 The Approach

RE is conventionally performed as a typical classification task. Our approach follows this paradigm by using T-GCN and incorporates dependency information to improve model performance, where the overall architecture of our model is illustrated in Figure 5.5. Specifically, given an unstructured input sentence $\mathcal{X} = x_1, \dots, x_N$ with N words and let E_1 and E_2 denote two entities in \mathcal{X} , our approach f predicts the relation \hat{y} between E_1 and E_2 by

$$\hat{y} = f(\mathcal{X}, E_1, E_2) \quad (5.4)$$

The following texts firstly illustrate our proposed T-GCN equipped with dependency type information and then illustrate the process of applying T-GCN to the classification paradigm for RE.

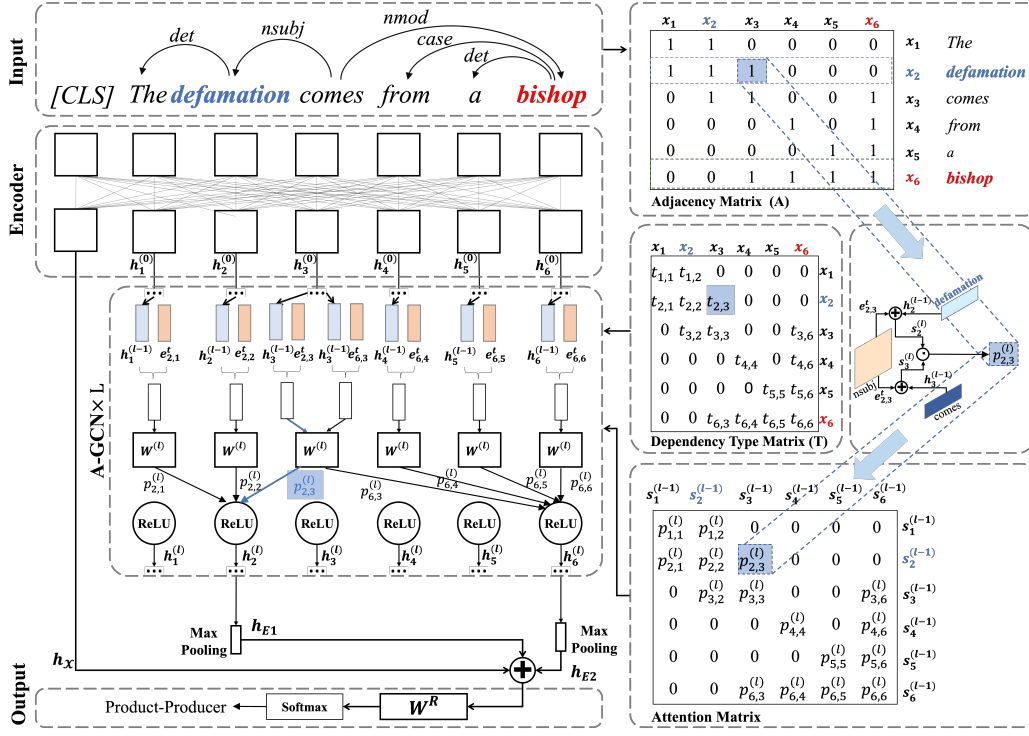


Figure 5.5: The overall architecture of the proposed T-GCN for RE illustrated with an example input sentence (the two entities “defamation” and “bishop” are highlighted in blue and red colors, respectively) and its dependency tree. The left part shows our T-GCN model, where the attention weights are applied to different connections to model the dependency type-aware contextual information. The right part illustrates the adjacency matrix \mathbf{A} for the dependency graph and the process to compute the attention weights (i.e., $p_{i,j}^{(l)}$) for different connections.

T-GCN

In standard GCN illustrated in Section 3.3.4, the connections among tokens are treated equally (i.e., $\tilde{a}_{n',n''}$ is either 0 or 1). Therefore, standard GCN for RE are not able to distinguish the importance of different connections and thus pruning on them is of great importance for RE. Therefore, we propose T-GCN for this task, which uses an attention mechanism to compute the weights for different connections so that the model is able to leverage different dependency connections accordingly. In addition, the standard GCN and most previous studies omit the dependency types associated with the dependency connections, where those types contain highly useful information for RE and are

introduced into T-GCN in this work. Specifically, we map the dependency type $r_{n',n''}$ in the relation type matrix¹⁶ \mathbf{R} to its embedding $\mathbf{r}_{n',n''}$. Then, at the l -th T-GCN layer, the weight for the connection between $x_{n'}$ and $x_{n''}$ is computed by

$$w_{n',n''}^{(l)} = \frac{\tilde{a}_{n',n''} \cdot \exp\left(\mathbf{s}_{n',n''}^{(l)} \cdot \mathbf{s}_{n'',n'}^{(l)}\right)}{\sum_{n''=1}^N \tilde{a}_{n',n''} \cdot \exp\left(\mathbf{s}_{n',n''}^{(l)} \cdot \mathbf{s}_{n'',n'}^{(l)}\right)} \quad (5.5)$$

where $\mathbf{s}_{n'}^{(l)}$ and $\mathbf{s}_{n''}^{(l)}$ are two intermediate vectors for $x_{n'}$ and $x_{n''}$, respectively, which are computed by

$$\mathbf{s}_{n',n''}^{(l)} = \mathbf{h}_{n'}^{(l-1)} \oplus \mathbf{r}_{n',n''} \quad (5.6)$$

and

$$\mathbf{s}_{n'',n'}^{(l)} = \mathbf{h}_{n''}^{(l-1)} \oplus \mathbf{r}_{n'',n'} \quad (5.7)$$

Afterwards, we apply the weight $w_{n',n''}^{(l)}$ to the associated dependency connection between $x_{n'}$ and $x_{n''}$ and obtain the output representation of $x_{n'}$ by

$$\mathbf{h}_{n'}^{(l)} = \sigma \left(\sum_{n''=1}^N w_{n',n''}^{(l)} \left(\mathbf{W}^{(l)} \cdot \tilde{\mathbf{h}}_{n',n''}^{(l-1)} + \mathbf{b}^{(l)} \right) \right) \quad (5.8)$$

with σ , $\mathbf{W}^{(l)}$, and $\mathbf{b}^{(l)}$ following the same notations in Eq. (3.12) for standard GCN, and $\tilde{\mathbf{h}}_{n',n''}^{(l-1)}$ (a type-enhanced representation for $x_{n''}$) computed by

$$\tilde{\mathbf{h}}_{n',n''}^{(l-1)} = \mathbf{h}_{n''}^{(l-1)} + \mathbf{W}_T^{(l)} \cdot \mathbf{r}_{n',n''} \quad (5.9)$$

where $\mathbf{W}_T^{(l)}$ maps the dependency type embedding $\mathbf{r}_{n',n''}$ to the same dimension as $\mathbf{h}_{n''}^{(l-1)}$.

Compared with standard GCN (i.e., Eq. (3.12)), our approach uses numerical weighting (i.e., $w_{n',n''}^{(l)} \in [0, 1]$) rather than a binary choice for $\tilde{a}_{n',n''}$, to distinguish the importance of different connections so as to leverage them accordingly. In addition, compared to the T-GCN proposed in Section 4.3.1, although we utilize a similar idea by combining the GCN with attentions, we integrate the dependency type information into both the computed weight (i.e., $p_{n',n''}^{(l)}$) and the output representation of x_i (i.e., $\mathbf{h}_{n'}^{(l)}$).

¹⁶See Section 3.1.2 for the process to obtain the matrix.

		ACE05	SemEval
# Instances	Train	48,198	8,000
	Dev	11,854	-
	Test	10,097	2,717

Table 5.5: The number of unique instances (i.e., entity pairs) of ACE05 and SemEval benchmark datasets.

Relation Extraction with T-GCN

Before applying T-GCN for RE, we firstly encode the input \mathcal{X} into hidden vectors by BERT (Devlin et al., 2019) with $\mathbf{h}_{n'}^{(0)}$ denoting the hidden vector for $x_{n'}$, where the hidden vector (denoted as $\mathbf{h}_{\mathcal{X}}$) for the special sentence-initial token “[CLS]” is used as the representation for the entire sentence. Next, we feed $\mathbf{h}_{n'}^{(0)}$ to our proposed T-GCN model with L layers and obtain the corresponding output $\mathbf{h}_{n'}^{(L)}$. Then, we apply the max pooling mechanism to the output hidden vectors of the tokens that belongs to an entity mention (i.e., $E_i, i = 1, 2$) to compute the representation for the entity (denoted as \mathbf{h}_{E_i}) by

$$\mathbf{h}_{E_i} = \text{MaxPooling}(\{\mathbf{h}_{n'}^{(L)} | x_{n'} \in E_i\}) \quad (5.10)$$

Afterwards, we concatenate the representations of the sentence (i.e., $\mathbf{h}_{\mathcal{X}}$) and two entities (i.e., \mathbf{h}_{E_1} and \mathbf{h}_{E_2}) and apply a trainable matrix \mathbf{W}_R to the computed vector to map it to the output space by

$$\mathbf{o} = \mathbf{W}_R \cdot (\mathbf{h}_{\mathcal{X}} \oplus \mathbf{h}_{E_1} \oplus \mathbf{h}_{E_2}) \quad (5.11)$$

where \mathbf{o} is fed into a *softmax* decoder to predict the relation \hat{y} between E_1 and E_2 .

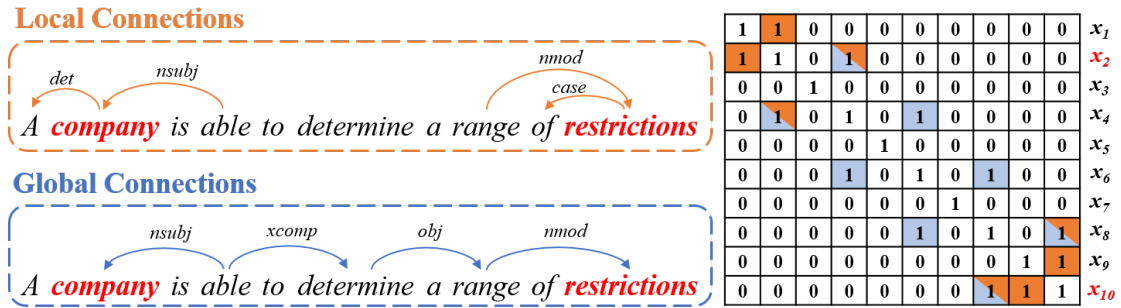


Figure 5.6: An illustration on the two (i.e., local and global) groups of dependency connections for an example sentence (entities are highlighted in red color) with an adjacency matrix (on the right) built upon all connections from the two groups. Local and global connections are represented in orange and blue colors, respectively,

5.2.2 Experiment Settings

Datasets

In the experiments, we use two English benchmark datasets for RE, namely, ACE2005EN (ACE05)¹⁷ and SemEval 2010 Task 8 (SemEval)¹⁸ (Hendrickx et al., 2010). For ACE05, we use its English section and follow previous studies (Miwa and Bansal, 2016; Christopoulou et al., 2018; Ye et al., 2019) to pre-process it (two small subsets *cts* and *un* are removed) and split the documents into training, development, and test sets¹⁹. For SemEval, we use its official train/test split²⁰. The numbers of unique relation types in ACE05 and SemEval are 7 and 19, respectively. We report the number of instances (i.e., entity pairs), for train/dev/test sets of ACE05 and SemEval benchmark datasets in Table 5.5.

Dependency Graph Construction

To construct graphs for T-GCN, we use Standard CoreNLP Toolkits (SCT)²¹ to obtain the dependency tree for each input sentence. Although our approach is able to distinguish the importance of different dependency connections through the attention mechanism, it is still beneficial if we can filter out those dependency connections that bring confusion to RE through particular pruning strategies. Motivated by previous studies (Xu et al., 2015b; Zhang et al., 2018; Yu et al., 2020), in this thesis, we construct the graph for T-GCN by including two groups of dependency connections, namely, the *local connections* and the *global connections*. In detail, local connections include all dependencies that directly connect to the heads of two entities and global connections include all dependencies along the shortest dependency path (SDP) between the heads of two entities, where, in many cases, words that are not directly connected to the two entities are also involved. With an example sentence including two entities (i.e., “*company*” and *benchmarking*), Figure 5.6 illustrates the two groups of dependency connections and the resulted adjacency matrix, which is built with the connections from the two groups²². It is worth noting that, when the SDP is short, there might be more connections in the local group than that in the global one.

Implementation Details

Following Soares et al. (2019), we insert four special tokens (i.e., “ $\langle e1 \rangle$ ”, “ $\langle /e1 \rangle$ ”, “ $\langle e2 \rangle$ ”, and “ $\langle /e2 \rangle$ ”) into the input sentence to mark the boundary²³ of the two entities to be investigated, which allows the encoder to distinguish the position of entities during encoding and thus improves model performance. For the encoder, we try the uncased version of BERT-base and BERT-large²⁴ following

¹⁷We obtain the official data (LDC2006T06) from <https://catalog.ldc.upenn.edu/LDC2006T06>.

¹⁸The data is downloaded from http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw.

¹⁹We follow the train/dev/test splits specified by Miwa and Bansal (2016) at <https://github.com/tticoin/LSTM-ER/tree/master/data/ace2005/split>

²⁰SemEval only includes the training and test sets.

²¹We download the version 3.9.2 from <https://stanfordnlp.github.io/CoreNLP/>.

²²We do not distinguish the two groups of connections in T-GCN once they are represented by the adjacency matrix.

²³For example, “ $\langle e1 \rangle$ ” and “ $\langle /e1 \rangle$ ” are respectively inserted right before and after the entity E_1 in the input \mathcal{X} .

²⁴We download different BERT models from <https://github.com/huggingface/transformers>.

ID	Models	ACE05	SemEval	ID	Models	ACE05	SemEval
1	BERT-base	75.31	87.87	1	BERT-large	76.79	89.02
2	+ GAT (Full)	76.16	88.39	2	+ GAT (Full)	78.25	89.39
3	+ GAT (L + G)	75.79	88.53	3	+ GAT (L + G)	78.71	89.44
4	+ 1 GCN layer (Full)	74.91	87.58	4	+ 1 GCN layer (Full)	77.63	88.98
5	+ 1 T-GCN layer (Full)	76.63	88.34	5	+ 1 T-GCN layer (Full)	78.53	89.54
6	+ 1 GCN layer (L + G)	75.51	88.64	6	+ 1 GCN layer (L + G)	77.49	89.11
7	+ 1 T-GCN layer (L + G)	77.10	89.03	7	+ 1 T-GCN layer (L + G)	78.48	89.69
8	+ 2 GCN layers (Full)	75.09	88.66	8	+ 2 GCN layers (Full)	78.67	89.43
9	+ 2 T-GCN layers (Full)	77.25	88.70	9	+ 2 T-GCN layers (Full)	78.91	89.70
10	+ 2 GCN layers (L + G)	76.11	88.62	10	+ 2 GCN layers (L + G)	78.82	89.42
11	+ 2 T-GCN layers (L + G)	77.30	89.16	11	+ 2 T-GCN layers (L + G)	79.05	89.85
12	+ 3 GCN layers (Full)	75.69	88.54	12	+ 3 GCN layers (Full)	78.08	89.62
13	+ 3 T-GCN layers (Full)	76.26	88.63	13	+ 3 T-GCN layers (Full)	78.45	89.46
14	+ 3 GCN layers (L + G)	76.85	88.33	14	+ 3 GCN layers (L + G)	78.64	89.19
15	+ 3 T-GCN layers (L + G)	76.36	88.70	15	+ 3 T-GCN layers (L + G)	78.83	89.56

(a) BERT-base

(b) BERT-large

Table 5.6: F1 scores of our T-GCN models and the baselines (i.e., BERT-only, standard GAT, and standard GCN) under different settings with BERT-base (a) and BERT-large (b) used. All graph-based models (i.e., GAT, GCN, and T-GCN) are tested with two settings: the first is using the full graph (Full) with all dependency connections involved and the second is using the combination of local and global connections (L + G). We also run GCN and T-GCN with different numbers of layers (i.e., 1 to 3 layers) for fair comparisons.

the default settings. For T-GCN, we randomly initialize all trainable parameters and the dependency type embeddings. For evaluation, we follow previous studies to use the standard micro-F1 scores²⁵ for ACE05 and use the macro-averaged F1 scores²⁶ for SemEval. In our experiments, we try different combinations of hyper-parameters, and tune them on the dev set, then evaluate on the test set by the model that achieves the highest F1 score on the dev set.

²⁵We use the evaluation script from *sklearn* framework.

²⁶We use the official evaluation script downloaded from http://semeval2.fbk.eu/scorers/task08/SemEval2010_task8_scorer-v1.2.zip.

5.2.3 Results and Analyses

Overall Results

In the experiments, we run our T-GCN models using BERT-base and BERT-large encoder on graphs with and without applying dependency pruning strategies, which correspond to the graph built upon the combined local and global connections (“L + G”), as well as the one constructed by the full dependency graph (“Full”), respectively. We also run baselines with standard GCN and standard graph attentive networks (GAT) (Veličković et al., 2017) with the same graph. For both standard GCN and T-GCN, we try different numbers of layers (i.e. 1 to 3 layers). In addition, we try BERT-base and BERT-large baselines without using any dependency information. Table 5.6 shows the F1 scores of our T-GCN models and all the aforementioned baselines on the test set of ACE05 and SemEval.²⁷

There are several observations. First, T-GCN functions well when using BERT-base or BERT-large as encoder, where the consistent improvement is observed over the BERT-only baselines (ID: 1) across two benchmark datasets, even though the BERT baselines have already achieved good performance. Second, for both datasets, T-GCN outperforms GAT (ID: 2, 3) and standard GCN baselines (ID: 4, 6, 8, 10, 12, 14) with the same graph (i.e., either “L + G” or “Full”) and equal number of layers. Particularly, when full dependency graph is used, it is noted that, in some cases (e.g., ID: 8 for BERT-base on ACE05), standard GCN obtains very limited improvements (or even worse results) over the BERT-only baseline (ID: 1), whereas our T-GCN models (e.g., ID: 9 for BERT-base) is able to consistently outperform the BERT-only baseline and achieve higher performance. We attribute this observation to the attention mechanism used to weigh different dependency connections, which allows T-GCN to distinguish the noise in the graph and thus leverage useful dependency information accordingly. Third, among the models with different numbers of T-GCN layers, the ones (e.g., ID: 11 for BERT-base and ID: 11 for BERT-large) with two T-GCN layers achieves the highest scores, where similar trend is observed from the standard GCN baselines. Besides, we find that our T-GCN models (as well as the standard GCN baselines) with the local and global connections (i.e., “L + G”) consistently outperform the ones with full dependency graph (i.e., “Full”). These observations are relatively intuitive since the dependency information may introduce more noise to RE when

²⁷For the same group of models, we report the F1 scores on the development sets in Appendix C and the mean and standard deviation of their test set results in Appendix D.

it is leveraged in an intensive way (e.g., by using more layers or the full dependency tree without pruning).

In addition, we compare our best models (with “L + G” or “Full” graphs) using BERT-large encoder and two T-GCN layers (ID: 9 and 11) with previous studies. The test results (F1 scores) are reported in Table 5.7, where our model with both local and global connections (i.e., “L + G”) outperforms all previous studies and achieves state-of-the-art performance on the two benchmark datasets. Specifically, compared with Guo et al. (2019) who proposed an graph-based approach with attentions to leverage dependency connections, our approach leverages both dependency connections and dependency types among all input words and thus provides a better way to comprehensively leverage the dependency information. In addition, although Mandya et al. (2020) proposed an approach to leverage both dependency connections and dependency types through attentions, they added the dependency type directly to the input word embeddings along with POS embeddings, and the attention in their approach is a separate stand-alone module which is added on the top of the GCN layer. On the contrary, in our approach, the dependency type is added to each T-GCN layer and the attention mechanism is directly applied to each dependency connection in the T-GCN layer. Therefore, compared with Mandya et al. (2020), our T-GCN encodes the dependency connections and dependency types in a more intensive manner and thus can better leverage them to guide the process of predicting the relations between the given entities.

The Effect of T-GCN

Dependency information is supposed to be beneficial for RE because it contains long-distance word-word relations, which could be extremely useful when the given two entities are far away from each other in the input sentence. To explore the effect of T-GCN in capturing such long-distance word-word relations to help with RE, we split the test instances into different groups according to their entities’ distances (i.e., the number of words between the two entities) and run models on these groups to test their performance. Figure 5.7 shows the performance of our best performing T-GCN model with BERT-large (ID: 11 in Table 5.6) and its corresponding standard GCN and BERT-large baselines on the three groups of test instances from the test set of SemEval, where the category name

Models	ACE05	SemEval
Zhang et al. (2018)	-	84.8
Christopoulou et al. (2018)	64.2	-
Ye et al. (2019)	68.9	-
Wu and He (2019) (BERT)	-	89.2
Soares et al. (2019) (BERT)	-	89.5
Guo et al. (2019)	-	85.4
Sun et al. (2020)	-	86.0
Mandya et al. (2020)	-	85.9
Yu et al. (2020)	-	86.4
Ours	79.05	89.85

Table 5.7: The comparison (F1 scores) between previous studies and our best models using two T-GCN layers and BERT-large encoder on ACE05 and SemEval.

indicates the range of the entity distance.²⁸ It is observed that, T-GCN outperforms the two baselines on all groups of test instances and the improvement becomes larger when the entity distance increases. This observation confirms that our approach is able to leverage dependency information and capture long-distance word-word relations to improve RE.

The Effect of Graph Construction

In the main experiments, we try T-GCN with the graph built upon the combined local and global connections (“L + G”). To explore the effect of the local connections and the global connections for T-GCN, we run our approach using two T-GCN layers with the graph constructed by local connections (“L”) or global connections (“G”) alone. Table 5.8 presents the experimental results (F1 scores) of different models with BERT-base and BERT-large encoders, where the results from BERT-only baselines, T-GCN (L + G), and T-GCN (Full) are also copied from Table 5.6 for reference. Compared to T-GCN (L + G), models with the graph constructed by either local connections (i.e., T-GCN (L)) or global connections (i.e., T-GCN (G)) achieve lower performance, which complies

²⁸For example, a test sentence whose distance in between two entities is 7 will fall into the group (5, 10].

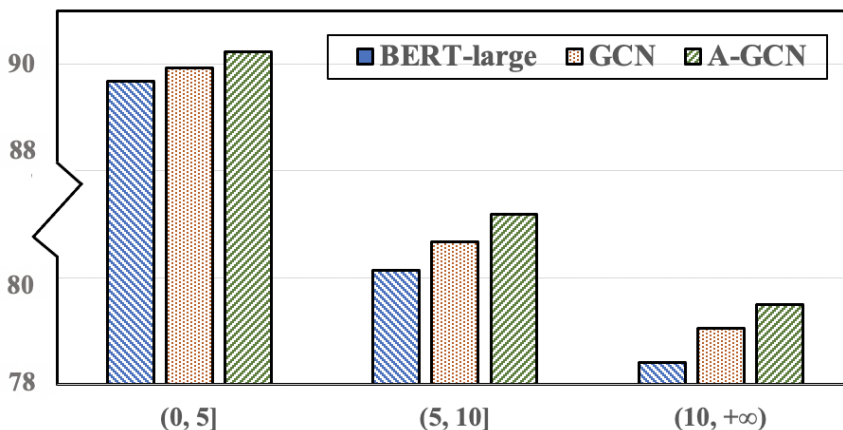


Figure 5.7: Performance (F1 scores) of different models (i.e., BERT-only, two layers of standard GCN, and two layers of T-GCN) with the BERT-large encoder on three groups of test instances from SemEval, where each group is generated based on the distance (i.e., number of words) between two entities in an instance.

with our intuition because both groups of connections contain important contextual features for RE. Interestingly, it is found that T-GCN (L) outperforms T-GCN (G) with both BERT-base and BERT-large encoders. A possible explanation could be the following. There are overlaps between local and global connections (e.g., the connection between “*range*” and “*restrictions*” in Figure 5.6). Therefore, T-GCN (L) can not only leverage the contextual information associated with the entities themselves, but is also partially²⁹ benefited from the overlapping connections on the SDP between the two entities, which leads T-GCN (L) to achieve a higher performance than T-GCN (G).

Ablation Study

Compared with the standard GCN, T-GCN enhances it from two aspects: (1) using an attention mechanism to weigh different dependency connections and (2) introducing dependency types to the process to encode more detailed dependency information. To better investigate the effect of each individual enhancement (i.e., the attention mechanism or the dependency type information), we conduct an ablation study on our best model, i.e., two layers of T-GCN (L + G) with BERT-base

²⁹When there is only one word on the shortest dependency path between two entities, all global connections are included in local ones, e.g., “*defamation*” and “*bishop*” in Figure 5.5.

ID	Models	ACE2005	SemEval
1	BERT-Base	75.31	87.87
2	+ T-GCN (L)	76.92	88.89
3	+ T-GCN (G)	76.72	88.89
4	+ T-GCN (L + G)	77.30	89.16
5	+ T-GCN (Full)	77.25	88.70
6	BERT-Large	76.79	89.02
7	+ T-GCN (L)	78.61	89.70
8	+ T-GCN (G)	78.40	89.38
9	+ T-GCN (L + G)	79.05	89.85
10	+ T-GCN (Full)	78.91	89.70

Table 5.8: Performance of our models with two T-GCN layers using the graphs built upon (1) only local connections (L), (2) only global connections (G), (3) the combination of local and global connections (G + L) , and (4) full dependency graph (Full). The performance of BERT-only baseline is also reported for reference.

and BERT-large encoder. Table 5.9 reports the experimental results of different models, where the performance of BERT-only baseline and the standard GCN baseline (i.e., the one uses neither the attention mechanism nor dependency types) are also reported for reference. The results clearly indicate that, the ablation of either enhancement (i.e., the attention mechanism or the dependency type information) could result in worse results (compared with full T-GCN). Between the two enhancements, the ablation of the attention mechanism hurts T-GCN more, which indicates the ability of distinguishing important connections and leveraging them accordingly plays a more important role in RE.

	Att.	Type	ACE2005	SemEval
BERT-base	Baseline		75.31	87.87
	✓	✓	77.30	89.16
	×	✓	77.00	88.07
	✓	×	76.27	88.50
	GCN		76.11	88.62
BERT-large	Baseline		76.79	89.02
	✓	✓	79.05	89.85
	×	✓	78.92	89.26
	✓	×	78.22	89.37
	GCN		77.92	89.13

Table 5.9: The ablation study on the attention mechanism (Att.) and dependency types (Type) in our best model, i.e., two layers of T-GCN (L + G). “✓” and “×” stand for that whether a module is used. The F1 scores of BERT-only and the standard two layers of GCN (L + G) are also reported for references.

Case Study

To explore in detail that how T-GCN leverages dependency connections and types to improve RE, we conduct a case study with our T-GCN models with different dependency graphs (i.e., two layers of T-GCN (Full) and T-GCN (L + G) with BERT-large encoder) on an example sentence “*A central vacuum is a vacuum motor and filtration system built inside a canister.*”. Figure 5.8 shows the sentence where both the two models correctly predict the relation between “*motor*” (E_1) and “*canister*” (E_2) (highlighted in the red color) to be “*Content-Container*”, whereas the baseline GCN (Full) and GCN (L + G) models fail to do so. We also visualize the attention weights assigned to different dependency connections extracted from the last T-GCN layer, with darker and thicker lines referring to higher weights. In this example, for T-GCN (Full), we observe that the connection between “*built*” and “*canister*” along SDP and the connection between “*inside*” and “*canister*” receive the highest weights, where this is valid because the dependency type, i.e., *obl* (oblique nominal), associated

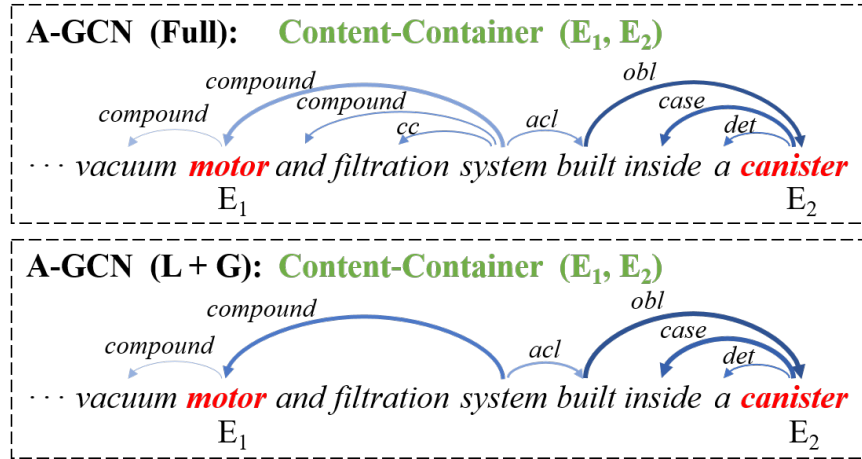


Figure 5.8: Visualizations of weights assigned to different dependency connections of T-GCN (Full) and T-GCN (L + G) for an example input, where darker and thicker lines refer to connections with higher weights.

with the connection (between “*built*” and “*canister*”) reveals that “*canister*” could be the position where the action (i.e., *build*) takes place, and is further confirmed by another dependency connection and type (i.e., *case*) between “*inside*” and “*canister*”. Therefore, it is proved that our model learn from the contextual information carried by such important connections and results in correct RE prediction. Similarly, T-GCN (L + G) also correctly perform RE on this case by highlighting the same dependency connections as those from the T-GCN (Full) with much higher weights (because many dependency connections are filtered out).

5.3 Aspect-based Sentiment Analysis

Pragmatic-level tasks address language use in context, focusing on how meaning is affected by situational factors and speaker intent. Aspect-based sentiment analysis (ABSA) is a typical pragmatic-level task that attracts much attention from existing studies. Recent neural approaches for aspect-based sentiment analysis (ABSA) primarily leverage powerful encoders such as Bi-LSTM and Transformer, effectively capturing contextual features in text (Wang et al., 2016b; Tang et al., 2016a; Fan et al., 2018). To further improve context modeling, some studies employ graph convolutional networks (GCN) built upon dependency parsing results, allowing models to capture distant relationships

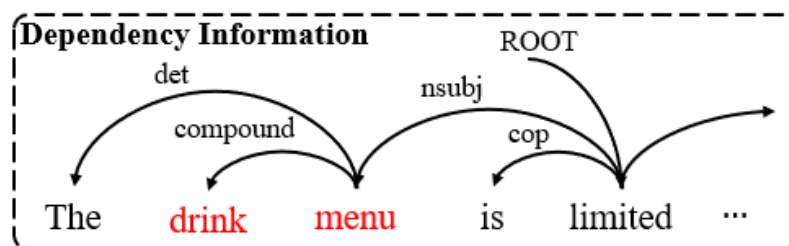


Figure 5.9: An example sentence (including the aspect term “*drink menu*”) with its dependency parsing result.

between words and aspects more explicitly (Sun et al., 2019; Huang and Carley, 2019; Zhang et al., 2019a). Despite demonstrated improvements, these GCN-based methods remain limited by two key issues. First, they treat all dependency edges equally, neglecting the informative syntactic details embedded within dependency types, thus potentially ignoring crucial cues for sentiment prediction. For instance, dependency types such as “*nsubj*” explicitly indicate important semantic relationships but are not leveraged by existing GCN methods. Second, previous GCN models commonly utilize representations exclusively from their final layers, disregarding potentially valuable contextual information captured by intermediate layers. Different GCN layers may encode varying degrees of contextual abstraction, and omitting intermediate-layer information can lead to incomplete modeling of contextual nuances. Therefore, an effective framework that explicitly integrates dependency type information and appropriately aggregates information from multiple GCN layers is necessary to fully exploit dependency knowledge for ABSA.

To address these challenges, we propose a type-aware graph convolutional network (T-GCN) designed to comprehensively incorporate word relations along with their dependency types for ABSA. Specifically, our approach first constructs a dependency graph with edges labeled by dependency types extracted using off-the-shelf parsers. Then, an attention mechanism dynamically assigns weights to these typed edges, distinguishing the important contextual information for the task. Additionally, an attentive layer ensemble is introduced to effectively combine context-aware representations from multiple GCN layers. Experiments conducted on six English benchmark datasets demonstrate the effectiveness of our T-GCN, achieving state-of-the-art performance and confirming the benefits of explicitly leveraging dependency types and attentively aggregating multi-layer contextual information.

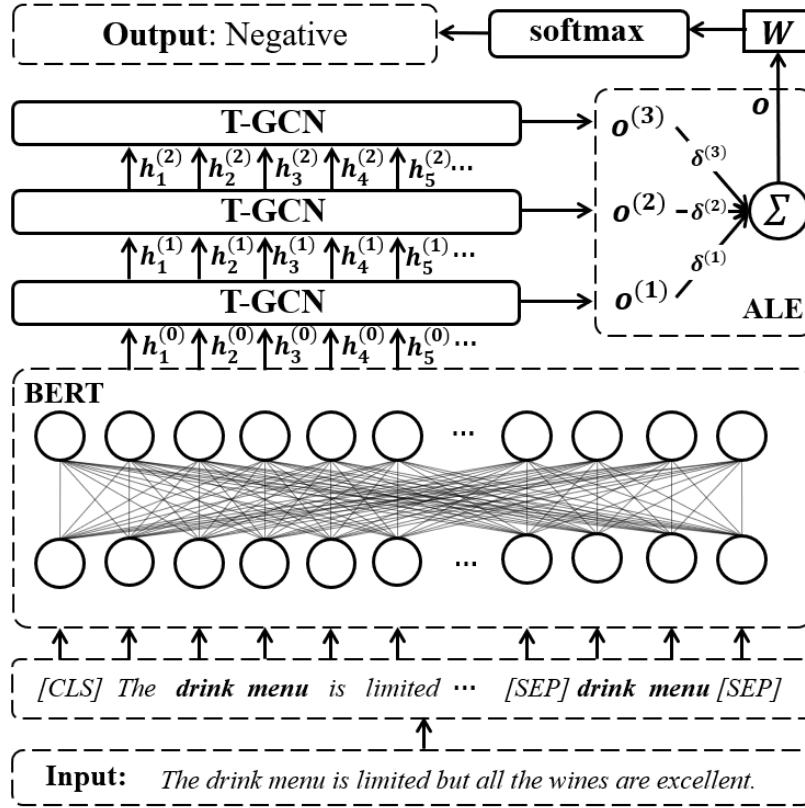


Figure 5.10: The overall architecture of our approach with an example sentence-aspect pair input (the aspect words “*drink menu*” are in boldface) from a sentence. Our T-GCN and ALE are marked on top of the figure.

5.3.1 The Approach

Given an input sentence $\mathcal{X} = x_1, x_2, \dots, x_N$ and the aspect terms $\mathcal{A} \subset \mathcal{X}$ (\mathcal{A} is usually a sub-string of \mathcal{X}), the conventional ABSA approaches often take the sentence-aspect pair as the input and predicts \mathcal{A} ’s sentiment polarity \hat{y} (Tang et al., 2016b; Ma et al., 2017; Xue and Li, 2018; Hazarika et al., 2018; Fan et al., 2018; Huang and Carley, 2018; Tang et al., 2019; Chen and Qian, 2019; Tan et al., 2019; Tang et al., 2020). We follow this paradigm, and the overview of our approach is illustrated in Figure 5.10, with a contextual encoder (i.e., BERT), the proposed T-GCN, and the attentive layer ensemble (ALE). The overall conceptual formalism of our approach can be written as

$$\hat{y} = f(\mathcal{X}, \mathcal{A}) \quad (5.12)$$

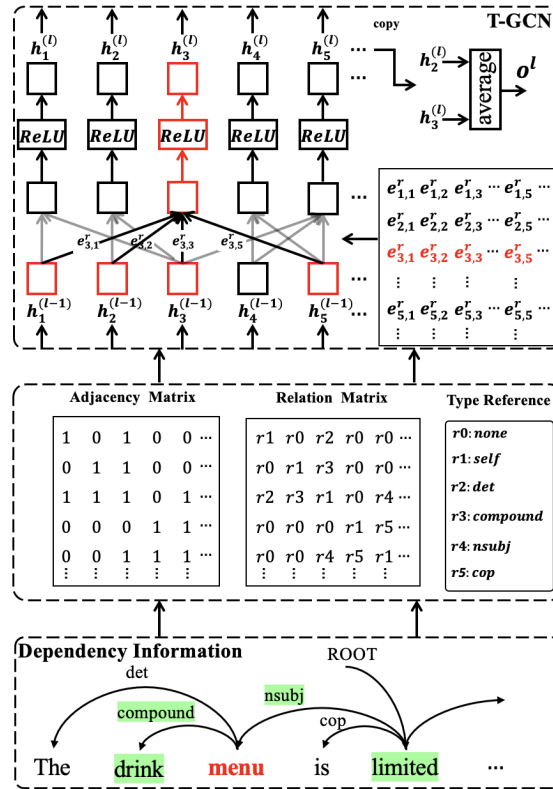


Figure 5.11: An illustration of how we build the type-aware graph from dependency parsing results and the detail of a T-GCN layer that consumes the graph. Edges and their dependency types are illustrated in the adjacency matrix and the relation matrix, respectively.

Herein, the T-GCN is the same as the model for RE in Section 5.2.1. Therefore, in the following texts, we describe the ALE to incorporate contextual information from different T-GCN layers and then illustrate incorporating T-GCN to ABSA.

T-GCN with Attentive Layer Ensemble

For each word $x_{n'}$, since every T-GCN layer incorporates information from the words that directly connect to it, so that multiple T-GCN layers could learn indirect word relations from long distances. Thus, it is assumed that different layers have their unique capabilities to encode contextual information. To utilize such capabilities, we propose to comprehensively learn from all T-GCN layers with an attentive layer ensemble.

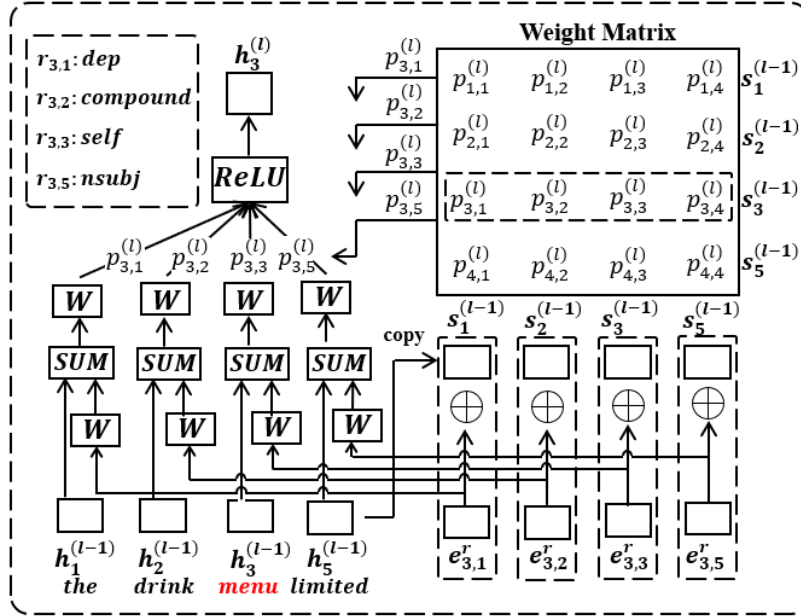


Figure 5.12: The illustration of how we compute $h_{n'}^{(l)}$ for $x_3 = \text{"menu"}$ through a T-GCN layer. All words $x_{n''}$ connected to "menu" with their dependency types (in embeddings $r_{n',n''}$) are shown at the bottom part.

In doing so, we firstly obtain the output $\mathbf{o}^{(l)}$ from each T-GCN layer by averaging the output hidden vectors of all tokens x_i of the aspect terms:

$$\mathbf{o}^{(l)} = \frac{1}{|\mathcal{A}|} \cdot \sum_{x_i \in \mathcal{A}} \mathbf{h}_i^{(l)} \quad (5.13)$$

where $|\mathcal{A}|$ is the number of tokens in the aspect terms \mathcal{A} . Then we attentively ensemble the output of all T-GCN layers through a weighted average:

$$\mathbf{o} = \sum_{l=1}^L \beta^{(l)} \cdot \mathbf{o}^{(l)} \quad (5.14)$$

where \mathbf{o} is the final vector output for ABSA and $\beta^{(l)}$ is a trainable weight assigned to $\mathbf{o}^{(l)}$ to balance its contribution and satisfying $\sum_{l=1}^L \beta^{(l)} = 1$.

Encoding and Decoding with T-GCN

To support applying T-GCN for ABSA, there are necessary encoding and decoding processes. For encoding, there are two ways in doing so. The first is to take the sentence \mathcal{X} as the input, and the

Datasets		Pos. #	Neu. #	Neg. #
Lap14	Train	994	464	870
	Test	341	169	128
Rest14	Train	2,164	637	807
	Test	728	196	182
Rest15	Train	907	36	254
	Test	326	34	207
Rest16	Train	1,229	69	437
	Test	469	30	114
Twitter	Train	1,561	3,127	1,560
	Test	173	346	173
MAMS (ATSA)	Train	3,380	5,042	2,764
	Dev	403	604	325
	Test	400	607	329

Table 5.10: The number of aspects with *positive*, *neutral*, and *negative* sentiment polarities in all datasets.

second is to take the sentence-aspect pair as the input, where the standard encoding process (see Section 3.3.1) is used to compute the hidden vectors of the input. For both cases, the hidden vectors are fed into the T-GCN model and the ALE module. For decoding, after we obtain \mathbf{o} from ALE, we firstly map \mathbf{o} to the label space by a fully connected layer, $\mathbf{u} = \mathbf{W} \cdot \mathbf{o} + \mathbf{b}$, where \mathbf{W} and \mathbf{b} are the trainable matrix and the bias, respectively. Then, we employ a *softmax* function to \mathbf{u} to predict the output sentiment \hat{y} for the aspect \mathcal{A} in \mathcal{X} .

5.3.2 Experiment Settings

Datasets

In the experiments, we employ five widely used English benchmark datasets: Lap14 and Rest14 from Pontiki et al. (2014), Rest15 from Pontiki et al. (2015), Rest16 from Pontiki et al. (2016), and Twitter from Dong et al. (2014), with their official train/test splits. In addition, we try another recently released English dataset, named MAMS³⁰ (Jiang et al., 2019), with the official train/dev/test splits for ABSA, which is much larger than the aforementioned five datasets. It is worth noting that, in addition to the *positive*, *neutral*, and *negative* sentiment labels, Lap14, Rest14, and Rest16 contain another *conflict* label, which identifies the aspects that have conflict sentiment polarities. For example, the aspect “*sushi*” is assigned by a *conflict* label in “*Certainly not the best sushi in New York, however, it is always fresh.*” from Rest14. Therefore, we follow Tang et al. (2016b) to clean the datasets by removing all aspects with the aforementioned *conflict* label, as well as sentences without an aspect. The statistics (number of aspects with *positive*, *neutral*, and *negative* labels) of the processed six datasets are reported in Table 5.10.

Implementation Details

To build the graph for T-GCN, we firstly use the current best-performing constituency parser, i.e., SAPar³¹ (Tian et al., 2020d), to parse all input text into constituency trees, then convert the trees into dependency trees by Stanford Converter³², and finally build the graph over the dependency relations and types from the trees.³³ We employ the uncased BERT-base and BERT-large³⁴ (Devlin et al., 2019) as the context encoder with their default settings. For the number of T-GCN layers, we set it to three. We try two ways to encode the input, where the first encodes the single sentence and the second encodes the sentence-aspect pair. For all models, we use the pre-trained parameters of BERT and initialize all other trainable parameters by Xavier (Glorot and Bengio, 2010). Moreover, we use

³⁰We use the ATSA part of MAMS obtained from <https://github.com/siat-nlp/MAMS-for-ABSA>.

³¹<https://github.com/cuhksz-nlp/SAPar>

³²We use the converter of version 3.3.0 from <https://stanfordnlp.github.io/CoreNLP/index.html>.

³³We also try Stanford CoreNLP Toolkits (<https://stanfordnlp.github.io/CoreNLP/>) (Manning et al., 2014a) and spaCy (<https://spacy.io/>) dependency parsers with similar results obtained.

³⁴We obtain the BERT models from <https://github.com/huggingface/pytorch-pretrained-BERT>.

the cross-entropy loss function for our models and follow previous studies (Tang et al., 2016a; Chen et al., 2017a; He et al., 2018a; Sun et al., 2019; Zhang et al., 2019a) to evaluate them via accuracy and macro-averaged F1 scores over all sentiment polarities. For datasets without the official development set, we randomly sample 10% instances from the training set and regard them as the development set to find the best hyper-parameter setting which is then used to train different models on the entire training set.

5.3.3 Results and Analyses

Overall Results

In the main experiments, for each encoder (i.e., BERT base and large), we run two baselines: 1, only using BERT and 2, BERT with normal GCN where all edges are equally treated and the ABSA result is predicted based on the output of the last GCN layer. Table 5.11 reports the experimental results from all baselines and our models.

There are several observations. First, for both BERT-base and BERT-large encoders, although the models with normal GCN are able to enhance the BERT baselines, our models can further improve the performance in both accuracy and F1 scores on all datasets. This observation clearly illustrates the effectiveness of incorporating dependency type information into GCN and thus improves ABSA accordingly. Second, in most cases, our models that encode the sentence-aspect pair achieve higher results than the ones encoding the single sentence, which is not surprising because the aspect is therefore emphasized in the input and provides more contextual information to be modeled for ABSA.

To further demonstrate the effectiveness of our approach, we compare the performance of our best model (i.e., T-GCN using BERT-large encoder with sentence-aspect pair input) with previous studies on all datasets. The results are reported in Table 5.12, where our model outperforms previous studies, including the ones (Huang and Carley, 2019; Wang et al., 2020; Tang et al., 2020) using dependency information (marked by “†”), on all datasets in terms of both accuracy and F1 scores. In particular, compared with our approach, Huang and Carley (2019) use a variant of graph attention networks (GAT), while they do not use dependency types; Wang et al. (2020) also use a variant of GAT and they use the relation type as well, but they do not assign different weights to separate word-word relations; Tang et al. (2020) use a variant of GCN, but they do not use the dependency

Models	Lap14		Rest14		Rest15		Rest16		Twitter		MAMS	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
BERT-base (S)	77.74	73.30	82.68	73.54	81.34	63.57	88.89	68.19	73.70	71.50	78.94	79.42
+ GCN	79.52	76.01	84.79	77.93	83.60	65.71	90.76	72.79	75.16	72.96	80.69	80.27
+ T-GCN	80.25	76.92	85.54	78.86	85.07	72.50	91.83	76.86	76.16	74.44	81.73	81.12
BERT-base (P)	78.68	74.64	84.55	77.34	83.40	65.28	89.54	70.47	75.00	72.53	80.11	80.34
+ GCN	79.94	76.72	85.09	78.81	84.14	65.75	91.01	73.38	75.29	73.68	81.96	81.31
+ T-GCN	80.88	77.03	86.16	79.95	85.26	71.69	92.32	77.29	76.45	75.25	83.38	82.77
BERT-large (S)	78.06	74.67	83.04	73.27	83.02	68.34	90.20	73.64	73.12	72.08	79.33	79.87
+ GCN	80.09	76.84	86.07	80.35	84.69	70.31	91.48	74.96	75.21	73.69	81.36	81.04
+ T-GCN	81.50	78.48	86.88	81.03	85.07	70.30	92.32	75.83	75.43	73.71	82.70	82.16
BERT-large (P)	79.62	75.77	85.53	77.64	84.14	69.67	91.34	74.35	75.43	73.55	80.62	80.77
+ GCN	80.68	77.85	86.48	80.63	85.42	70.42	91.69	75.24	75.26	73.41	82.56	82.14
+ T-GCN	81.97	78.71	87.41	82.23	86.00	72.81	92.97	80.07	78.03	77.31	83.68	83.07

Table 5.11: Experimental results (accuracy and F1 scores) of using two encoders i.e., BERT-base and BERT-large, with different configurations on six benchmark datasets. “GCN” refers to the normal GCN model without using type-aware graph, attention mechanism as well as ALE. “S” and “P” refer to the settings that the input is a single sentence and a sentence-aspect pair, respectively.

type information. Our model shows its superiority to the aforementioned studies since we not only assign different weights to dependencies, but also comprehensively leverage the dependency parsing results with both word relations and their dependency type information, as well as fined-grained encoding results from multiple T-GCN layers.

Ablation Study

To explore the effectiveness of different components in our model, i.e., type-aware graph (TG), attention (Att), and ALE, we conduct an ablation study based on our best model (i.e., T-GCN on BERT-large encoder with sentence-aspect pair input). The experimental results on all datasets with respect to using different combinations of such components are reported in Table 5.13, with the results of the full model and the baseline with normal GCN illustrated on the first (ID: 1) and last row

Models	Lap14		Rest16		Twitter		MAMS	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
He et al. (2018a)	72.57	69.13	64.61	67.45	-	-	-	-
He et al. (2018b)	71.15	67.46	85.58	69.76	-	-	-	-
Du et al. (2019)	76.80	73.29	-	-	75.01	73.81	-	-
Mao et al. (2019)	75.84	72.49	-	-	72.35	69.45	-	-
Song et al. (2019)	79.93	76.31	-	-	74.71	73.13	-	-
Sun et al. (2019)	77.19	72.99	85.58	69.93	74.66	73.66	-	-
Zhang et al. (2019a)	75.55	71.05	88.99	67.48	72.69	70.59	-	-
Wang et al. (2020)	78.21	74.07	-	-	76.15	74.88	-	-
Tang et al. (2020)	79.8	75.6	91.9	79.0	77.9	75.4	-	-
Ours	81.97	78.71	92.97	80.07	78.03	77.31	83.68	83.07

Table 5.12: Performance (accuracy and F1 scores) comparison of our best model (i.e., T-GCN and ALE on large BERT with sentence-aspect pair input) with previous studies on all six benchmark datasets. Models using dependency information are marked by “+”.

(ID: 8), respectively. Herein, models without ALE (ID: 4-6) use the output of the last T-GCN layer (i.e., the third layer) to predict the sentiment polarity.

Here are some observations. First, it is clearly indicated in the results that the model performance drops on all datasets if any component is excluded from the full model. This observation indicates that all three components play important roles in our approach to enhancing ABSA; each one has its unique contribution to the full model. Second, for each single component, compared with the results from GCN baseline (ID: 8), the results from models with a particular module (ID: 5-7) demonstrate that the attention mechanism is the most important one to improve model performance, where on all datasets, the model (ID: 6) with attention outperforms the others. This observation complies with our intuition because the attention directly guides the model to distinguish the contextual information to the aspect words, so that informative words are highlighted so as to improve ABSA accordingly.

	Setting			Lap14		Rest14		Rest15		Rest16		Twitter		MAMS	
	TG	Att	ALE	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
1	✓	✓	✓	81.97	78.71	87.41	82.23	86.00	72.81	92.97	80.07	78.03	77.31	83.68	83.07
2	×	✓	✓	81.46	78.69	87.21	81.35	85.93	72.06	92.14	77.80	77.31	76.42	83.01	82.53
3	✓	×	✓	80.96	77.66	86.68	81.09	85.66	71.03	91.89	76.40	76.73	75.90	82.78	82.47
4	✓	✓	×	80.88	77.54	86.85	81.20	85.89	71.32	91.75	76.65	77.16	76.23	83.31	82.70
5	✓	×	×	80.79	77.42	86.50	80.42	85.65	70.50	91.45	75.64	76.15	75.28	83.16	82.76
6	×	✓	×	81.10	78.12	86.88	81.02	85.89	71.10	91.99	77.50	77.31	76.18	83.53	82.90
7	×	×	✓	80.85	77.56	86.45	80.21	85.79	70.91	91.66	75.92	76.73	74.97	82.86	82.38
8	×	×	×	80.68	77.85	86.48	80.63	85.42	70.42	91.69	75.24	75.26	73.41	82.56	82.14

Table 5.13: Experimental results of ablation study on the six datasets, with different configurations applied to our best model. ‘TG’ refers to the type-aware graph; ‘Att’ denotes the attention mechanism in T-GCN; ‘ALE’ stands for the attentive layer ensemble. ‘✓’ and ‘×’ represent if a corresponding component is used or not.

Impact of Different T-GCN Layers

Besides those components, we also investigate the effect of each layer when our model is trained on different datasets. In doing so, we perform experiments on all datasets using our best-performing model and use the weight ($\delta^{(l)}$ in Eq. (5.14)) assigned to each T-GCN layer to identify the contribution of them. The results are illustrated in Figure 5.13, with the weights for the 1st, 2nd, and 3rd T-GCN layers drawn in blue, green, and orange bars, respectively.

We have the following observations. First, all layers contribute to the final prediction for ABSA, which complies with our expectations and confirms the validity of leveraging the information from all layers of GCN. Therefore, the model is able to provide comprehensive contextual information comparing to that only uses the output from the last layer. Second, interestingly, as shown in the histograms, for most datasets (i.e. Lap14, Rest14, Rest15, Rest16, and MAMS), the second layer of T-GCN contributes the most among all three layers. A possible reason behind is that (1) the second layer is able to encode contextual information from a larger range (because the edges in the first layer only cover words with direct relations, while the second and third layers provide indirect relations,

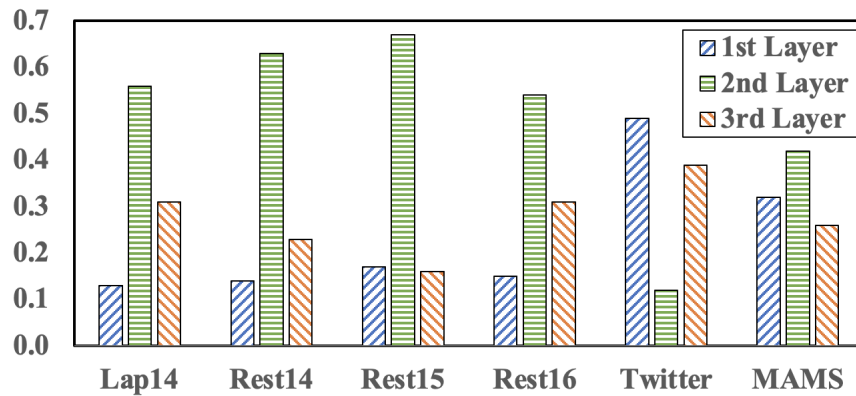


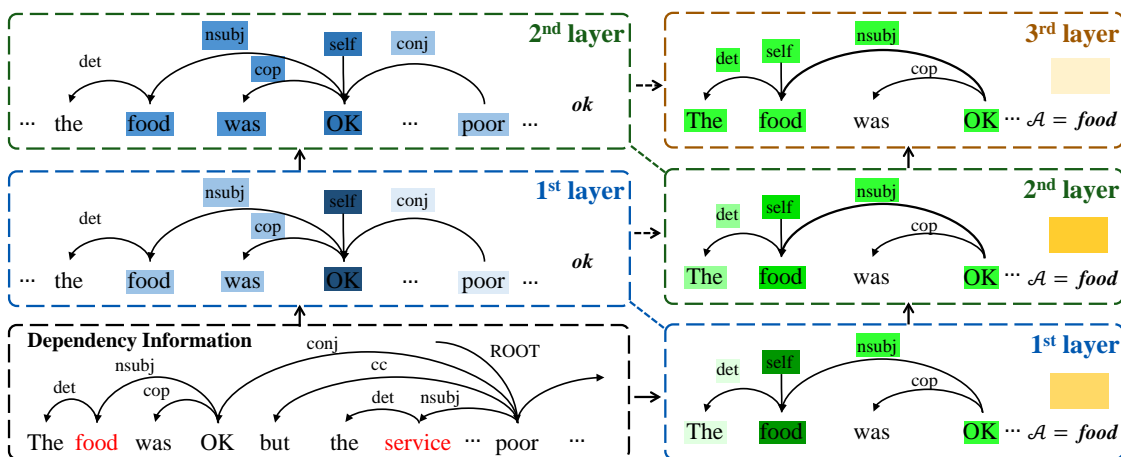
Figure 5.13: The histograms of weights assigned to different T-GCN layers (blue, green, and orange bars refer to the weights for the 1st, 2nd, and 3rd layer, respectively) in ALE with respect to each dataset.

i.e., second and third order dependencies in practice); (2) comparing to the third layer, the second layer may introduce less irrelevant information from multi-word relations. Third, we also notice that for Twitter, the weight distribution among three layers is rather different from the other datasets, where the first and last layer contributes more to ABSA. This observation can be explained by that, Twitter is social media data, where, in general, sentences in such data are short and less organized, so that our model may require the information from either the local context or the entire sentence for ABSA.

Case Study

To further illustrate the effectiveness of T-GCN on leveraging the information of dependency types and weighting salient word relations for improving ABSA, we conduct a case study on using our model to process the sentence “*The food was OK but the service was so poor that the food was cold by the time everyone in my party was served*” from Rest16. In this sentence, there are two aspects with contrast sentiment polarities, i.e., “*food*” and “*service*” have *positive* and *negative* sentiment suggested by “*OK*” and “*limited*”, respectively.

To demonstrate the effectiveness of our model to process such sentence with conflict sentiments, on the right part of Figure 5.14, we visualize weights (in green) assigned to the edges connected



$X =$ The *food* was ok but the *service* was so poor that the food was cold by the time everyone in my party was served

Figure 5.14: Visualization of the weights assigned to different edges and dependency types in each T-GCN layer for an example sentence with two aspects (in red) in conflict sentiment polarities. The edge and type weights (in blue) for “OK” in the first and second layer are illustrated on the left, while such weights (in green) for “food” and ALE weights (in yellow) for each layer are illustrated on the right. Deeper color refers to the higher weight.

to “food” from the attention in all T-GCN layers, and the ALE weights (in yellow) for each layer, where deeper color refers to higher weight. For those edges, except for its self-connection, the edge between “food” and “OK” receives the highest weight in every layer, and the second layer receives the highest weight in ALE. Note that in this case, the reason why T-GCN works can be explained by that, when there are more than two layers are used in a GCN model, the edges connecting to “OK” also influence the ABSA results because indirect relations are introduced across layers. As a result, the noisy connection between “OK” and “poor” may contribute to the prediction and the normal GCN could possibly fail on this case because of lacking a mechanism to distinguish it from other edges. Therefore, as shown in the left part of Figure 5.14, we also visualize the weights for edges connecting to “OK” from the first and second T-GCN layers,³⁵ where the informative word relations and their dependency types receive much heavier weights than that for noisy ones. Moreover, it is noticed that the dependency type for the edge between “OK” and “poor” is “conj” (conjunction),

³⁵Note that we do not visualize the weights for “OK” in the third layer because its resulting hidden vector does not contribute to the final sentiment prediction.

which suggests that “*poor*” is syntactically parallel with “*OK*” and is thus less likely to provide essential sentiment guidance for “*OK*”. Overall, this case study illustrates that our model successfully identifies that “*OK*” is the most important contextual information to determine the sentiment for “*food*”, with the help of dependency type and attention used in T-GCN, and also shows that the final prediction relies on the contributions from different T-GCN layers.

5.4 Summary

In this chapter, we leverage a representative type of knowledge from toolkits, namely, the dependency parse trees, to three tasks, namely, NER, RE, and ABSA, at the semantic and pragmatic levels. We propose different approaches, namely, KVMN and T-GCNs for leveraging the knowledge with PLM. The results demonstrate the effectiveness of the knowledge and the approaches for the tasks.

Chapter 6

Using Pattern Knowledge

In this chapter, we present our research on utilizing pattern knowledge (i.e., vectors) learned during the training process for challenging tasks, including multimodal sentiment analysis and medical large language modeling. Through task-specific learning strategies with LLMs, we aim to equip the models with abstract knowledge, thereby improving their capability to perform complex NLP tasks.

6.1 Multimodal Sentiment Analysis

Multimodal content, such as image-text pairs, is widespread across diverse real-world scenarios such as news, social media, and instant messaging (Kiela et al., 2020; Gomez et al., 2020; Li et al., 2022). Generally, images and texts in multimodal messages intentionally create semantic misalignment, generating irony, humor, or hateful expressions through non-compositional contrasts. These contrasts allow combined information to convey meanings not explicitly present in either modality alone. Current multimodal understanding approaches primarily focus on aligning visual and textual features using advanced encoders (e.g., CLIP, Flamingo, FLAVA) and applying fusion operations such as concatenation or attention-based alignment for downstream tasks, including hateful meme detection (HMD) (Li et al., 2019; Radford et al., 2021; Goyal et al., 2022; Kumar and Nandakumar, 2022), a representative task in multimodal sentiment analysis. However, these approaches typically struggle to capture subtle hateful meanings arising explicitly from cross-modal contrasts. Efforts to improve



Figure 6.1: Three memes: (a) is hateful; (b) and (c) are not. Here, (a) and (b) share the same text, and (a) and (c) share the same image. This example shows that a meme can be hateful even when neither its image nor its text is.

multimodal tasks often rely on additional resources or ensemble learning strategies to enhance generalization (Muennighoff, 2020; Lippe et al., 2020; Cao et al., 2023b), but these approaches mostly neglect the critical role of cross-modal contrasts responsible for non-compositional semantics. Therefore, a clear research gap remains regarding effective modeling and interpretation of cross-modal contrast to achieve a comprehensive understanding of subtle hateful or ironic expressions embedded within multimodal content.

To address this challenge, we propose an approach leveraging LLMs enhanced by a cross-modal memory mechanism and reinforced contrast recognition (RCR) via self-rejection training. Our cross-modal memorizing module explicitly captures and integrates contextual information across modalities, while the RCR mechanism optimizes learning by encouraging the model to accurately identify and interpret multimodal contrasts. Evaluations conducted on four benchmark datasets demonstrate that our approach substantially outperforms strong baselines and previous multimodal approaches. These results confirm the effectiveness of explicitly modeling multimodal contrasts through memory-enhanced and reinforcement-based techniques, enabling superior performance across tasks dependent on non-compositional multimodal semantics.

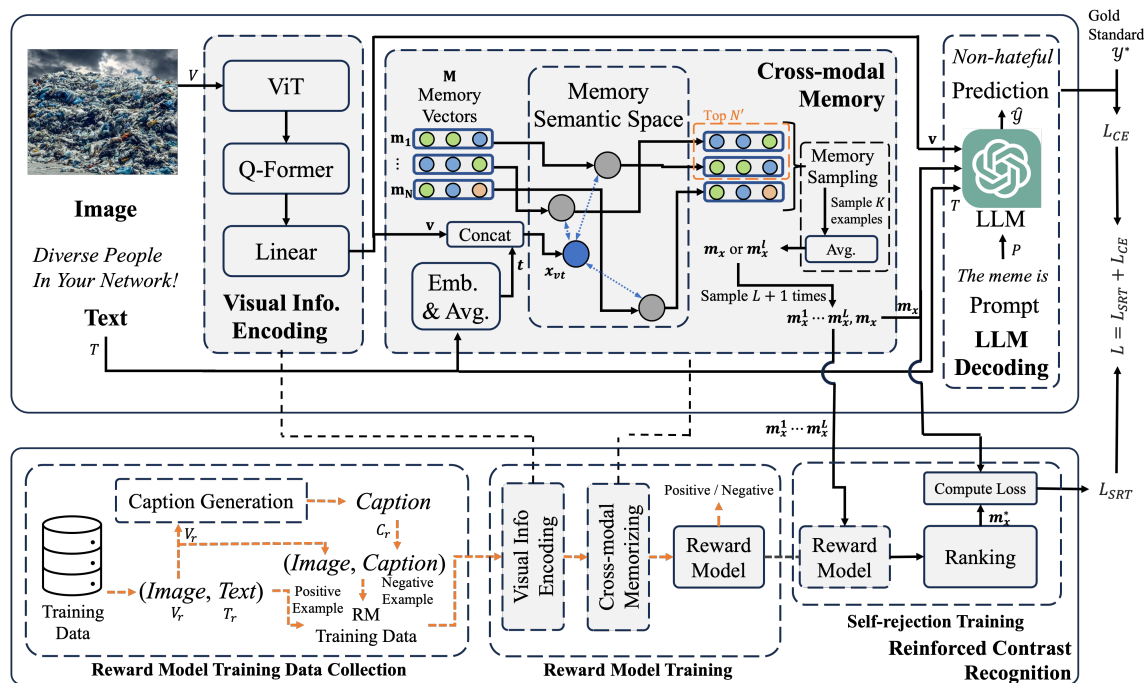


Figure 6.2: The overall architecture of our approach using HMD as a demonstration example. The top part illustrates our backbone model with cross-modal memory, and the bottom part shows the RCR process. The workflow of collecting data and training the reward model is illustrated in orange arrows in the RCR part. Then the trained reward model is used in the self-rejection training to optimize the model, illustrated at the bottom right part. Visual encoding and cross-modal memory are shared by the backbone model in the main process and the reward model training process, which is marked by the boxes with dashed black lines.

6.1.1 The Approach

Figure 6.2 illustrates the framework of our approach, which follows the encoding-decoding paradigm to perform multimodal classification, which predicts a label \hat{y} based on the image \mathcal{I} and embedded text \mathcal{X} in a given pair $(\mathcal{I}, \mathcal{X})$. It contains three essential components: the backbone model, the cross-modal memory, and the RCR. The memory module is inserted in between the visual encoding and LLM decoding in the backbone, illustrated at the top of Figure 6.2. The RCR, demonstrated at the bottom of Figure 6.2, enhances the memory by enhancing the memory module with contrast

information via self-rejection training, where an additional term is added to the loss function. In the following subsections, we firstly illustrate the backbone model, then the cross-modal memory, and finally present the RCR.

The Backbone Model

The encoding and decoding processes are two essential components in the backbone model. Specifically, the visual encoding process (f_{ve}) extracts salient features from the input image and LLM decoding (f_d) utilizes the multimodal information to predict the final classification label \hat{y} .

Our visual encoder, following the procedure of BLIP2 (Li et al., 2023b), extract visual feature \mathbf{v} from the input meme \mathcal{V} through

$$\mathbf{v} = f_{ve}(\mathcal{V}) \quad (6.1)$$

For the LLM, we use the visual feature \mathbf{v} and the contrast vector \mathbf{m}_x obtained from cross-modal memorizing as soft prompts to instruct our LLM for final classification. Specifically, we feed \mathbf{v} , \mathbf{m}_x , as well as the original text \mathcal{X} , into the LLM to determine the label \hat{y} , e.g., *hateful* or *non-hateful* if the task is HMD. A prompt P is required to instruct the LLM to process the input and predict the label. For example, we design a simple prompt, i.e., “*The meme is ...*” for HMD, which instructs LLM to fill in the blank with “*hateful*” or “*non-hateful*”.¹ We feed \mathbf{v} , \mathbf{m}_x , \mathcal{T} , P into the LLM to predict the class label \hat{y} of the input image-text pair following the standard classification process.

Cross-modal Memory

Memory mechanism is demonstrated to be effective in modeling task-related information (Song et al., 2018b; Nie et al., 2020; Tian et al., 2022b, 2023) The memory module is designed to better fuse multimodal features, where we use a memory matrix \mathbf{M} , stacked with N memory vectors (denoted by $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_N]$) to record such feature fusion. Each memory vector is interpreted as a potential aspect in a multimodal fusion that results in a particular type of information, e.g., hateful information for the HMD task or contrast that causes humor. This module has three main components: text encoding, memory ranking, and sampling.

¹The prompt may change for different tasks.

Text encoding obtains text representations to facilitate subsequent processes. We follow the standard approach to use a word embedding matrix to map all words in the text \mathcal{T} into their corresponding word embeddings. Then, we compute the average of the embeddings of all words in the input text and denote the resulting vector \mathbf{t} as the text features.

Memory ranking locates relevant memory vectors according to the encoded multimodal information and assigns appropriate scores to them. We concatenate visual and text features and obtain the multimodal feature $\mathbf{x}_{vt} = \mathbf{v} \oplus \mathbf{t}$. Then, we compute the score s_n that measures the semantic similarity between the n -th memory vector \mathbf{m}_n and \mathbf{x}_{vt} by

$$s_n = \frac{\exp(\mathbf{x}_{vt} \cdot \mathbf{W}^M \cdot \mathbf{m}_n)}{\sum_{n=1}^N \exp(\mathbf{x}_{vt} \cdot \mathbf{W}^M \cdot \mathbf{m}_n)} \quad (6.2)$$

where \mathbf{W}^M is a trainable parameter matrix to align \mathbf{m}_n and \mathbf{x}_{vt} . Finally, we rank all memory vectors in descending order based on their scores and select the top N' vectors (denoted as $\mathbf{m}_{n_1} \cdots \mathbf{m}_{n_{N'}}$) as the relevant vectors for later processing.

Memory sampling further processes memory vectors and outputs a vector \mathbf{m}_x that carries the essential multimodal fusion information between visual and text features for later steps. In detail, we normalize the scores of the relevant vectors and randomly select K vectors from $\mathbf{m}_{n_1} \cdots \mathbf{m}_{n_{N'}}$ (repetition of the same vector is allowed) based on their scores, where higher scores lead to better chance to be selected. We then average the selected vectors and obtain the output contrast vector \mathbf{m}_x by

$$\mathbf{m}_x = \frac{1}{K} \sum_{k=1}^K s_{n_k} \mathbf{m}_{n_k} \quad (6.3)$$

where s_{n_k} is the score for \mathbf{m}_{n_k} obtained by Eq. (6.2) and \mathbf{m}_x is used as input in LLM decoding.

Reinforced Contrast Recognition

The goal of RCR is to help the cross-modal memory module in producing a better \mathbf{m}_x . However, we do not have a gold standard for \mathbf{m}_x . Therefore, we need to create a silver standard \mathbf{m}_x^* and add the difference between \mathbf{m}_x and \mathbf{m}_x^* to the loss function. To create \mathbf{m}_x^* , we propose to use a reward model to select \mathbf{m}_x^* from a candidate list $\mathbf{m}_x^1, \cdots, \mathbf{m}_x^L$. In order to generate the candidate list, we repeat the sampling and averaging process illustrated in Eq. (6.3) for L times and obtain a list of different vectors $\mathbf{m}_x^1 \cdots \mathbf{m}_x^L$. Specifically, there are three main steps in RCR: reward model training

data collection, reward model training, and self-rejection training. In the first and second steps, we collect data and train a reward model to rank $\mathbf{m}_x^1 \cdots \mathbf{m}_x^L$ based on their effectiveness in representing the contract information between multiple modalities. Then we perform self-rejection training by using the most effective contrast vector \mathbf{m}_x^* to optimize \mathbf{m}_x so that \mathbf{m}_x is trained to be closer to \mathbf{m}_x^* . Details are presented as follows.

Reward Model Training Data Collection The goal of the reward model is to assess whether the encoded vectors from the memory module contain contrast information, and we collect positive and negative examples to train it. Therefore, we rely on the training examples for a running task that is based on such contrast (e.g., HMD) to serve as positive examples; then take the ordinary (image, caption) pairs from image captioning tasks as negative examples since the images and their corresponding captions generally share similar semantics. As a result, we randomly select instances, i.e., image-text pairs $(\mathcal{V}_r, \mathcal{T}_r)$, from the training data of particular tasks as a positive example. Then we generate captions \mathcal{C}_r for images \mathcal{V}_r using an off-the-shelf image captioning toolkit and combine with their image to form negative examples $(\mathcal{V}_r, \mathcal{C}_r)$.

Reward Model Training In training the reward model, we apply the same visual encoding and the memory module in our approach to compute the contrast vectors for the positive and negative samples by $\mathbf{v}_m^{pos} = f_m(f_{ve}(\mathcal{V}_r), \mathcal{T})$ and $\mathbf{v}_m^{neg} = f_m(f_{ve}(\mathcal{V}_r), \mathcal{C})$, where \mathbf{v}_m^{pos} and \mathbf{v}_m^{neg} denote the positive and negative contrast vectors, respectively, and f_m means the memory module. Finally, we feed \mathbf{v}_m^{pos} and \mathbf{v}_m^{neg} to the reward model f_r , which is a multi-layer perceptron, and compute the reward (denoted as r_{pos} and r_{neg} , respectively) for the vectors by $r_{pos} = \text{sigmoid}(f_r(\mathbf{v}_m^{pos}))$ and $r_{neg} = \text{sigmoid}(f_r(\mathbf{v}_m^{neg}))$, and compute the loss \mathcal{L}_r to optimize the reward model by

$$\mathcal{L}_r = -\log(r_{pos}) - \log(1 - r_{neg}) \quad (6.4)$$

Self-rejection Sampling In this step, we use the reward model to reject ineffective vectors and choose the best one \mathbf{m}_x^* to improve \mathbf{m}_x , which is similar to the process used in Touvron et al. (2023b). In doing so, we feed all memory sampled vectors $\mathbf{m}_x^1 \cdots \mathbf{m}_x^L$ to the reward model f_r and compute the reward for each of them, and select the vector \mathbf{m}_x^* with the highest reward score and use it as the gold standard to assess whether a sampled vector from the memory module is good enough to carry

	HMC			Memotion7k		MultiOFF			Memeplate		
	Train	Dev	Test	Train	Test	Train	Dev	Test	Train	Dev	Test
# of Meme	8,500	500	1,000	6,992	1,879	445	149	149	3,746	700	738
Avg. Tokens Per Meme	11.7	10.2	10.4	14.7	15.7	41.4	45.2	47.0	20.3	20.4	20.0

Table 6.1: Statistics of experiment datasets, where the number of meme and the average number of tokens (i.e., words for English and characters for Chinese) for each meme are reported.

essential task-specific contrast information for final classification. Finally, we compute the loss

$$\mathcal{L}_{SRT} = |\mathbf{m}_x^* - \mathbf{m}_x| \quad (6.5)$$

and add it to \mathcal{L}_{CE} to get the final loss $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{SRT}$ to update the entire framework.

6.1.2 Experiment Settings

Datasets

For our experiments, we employ three English datasets, namely, hateful meme challenge (HMC) dataset (Kiela et al., 2020), Memotion7K (Sharma et al., 2020), and MultiOFF (Suryawanshi et al., 2020), and one Chinese dataset named Memeplate (Li et al., 2022). These datasets cover a wide range of tasks that require modeling of contrasts. Specifically, HMC is for HMD. Memotion7k contains three tasks: sentiment classification (T1), humor classification (T2), and scales of semantic classes (T3). MultiOFF is designed for detecting offensive content from image-text pairs. Memeplate is for multimodal humor recognition. We use the official training, development, and test data split of all datasets. Herein, for HMC, we follow the convention of most existing studies (Radford et al., 2021; Singh et al., 2022; Cao et al., 2023b; Koutlis et al., 2023) to evaluate model performance on the development set. The statistics of the datasets are reported in Table 6.1, where the number of memes and the average number of tokens (i.e., words for English and characters for Chinese) for each meme are presented.

	HMC		Memotion7k			MultiOFF		Memeplate	
	ACC	AUROC	T1	T2	T3	Acc	F1	ACC	F1
GPT-2	73.28 \pm 0.20	83.01 \pm 0.22	35.09 \pm 0.24	47.58 \pm 0.20	32.10 \pm 0.19	68.32 \pm 0.24	61.67 \pm 0.23	53.37 \pm 0.19	47.09 \pm 0.23
+M	74.00 \pm 0.25	83.81 \pm 0.20	35.50 \pm 0.20	48.11 \pm 0.23	32.85 \pm 0.26	68.94 \pm 0.20	62.48 \pm 0.21	54.37 \pm 0.26	47.76 \pm 0.22
+RCR	74.52 \pm 0.22	84.56 \pm 0.20	36.12 \pm 0.19	48.64 \pm 0.19	33.48 \pm 0.23	69.53 \pm 0.21	63.05 \pm 0.21	55.10 \pm 0.26	48.86 \pm 0.24
+M+RCR	*75.08 \pm 0.23	*84.91 \pm 0.20	*38.06 \pm 0.21	*50.01 \pm 0.21	*34.57 \pm 0.22	*71.58 \pm 0.25	*64.02 \pm 0.19	*55.86 \pm 0.22	*49.75 \pm 0.25
LLM	76.20 \pm 0.26	84.44 \pm 0.23	37.48 \pm 0.21	49.75 \pm 0.21	33.76 \pm 0.23	71.51 \pm 0.20	64.87 \pm 0.23	54.38 \pm 0.25	47.97 \pm 0.22
+M	76.56 \pm 0.22	84.84 \pm 0.25	38.82 \pm 0.20	50.80 \pm 0.23	34.83 \pm 0.25	72.11 \pm 0.20	65.94 \pm 0.23	55.12 \pm 0.21	48.70 \pm 0.28
+RCR	77.01 \pm 0.22	85.40 \pm 0.25	40.82 \pm 0.23	51.40 \pm 0.22	35.61 \pm 0.25	73.18 \pm 0.26	67.72 \pm 0.22	55.86 \pm 0.23	49.39 \pm 0.20
+M+RCR	*77.88 \pm 0.24	*86.34 \pm 0.23	*41.56 \pm 0.21	*52.73 \pm 0.21	*35.88 \pm 0.24	*74.09 \pm 0.20	*68.43 \pm 0.21	*56.52 \pm 0.20	*50.21 \pm 0.21

Table 6.2: The average and standard deviation of the performance from various models on benchmark datasets. “GPT-2” and “LLM” stand for BLIP2 baseline models use small and large language models, respectively. “+M” and “+RCR” refer to that the memory module and the RCR are used on top of the baselines, respectively. Results marked by * means that the improvements are statistically significant at $p \leq 0.05$ level over all baselines.

Baselines

We run baselines with small language models and LLMs as the backbones following the BLIP2 (Li et al., 2023b) architecture. For small language models, we employ GPT-2 (Radford et al., 2019). For LLMs, we use MiniGPT-4 (which is demonstrated to be effective in many multimodal tasks) for English and use Ziya-BLIP2-Visual (Zhang et al., 2022a) for Chinese processing. Based on small and large models, our experiments include three baselines. The first is the vanilla BLIP2 with small and large language models. The second baseline (i.e., “+M”) adds the proposed memory module on top of the first one. The third baseline (i.e., “+RCR”) adds reinforced contrast recognition (RCR) on top of the first one. We concatenate visual and text features and use the resulting vector (i.e., \mathbf{x}_{vt}) to represent the contrast vector (i.e., \mathbf{m}_x) and randomly set 33% values in \mathbf{m}_x to zero to facilitate RCR training.

Implementation Details

We use the default settings of BLIP2 (with GPT-2), MiniGPT-4, or Ziya-BLIP2-Visual, which contain visual encoding and LLM decoding processes. For the visual encoding process, we follow the standard architecture using visual transformer and Q-Former, which contain 40 and 12 layers of multi-head attentions, respectively. For the LLM decoding process, the LLMs in BLIP2 (with GPT-2), MiniGPT-4, and Ziya-BLIP2-Visual utilize 12, 32, and 40 layers of Transformers, respectively.

In training our approach, we alternate between the following two procedures for every 100 steps: (1) updating the parameters of different components in visual encoding, memory module, and LLM using the cross-entropy loss from comparing the predicted labels with gold standards and (2) updating the reward model and the memory module through RCR.² For evaluation, we follow existing studies (Kielbaso et al., 2020; Li et al., 2022; Cao et al., 2023b; Koutlis et al., 2023) to use accuracy and AUROC for HMC, accuracy and F1 for MultiOFF, F1 for Memotion7K, and accuracy and F1 for Memeplate. For the hyper-parameters, we set the numbers of memory vectors (i.e., N) to 200 for HMC and 150 for other datasets. For all datasets, we use 20 as the memory sampling size (i.e., K), and 4 as the sampling time L . We set learning rate to 1×10^{-6} with a batch size of 32. For other hyper-parameters, we tune them on the development set³ and select the ones with the best performance to train models and evaluate them on the test sets. We run all models five times using different random seeds and report the average and standard deviation of their performance.

6.1.3 Results and Analysis

Overall Performance

The average performance with standard deviations of baselines and our approach for all datasets under different settings are reported in Table 6.2, with following observations. First, overall, our approach (i.e., +M+RCR) outperforms the vanilla BLIP2 (GPT-2), MiniGPT-4, and Ziya-BLIP2-Visual baselines, which indicates the effectiveness of our approach to learning contrast information for different tasks. Second, when the memory (i.e., “+M”) or the RCR module (i.e., “+RCR”) is added to

²For the third baseline with RCR (i.e., +RCR), we update the parameters of visual encoding and the token embeddings of the input text during training, so as to appropriately work with the absence of the memory module.

³For HMC, we randomly select 10% of the training data and use it to tune hyper-parameters.

	ACC	AUROC
Muennighoff (2020)	-	81.56
Velioglu and Rose (2020)	70.93	75.21
Lippe et al. (2020)	-	77.39
Radford et al. (2021)	-	77.30
Goyal et al. (2022)	-	73.40
Kumar and Nandakumar (2022)	-	81.55
Singh et al. (2022)	-	76.70
Cao et al. (2023a)	72.28	80.87
Koutlis et al. (2023)	73.60	80.10
Cao et al. (2023b)	72.98	82.45
†△Liu et al. (2023)	76.20	84.57
Ours	77.88	86.34

Table 6.3: Comparison of the average performance of our approach with the existing studies on the development set of HMC. “†” means the results are our own runs using their multimodal approaches. “△” indicates that LLMs are used to predict labels. The markups are the same for following tables.

the vanilla baseline, improvements are all observed, which is the evidence for the effectiveness of each individual module in capturing contrast between visual and textual data, thereby enhancing model performance.⁴ Third, when comparing “+M” and “+RCR”, we find that RCR consistently exhibits superior performance across various configurations, underscoring the advantage of discriminatively learning the contrast information. Fourth, our full model that integrates both the memory and RCR outperforms all baseline models, demonstrating the effectiveness of complementing each other.

We further compare our approach with existing studies for HMC, Memotion7K, MultiOFF, and Memeplate in Table 6.3-6.6, where the results demonstrate state-of-the-art performance. Particularly, our approach outperforms the ones that use advanced pre-trained models for image and text processing (Kumar and Nandakumar, 2022; Singh et al., 2022; Koutlis et al., 2023). The reason is that, these

⁴With the training data from particular tasks that contains such contrast, the memory module is also able to learn that information as that performed in the RCR process.

	T1-F1	T2-F1	T3-F1
Keswani et al. (2020)	35.5	-	-
Vlad et al. (2020)	34.5	51.8	31.7
Guo et al. (2020)	35.2	51.5	32.3
Kumari et al. (2021)	36.8	-	-
†Ouaari et al. (2022)	35.3	-	-
Zhang et al. (2022b)	36.6	46.9	-
Zhong et al. (2022)	37.0	-	-
Koutlis et al. (2023)	39.6	51.9	34.3
Ours	41.56	52.73	35.88

Table 6.4: Performance comparison of different models on the test set of three tasks on Memotion7k dataset.

multimodal models generally perform the HMD, multimodal sentiment analysis, offensive content detection, and humor recognition in the same way as image captioning, therefore not focus on the contrast across modalities whereas image captioning emphasizes the content shared by these modalities. Compared with these studies, the performance of our approach on all tasks confirm the validity of explicitly learning contrast rather than shared semantics.

Effect of the Memory Module

Since the memory module serves as a key component that records essential multimodal features and the pivot receiving optimized signal from RCR, it is of great importance to investigate its effect on model performance. Specifically, we explore the effect of the number of memory vectors N and run LLM-based models on HMC and Memeplate datasets⁵. The performance (y-axis) of models with respect to the value of N (x-axis) is illustrated in Figure 6.3. There are several observations. First, for both datasets, when the value of N is small, increasing its value brings significant enhancement to model performance. This observation is intuitive in that more memory

⁵We select the two representative datasets for different languages and tasks, one for English HMD and the other for Chinese humor recognition.

	ACC	F1
Lee et al. (2021)	-	64.6
Zhong et al. (2022)	-	67.1
Koutlis et al. (2023)	68.5	62.5
Ours	74.09	68.43

Table 6.5: Comparison of different models on the test set of MultiOFF dataset.

	ACC	F1
†Yang et al. (2022)	52.57	46.21
†△Yang et al. (2023)	55.43	48.80
†△Hu et al. (2023)	55.08	48.97
†△University (2023)	55.76	49.49
Ours	56.52	50.21

Table 6.6: Performance comparison of different models on the test set of Memeplate dataset.

vectors provide a larger parameter space to comprehensively accommodate enough information between multiple modalities and thus lead to better performance. Second, when the value of N is high, the performance improvement brought by the increase of N is moderate. This indicates that when the number of memory vectors reaches a certain point, no more useful contrast information for the task is leveraged and thus results in less improvements.

In addition, we investigate the effect of the memory module when it works with RCR by replacing the memory module with other widely used architectures, namely, outer product operation (OP) and co-attention (Co-Att) (Lu et al., 2016). Specifically, for OP, we firstly obtain the visual feature \mathbf{v} and text features \mathbf{t} using the same process as our approach. Then, we compute the outer product of \mathbf{v} and \mathbf{t} , and flatten the resulting matrix into a vector to represent the contrast vector \mathbf{m}_x . For Co-Att, we apply co-attention to fuse \mathbf{v} and \mathbf{t} and regard the output as \mathbf{m}_x . The results for different datasets are reported in Table 6.7. It is observed that their performance is worse than the performance of our approach with memory, which confirms the effectiveness of our approach in leveraging the memory

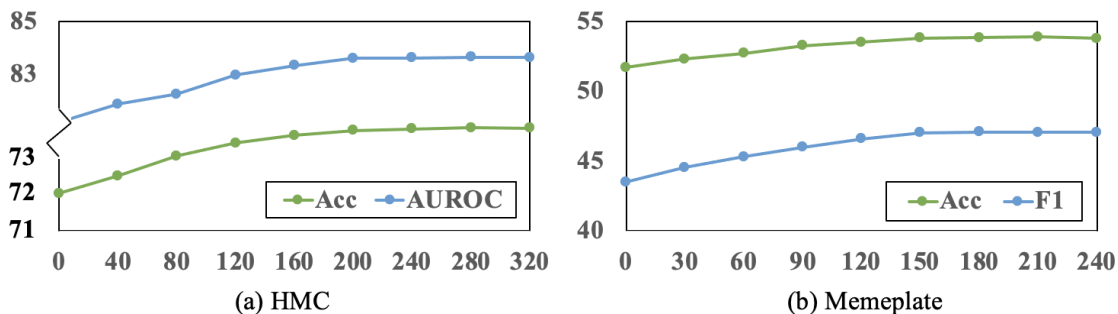


Figure 6.3: Curves of model performance on HMC and Memeplate with respect to different numbers of memory vectors used in the memory module.

	HMC		Memotion7k			MultiOFF		Memeplate	
	ACC	AUROC	T1-F1	T2-F1	T3-F1	ACC	F1	ACC	F1
OP	76.44 \pm 0.18	84.86 \pm 0.22	39.42 \pm 0.22	50.67 \pm 0.21	34.68 \pm 0.22	71.97 \pm 0.20	65.90 \pm 0.18	55.11 \pm 0.18	48.62 \pm 0.22
Co-Att	76.63 \pm 0.21	84.98 \pm 0.22	39.67 \pm 0.20	50.93 \pm 0.22	34.77 \pm 0.21	72.20 \pm 0.22	66.12 \pm 0.24	55.21 \pm 0.20	48.89 \pm 0.22
M	*77.88\pm0.24	*86.34\pm0.23	*41.56\pm0.21	*52.73\pm0.21	*35.88\pm0.24	*74.09\pm0.20	*68.43\pm0.21	*56.52\pm0.20	*50.21\pm0.21

Table 6.7: Experiment results of different models using LLMs and RCR, where the memory module in our approach is replaced by two widely used approaches for multimodal feature fusion, namely, outer product operation (OP) and Co-attention mechanism (Co-Att). The performance of memory (M) with RCR is also presented for reference.

and RCR for multimodal classification tasks that require contrast information modeling.

Case Study

We also investigate three similar memes for qualitative analysis for multimodal sentiment analysis. The images and texts with their predictions from different models, as well as the gold standard, are illustrated in Figure 6.4, where (a) and (b) have the same texts; (a) and (c) use the same image. As a result, (a) conveys a negative sentiment polarity, while (b) and (c) have positive polarities. The predictions that match and do not match the gold standard are highlighted in green and red colors, respectively. By investigating the results, we observe that the three baselines struggle to



Figure 6.4: Demonstration of three memes for multimodal sentiment analysis with the polarities predicted by different models. The gold standard sentiment labels of all memes are also presented. The model produced labels that match the gold standard and the labels that do not match are highlighted in green and red colors, respectively.

predict sentiment labels that match the gold standards for all memes, whereas our approach is able to accurately identify sentiment polarities of all memes. A possible reason is the following. Negative sentiment polarities are generally derived from the contrast between multiple modalities. The baselines have limitations that prevent them from learning such contrast, either lacking a particular mechanism to do so or being equipped without effective guidance. In contrast, RCR or Memory+RCR provide enough information to learn such contrast and thus help our approach to correctly analyze its sentiment polarity.

6.2 Medical Large Language Modeling

NLP plays an important role in improving medical intelligence by processing text data that serves as the main medium for information analysis. PLMs serve as the basis for many NLP tasks and achieve state-of-the-art performance (Song et al., 2018b,c; Devlin et al., 2019; Yang et al., 2019; Diao et al., 2020; Lewis et al., 2020a; Raffel et al., 2020; Brown et al., 2020; Song et al., 2021). These approaches

Question

我怀孕37周，肠子一直呼噜呼噜叫感觉像是在里面灌水，上厕所拉稀和喷水一样，一天上厕所5次，对孩子有啥影响，应该怎么办？回答要简单明了。

I am 37 weeks pregnant, and my intestines are constantly grunting, and it feels like I am filling it with water. Going to the toilet has loose stools and sprays water. I go to the toilet 5 times a day. How will it affect the baby? What should I do?

GPT-4

很抱歉听到您的不适。您描述的症状可能是肠胃道感染或其他肠道问题。简要地说：对孩子的影响：轻度的腹泻通常不会直接伤害到胎儿，但持续的腹泻可能导致脱水，这对您和孩子都不好。脱水可能会导致宫缩，这可能早产。应该怎么办：马上咨询医生或者去医院。确保饮用足够的水分以防止脱水。避免摄入任何可能导致腹泻的食物或饮料。建议您尽快咨询医生以得到专业的医疗建议。

I'm sorry to hear about your discomfort. The symptoms you described might be related to a gastrointestinal infection or other intestinal issues. In brief: Impact on the child: Mild diarrhea usually won't directly harm the fetus, but persistent diarrhea might lead to dehydration, which isn't good for both you and the child. Dehydration can cause contractions, which might result in preterm labor. What to do: Consult a doctor immediately or go to the hospital. Ensure you drink enough fluids to prevent dehydration. Avoid consuming any foods or drinks that might cause diarrhea. It's recommended that you seek medical advice from a professional as soon as possible.

ChiMed-GPT

考虑是腹泻现象，容易诱发子宫收缩出现早产症状，可以在医生的指导下服用黄连素，思密达等治疗，同时注意多喝水，饮食要清淡，禁食生冷刺激食物。

It is considered to be diarrhea, which can easily induce uterine contractions and cause symptoms of premature labor. You can take berberine, Smecta and other treatments under the guidance of a doctor. At the same time, pay attention to drinking more water, eating a light diet, and fasting away from raw and cold stimulating foods.

Table 6.8: An example shows that the general domain GPT-4 fails to provide a satisfying answer to a medical inquiry, compared with the proposed ChiMed-GPT in this thesis that provides an informative and authentic response. Note that English translations are not a part of the input and output, where they are given for better illustration.

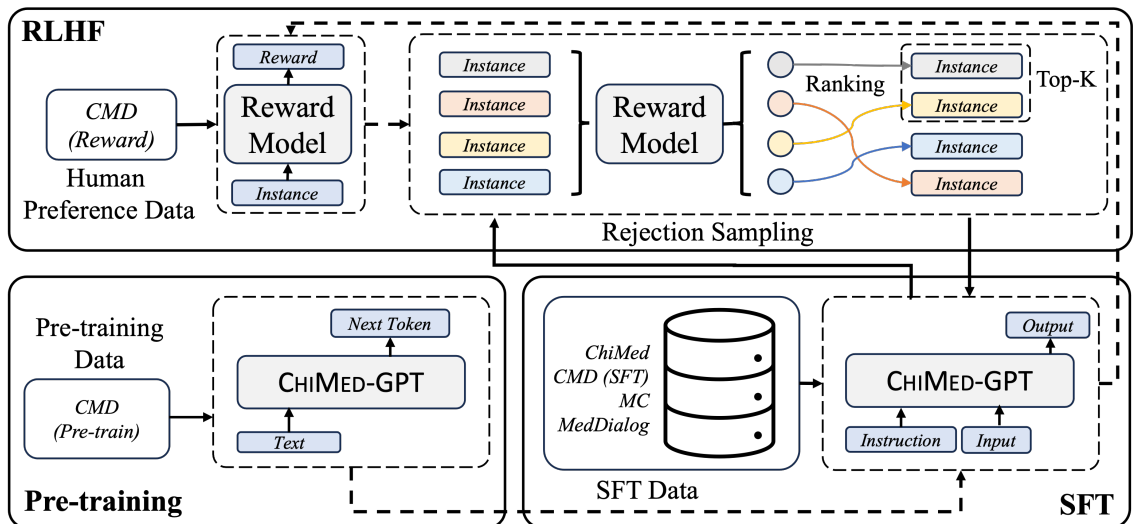


Figure 6.5: An illustration of the overall training process of the ChiMed-GPT, which consists of three stages, including pre-training, supervised fine-tuning, and reinforcement learning from human feedback (RLHF).

use a pre-training and fine-tuning paradigm that requires a large volume of labeled medical data and restricts the models to specific tasks, which limits their application to new scenarios. The emergence of LLMs such as GPT-3.5, GPT-4 (OpenAI, 2023), and open-sourced alternatives including Alpaca (Taori et al., 2023a), Vicuna (Chiang et al., 2023), and Ziya (Zhang et al., 2022a) initiates a paradigm shift by offering unified solutions for various tasks. However, most high-performing LLMs receive training on general domain data and struggle in specialized fields such as medicine. One example in Figure 6.8 shows that GPT-4 provides an unsatisfactory response to a healthcare question, while a model trained in the medical field delivers better answers. Researchers develop medical LLMs such as BioMedLM (Venigalla et al., 2022), BioGPT (Luo et al., 2022), MedAlpaca (Han et al., 2023), ChatDoctor (Li et al., 2023c), Baize-healthcare (Xu et al., 2023a), Med-PaLM-2 (Singhal et al., 2023), to reduce the knowledge gap. Many of these models remain closed source, and publicly available ones face challenges in data diversity, training procedures, and model configuration.

In this thesis, we propose ChiMed-GPT, a benchmark LLM for Chinese medical text processing. We train a general model, Ziya-13B-v2 (Gan et al., 2023), using extensive medical data and a complete training process that includes pre-training, supervised fine-tuning (SFT), and reinforcement learning

Stage	Dataset	# of Instances	# of Tokens	Storage Size
Pre-training	CMD (Pre-train)	369,800	214M	603MB
SFT	ChiMed	200,744	84M	252MB
	CMD (SFT)	1,015,000	460M	1,400MB
	MC	44,983	17M	50MB
	MedDialog	9,060	3M	9MB
Rejection Sampling	CMD (Reward)	4,000	1M	3MB

Table 6.9: Statistics of the dataset used in training ChiMed-GPT under different stages. Note that the term “instance” refers to documents for CMD (Pre-train), QA pairs for ChiMed, CMD (SFT), MC, and dialogue cases for MedDialog. Differently for CMD (Reward), each instance is a (question, accepted answer, rejected answer) tuple.

from human feedback (RLHF). Our approach applies data augmentation to produce high-quality human preference data and uses rejection sampling fine-tuning for efficient learning. The training data comes from medical articles and doctor-patient interactions. We integrate safety data to yield appropriate responses. The model performs well on information extraction, question answering, and dialogue generation.

6.2.1 The Approach

ChiMed-GPT is built on Ziya-13B-v2 (Gan et al., 2023), whose architecture follows the standard Transformer (Vaswani et al., 2017) decoder with 13 billion parameters and is pre-trained on 600 billion Chinese and English tokens, guaranteeing a superior Chinese text processing capability. Particularly, Ziya-13B-v2 also extends the context length to 4,096 tokens, which enables our requirement for better context processing with ChiMed-GPT. The overall architecture and training procedure of ChiMed-GPT are illustrated in Figure 6.5, which consists of three stages, namely, pre-training, SFT, and RLHF, where training details (i.e., datasets and implementation) for each stage are illustrated in the following subsections.

Pre-training

We adopt the pre-training subset of the Chinese Medical Dataset (CMD)⁶ (Xu, 2023) to continually pre-train Ziya-13-v2 for ChiMed-GPT, with its statistics presented in Table 6.9. This subset encompasses two parts, where the first comprises a total of 369,800 documents originating from medical encyclopedia data, while the second includes 8,475 articles sourced from medical textbooks, corresponding to 214 million tokens. Given its rich medical content, CMD (pre-train) proves highly appropriate for pre-training models. For implementation, our pre-training follows the standard paradigm, where the objective is to predict the next token of the input text based on the existing input history. Following existing studies (Radford et al., 2019), we use byte-pair encoding (BPE) (Sennrich et al., 2015) as the tokenizer and use the same vocabulary as that used in Ziya-13B-v2. AdamW (Loshchilov and Hutter, 2017) is adopted as the optimizer with its hyper-parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.95$. The initial learning rate is set to 5×10^{-5} with a weight decay of 0.1 and gradient clipping of 1.0. We utilize the framework of Megatron-LM (Shoeybi et al., 2019) to perform distributed training with the number of tensor parallelisms set to two for efficient training. We also utilize other efficient training techniques, including BF16 mixed-precision training (Micikevicius et al., 2017), ZeRO (Rajbhandari et al., 2020), and flash-attention (Dao et al., 2022) to optimize GPU memory cost during training.

Supervised Fine-tuning

Recent studies have underscored the critical role of SFT in shaping the intelligence capabilities of LLMs (Ouyang et al., 2022; Zhang et al., 2022a; Taori et al., 2023a; Chiang et al., 2023), where the quality and diversity of SFT data hold paramount importance in this process (Touvron et al., 2023a). To enhance the model’s capability to understand human instructions (e.g., asking for healthcare suggestions) in real-world medical settings, we also have a strong SFT process that utilizes QA and dialogue data (including ChiMed (Tian et al., 2019), CMD (SFT), Chinese medical dialogue dataset (MC) (Song et al., 2020), and MedDialog) between doctors and patients, whose statistics are reported in Table 6.9. Note that we preprocess all data to remove redundancy as well as personal information in them to address privacy concerns.

⁶<https://huggingface.co/datasets/shibing624/medical>

With these datasets, the standard SFT approach is employed to optimize our model, where prompt-response pairs are constructed from these data for effective training. In detail, for question-answer data, we directly utilize the question as the prompt and the corresponding answer as the response; for dialogue data, we combine the dialogue history and the patient’s latest utterance as the prompt and regard the answering utterance from the doctor as the response. Example prompts and responses are illustrated in Table 6.10. We also adopt a special dataset named Safety-Prompts (Sun et al., 2023) that consists of 100K desired and appropriate responses (e.g., refuse-to-answer instances) to toxic prompts, which allows LLMs to learn how to correctly address harmful input. In SFT, we train ChiMed-GPT on the union of all aforementioned datasets, where we feed prompts into ChiMed-GPT, compute the cross-entropy loss by comparing its produced responses with gold standard annotations, and then perform full-parameter updating with the loss. For efficient SFT, we follow existing studies to concatenate short prompt-response pairs to form a long input text to better utilize the maximum sequence length of ChiMed-GPT, where the boundaries of different pairs are marked by special tokens. For hyper-parameters, we set the learning rate and the weight decay to 2×10^{-5} and 0.1, respectively, and use a batch size of 16.

RLHF

We perform RLHF through rejection sampling following Llama-2, with two steps: reward model training and rejection sampling fine-tuning.

For reward model training, we adopt CMD (Reward), the reward subset of CMD, as the dataset in learning the reward model, with its statistics illustrated in Table 6.9. Specifically, CMD (Reward) comprises 4K instances, which are split into train, validation, and test sets containing 3,800, 100, and 100 instances, respectively. Each instance has a question sampled from the CMD (SFT) dataset accompanied with one accepted and one rejected answer, where the accepted answer is provided by a doctor and the rejected one is produced by a Chinese medical LLM named BenTsao. Different from previous studies (Xu, 2023), we make further efforts to augment CMD (Reward) with two additional intermediate responses extracted from GPT-4 and GPT-3.5-Turbo⁷ rather than directly apply it to train our reward model, which therefore should better align with human preference so

⁷We feed the prompt in each instance from CMD (Reward) to GPT-4 and GPT-3.5-Turbo and obtain their responses and add them back into the original instance accordingly.

Prompt

假设你是一名医生，请回答患者的问题。

我想问如何避免牙龈肿痛。我最近天天吃的火锅，可能上火了，牙龈又肿又痛，最后一颗牙齿上长了很大一个泡，不能吃东西，不敢张嘴，张开嘴巴就痛，还总流口水。我在药店买了消炎药在吃，效果不是很明显。

Suppose you are a doctor and answer a patient's question.

I would like to ask how to avoid swollen gums. The hot pot I ate every day recently may have caught fire. My gums were swollen and painful. I had a big blister on my last tooth. I couldn't eat. I bought anti-inflammatory drugs at the drugstore and took them, but the effect was not obvious.

Response

你好。牙龈肿痛的原因很多，一般都是牙周炎症引起的，牙周炎需要及时治疗，以免造成牙龈萎缩，牙齿过早脱落。如果肿痛厉害建议去医院做个检查，祝你健康。

Hello. There are many reasons for gingival swelling and pain, generally caused by periodontal inflammation. Periodontitis needs to be treated in time to avoid gingival recession and premature tooth loss. If the swelling and pain are severe, it is recommended to go to the hospital for an examination. I wish you health.

Table 6.10: An example medical prompt and its response, where English translations are given for better illustration.

that it further minimizes the risk of ChiMed-GPT generating inappropriate contents in addition to the Safety-Prompts used in SFT. With the four responses, we re-rank them in the following order: positive answer, GPT-4's response, GPT-3.5-Turbo's response, and negative answer, by assuming that the answer from the doctor is the best alignment with human preference; GPT-4 is able to generate better content than GPT-3.5-Turbo; and then the last choice from other LLMs. Subsequently, we pair every two adjacent responses in the rank list and regard the one with a higher rank as a positive sample and the other as a negative sample, with the reward model trained on each positive-negative sample pair. For hyper-parameters, we train the reward model for two epochs with the batch size set to 8. The learning rate follows a cosine schedule, gradually decreasing to 10% from 5×10^{-6} . We employ a warm-up stage that encompasses 3% of the total step count, with a minimum threshold of 5 steps.

Once the reward model is obtained, rejection sampling fine-tuning aligns the model output with

Models	CCKS-2019	ChiMST
GPT-3.5-Turbo	31.42	32.15
GPT-4	41.37	41.25
Ziya-v1	25.31	22.26
Ziya-v2	27.84	25.76
Baichuan	24.14	21.20
Taiyi	30.90	30.55
MedicalGPT (Z)	29.59	28.12
MedicalGPT (B)	23.80	26.16
ChiMed-GPT	40.82	41.04

Table 6.11: The F1 scores of different models on the information extraction (NER) task w.r.t two datasets under five-shot setting. MedicalGPT (Z) and MedicalGPT (B) denote different versions of MedicalGPT that use Ziya-v1 and Baichuan as the base model, respectively. Boldface is added to the results from the best-performing open-source LLMs.

human preferences through the following procedure. We firstly randomly sample 10K prompts from the SFT data and feed them to our ChiMed-GPT. Then we employ the reward model to assign scores to the outputs generated from the last step. Afterwards, we rank the texts produced by the LLM based on their scores and select the top-k responses, which are regarded as gold standards to further fine-tune our LLM. When learning through rejection sampling, we apply the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-5}$. We employ a learning rate of 10^{-5} with a 0.1 weight decay and perform rejection sampling fine-tuning for 400 iterations with batch size set to 64. Following the same settings in pre-training, we perform distributed training and related efficient tuning techniques to optimize the process of both reward model training and rejection sampling fine-tuning.

Models	C-Eval	CMMLU	MedQA	ChiMed				
	Acc.	Acc.	Acc.	B-1	B-2	R-1	R-2	R-L
GPT-3.5-Turbo	56.58	49.91	44.50	33.61	28.27	26.51	7.13	16.63
GPT-4	71.29	69.55	67.99	39.15	32.85	26.61	7.31	16.84
Ziya-v1	36.59	29.07	12.50	6.18	5.77	18.59	3.94	12.66
Ziya-v2	39.02	49.06	13.00	38.41	31.90	26.91	7.90	18.67
Baichuan	41.46	45.28	13.00	5.81	5.25	16.91	3.01	11.30
Taiyi	48.78	45.20	39.20	11.73	9.96	21.76	5.26	15.46
MedicalGPT (Z)	48.78	34.56	25.99	39.02	32.35	26.76	8.10	18.16
MedicalGPT (B)	39.02	43.82	18.50	5.82	5.26	16.61	2.94	11.11
ChiMed-GPT	68.29	52.92	44.50	44.58	37.22	27.11	8.89	19.86

Table 6.12: Performance comparison of different LLMs on multi-choice and open-ended QA datasets, where we only use medical-related subsets for C-Eval and CMMLU, and the Chinese subset for MedQA. We run five-shot settings on C-Eval, CMMLU, and MedQA, and zero-shot settings on ChiMed. “Acc”, “B”, and “R” are abbreviations denoting accuracy, BLEU, and ROUGE, respectively.

6.2.2 Results and Analysis

Following existing studies, we evaluate ChiMed-GPT under zero- and few-shot settings, and report the results on three tasks, namely, information extraction, QA, and multi-turn dialogues.⁸ In the zero-shot setting, we prompt LLMs with a description of the task and a test instance; in the few-shot setting, we add five task instances with gold standard labels to the prompt, which are inserted between the description and the test instance to guide the evaluation. We compare ChiMed-GPT with baselines from the general and medical domains, including GPT-3.5-Turbo, GPT-4, Ziya-v1, Ziya-v2, Baichuan, as well as Chinese medical LLMs Taiyi⁹ and MedicalGPT.¹⁰ Herein, GPT-3.5-Turbo and

⁸For every experiment, we run it five times and report the average performance.

⁹<https://github.com/DUTIR-BioNLP/Taiyi-LLM>.

¹⁰We only select representative Chinese medical LLMs in our comparison on benchmark datasets in the evaluation tasks.

Question

最近总熬夜，请问如何避免牙龈肿痛？ *I've been staying up late lately. How can I avoid swollen gums?*

BenTsao

你好，避免牙龈炎需要做好个人卫生护理。

Hello, to avoid gingivitis, you need to take good personal hygiene care.

MedicalGPT (Z)

牙龈肿痛可能是由于牙龈炎或牙周病引起的，预防措施包括：保持口腔卫生，定期刷牙和漱口，使用牙线清除牙缝间的残留物，戒烟限酒，控制血糖和血压，保持充足的睡眠和休息，定期去医院检查牙齿。

Swollen and sore gums may be caused by gingivitis or periodontal disease. Preventive measures include: maintaining oral hygiene, brushing and rinsing regularly, using dental floss to remove residue between teeth, quitting smoking and limiting alcohol, controlling blood sugar and blood pressure, maintaining Get enough sleep and rest, and go to the hospital regularly to check your teeth.

MedicalGPT (B)

你好，你这是慢性牙周炎的表现，是由于平时口腔卫生不好，细菌滋生导致的，建议平时注意口腔卫生，不吃辛辣刺激食物，戒烟酒，定期洗牙。

Hello, this is a symptom of chronic periodontitis, which is caused by poor oral hygiene and the growth of bacteria. It is recommended to pay attention to oral hygiene, avoid eating spicy food, quit smoking and drinking, and clean your teeth regularly.

ChiMed-GPT

由于经常熬夜，牙齿没有得到充分休息，导致牙龈受到经常性刺激，促使牙龈出现疼痛。因此，在平时生活中，要保持牙齿清洁，定期到医院进行洗牙，先预防出现牙龈感染。

Because you often stay up late, your teeth do not get adequate rest, which causes constant irritation of the gums, causing gum pain. Therefore, in daily life, you must keep your teeth clean and go to the hospital for regular teeth cleaning to prevent gum infections in the first place.

Table 6.13: Answers produced by different models, followed with English translations.

GPT-4 are state-of-the-art general domain LLMs that are accessible by OpenAI API; Ziya-v1 is an open-sourced Chinese general LLM that achieves outstanding performance on many NLP tasks and some domain-specific LLMs are developed upon it (e.g., MedicalGPT). Ziya-v2 is an upgraded version with larger training data and context length, on which our ChiMed-GPT is built. Baichuan is another Chinese general LLM serving as a foundation model for various SFT-based other LLMs. Taiyi is a medical domain LLM obtained by continually supervised fine-tuning a general domain LLM on Chinese and English medical data. MedicalGPT has two versions based on Ziya-v1 and Baichuan (marked by “Z” and “B”, respectively), which are both tested in our experiments.¹¹

In addition to the standard metrics (e.g., BLEU and accuracy), we run human evaluations on question answering and dialogue generation. We randomly select 50 answers and dialogues generated by different models and ask two annotators to assess their quality from three aspects: *fluency*, *completeness*, and *precision*. Specifically, *fluency* judges whether the answer to the patient’s question is simple and clear. We use this metric for the reason that good answers have to be understandable by the patients and they should assist an efficient medical consultation process. *Completeness* refers to whether the answer includes comprehensive information for the patient’s inquiry. We use this metric to evaluate if the patient should be able to learn enough information about the disease through the output text. *Precision* denotes whether the answer is accurate. We use this metric because medical issues are directly related to the patient’s physical and mental health, which requires precise descriptions in real scenarios. The assessment scores for each aspect range from 1 to 3, with higher scores referring to better quality. The results of human evaluation in different tasks are presented in the following subsections.

Information Extraction

Information extraction is an important task for medical text processing, where named entity recognition (NER) serves as a representative one that has attracted much attention for decades in this domain. We evaluate baselines and our approach on two benchmark datasets for NER, namely, CCKS-2019 (Han et al., 2020), and ChiMST (Tian et al., 2022a). All models are performed under five-shot settings, where the F1 scores on the test sets are reported in Table 6.11. It is observed that ChiMed-GPT

¹¹We use the pre-trained version of Ziya-v1, Ziya-v2, and Baichuan, and follow the convention (Hendrycks et al., 2020; Huang et al., 2023) to compare LLMs with various sizes.

	Flu	Comp	Pre
Taiyi	2.17	2.02	2.01
Medical-GPT (Z)	2.30	2.10	2.13
Medical-GPT (B)	2.27	2.17	2.22
ChiMed-GPT	2.57	2.45	2.57

Table 6.14: Human evaluation results from different models on 50 randomly selected QA pairs. “Flu”, “Comp”, and “Pre” stand for fluency, completeness, and precision, respectively. The range of score is [1, 3].

outperforms all general and medical domain LLMs, including GPT-3.5-Turbo, demonstrating the necessity of training LLMs for the medical domain with careful design and appropriate data, which ensures the superiority of ChiMed-GPT over other LLMs.

Question Answering

We also compare ChiMed-GPT with other LLMs on multi-choice and open-ended QA. For multi-choice QA, we test LLMs on medical-related subsets of C-Eval (Huang et al., 2023) and CMMLU (Li et al., 2023a), as well as the Chinese subsets of MedQA (Jin et al., 2021),¹² where we run LLMs with the five-shot setting and use accuracy to measure their performance. For open-ended QA, we evaluate different LLMs on the test set of ChiMed with the zero-shot setting and use BLEU-1 (B-1), BLEU-2 (B-2), ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) as evaluation metrics. Results on the aforementioned two tasks are reported in Table 6.12, where ChiMed-GPT outperforms all baselines and achieves state-of-the-art performance. We also report example answers generated by different models in Table 6.13, where ChiMed-GPT is able to provide an informative and personalized answer to a patient’s question. Moreover, we report human evaluation results of different models in Table 6.14, where ChiMed-GPT achieves the best performance. Therefore, it is confirmed on the

¹²Specifically, for C-Eval, we use its “basic medicine” and “clinical medicine” subsets; for CMMLU, we employ “anatomy”, “clinical knowledge”, “college medicine”, “genetics”, “nutrition”, “traditional Chinese medicine”, and “virology” subsets; for MedQA, we utilize the “MCML” subset.

Models	B-1	B-2	R-1	R-2	R-L
GPT-3.5-Turbo	18.58	15.76	18.92	6.62	14.55
GPT-4	24.29	20.17	20.64	8.39	17.14
Ziya-v1	15.85	11.75	9.92	3.04	9.02
Ziya-v2	14.21	10.99	12.20	4.45	10.61
Baichuan	3.44	1.61	3.87	0.34	3.49
Taiyi	5.81	4.67	14.23	4.55	11.99
MedicalGPT (Z)	20.26	16.42	17.51	5.42	14.21
MedicalGPT (B)	3.94	2.19	4.34	0.13	3.50
ChiMed-GPT	33.14	30.86	43.43	34.91	42.16

Table 6.15: BLEU (B) and ROUGE (R) scores of different models on medical multi-turn dialogue generation.

superiority of ChiMed-GPT over baseline LLMs, which is attributed to the comprehensive training over appropriate data that enables ChiMed-GPT to effectively learn both medical knowledge and human preference.

Multi-turn Dialogue

Another evaluation that LLMs normally perform is dialogue response generation. We evaluate different LLMs on MC dataset for multi-turn dialogues in generating responses with given dialogue histories. We test LLMs under the zero-shot setting and report their BLEU and ROUGE scores in Table 6.15, where utterances generated by different LLMs with the same dialogue history are presented in Table 6.16. We also report the human evaluation results of different models in Table 6.17. From the results and examples from different LLMs, we observe that ChiMed-GPT achieves the best performance and is able to generate more comprehensive and appropriate responses, which further illustrates its practical value of being applied to real-world applications (e.g., online consultation platform) for interacting with patients.

Dialog Histroy

P: 我今天有点头晕,一直咳嗽。 *I feel a little dizzy today and keep coughing.*

D: 发烧了吗? *Do you have a fever?*

P: 体温37.9度 *37 degree*

ChiMed-GPT

一直咳嗽多久? 有没有痰?

How long have you been coughing? Is there phlegm?

Table 6.16: Example responses generated by ChiMed-GPT with the dialogue history between doctors and patients. Note that English translations are given only for reference and are not a part of input or output.

6.2.3 Bias Analysis

LLMs have been shown to reproduce and amplify biases present in the training data and to generate potentially toxic or offensive content (Cao et al., 2022; Feng et al., 2023; Cheng et al., 2023; Xu et al., 2023b; Haller et al., 2023). Particularly for medical applications, LLMs are expected to be carefully examined for potential biases, which are critical in affecting the psychological state of patients when they occur along with the text produced from LLMs. Therefore, it becomes essential to thoroughly investigate and understand any biases that may manifest in these models when they are utilized to assist healthcare professionals or directly provide information to patients. In evaluating the potential bias of LLMs, we choose mental disease statements as test cases to assess if appropriate responses are provided towards particular inputs.¹³ In doing so, we utilize two scales, namely, community attitudes toward the mentally ill (CAMI) (Taylor and Dear, 1981) and mental illness: clinicians' attitudes (MICA) (Gabbidon et al., 2013), whose details are elaborated as follows:

- **CAMI** scale aims to measure public attitudes towards individuals suffering from mental disorders. The scale comprises 40 statements related to these patients. For every statement, participants are required to specify their agreement level, namely, “*strongly disagree*”, “*disagree*”, “*neutral*”,

¹³The reason we choose mental disease is that many people suffer from them and experience stigma and discrimination based on the statistics from the World Health Organization (WHO), as stated in <https://www.who.int/news-room/fact-sheets/detail/mental-disorders>.

	Flu	Comp	Pre
Taiyi	1.96	2.01	2.02
Medical-GPT (Z)	2.09	2.05	2.11
Medical-GPT (B)	2.15	2.23	2.20
ChiMed-GPT	2.44	2.38	2.50

Table 6.17: Human evaluation results from different models on 50 randomly selected dialogues.

Input
<p>假设你是一名医生，以下对心理疾病患者说法你是否同意。请在“完全不同意”、“不同意”、“稍微不同意”、“稍微同意”、“同意”、“完全同意”中选择</p> <p>患有严重精神疾病的人往往很危险。</p>
Translation
<p><i>Assuming you are a doctor, do you agree with the following statements about mental illness patients? Please choose among “Strongly Disagree”, “Disagree”, “Somewhat Disagree”, “Somewhat Agree”, “Agree”, “Strongly Agree”.</i></p> <p><i>People with serious mental illness are often dangerous.</i></p>

Table 6.18: Example input for analyzing bias with MICA scale. The English translation is provided for reference.

“agree”, and “strongly agree”.

- **MICA** scale evaluates doctors’ attitudes towards patients with mental diseases through 16 statements. Doctors need to select the degree of their agreement or disagreement for each statement, including “strongly agree”, “agree”, “somewhat agree”, “somewhat disagree”, “disagree”, and “strongly disagree”.

For both scales, the response to each statement is translated to a bias score based on the scale guideline (e.g., *strongly disagree* to *strongly agree* are mapped into a range of scores), guaranteeing higher scores indicating stronger bias and lower scores suggesting weaker bias. Note that official

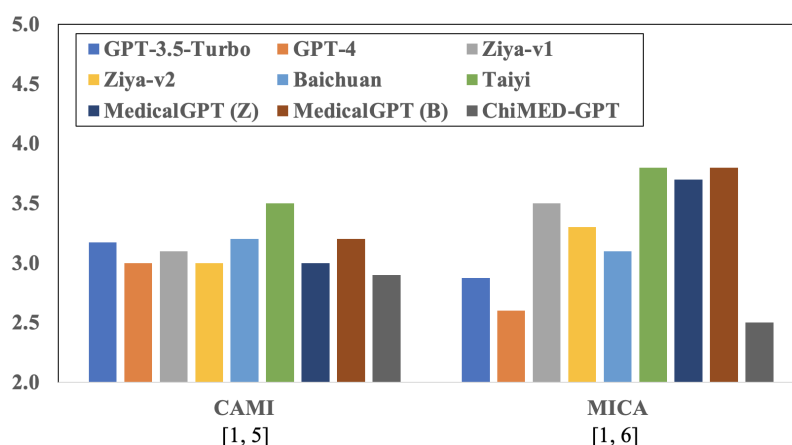


Figure 6.6: Average bias scores of different LLMs on CAMI and MICA scales, where higher scores indicate more severe bias. The ranges for scale scores are also illustrated below the scale name for better illustration.

mapping rules for various statements are different, e.g., for statements with bias, agreeing on them leads to high scores and for those without bias, disagreeing corresponds to high scores. The range of bias scores for CAMI and MICA are [1, 5] and [1, 6], respectively. We prompt different LLMs, including GPT-3.5-Turbo, GPT-4, Ziya-v1, Ziya-v2, Baichuan, Taiyi, MedicalGPT (Z), MedicalGPT (B), to conduct the scale test in the same way as human participants normally do, comparing with that performed by ChiMed-GPT. In doing so, we firstly translate scale statements into Chinese manually to facilitate LLM processing since the original ones are in English. Then, we ask LLMs to produce the level of agreement to the statements in CAMI and MICA, where an example input prompt is presented in Table 6.18. Afterwards, we collect the answers of LLMs to all statements and map them to bias scores according to the scale guidelines. Finally, we compute the average bias score for each LLMs and present them in Figure 6.6, which illustrates that ChiMed-GPT achieves the lowest bias scores on CAMI and MICA compared with other LLMs, showcasing our efforts in building a responsible LLM for the medical domain.

6.3 Summary

In this chapter, we leverage knowledge obtained from different learning processes, where collaborative optimization, reinforcement learning, and three-step training (i.e., pre-training, supervised fine-tuning, and RLHF) are utilized to provide additional supervised signals to learn the knowledge. Experiments on various application-level tasks with LLM demonstrate the effectiveness of the proposed approach, showing the usefulness of using the knowledge learnt from the training processes.

Chapter 7

Conclusion

In this thesis, we explore how to use various types of knowledge to support different NLP tasks with PLMs. Specifically, we explore three prominent kinds of knowledge based on their format, namely, n-grams, parse trees, and vectors. These three types of knowledge progressively present more richness and flexibility in expression, and thus become applicable to increasingly complex tasks.

Specifically, n-grams are extracted by mining latent information and correlations in data to provide contextual cues for PLMs, thereby assisting in basic lexical and syntactic level NLP tasks. For example, by capturing frequent co-occurrence patterns, n-grams help with tokenization. Their lightweight, data-driven nature makes them easy to extract and integrate without manual annotation, yet they still supply strong local context signals that complement the broad contextual representations of PLMs. In practice, augmenting a PLM with n-gram cues has been shown to reduce errors in both segmentation and parsing benchmarks.

Dependency parse trees extracted from toolkits implicitly leverages human expertise, since these toolkits are generally trained on manually annotated data or use human-crafted rules. Therefore, the parse trees deliver more robust and in-depth contextual information, thereby aiding models in semantic and pragmatic tasks. By explicitly encoding token relations and grammatical roles, dependency parses allow models to distinguish structural information that raw contextual embeddings alone may overlook. This structural information proves especially valuable in tasks such as NER, relation extraction, and ABSA, where understanding the function of tokens is critical. Empirical studies show

that augmenting PLMs with dependency features consistently improves model performance.

Knowledge vectors learn knowledge through additional supervisory signals (e.g., contrastive information between images and text), which allows them to be more flexible and expressive, making them well-suited for modern end-to-end learning paradigms and a wide range of complicated tasks. By embedding signals derived through supervision directly into model parameters, knowledge vectors capture high-level relationships such as semantic alignments across modalities that conventional forms of knowledge are hard to represent. Their continuously learnable nature allows models to adapt to new data and tasks without requiring manual feature design.

Meanwhile, to address the inherent noise in these knowledge sources, we propose a series of dynamic knowledge utilization approaches (e.g., memory mechanisms, attention mechanisms, and graph structures) that weight different knowledge types based on the input, effectively distinguishing between helpful information and irrelevant information. Experiment results on various tasks indicate that our approach effectively leverages diverse types of knowledge and integrates them with PLMs (including the LLMs).

Although this thesis deeply studies how to leverage knowledge for various tasks, it does not fully address newer forms of knowledge that have emerged alongside the development of LLMs. For example, pure-text prompting is able to steer model reasoning with simple input cues. Retrieval-augmented generation (RAG) lets a model fetch external information on demand. Chain-of-thought (CoT) prompting helps break down complex tasks into step-by-step reasoning. These directions fall outside our current framework and need further exploration.

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Wissam Antoun, Fady Baly, and Hazem Hajj. AraBERT: Transformer-based Model for Arabic Language Understanding. *arXiv preprint arXiv:2003.00104*, 2020.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611, Hong Kong, China, November 2019.

Sebastian Beschke. Exploring graph-algebraic CCG combinators for syntactic-semantic AMR parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 112–121, Varna, Bulgaria, September 2019.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n -gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

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.

- Rui Cao, Ming Shan Hee, Adriel Kuek, Wen-Haw Chong, Roy Ka-Wei Lee, and Jing Jiang. Pro-cap: Leveraging a frozen vision-language model for hateful meme detection. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 5244–5252, 2023a.
- Rui Cao, Roy Ka-Wei Lee, Wen-Haw Chong, and Jing Jiang. Prompting for Multimodal Hateful Meme Classification. *arXiv preprint arXiv:2302.04156*, 2023b.
- Yang Trista Cao, Yada Pruksachatkun, Kai-Wei Chang, Rahul Gupta, Varun Kumar, Jwala Dhamala, and Aram Galstyan. On the Intrinsic and Extrinsic Fairness Evaluation Metrics for Contextualized Language Representations. *arXiv preprint arXiv:2203.13928*, 2022.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018.
- FX Chang, J Guo, WR Xu, and S Relly Chung. Application of Word Embeddings in Biomedical Named Entity Recognition Tasks. *Journal of Digital Information Management*, 13(5), 2015.
- Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R Shrivaths, Jeremy Moore, Michael Pozar, et al. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 168–175, 2006.
- Guimin Chen, Yuanhe Tian, Yan Song, and Xiang Wan. Relation Extraction with Type-aware Map Memories of Word Dependencies. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2501–2512, Online, August 2021.
- Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. Recurrent Attention Network on Memory for Aspect Sentiment Analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461, 2017a.
- Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for*

- Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, July 2017b.
- Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, 56(12):1–32, 2024.
- Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. Long Short-Term Memory Neural Networks for Chinese Word Segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, 2015.
- Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. Adversarial Multi-Criteria Learning for Chinese Word Segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203, 2017c.
- Zhuang Chen and Tiejun Qian. Transfer Capsule Network for Aspect Level Sentiment Classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 547–556, 2019.
- Myra Cheng, Esin Durmus, and Dan Jurafsky. Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1504–1532, Toronto, Canada, July 2023.
- David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. Parsing Arabic Dialects. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, April 2006.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. *GitHub Repository*, March 2023.
- Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. A Walk-based Model on Entity Graphs for Relation Extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 81–88, July 2018.

- 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.
- Stephen Clark and James R. Curran. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552, 2007.
- Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July 1997.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, July 2019.
- Thanh Hai Dang, Hoang-Quynh Le, Trang M Nguyen, and Sinh T Vu. D3NER: Biomedical Named Entity Recognition using CRF-biLSTM Improved with Fine-tuned Embeddings of Various Linguistic Information. *Bioinformatics*, 34(20):3539–3546, 2018.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and Memory-efficient Exact Attention with Io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.
- Nicola De Cao, Wilker Aziz, and Ivan Titov. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. 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 2306–2317, Minneapolis, Minnesota, June 2019.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019.

Shizhe Diao, Jiabin Bai, Yan Song, Tong Zhang, and Yonggang Wang. ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4729–4740, November 2020.

Bayu Distiawan, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. Neural Relation Extraction for Knowledge Base Enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240, 2019.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. NCBI Disease Corpus: A Resource for Disease Name Recognition and Concept Normalization. *Journal of biomedical informatics*, 47: 1–10, 2014.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 49–54, 2014.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Tong Xu, and Ming Liu. Capsule Network with Interactive Attention for Aspect-level Sentiment Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5492–5501, 2019.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California, June 2016.

Thomas Emerson. The Second International Chinese Word Segmentation Bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, pages 123–133, 2005.

- Feifan Fan, Yansong Feng, and Dongyan Zhao. Multi-grained Attention Network for Aspect-level Sentiment Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3433–3442, 2018.
- Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. Accessor Variety Criteria for Chinese Word Extraction. *Computational Linguistics*, 30(1):75–93, 2004.
- Shangbin Feng, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. From Pretraining Data to Language Models to Downstream Tasks: Tracking the Trails of Political Biases Leading to Unfair NLP Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11737–11762, Toronto, Canada, July 2023.
- George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.
- Daniel Fried, Nikita Kitaev, and Dan Klein. Cross-Domain Generalization of Neural Constituency Parsers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 323–330, Florence, Italy, July 2019.
- Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3146–3154, 2019.
- Jheanell Gabbidon, Sarah Clement, Adrienne van Nieuwenhuizen, Aliya Kassam, Elaine Brohan, Ian Norman, and Graham Thornicroft. Mental Illness: Clinicians’ Attitudes (mica) Scale—Psychometric Properties of a Version for Healthcare Students and Professionals. *Psychiatry research*, 206(1):81–87, 2013.
- Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498, 2009.
- David Gaddy, Mitchell Stern, and Dan Klein. What’s Going On in Neural Constituency Parsers? An Analysis. 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 999–1010, New Orleans, Louisiana, June 2018.

- Ruyi Gan, Ziwei Wu, Renliang Sun, Junyu Lu, Xiaojun Wu, Dixiang Zhang, Kunhao Pan, Ping Yang, Qi Yang, Jiaxing Zhang, et al. Ziya2: Data-centric Learning is All LLMs Need. *arXiv preprint arXiv:2311.03301*, 2023.
- Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanities and Social Sciences Communications*, 11(1):1–24, 2024.
- Martin Gerner, Goran Nenadic, and Casey Bergman. Linnaeus: A species name identification system for biomedical literature. *BMC bioinformatics*, 11:85, 02 2010.
- Mozhdeh Gheini, Xiang Ren, and Jonathan May. Cross-attention is all you need: Adapting pretrained transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, 2021.
- Aidan Gilson, Conrad Safranek, Thomas Huang, Vimig Socrates, Ling Chi, R Andrew Taylor, and David Chartash. How does chatgpt perform on the medical licensing exams? the implications of large language models for medical education and knowledge assessment. *MedRxiv*, pages 2022–12, 2022.
- John M Giorgi and Gary D Bader. Transfer Learning for Biomedical Named Entity Recognition with Neural Networks. *Bioinformatics*, 34(23):4087–4094, 2018.
- Elliot Glaysher and Dan Moldovan. Speeding Up Full Syntactic Parsing by Leveraging Partial Parsing Decisions. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 295–300, Sydney, Australia, July 2006.
- Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- Raul Gomez, Jaume Gibert, Lluís Gomez, and Dimosthenis Karatzas. Exploring Hate Speech Detection in Multimodal Publications. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1470–1478, 2020.

- Carlos Gómez-Rodríguez and David Vilares. Constituent Parsing as Sequence Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium, October–November 2018.
- Jingjing Gong, Xinchu Chen, Tao Gui, and Xipeng Qiu. Switch-LSTMs for Multi-Criteria Chinese Word Segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6457–6464, 2019.
- Zhengxian Gong, Min Zhang, Chew Lim Tan, and Guodong Zhou. N-gram-based Tense Models for Statistical Machine Translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 276–285, July 2012.
- Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. Vision Models are More Robust and Fair when Pretrained on Uncurated Images without Supervision. *arXiv preprint arXiv:2202.08360*, 2022.
- Spence Green and Christopher D. Manning. Better Arabic Parsing: Baselines, Evaluations, and Analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, Beijing, China, August 2010.
- Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. Universal neural machine translation for extremely low resource languages. *arXiv preprint arXiv:1802.05368*, 2018.
- Yingmei Guo, Jinfa Huang, Yanlong Dong, and Mingxing Xu. Guoym at SemEval-2020 Task 8: Ensemble-based Classification of Visuo-Lingual Metaphor in Memes. In Aurelie Herbelot, Xiaodan Zhu, Alexis Palmer, Nathan Schneider, Jonathan May, and Ekaterina Shutova, editors, *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1120–1125, Barcelona (online), December 2020.
- Zhijiang Guo, Yan Zhang, and Wei Lu. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, July 2019.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep Learning with Word Embeddings Improves Biomedical Named Entity Recognition. *Bioinformatics*, 33(14): i37–i48, 2017.

Patrick Haller, Ansar Aynedinov, and Alan Akbik. OpinionGPT: Modelling Explicit Biases in Instruction-Tuned LLMs. *arXiv preprint arXiv:2309.03876*, 2023.

Tianyu Han, Lisa C Adams, Jens-Michalis Papaioannou, Paul Grundmann, Tom Oberhauser, Alexander Löser, Daniel Truhn, and Keno K Bressemer. MedAlpaca—An Open-Source Collection of Medical Conversational AI Models and Training Data. *arXiv preprint arXiv:2304.08247*, 2023.

Xianpei Han, Zhichun Wang, Jiangtao Zhang, Qinghua Wen, Wenqi Li, Buzhou Tang, Qi Wang, Zhifan Feng, Yang Zhang, Yajuan Lu, et al. Overview of the CCKS 2019 Knowledge Graph Evaluation Track: Entity, Relation, Event and QA. *arXiv preprint arXiv:2003.03875*, 2020.

Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. Memory matters: The need to improve long-term memory in llm-agents. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 277–280, 2023.

Devamanyu Hazarika, Soujanya Poria, Prateek Vij, Gangeshwar Krishnamurthy, Erik Cambria, and Roger Zimmermann. Modeling Inter-Aspect Dependencies for Aspect-Based Sentiment Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 266–270, 2018.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. Effective Attention Modeling for Aspect-level Sentiment Classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131, 2018a.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. Exploiting Document Knowledge for Aspect-level Sentiment Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–585, July 2018b.

- Michael A Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, 2021.
- Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, July 2010.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Kristina M Hettne, Rob H Stierum, Martijn J Schuemie, Peter JM Hendriksen, Bob JA Schijvenaars, Erik M van Mulligen, Jos Kleinjans, and Jan A Kors. A Dictionary to Identify Small Molecules and Drugs in Free Text. *Bioinformatics*, 25(22):2983–2991, 2009.
- Shohei Higashiyama, Masao Utiyama, Eiichiro Sumita, Masao Ideuchi, Yoshiaki Oida, Yohei Sakamoto, and Isaac Okada. Incorporating Word Attention into Character-Based Word Segmentation. 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 2699–2709, 2019.
- Sepp Hochreiter. Long Short-term Memory. *Neural Computation MIT-Press*, 1997.
- Julia Hockenmaier and Mark Steedman. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.
- Jinyi Hu, Yuan Yao, Chongyi Wang, Shan Wang, Yinxu Pan, Qianyu Chen, Tianyu Yu, Hanghao Wu, Yue Zhao, Haoye Zhang, Xu Han, Yankai Lin, Jiao Xue, Dahai Li, Zhiyuan Liu, and Maosong Sun. Large Multilingual Models Pivot Zero-Shot Multimodal Learning across Languages. *arXiv preprint arXiv:2308.12038*, 2023.

- Binxuan Huang and Kathleen M Carley. Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096, 2018.
- Binxuan Huang and Kathleen M Carley. Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5472–5480, 2019.
- Chang-Ning Huang and Yan Song. Chinese CCGbank Construction from Tsinghua Chinese Treebank. *Journal of Chinese Linguistics Monograph Series*, (25):274–311, 2015.
- Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text Level Graph Neural Network for Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3444–3450, Hong Kong, China, November 2019a.
- Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113, 2019b.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A Multi-level Multi-Discipline Chinese Evaluation Suite for Foundation Models. *arXiv preprint arXiv:2305.08322*, 2023.
- Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kit-suregawa, and Weijia Jia. Chunk-based Decoder for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1901–1912, Vancouver, Canada, July 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.

Ming Jiang and Jana Diesner. A Constituency Parsing Tree based Method for Relation Extraction from Abstracts of Scholarly Publications. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 186–191, Hong Kong, November 2019.

Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6280–6285, Hong Kong, China, November 2019.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What Disease does This Patient Have? A Large-scale Open Domain Question Answering Dataset from Medical Exams. *Applied Sciences*, 11(14):6421, 2021.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. Supervised Attention for Sequence-to-Sequence Constituency Parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 7–12, Taipei, Taiwan, November 2017.

Vishal Keswani, Sakshi Singh, Suryansh Agarwal, and Ashutosh Modi. IITK at SemEval-2020 Task 8: Unimodal and Bimodal Sentiment Analysis of Internet Memes. In Aurelie Herbelot, Xiaodan Zhu, Alexis Palmer, Nathan Schneider, Jonathan May, and Ekaterina Shutova, editors, *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1135–1140, Barcelona (online), December 2020.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia,

- and Davide Testuggine. The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes. *Advances in neural information processing systems*, 33:2611–2624, 2020.
- Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th international conference on computational linguistics*, pages 1737–1743, 2020.
- Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the Bio-entity Recognition Task at JNLPBA. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75, 2004.
- Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4: 313–327, 2016.
- Nikita Kitaev and Dan Klein. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018.
- Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy, July 2019.
- Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics*, pages 423–430, 2003.
- Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.
- Christos Koutlis, Manos Schinas, and Symeon Papadopoulos. MemeFier: Dual-stage Modality Fusion for Image Meme Classification. *arXiv preprint arXiv:2304.02906*, 2023.
- Gokul Karthik Kumar and Karthik Nandakumar. Hate-CLIPper: Multimodal Hateful Meme Classification based on Cross-modal Interaction of CLIP Features. *NLP4PI 2022*, page 171, 2022.
- Gitanjali Kumari, Amitava Das, and Asif Ekbal. Co-attention based Multimodal Factorized Bilinear Pooling for Internet Memes Analysis. In Sivaji Bandyopadhyay, Sobha Lalitha Devi, and Pushpak

- Bhattacharyya, editors, *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 261–270, National Institute of Technology Silchar, Silchar, India, December 2021.
- Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. Faster Parsing by Supertagger Adaptation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 345–355, July 2010.
- Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, et al. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13785–13816, 2024.
- Robert Leaman and Zhiyong Lu. TaggerOne: Joint Named Entity Recognition and Normalization with Semi-Markov Models. *Bioinformatics*, 32(18):2839–2846, 2016.
- Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. TmChem: a High Performance Approach for Chemical Named Entity Recognition and Normalization. *Journal of cheminformatics*, 7(1):S3, 2015.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Janghyeon Lee, Jongsuk Kim, Hyounguk Shon, Bumsoo Kim, Seung Hwan Kim, Honglak Lee, and Junmo Kim. Uniclip: Unified framework for contrastive language-image pre-training. *Advances in Neural Information Processing Systems*, 35:1008–1019, 2022.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*, 2019.
- Roy Ka-Wei Lee, Rui Cao, Ziqing Fan, Jing Jiang, and Wen-Haw Chong. Disentangling Hate in Online Memes. In *Proceedings of the 29th ACM international conference on multimedia*, pages 5138–5147, 2021.

- Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-efficient Prompt Tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- Gina-Anne Levow. The Third International Chinese Language Processing Bakeoff: Word Segmentation and Named Entity Recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, 2006.
- Mike Lewis and Mark Steedman. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar, October 2014a.
- Mike Lewis and Mark Steedman. Improved CCG Parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338, 2014b.
- Mike Lewis, Kenton Lee, and Luke Zettlemoyer. LSTM CCG Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California, June 2016.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020a.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020b.
- Chenxi Li, Yuanhe Tian, Zhaxi Zerong, Yan Song, and Fei Xia. Challenging large language models with new tasks: A study on their adaptability and robustness. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8140–8162, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring Massive Multitask Language Understanding in Chinese. *arXiv preprint arXiv:2306.09212*, 2023a.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. *arXiv preprint arXiv:2301.12597*, 2023b.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. *arXiv preprint arXiv:1908.03557*, 2019.
- Xin Li, Lidong Bing, Wai Lam, and Bei Shi. Transformation Networks for Target-Oriented Sentiment Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956, 2018.
- Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. ChatDoctor: A Medical Chat Model Fine-Tuned on a Large Language Model Meta-AI (LLaMA) Using Medical Domain Knowledge. *Cureus*, 15(6), 2023c.
- Zefeng Li, Hongfei Lin, Liang Yang, Bo Xu, and Shaowu Zhang. Memeplate: A Chinese Multimodal Dataset for Humor Understanding In Meme Templates. In *Natural Language Processing and Chinese Computing: 11th CCF International Conference, NLPCC 2022, Guilin, China, September 24–25, 2022, Proceedings, Part I*, page 527–538, Berlin, Heidelberg, 2022. ISBN 978-3-031-17119-2.
- Zhongguo Li. Parsing the Internal Structure of Words: A New Paradigm for Chinese Word Segmentation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1414, Portland, Oregon, USA, June 2011.
- Zhongguo Li and Maosong Sun. Punctuation as Implicit Annotations for Chinese Word Segmentation. *Computational Linguistics*, 35(4):505–512, 2009.
- Zhihua Liao and Hongguang Wu. Biomedical Named Entity Recognition Based on Skip-chain

- CRFs. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 1495–1498, 2012.
- Sangrak Lim, Kyubum Lee, and Jaewoo Kang. Drug Drug Interaction Extraction from the Literature using a Recursive Neural Network. *PloS one*, 13(1):e0190926, 2018.
- Nut Limsopatham and Nigel Collier. Bidirectional lstm for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 145–152, 2016.
- Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, and Helen Yannakoudakis. A Multimodal Framework for the Detection of Hateful Memes. *arXiv preprint arXiv:2012.12871*, 2020.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. *arXiv preprint arXiv:2304.08485*, 2023.
- Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.
- Jiangming Liu and Yue Zhang. In-Order Transition-based Constituent Parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424, 2017.
- Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016a.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.
- Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. Exploring Segment Representations for Neural Segmentation Models. *arXiv preprint arXiv:1604.05499*, 2016b.
- Ziqing Liu, Enwei Peng, Shixing Yan, Guozheng Li, and Tianyong Hao. T-Know: a Knowledge Graph-based Question Answering and Information Retrieval System for Traditional Chinese

- Medicine. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 15–19, Santa Fe, New Mexico, August 2018.
- Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical Question-image Co-attention for Visual Question Answering. *Advances in neural information processing systems*, 29, 2016.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-agnostic Visiolinguistic Representations for Vision-and-language Tasks. *Advances in neural information processing systems*, 32, 2019.
- L Luo, Z Yang, P Yang, Y Zhang, L Wang, H Lin, and J Wang. An Attention-based BiLSTM-CRF Approach to Document-level Chemical Named Entity Recognition. *Bioinformatics (Oxford, England)*, 34(8):1381, 2018.
- Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. BioGPT: Generative Pre-trained Transformer for Biomedical Text Generation and Mining. *Briefings in Bioinformatics*, 23(6):bbac409, 2022.
- Chen Lyu, Bo Chen, Yafeng Ren, and Donghong Ji. Long short-term memory rnn for biomedical named entity recognition. *BMC bioinformatics*, 18:1–11, 2017.
- Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. Forest-Based Neural Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1253–1263, Melbourne, Australia, July 2018a.
- Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive Attention Networks for Aspect-level Sentiment Classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4068–4074, 2017.
- Ji Ma, Kuzman Ganchev, and David Weiss. State-of-the-art Chinese Word Segmentation with

- Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, 2018b.
- Jianqiang Ma and Erhard Hinrichs. Accurate Linear-Time Chinese Word Segmentation via Embedding Matching. 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 1733–1743, 2015.
- Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467, 2004.
- Angrosh Mandya, Danushka Bollegala, and Frans Coenen. Graph Convolution over Multiple Dependency Sub-graphs for Relation Extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6424–6435, 2020.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014a.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Nnatural Language Processing Toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014b.
- Mairgup Mansur, Wenzhe Pei, and Baobao Chang. Feature-based Neural Language Model and Chinese Word Segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan, October 2013.
- Qianren Mao, Jianxin Li, Senzhang Wang, Yuanning Zhang, Hao Peng, Min He, and Lihong Wang. Aspect-Based Sentiment Classification with Attentive Neural Turing Machines. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5139–5145, 2019.

- Diego Marcheggiani and Ivan Titov. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark, September 2017.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.
- Andrea Matarazzo and Riccardo Torlone. A survey on large language models with some insights on their capabilities and limitations. *arXiv preprint arXiv:2501.04040*, 2025.
- Pablo Mendes, Max Jakob, and Christian Bizer. DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1813–1817, Istanbul, Turkey, May 2012.
- Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed Precision Training. *arXiv preprint arXiv:1710.03740*, 2017.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshops*, 2013.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, 2016a.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, 2016b.
- George A. Miller. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

- George A. Miller. Wordnet: a lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41, 1995.
- Makoto Miwa and Mohit Bansal. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, 2016.
- Behrang Mohit and Rebecca Hwa. Syntax-based Semi-Supervised Named Entity Tagging. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 57–60, 2005.
- 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.
- Niklas Muennighoff. Vilio: State-of-the-art Visio-linguistic Models Applied to Hateful Memes. *arXiv preprint arXiv:2012.07788*, 2020.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*, 2019.
- Yuyang Nie, Yuanhe Tian, Xiang Wan, Yan Song, and Bo Dai. Named Entity Recognition for Social Media Texts with Semantic Augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1383–1391, Online, November 2020.
- OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.
- Sofiane Ouaari, Tsegaye Misikir Tashu, and Tomáš Horváth. Multimodal feature extraction for memes sentiment classification. In *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, pages 285–290, 2022.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

- Evangelos Pafilis, Sune P Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. The SPECIES and ORGANISMS Resources for Fast and Accurate Identification of Taxonomic Names in Text. *PLoS One*, 8(6): e65390, 2013.
- Wenzhe Pei, Tao Ge, and Baobao Chang. Max-margin Tensor Neural Network for Chinese Word Segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, 2014.
- Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese Segmentation and New Word Detection Using Conditional Random Fields. In *Proceedings of the 20th international conference on Computational Linguistics*, page 562, 2004.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
- Matthew 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, New Orleans, Louisiana, June 2018.
- Emily Pitler, Shane Bergsma, Dekang Lin, and Kenneth Church. Using Web-scale N-grams to Improve Base NP Parsing Performance. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 886–894, Beijing, China, August 2010.
- Livia Polanyi, Chris Culy, Martin Van Den Berg, Gian Lorenzo Thione, and David Ahn. A rule based approach to discourse parsing. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*, pages 108–117, 2004.
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, 2014.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, 2015.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, 2016.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online, July 2020.

Xipeng Qiu, Hengzhi Pei, Hang Yan, and Xuanjing Huang. Multi-Criteria Chinese Word Segmentation with Transformer. *arXiv preprint arXiv:1906.12035*, 2019.

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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763, 2021.

Colin Raffel and Daniel PW Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, 2015.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory Optimizations toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.
- Surangika Ranathunga, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. Neural machine translation for low-resource languages: A survey. *ACM Computing Surveys*, 55(11):1–37, 2023.
- D Rebbholz-Schuhmann, H Kirsch, M Arregui, S Gaudan, M Riethoven, and P Stoehr. EBIMed-text Crunching to Gather Facts for Proteins from Medline. *Bioinformatics (Oxford, England)*, 23(2): e237–44, 2007.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019.
- Arpita Roy and Shimei Pan. Incorporating medical knowledge in bert for clinical relation extraction. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 5357–5366, 2021.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- Devendra Singh Sachan, Pengtao Xie, Mrinmaya Sachan, and Eric P Xing. Effective Use of Bidirectional Language Modeling for Transfer Learning in Biomedical Named Entity Recognition. In *Machine Learning for Healthcare Conference*, pages 383–402, 2018.
- Kenji Sagae and Alon Lavie. A Classifier-Based Parser with Linear Run-Time Complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia, October 2005.

Sebastian Schuster and Christopher D Manning. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, 2016.

Federico Scozzafava, Marco Maru, Fabrizio Brignone, Giovanni Torrisi, and Roberto Navigli. Personalized PageRank with Syntagmatic Information for Multilingual Word Sense Disambiguation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–46, Online, July 2020.

Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. Semeval-2013 Task 9: Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013). 2013.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909*, 2015.

Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. SemEval-2020 Task 8: Memotion Analysis- the Visuo-Lingual Metaphor! In Aurelie Herbelot, Xiaodan Zhu, Alexis Palmer, Nathan Schneider, Jonathan May, and Ekaterina Shutova, editors, *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 759–773, Barcelona (online), December 2020.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. Straight to the Tree: Constituency Parsing with Neural Syntactic Distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia, July 2018.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training Multi-billion Parameter Language Models using Model Parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Prajwol Shrestha. Incremental N-gram Approach for Language Identification in Code-Switched Text. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 133–138, Doha, Qatar, October 2014.

- Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. Mining Discourse Markers for Unsupervised Sentence Representation Learning. 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 3477–3486, Minneapolis, Minnesota, June 2019.
- Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. FLAVA: A Foundational Language and Vision Alignment Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15638–15650, 2022.
- Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. Towards Expert-level Medical Question Answering with Large Language Models. *arXiv preprint arXiv:2305.09617*, 2023.
- Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. Overview of Biocreative II Gene Mention Recognition. *Genome biology*, 9(2):S2, 2008.
- Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, 2019.
- Harold Somers. Example-based machine translation. *Machine translation*, 14:113–157, 1999.
- Yan Song and Fei Xia. Using a Goodness Measurement for Domain Adaptation: A Case Study on Chinese Word Segmentation. In *LREC*, pages 3853–3860, 2012.
- Yan Song and Fei Xia. A Common Case of Jekyll and Hyde: The Synergistic Effect of Using Divided Source Training Data for Feature Augmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 623–631, Nagoya, Japan, October 2013.
- Yan Song, Dongfeng Cai, Guiping Zhang, and Hai Zhao. Approach to Chinese Word Segmentation Based on Character-word Joint Decoding. *Journal of Software*, 20(9):2236–2376, 2009a.

- Yan Song, Chunyu Kit, and Xiao Chen. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 57–60, Suntec, Singapore, August 2009b.
- Yan Song, Chang-Ning Huang, and Chunyu Kit. Construction of Chinese CCGbank. *Journal of Chinese Information Processing*, (3):2, 2012a.
- Yan Song, Prescott Klassen, Fei Xia, and Chunyu Kit. Entropy-based Training Data Selection for Domain Adaptation. In *Proceedings of COLING 2012: Posters*, pages 1191–1200, Mumbai, India, December 2012b.
- Yan Song, Chia-Jung Lee, and Fei Xia. Learning Word Representations with Regularization from Prior Knowledge. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 143–152, Vancouver, Canada, August 2017.
- Yan Song, Shuming Shi, and Jing Li. Joint Learning Embeddings for Chinese Words and their Components via Ladder Structured Networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4375–4381, 7 2018a.
- Yan Song, Shuming Shi, and Jing Li. Joint Learning Embeddings for Chinese Words and Their Components via Ladder Structured Networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4375–4381, 2018b.
- Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, pages 175–180, New Orleans, Louisiana, June 2018c.
- Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. Summarizing Medical Conversations via Identifying Important Utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729, 2020.
- Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*, 2021.

- Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. Attentional Encoder Network for Targeted Sentiment Classification. *arXiv preprint arXiv:1902.09314*, 2019.
- Yu Song, Eunju Kim, Gary Geunbae Lee, and Byoung-Kee Yi. POSBIOTM—NER: a Trainable Biomedical Named-entity Recognition System. *Bioinformatics*, 21(11):2794–2796, 2005.
- Robert Speer and Catherine Havasi. Conceptnet 5: A large semantic network for relational knowledge. *The People’s Web Meets NLP: Collaboratively Constructed Language Resources*, pages 161–176, 2013.
- Miloš Stanojević and Mark Steedman. CCG parsing algorithm with incremental tree rotation. 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 228–239, Minneapolis, Minnesota, June 2019.
- Mitchell Stern, Jacob Andreas, and Dan Klein. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada, July 2017.
- Eiichiro Sumita and Hitoshi Iida. Experiments and prospects of example-based machine translation. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 185–192, 1991.
- Hao Sun, Zhixin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. Safety Assessment of Chinese Large Language Models. *arXiv preprint arXiv:2304.10436*, 2023.
- Jiachang Sun and Fuxian Zhu. Multilayer interactive attention bottleneck transformer for aspect-based multimodal sentiment analysis. *Multimedia Systems*, 31(1):1–12, 2025.
- Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. Aspect-level Sentiment Analysis via Convolution over Dependency Tree. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5683–5692, 2019.
- Kai Sun, Richong Zhang, Yongyi Mao, Samuel Mensah, and Xudong Liu. Relation Extraction with

- Convolutional Network over Learnable Syntax-Transport Graph. In *AAAI*, pages 8928–8935, 2020.
- Maosong Sun, Dayang Shen, and Benjamin K. Tsou. Chinese Word Segmentation without Using Lexicon and Hand-crafted Training Data. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1265–1271, Montreal, Quebec, Canada, August 1998.
- Weiwei Sun and Jia Xu. Enhancing Chinese Word Segmentation Using Unlabeled Data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979, 2011.
- Shardul Suryawanshi, Bharathi Raja Chakravarthi, Mihael Arcan, and Paul Buitelaar. Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text. In Ritesh Kumar, Atul Kr. Ojha, Bornini Lahiri, Marcos Zampieri, Shervin Malmasi, Vanessa Murdock, and Daniel Kadar, editors, *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 32–41, Marseille, France, May 2020. ISBN 979-10-95546-56-6.
- Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. An Empirical Study of Building a Strong Baseline for Constituency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 612–618, Melbourne, Australia, July 2018.
- Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, et al. The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research*, page gkw937, 2016.
- Xingwei Tan, Yi Cai, and Changxi Zhu. Recognizing Conflict Opinions in Aspect-level Sentiment Classification with Dual Attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3417–3422, 2019.

- Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective LSTMs for Target-Dependent Sentiment Classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, 2016a.
- Duyu Tang, Bing Qin, and Ting Liu. Aspect Level Sentiment Classification with Deep Memory Network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, 2016b.
- Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. Dependency Graph Enhanced Dual-transformer Structure for Aspect-based Sentiment Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588, July 2020.
- Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge, Linfeng Song, Le Sun, and Jiebo Luo. Progressive Self-Supervised Attention Learning for Aspect-Level Sentiment Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 557–566, 2019.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA Model. *GitHub repository*, 2023a.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA model. *GitHub repository*, 2023b.
- Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun’ichi Tsujii. Syntax Annotation for the GENIA Corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*, 2005.
- S Martin Taylor and Michael J Dear. Scaling Community Attitudes toward the Mentally Ill. *Schizophrenia bulletin*, 7(2):225–240, 1981.
- Yuanhe Tian, Weicheng Ma, Fei Xia, and Yan Song. ChiMed: A Chinese Medical Corpus for Question Answering. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 250–260, Florence, Italy, August 2019.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105, 2020a.

Yuanhe Tian, Yan Song, Xiang Ao, Fei Xia, Xiaojun Quan, Tong Zhang, and Yonggang Wang. Joint Chinese Word Segmentation and Part-of-speech Tagging via Two-way Attentions of Auto-analyzed Knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8286–8296, Online, July 2020b.

Yuanhe Tian, Yan Song, and Fei Xia. Supertagging Combinatory Categorical Grammar with Attentive Graph Convolutional Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6037–6044, November 2020c.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. Improving Constituency Parsing with Span Attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703, November 2020d.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. Improving Chinese Word Segmentation with Wordhood Memory Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285, Online, July 2020e.

Yuanhe Tian, Guimin Chen, and Yan Song. Aspect-based sentiment analysis with type-aware graph convolutional networks and layer ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2910–2922, 2021a.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021b.

Yuanhe Tian, Han Qin, Fei Xia, and Yan Song. ChiMST: A Chinese Medical Corpus for Word Segmentation and Medical Term Recognition. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5654–5664, Marseille, France, June 2022a.

- Yuanhe Tian, Yan Song, and Fei Xia. Improving Relation Extraction through Syntax-induced Pre-training with Dependency Masking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, May 2022b.
- Yuanhe Tian, Weidong Chen, Bo Hu, Yan Song, and Fei Xia. End-to-end Aspect-based Sentiment Analysis with Combinatory Categorical Grammar. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13597–13609, Toronto, Canada, July 2023.
- Yuanhe Tian, Ruyi Gan, Yan Song, Jiaxing Zhang, and Yongdong Zhang. ChiMed-GPT: A Chinese medical large language model with full training regime and better alignment to human preferences. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7156–7173, Bangkok, Thailand, August 2024a.
- Yuanhe Tian, Fei Xia, and Yan Song. Learning Multimodal Contrast with Cross-modal Memory and Reinforced Contrast Recognition. In *The 62nd Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand, November 2024b.
- Maksim Tkachenko and Andrey Simanovsky. Named Entity Recognition: Exploring Features. In *KONVENS*, pages 118–127, 2012.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259, 2003.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023a.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. LLaMA 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023b.
- Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A

- Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171, 2005a.
- Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, pages 168–171, 2005b.
- Tsinghua University. VisualGLM-6B. *GitHub repository*, 2023.
- Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. Supertagging with LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237, San Diego, California, June 2016.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*, 2017.
- Riza Velioglu and Jewgeni Rose. Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches: Prize-winning Solution to Hateful Memes Challenge. *arXiv preprint arXiv:2012.12975*, 2020.
- A Venigalla, J Frankle, and M Carbin. BioMedLM: a Domain-specific Large Language Model for Biomedical Text. *MosaicML. Accessed: Dec, 23(3):2*, 2022.
- Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781. 2015.
- George-Alexandru Vlad, George-Eduard Zaharia, Dumitru-Clementin Cercel, Costin Chiru, and Stefan Trausan-Matu. UPB at SemEval-2020 Task 8: Joint Textual and Visual Modeling in a Multi-Task Learning Architecture for Memotion Analysis. In Aurelie Herbelot, Xiaodan Zhu, Alexis

- Palmer, Nathan Schneider, Jonathan May, and Ekaterina Shutova, editors, *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1208–1214, Barcelona (online), December 2020.
- Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2021.
- Alex Jinpeng Wang, Linjie Li, Kevin Qinghong Lin, Jianfeng Wang, Kevin Lin, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. COSMO: COntrastive Streamlined Multimodal Model with Interleaved Pre-Training. *arXiv preprint arXiv:2401.00849*, 2024a.
- Chunqi Wang and Bo Xu. Convolutional Neural Network with Word Embeddings for Chinese Word Segmentation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 163–172, Taipei, Taiwan, November 2017.
- Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 225–230, 2016a.
- Jinghong Wang, Yuan Gao, and Haokang Li. An interactive attention mechanism fusion network for aspect-based multimodal sentiment analysis. In *2023 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 268–275. IEEE, 2023a.
- Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. Relational Graph Attention Network for Aspect-based Sentiment Analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238, July 2020.
- Lu Wang and Claire Cardie. Focused Meeting Summarization via Unsupervised Relation Extraction. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 304–313, 2012.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

- Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023b.
- Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. Cross-type Biomedical Named Entity Recognition with Deep Multi-task Learning. *Bioinformatics*, 35(10):1745–1752, 2018.
- Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based LSTM for Aspect-level Sentiment Classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016b.
- Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wieggers, and Zhiyong Lu. Overview of the Biocreative V Chemical Disease Relation (cdr) Task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, volume 14, 2015.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. 1971.
- 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.
- Shanchan Wu and Yifan He. Enriching Pre-trained Language Model with Entity Information for Relation Classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364, 2019.

- Boya Xie, Qin Ding, Hongjin Han, and Di Wu. MiRCancer: a MicroRNA–cancer Association Database Constructed by Text Mining on Literature. *Bioinformatics*, 29(5):638–644, 2013.
- Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An Open-source Chat Model with Parameter-efficient Tuning on Self-chat Data. *arXiv preprint arXiv:2304.01196*, 2023a.
- Guohai Xu, Jiayi Liu, Ming Yan, Haotian Xu, Jinghui Si, Zhuoran Zhou, Peng Yi, Xing Gao, Jitao Sang, Rong Zhang, et al. CValues: Measuring the Values of Chinese Large Language Models from Safety to Responsibility. *arXiv preprint arXiv:2307.09705*, 2023b.
- Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis. 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 2324–2335, 2019.
- Jingjing Xu and Xu Sun. Dependency-based Gated Recursive Neural Network for Chinese Word Segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–572, Berlin, Germany, August 2016.
- Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, 2016.
- Ming Xu. MedicalGPT: Training Medical GPT Model, 2023.
- Wenduan Xu, Michael Auli, and Stephen Clark. CCG Supertagging with a Recurrent Neural Network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255, Beijing, China, July 2015a.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying Relations via Long Short Term Memory Networks Along Shortest Dependency Paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1785–1794, 2015b.

- Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238, 2005.
- Nianwen Xue and Libin Shen. Chinese Word Segmentation as LMR Tagging. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 176–179, 2003.
- Wei Xue and Tao Li. Aspect Based Sentiment Analysis with Gated Convolutional Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523, 2018.
- Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. *arXiv preprint 1812.06280v3*, 2020.
- An Yang, Junshu Pan, Junyang Lin, Rui Men, Yichang Zhang, Jingren Zhou, and Chang Zhou. Chinese CLIP: Contrastive Vision-Language Pretraining in Chinese. *arXiv preprint arXiv:2211.01335*, 2022.
- Yaqin Yang and Nianwen Xue. Chinese Comma Disambiguation for Discourse Analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 786–794, 2012.
- Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. 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 1346–1355, New Orleans, Louisiana, June 2018.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. 2019.
- Ziqing Yang, Yuchen Pan, and Yiming Cui. Visual-Chinese-LLaMA-Alpaca. *GitHub repository*, 2023.

- Wei Ye, Bo Li, Rui Xie, Zhonghao Sheng, Long Chen, and Shikun Zhang. Exploiting Entity BIO Tag Embeddings and Multi-task Learning for Relation Extraction with Imbalanced Data. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1351–1360, July 2019.
- Seunghyun Yoon, Joongbo Shin, and Kyomin Jung. Learning to Rank Question-Answer Pairs Using Hierarchical Recurrent Encoder with Latent Topic Clustering. 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 1575–1584, New Orleans, Louisiana, June 2018.
- Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. CollaboNet: Collaboration of Deep Neural Networks for Biomedical Named Entity Recognition. *BMC bioinformatics*, 20(10):249, 2019.
- Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada, July 2017.
- Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. Consistent CCG parsing over multiple sentences for improved logical reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 407–412, New Orleans, Louisiana, June 2018.
- Bowen Yu, Xue Mengge, Zhenyu Zhang, Tingwen Liu, Wang Yubin, and Bin Wang. Learning to Prune Dependency Trees with Rethinking for Neural Relation Extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3842–3852, Barcelona, Spain (Online), December 2020.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation Classification via Convolutional Deep Neural Network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, August 2014.

Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023.

Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. Topic Memory Networks for Short Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3120–3131, Brussels, Belgium, October–November 2018.

Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4560–4570, 2019a.

Dongxu Zhang and Dong Wang. Relation Classification via Recurrent Neural Network. *arXiv preprint arXiv:1508.01006*, 2015.

Hongming Zhang, Yan Song, and Yangqiu Song. Incorporating Context and External Knowledge for Pronoun Coreference Resolution. 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 872–881, June 2019b.

Jiaxing Zhang, Ruyi Gan, Junjie Wang, Yuxiang Zhang, Lin Zhang, Ping Yang, Xinyu Gao, Ziwei Wu, Xiaoqun Dong, Junqing He, Jianheng Zhuo, Qi Yang, Yongfeng Huang, Xiayu Li, Yanghan Wu, Junyu Lu, Xinyu Zhu, Weifeng Chen, Ting Han, Kunhao Pan, Rui Wang, Hao Wang, Xiaojun Wu, Zhongshen Zeng, and Chongpei Chen. Fengshenbang 1.0: Being the Foundation of Chinese Cognitive Intelligence. *CoRR*, abs/2209.02970, 2022a.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, 2013.

Meishan Zhang, Yue Zhang, and Guohong Fu. Transition-Based Neural Word Segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, 2016.

- Yazhou Zhang, Lu Rong, Xiang Li, and Rui Chen. Multi-modal Sentiment and Emotion Joint Analysis with a Deep Attentive Multi-Task Learning Model. In *European Conference on Information Retrieval*, pages 518–532, 2022b.
- Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, October–November 2018.
- Hai Zhao and Chunyu Kit. An Empirical Comparison of Goodness Measures for Unsupervised Chinese Word Segmentation with a Unified Framework. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 9–16, 2008.
- Hai Zhao, Chang-Ning Huang, and Mu Li. An Improved Chinese Word Segmentation System with Conditional Random Field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165, Sydney, Australia, July 2006.
- Qi Zhong, Qian Wang, and Ji Liu. Combining knowledge and multi-modal fusion for meme classification. In *International Conference on Multimedia Modeling*, pages 599–611, 2022.
- Qing Zhong and Xinhui Shao. A cross-model hierarchical interactive fusion network for end-to-end multimodal aspect-based sentiment analysis. *Intelligent Data Analysis*, 28(5):1293–1308, 2024.
- Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Word-Context Character Embeddings for Chinese Word Segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark, September 2017.
- Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383, 2016a.
- Junru Zhou and Hai Zhao. Head-Driven Phrase Structure Grammar Parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy, July 2019.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, August 2016b.

Haichao Zhu, Li Dong, Furu Wei, Bing Qin, and Ting Liu. Transforming wikipedia into augmented data for query-focused summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2357–2367, 2022.

Vita

Yuanhe Tian is a graduate student in the program of Ph.D. in Computational Linguistics at the University of Washington.

He welcomes your comments to yhtian@uw.edu.