Incorporating external knowledge into machine learning algorithms for NLP applications

Li, Pengfei

2020

Li, P. (2020). Incorporating external knowledge into machine learning algorithms for NLP applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/144577>

<https://doi.org/10.32657/10356/144577>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:21:09 SGT

INCORPORATING EXTERNAL KNOWLEDGE INTO MACHINE LEARNING ALGORITHMS FOR NLP APPLICATIONS

PENGFEI LI

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

01/10/2020

.....

Date

.....

PENGFEI LI

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

01/10/2020

.....

Date

Mao Kezhi

.....

Prof. KEZHI MAO

Authorship Attribution Statement

This thesis contains material from 4 papers published in the following peer-reviewed journals / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Pengfei Li, Kezhi Mao, Yuecong Xu, Jiaheng Zhang and Qi Li. “Bag-of-Concepts Representation for Document Classification based on Automatic Knowledge Acquisition from Probabilistic Knowledge Base.” *Knowledge-based Systems*, vol. 193, p. 105436. 2020.

The contributions of the co-authors are as follows:

- I proposed the idea, constructed the model, designed the experiments, and prepared the manuscript.
- Prof Kezhi Mao provided ideas for refined approaches and revised the manuscript.
- Yuecong Xu, Jiaheng Zhang, Qi Li, and I conducted the experiments.

Chapter 4 is published as Pengfei Li and Kezhi Mao. “Knowledge-oriented Convolutional Neural Network for Causal Relation Extraction from Natural Language Texts.” *Expert Systems with Applications*, vol. 115, pp.512-523. 2019.

The contributions of the co-authors are as follows:

- I proposed detailed methodologies, constructed the model, designed the experiments, and prepared the manuscript.
- Prof Kezhi Mao provided the initial idea and revised the manuscript.

Chapter 5 is published as Pengfei Li, Kezhi Mao, Xuefeng Yang and Qi Li. “Improving Relation Extraction with Knowledge-attention.” 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)*, pp. 229-239. 2019.

The contributions of the co-authors are as follows:

- I proposed the idea, constructed the model, designed the experiments, and prepared the manuscript.
- Prof Kezhi Mao provided the project direction and suggestions.
- Qi Li and I conducted the experiments.
- Xuefeng Yang and Kezhi Mao revised the manuscript.

Chapter 6 is published as Pengfei Li, Peixiang Zhong, Jiaheng Zhang and Kezhi Mao. “Convolutional Transformer with Sentiment-aware Attention for Sentiment Analysis.” *International Joint Conference on Neural Networks (IJCNN)*. 2020.

The contributions of the co-authors are as follows:

- I proposed the idea, constructed the model, designed the experiments, and prepared the manuscript.
- Prof Kezhi Mao provided the project direction and suggestions.
- Peixiang Zhong and I conducted the experiments.
- Jiaheng Zhang and Kezhi Mao revised the manuscript.

01-10-20

.....

Date

.....

PENGFEI LI

Acknowledgements

First of all, I wish to express my sincerest gratitude to my supervisor Prof. Kezhi Mao for his continuous guidance and support throughout my Ph.D. study. His motivation, immense knowledge, and attitude towards research have influenced me deeply to be a better researcher. Prof. Mao has patiently taught me the methodologies of carrying out research works and generously spent much time and effort in discussing research ideas with me, offering advice and recommendations, reading and revising my draft papers. Almost all the research works presented in this thesis were accomplished under his guidance, I am extremely grateful for what he has offered me.

Besides my supervisor, I would also like to thank the rest of my thesis advisory committee: Prof. Lihui Chen and Prof. Chng Eng Siong, for their encouragement, insightful comments, and invaluable suggestions.

Special thanks to my seniors: Dr. Rui Zhao, Dr. Dongzhe Wang, Dr. Zuobin Wu, and Dr. Xuefeng Yang, who have provided me a lot of invaluable advice for research, career, and even life. Whenever I need support, they are always there willing to help. As excellent researchers, they also set good examples to me which motivate me a lot during my Ph.D. study.

I would also like to thank my fellow lab mates in the Internet of Things Laboratory: Dr. Qinxu Ding, Dr. Zhenghua Chen, Dr. Jie Ding, Dr. Xueheng Qiu, Dr. Xuhao Li, Dr. Qingchang Zhu, Dr. Yong Zhang, Dr. Hongxin Shao, Dr. Xiaolei Li, Dr. Chao Deng, Dr. Ci Chen, Dr. Qianwen Xu, Dr. Kemi Ding, Zhen Cao, Jianfei Yang, Jiaqi Yan, Xu Fang, Yuecong Xu, Qi Li, Jiaheng Zhang, and Haozhi Cao. It was a great privilege and honor to work with these talented and amazing guys. It was also these guys who made my graduate study memorable and joyful!

Last but not least, I would like to thank my beloved family members. This work would not have been possible without the endless love, care, support, and understanding provided by my family. Special thanks to my parents who give birth to

me in the first place, and sacrifice themselves to educate and prepare me for the future. Special thanks to my aunt and uncle-in-law who provided me a lot of help and guidance throughout my life. Special thanks to my wife, who has spent a lot of memorable time with me and helped me survive all the stress. Long last our love!

Pengfei Li, June 2020

“If you know you are on the right track, if you have this inner knowledge, then nobody can turn you off... no matter what they say..”

—Barbara McClintock

To my dear family and friends

Abstract

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI) that mainly uses machine learning algorithms to process and analyze large amounts of text data. It gives machines the ability to read, understand, derive meaning from human languages, and potentially generate human languages. The key issue in the modern statistical NLP is text representation learning that transforms unstructured text data into structured numerical representations. A good text representation shall capture important lexical, syntactic, and semantic information for certain NLP tasks, such as keywords and cue phrases, conceptual information, and long-distance dependencies.

Traditional Bag-of-Words (BoW) model represents text as a fixed-length vector of words, where each dimension is a numerical value such as frequency or tf-idf weight. However, BoW simply looks at the surface form of words and suffers from high dimensionality and sparsity issues. Deep neural networks have shown to be more effective since word order information can be utilized and more semantic features can be captured. The commonly adopted deep neural network architectures include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Transformer. However, deep neural network normally requires large amount of training data, heavy computations, and sufficient CPU/GPU memory. The lack of high-quality training data can easily lead to under-fitting or over-fitting issue, especially for data-driven deep neural networks. Besides, hardware constraints as well as poor interpretability often become obstacles for deep neural networks to be applied in real-world NLP applications.

External knowledge is proven to be beneficial for machine learning algorithms to reduce the reliance on training data and provide additional useful information. For natural language, abundant publicly available knowledge bases such as WordNet, FrameNet, Wikipedia, etc. can be utilized for various NLP tasks. However, different tasks require different knowledge, and different machine learning models

have different architectures and operations. How to effectively incorporate useful external knowledge into machine learning algorithms remains an open research question.

This thesis focuses on incorporating existing knowledge from external knowledge bases into machine learning algorithms as prior knowledge for NLP applications. By utilizing external knowledge, we aim at obtaining better text representations, reducing the model’s reliance on training data, and improving model interpretability. We demonstrate the advantages of leveraging both data and knowledge in machine learning systems and provide general frameworks of incorporating external knowledge into different machine learning algorithms to improve their performance on various NLP tasks. Specifically,

- For BoW model, we show how to utilize the conceptual knowledge from a probabilistic knowledge base (Probase) and construct a Bag-of-Concepts (BoC) representation to provide more semantic and conceptual information of text and better interpretability for document classification.
- For CNN, we demonstrate how to automatically generate convolutional filters from lexical knowledge bases such as WordNet and FrameNet to improve its ability to capture the keywords and cue phrases for causal relation extraction.
- For Transformer, we propose a complementary knowledge-attention encoder which incorporates prior knowledge from lexical knowledge bases to better capture the important linguistic clues. We also propose three effective ways of integrating knowledge-attention with the self-attention in Transformer to maximize the utilization of both knowledge and data for relation extraction.
- For neural networks using attention mechanism, we show how to incorporate word’s sentiment intensity information from SentiWordNet into attention mechanism for sentiment analysis task. In addition, we propose two novel neural architectures including Convolutional Transformer (ConvTransformer) and Attentive Convolutional Transformer (ACT) which take the advantages of both CNN and Transformer for efficient text representation.

Contents

Acknowledgements	i
Abstract	iii
List of Figures	ix
List of Tables	xi
Acronyms and Abbreviations	xiii
1 Introduction	1
1.1 Overview and Motivation	1
1.2 Scope and Objectives	4
1.3 Major Contributions	5
1.4 Organization of the Thesis	7
2 Literature Review of Text Representations	9
2.1 BoW-based Models	10
2.1.1 Bag-of-Words Model	10
2.1.2 Topic Modeling	11
2.2 Word Embedding	14
2.2.1 Word2Vec and GloVe	15
2.2.2 Contextualized Word Representations	17
2.3 Deep Neural Networks	18
2.3.1 Convolutional Neural Network	19
2.3.2 Recurrent Neural Network	21
2.3.3 Attention Mechanism	25
2.3.4 Transformer	28
3 Incorporate Knowledge into Bag-of-Words	31
3.1 Introduction	31
3.2 Related Work	34
3.2.1 Representation based on data	35
3.2.2 Representation leveraging knowledge bases	36
3.3 Bag-of-Concepts (BoC) representation	38

3.3.1	Offline Knowledge Acquisition	38
3.3.2	Online Representation Learning	43
3.4	Refined Approaches	47
3.4.1	Bag-of-Concept-Clusters (BoCCL)	47
3.4.2	Conceptualization with entity sense disambiguation	49
3.4.3	Combination of BoCCL and BoW Representations	50
3.5	Experiments	52
3.5.1	Datasets	52
3.5.2	Experimental Settings	53
3.5.3	Results and Analyses	55
3.6	Conclusion	64
4	Incorporate Knowledge into Convolutional Neural Network	65
4.1	Introduction	65
4.2	Related Work	68
4.2.1	Models with Sophisticated Feature Engineering	68
4.2.2	Deep Neural Network Models	70
4.3	Knowledge-oriented CNN (K-CNN)	72
4.3.1	Knowledge-oriented Channel	72
4.3.2	Data-oriented Channel	80
4.3.3	Additional Features for Causal Relation Extraction	81
4.3.4	Regularization and Classification	83
4.4	Experiments	84
4.4.1	Datasets	84
4.4.2	Experimental Settings	85
4.4.3	Results and Analyses	86
4.5	Conclusion	91
5	Incorporate Knowledge into Transformer	93
5.1	Introduction	93
5.2	Related Work	94
5.3	Knowledge-attention Encoder	95
5.3.1	Relation Indicators Generation	96
5.3.2	Knowledge Attention	97
5.3.3	Position-aware Attention	98
5.4	Integrate Knowledge-attention with Self-attention	99
5.4.1	Multi-channel Attention	100
5.4.2	Softmax Interpolation	101
5.4.3	Knowledge-informed Self-attention	101
5.5	Experiments	102
5.5.1	Baseline Models	102
5.5.2	Experimental Settings	103
5.5.3	Results and Analyses	104

5.6	Conclusion	109
6	Incorporate Knowledge into Attention Mechanism	111
6.1	Introduction	111
6.2	Related Work	113
6.3	ConvTransformer with Sentiment-aware Attention	115
6.3.1	Convolutional Transformer Encoder	115
6.3.2	Sentiment Embedding Generation	118
6.3.3	Sentiment-aware Attention	119
6.3.4	Classification and Optimization	120
6.4	Experiments	121
6.4.1	Datasets	121
6.4.2	Experimental Settings	121
6.4.3	Results and Analyses	122
6.4.4	Ablation Study	124
6.4.5	Attention Visualization	125
6.5	Conclusion	125
7	Attentive Convolutional Transformer	127
7.1	Introduction	127
7.2	Methodology	128
7.2.1	Attentive Convolution Mechanism	128
7.2.2	Multi-head Multi-layer Attentive Convolution	131
7.2.3	Global Attention and Classification	132
7.3	Experiments	133
7.3.1	Datasets	133
7.3.2	Baseline Models	134
7.3.3	Experimental Settings	135
7.3.4	Results and Analyses	135
7.4	Discussions	137
7.4.1	Ablation Study	137
7.4.2	Hyper-parameter Study	138
7.4.3	Attention Visualization	139
7.4.4	Model Size and Inference Speed	140
7.5	Conclusion	140
8	Conclusion and Recommendations	141
8.1	Conclusion	141
8.2	Recommendations for Future Research	142
	List of Author's Publications	145
	Bibliography	147

Appendices	167
A Proof of the Cosine Similarity Value in K-CNN Feature Map . . .	169
B FramNet Frames Used for Relation Extraction on TACRED Dataset	170
C Hyper-parameter Settings and Training Details of Knowledge-attention	
Encoder	171
D Attention Visualization	172

List of Figures

1.1	Human Learning vs Machine Learning. This thesis focuses on automatically incorporate prior knowledge from open knowledge bases into machine learning algorithms.	3
1.2	Overall structure and mind map of the thesis.	7
2.1	Example of BoW representations for a corpus of two documents. . .	11
2.2	LSA using truncated Singular Value Decomposition.	12
2.3	Graphical models of pLSA (left) and LDA (right).	13
2.4	Comparison between one-hot word vector and word embedding [1]. .	15
2.5	Word embeddings are usually used as the input to deep neural networks for text representation learning.	15
2.6	Continuous Bag-of-Words and Skip-Gram models for Word2Vec training [2].	16
2.7	Model architectures of BERT, OpenAI GPT, and ELMo [3].	18
2.8	How does the CNN work on text data [4]. 6 convolutional filters with different region sizes are used here for illustration.	20
2.9	Sequential processing in RNN [5].	22
2.10	Structures of bidirectional RNN (left) and deep bidirectional RNN (right).	24
2.11	A typical implementation of attention mechanism [6]	26
2.12	Architecture of hierarchical attention network [7].	27
2.13	Model architecture of Transformer [8]. Scaled dot-product attention is used as the basic operation in the multi-head attention of Transformer.	29
3.1	Overall architecture of BoC model.	38
3.2	Example of concept and entity extraction from Probase using 3-level iterations based on the class label “hardware”.	42
3.3	The flow chart of Backward Maximum Matching (BMM) for entity and concept recognition.	44
3.4	Illustration of clustering of similar concepts.	48
3.5	Combination of BoCCl and BoW representations using attentional neural network.	51
4.1	Overall architecture of K-CNN.	72
4.2	Knowledge-oriented channel of K-CNN.	73

4.3	Effect of the number of filters in data-oriented channel. “SemEval”, “Event-SL”, “Causal-TB” indicate SemEval-2010 task 8, Event StoryLine and Causal-TimeBank datasets respectively.	86
4.4	Learning curves of K-CNN and CNN_multiple for SemEval-2010 task 8 (top), Causal-TimeBank (middle) and Event StoryLine (bottom) datasets. Squared dot indicates the highest score achieved on validation set.	89
5.1	Knowledge-attention process (left) and multi-head structure (right) of knowledge-attention encoder.	96
5.2	Three ways of integrating knowledge-attention with self-attention: multi-channel attention and softmax interpolation (top), as well as knowledge-informed self-attention (bottom).	100
5.3	Knowledge-informed self-attention structure. Q , K represent input matrix and knowledge indicators respectively, h is the number of attention heads.	102
5.4	Change of precision, recall and F_1 score on dev set as the priority weight β in softmax interpolation changes.	105
6.1	Overall architecture of the proposed sentiment analysis system. . . .	115
6.2	Convolutional Transformer (ConvTransformer).	116
6.3	Sentiment-aware attention mechanism. Attention weights α_i are calculated based on ConvTransformer output $\mathbf{o}_i$, sentiment embedding $\mathbf{s}_i$ and position embedding $\mathbf{p}_i$ of each token.	120
7.1	Left: attentive convolution mechanism. Outputs are obtained by combining convolutional filters attentively utilizing feature map as attention weights. Right: multi-head multi-layer structure of ACT. h and N indicate number of attentive convolution heads and layers respectively.	129
7.2	Global attention mechanism. Attention weights α_i are calculated based on local representation $\mathbf{o}_i$, global representation $\mathbf{g}$, and position embedding $\mathbf{p}_i$ of each token.	132
7.3	Hyper-parameter study on ACT. X-coordinate indicates the hyper-parameters studied, Y-coordinate indicates classification accuracy for Yelp F. dataset and micro-averaged F_1 score for TACRED dataset.	138

List of Tables

3.1	Statistics of six document classification datasets.	53
3.2	Number of concepts and entities automatically acquired from Probase for six different document classification tasks.	55
3.3	Classification accuracies (%) of BoW, LSA and LDA using different weighting schemes.	56
3.4	Classification accuracies (%) of BoC using different weighting schemes.	56
3.5	Classification accuracies (%) of various models on six document classification datasets.	58
3.6	Pair-wise comparison of algorithms using Wilcoxon Signed-Rank Test with $\alpha = 0.05$. Z statistics are shown above the diagonal, bold indicates significantly different between algorithms with $z < -1.96$. Comparisons of the algorithms in the row with the column are shown below the diagonal, + and - indicate significantly better or worse respectively; = indicates no statistical significant difference.	58
3.7	Dimensionality of different document representations learned from six datasets.	60
3.8	Top 3 features of different representations for two documents from “sport” and “entertainment” categories respectively.	62
3.9	Classification accuracy (%) of 50 samples from BBC dataset based on human judgment of the top 5 weighted features of each algorithm. “avg” and “std” indicate average and standard deviation values of the three evaluators respectively.	63
4.1	Macro-averaged precision (P), recall (R) and F1 scores of K-CNN without and with data-oriented channel. Scores are shown in percentage (%).	87
4.2	Macro-averaged F1 score (%) of various model architectures for causal relation extraction task on three datasets.	87
4.3	Macro-averaged precision (P), recall (R) and F1 scores of K-CNN without and with semantic features (SF). Scores are shown in percentage (%).	91
5.1	Micro-averaged precision (P), recall (R) and F_1 score on TACRED dataset. †, ‡ and †† mark the results reported in [9], [10] and [11] respectively. * marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.	104

5.2	Results of adding NER embeddings and entity categorical embeddings to the multi-channel attention (MCA) integrated model. . . .	106
5.3	Macro-averaged F_1 score on SemEval2010-Task8 dataset. * marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.	106
5.4	Ablation study on knowledge-attention encoder. Results are the median F_1 scores of 5 independent runs on dev set of TACRED. . .	107
5.5	Attention visualization for knowledge-attention encoder (first) and self-attention encoder (second). Words are highlighted based on the attention weights assigned to them. Best viewed in color.	108
6.1	Example words and their sentiment intensities obtained from SentiWordNet. The (POS,ID) pair uniquely identifies a WordNet synset; <i>PosScore</i> and <i>NegScore</i> represent positivity score and negativity score respectively.	119
6.2	Statistics of the three sentiment analysis datasets used in our experiments.	121
6.3	Classification accuracy (%) on three sentiment analysis datasets. * means the results are obtained from our implementation. – means not reported. All other results are directly cited from the respective papers.	123
6.4	Ablation study on ConvTransformer. Classification accuracy is reported on the development set of IMDB.	124
6.5	Visualization of attention weights for sentiment-aware attention. Words are highlighted based on the attention weights assigned to them. Best viewed in color.	125
7.1	Statistics of the six text classification datasets used in our experiments.	133
7.2	Left: classification accuracy (%) on sentiment analysis and topic categorization tasks. Right: F_1 scores on relation extraction task, official micro-averaged and macro-averaged F_1 scores are used for TACRED and SemEval2010-task8 datasets respectively. * means the results are obtained from our implementation. / means not reported. All other results are directly cited from the respective papers mentioned in Section 7.3.2.	136
7.3	Ablation study on ACT. Accuracy (%) and micro-averaged F_1 score are reported on the development sets of Yelp F. and TACRED respectively.	138
7.4	Attention visualization for Transformer and ACT. For each sample, the visualization of Transformer is presented first, followed by our proposed ACT. Words are highlighted based on the attention weights assigned to them. Best viewed in color.	139
7.5	Comparison of model parameters and inference time per batch on Yelp F. dataset.	140

Acronyms and Abbreviations

AI	Artificial Intelligence
ANOVA	Analysis of Variance
BMM	Backward Maximum Matching
BoW	Bag-of-Words
BPTT	Back-Propagation Through Time
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
CV	Cross Validation
ELMo	Embedding from Language Models
FFNN	Feed-Forward Neural Network
GloVe	Global Vectors for Word Representation
GPT	Generative Pre-Training
GRU	Gated Recurrent Unit
IE	Information Extraction
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NN	Neural Network
pLSA	Probabilistic Latent Semantic Analysis
POS	Part-of-Speech
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machine
tanh	Hyperbolic Tangent

ACT	Attentive Convolutional Transformer
BoC	Bag-of-Concepts
BoCCI	Bag-of-Concept-Clusters
ConvTransformer	Convolutional Transformer
FE	Frame Element
K-CNN	Knowledge-oriented Convolutional Neural Network
KISA	Knowledge-Informed Self-Attention
Knwl-attn	Knowledge-attention Encoder
MCA	Multi-Channel Attention
SI	Softmax Interpolation

Chapter 1

Introduction

1.1 Overview and Motivation

In this era of information explosion, text data is everywhere, from news articles, blogs, and social media, to customer feedback, machine logs, and medical reports. Irrespective of industry, company, or individual, the demands of automatic text analyses using Natural Language Processing (NLP) techniques are continuously increasing for better-informed decisions and uncovering valuable insights from texts. Some important NLP applications include document categorization, relation extraction, sentiment analysis, question answering, machine translation, natural language generation, etc.

In the early days, most NLP systems were rule-based systems that consist of complex sets of hand-written rules based on human prior knowledge [12, 13]. In spite of extensive human effort, the performance of such rigid rule-based systems is not satisfying. With the development of machine learning techniques, statistical NLP becomes dominant nowadays [14, 15]. The machine-learning paradigm enables automatically learning of important rules through statistical analysis and inference based on a large amount of real-world examples (corpus). It is known that a good data representation determines the upper-bound performance of machine learning algorithms [16]. The key issue in statistical NLP is text representation learning that transforms unstructured text data into structured numerical representations, where important lexical, syntactic, and semantic information for certain NLP tasks can be entangled [17].

Traditional machine learning algorithms such as Naive Bayes, SVM, and decision trees rely on sophisticated feature engineering to construct hand-crafted features (lexical, syntactic, or semantic features) for text representation. Such representations require extensive human effort, and they are normally domain-dependent since domain knowledge is needed to construct the features. The commonly adopted Bag-of-Words (BoW) representation is able to automatically generate lexical features from text corpus [18–20]. However, BoW model simply looks at the surface form of words and ignores the semantic and order information of texts. Besides, BoW representation suffers from high dimensionality and sparsity problems.

With the development of word embeddings, a distributed representation of words in vector space, semantic and syntactic properties of words can be captured in low-dimensional and dense space [2, 21–24]. Meanwhile, with the prevalence of deep learning [25], deep neural networks based on word embeddings as sequential input have been widely applied in NLP and achieved promising performances. A deep neural network contains multi-layer neural networks with various forms, including convolutional [26, 27], recurrent [28–30], recursive [31, 32] neural networks, as well as attention-based Transformers [3, 8] to learn text representations by performing compositions over word embeddings. The advantages of deep neural networks are that word order and contextual information are preserved, and more complex latent semantics can be discovered by training the models with a large amount of data.

However, deep neural networks suffer from the following shortcomings in NLP applications:

- **Data-driven:** Deep neural network normally requires sufficient high-quality and annotated training corpus to train its enormous amount of model parameters. This is often impractical due to the complexity of natural language, including lexical, morphological, and syntactic variations. The lack of training data can easily lead to under-fitting or over-fitting issue.
- **Computational resource demand:** Due to the large number of model parameters, deep neural networks normally involve heavy computations and require sufficient CPU/GPU memory. The memory and speed constraints will become obstacles for real applications.

- **Poor interpretability:** The text representations obtained from deep neural networks lack interpretability since the features are computed through the complex composition of neural networks.

FIGURE 1.1: Human Learning vs Machine Learning. This thesis focuses on automatically incorporate prior knowledge from open knowledge bases into machine learning algorithms.

Despite that different classes of machine-learning algorithms have been applied to NLP tasks, the majority of them are data-driven. The motivation of our work comes from the human’s source of learning, which is from both our own experiences and the guide of teachers [33]. A good machine learning system shall mimic the human way of learning, as illustrated in Figure 1.1. For a machine learning system, the experiences come from a large amount of data, which is exactly the learning paradigm of majority machine learning algorithms. The teacher corresponds to the prior knowledge in the system which is incorporated from external knowledge resources. Prior knowledge provides additional reliable information and able to improve the performance of machine learning systems, especially when short of training data [34–36].

Although prior knowledge is widely applied in rule-based NLP systems by manually constructing rules from expert knowledge, they are seldom applied in machine learning algorithms as a unified data-knowledge co-learning paradigm. The reasons are: firstly, different tasks require different knowledge, manually generating such knowledge require domain expert and extensive human effort; secondly, different machine learning models have different architectures and operations, how to effectively incorporate prior knowledge into machine learning algorithms remains an open research question. Thanks to the abundant publicly available knowledge bases for natural language, we can easily obtain prior knowledge from these knowledge bases for NLP applications. Our work focus on automatically extracting

useful knowledge from open knowledge bases and investigating effective ways of incorporating such external knowledge into different machine learning algorithms.

1.2 Scope and Objectives

The machine learning algorithms considered in this thesis are the main-stream machine learning algorithms for NLP, including BoW, CNN, Transformer, and neural networks with attention mechanisms. RNN is not considered here since Transformer has shown superior performance than RNN in terms of both computational speed and long-distance dependency modeling capability [8]. We think Transformer is able to fully replace RNN for NLP applications.

The main objective of this work is to investigate the most effective ways of incorporating external knowledge into these machine learning algorithms and to solve or alleviate the limitations of the existing models. Specifically, by utilizing external knowledge, we aim at obtaining better text representations, alleviating overfitting issues, reducing the model’s reliance on training data, and improving model interpretability. To save us from manually constructing knowledge, the external knowledge used in our work are automatically extracted from open knowledge bases for natural language, including WordNet [37], FrameNet [38], SentiWordNet [39], Probase [40], and Thesaurus.com¹.

To verify the performance of our proposed approaches and demonstrate how external knowledge is incorporated into each machine learning algorithm, various NLP tasks are presented, including document classification, relation extraction (including causal relation extraction), and sentiment analysis. Note that the proposed approach for each machine learning algorithm is a general framework and a unified data-knowledge co-learning paradigm, whose applications are not limited to the tasks presented in this thesis. Readers can apply the ideas in this thesis to other NLP applications as well.

¹<https://www.thesaurus.com>

1.3 Major Contributions

The main contributions of this thesis are stated as follows:

- Bag-of-Concepts (BoC) Representation:** We first propose an efficient way of automatically acquiring useful knowledge from a large probabilistic knowledge base to reduce the search space as well as querying time, and also preserve the useful information for the certain classification task. Based on the acquired knowledge, we propose a text conceptualization scheme which reveals the higher-level semantics of text, and constructs a Bag-of-Concepts (BoC) document representation using a novel concept score - inverse soft document frequency ($cs - idf^s$) weighting scheme. Compared with BoW, BoC representation has lower dimensionality and sparsity, and able to capture the semantic relatedness and conceptual information of texts. In addition, we propose concept clustering and entity sense disambiguation techniques, and construct a Bag-of-Concept-Clusters (BoCCl) model to further improve BoC representation. Comprehensive experiments are conducted to evaluate both statistical and semantic qualities of the proposed BoC and BoCCl representations. Results demonstrate that Bag-of-Concepts representation is able to improve document classification performance and also provide great interpretability for human understanding.
- Knowledge-oriented CNN (K-CNN):** We propose an effective way of combining human prior knowledge and information from data for CNN. The convolutional filters in the knowledge-oriented channel of K-CNN are automatically generated from lexical knowledge bases, we call them word filters. Compared with conventional CNN, the proposed word filters are able to represent keywords and cue phrases more accurately. The specially designed convolutional operations allow the model to extract these linguistic clues from texts and output a similarity score for each word filter, this overcomes the problem of rigid pattern matching of rule-based methods. As word filters are static, the number of free parameters in K-CNN is significantly reduced compared with conventional CNN. Hence, overfitting issues can be alleviated and training efficiency can be improved. Besides, the word filters are more interpretable than random initialized convolutional filters since each word filter has a specific semantic meaning. Furthermore, two dimensionality reduction

techniques are proposed to improve the performance of K-CNN, including word filter selection and clustering techniques to remove insignificant and redundant word filters. We evaluate the proposed K-CNN on causal relation extraction task and demonstrate that K-CNN is able to extract causal relations more effectively and efficiently than conventional CNN models.

- **Knowledge-attention Encoder:** We propose a novel attention mechanism called knowledge-attention which incorporates prior knowledge from external lexical resources to effectively capture the informative linguistic clues for relation extraction. Furthermore, to maximize the utilization of both knowledge and data, we present three effective ways of integrating knowledge-attention with the self-attention in Transformer, including multi-channel attention, softmax interpolation, and knowledge-informed self-attention. Experiments and detailed analyses show that the proposed knowledge-attention mechanism has complementary strengths with self-attention, and the integrated models outperform existing CNN, RNN, and self-attention based models.
- **Sentiment-aware Attention:** We demonstrate how external knowledge can be incorporated into neural networks with attention mechanisms by taking sentiment analysis task as an example. We generate sentiment embeddings from an external knowledge base (SentiWordNet), which contains sentiment intensity information of words. A sentiment-aware attention mechanism is proposed to effectively incorporate sentiment embeddings while computing the attention weight of each word. By taking sentiment intensity information into consideration, the proposed sentiment-aware attention is able to capture the sentiment of text more accurately and also reduce the model’s reliance on training data. In addition, we propose a novel neural architecture, Convolutional Transformer (ConvTransformer), which combines the advantages of both CNN and Transformer for sentiment analysis.
- **Attentive Convolutional Transformer (ACT):** Based on ConvTransformer, we introduce another neural architecture called ACT which is able to perform efficient text classification. Specifically, we propose an attentive convolution mechanism that utilizes the semantic meaning of convolutional filters to transform text from complex word space to a more informative convolutional filter space where important n-grams are captured. ACT is able

to capture both local and global dependencies effectively while preserving sequential information of texts. Experiments on various text classification tasks demonstrate that the proposed ACT is a lightweight and effective universal text classifier, outperforming existing CNN, RNN, and attentive models including Transformer.

1.4 Organization of the Thesis

The overall structure of this thesis is illustrated in Figure 1.2 as a mind map.

FIGURE 1.2: Overall structure and mind map of the thesis.

Chapter 1 gives an overview of the field of NLP including major text representation methods and their limitations, which essentially motivates this research work. The scope and objectives of our work are briefly described, and the main contributions of this thesis are also stated.

Chapter 2 reviews the text representation methods in a systematic way, including BoW-based methods, word embeddings, and deep neural network based methods.

Chapter 3, Chapter 4, Chapter 5, and Chapter 6 present general frameworks of incorporating external knowledge into BoW, CNN, Transformer, and attention mechanism respectively. Specifically:

- Chapter 3 addresses the limitations of BoW representation and proposes Bag-of-Concepts (BoC) representations for document classification. The conceptual knowledge is automatically acquired from Probase, a probabilistic knowledge base.
- Chapter 4 introduces a knowledge-oriented CNN (K-CNN) for causal relation extraction. The word filters in K-CNN are automatically constructed from large lexical knowledge bases (WordNet and FrameNet), which are able to capture linguistic clues of causal relation effectively, and also reduce free parameters of the model significantly.
- Chapter 5 proposes a knowledge-attention encoder to improve Transformer’s performance on relation extraction task. The relation indicators in knowledge-attention are generated from WordNet and Thesaurus.com, which represent the keywords and cue phrases of various relations. Knowledge-attention is shown to have complementary strengths with self-attention, and three integration methods are proposed to achieve optimal performance.
- Chapter 6 presents sentiment-aware attention for sentiment analysis. Word’s sentiment intensity information is extracted from SentiWordNet and incorporated into the attention mechanism to better capture the overall sentiment of the text. In addition, a novel neural architecture called Convolutional Transformer (ConvTransformer) that takes the advantages of both CNN and Transformer is introduced in this chapter.

Chapter 7 proposes an effective, efficient, and universal text classifier called Attentive Convolutional Transformer (ACT), which is based on ConvTransformer. Instead of conventional convolution operation, a novel attentive convolution mechanism is proposed in this chapter which represents text in convolutional filter space.

Chapter 8 summarizes the thesis and provides recommendations for future research.

Chapter 2

Literature Review of Text Representations

In this chapter, we systematically review the popular text representation methods in the literature. In Section 2.1, Bag-of-Words (BoW) representations with different weighting schemes are introduced. Besides, topic models based on BoW representation are also reviewed, including Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA); In Section 2.2, word embedding is introduced, including the classic Word2Vec and GloVe models, as well as the contextualized word representations. The development of word embedding has made deep learning approaches possible for NLP applications. In Section 2.3, the mainstream deep neural networks in NLP are reviewed, including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), attention mechanism, and Transformer. Besides the introduction to the methodologies of these major text representation models, we also highlight the advantages and potential concerns behind these models, which essentially motivate our research work.

2.1 BoW-based Models

2.1.1 Bag-of-Words Model

BoW is a simple and intuitive method to represent unstructured text as a fixed-length vector [41]. The model represents a text (sentence or document) as a multi-set (bag) of its words, disregarding word order and grammar but keeping lexical features. There are two steps involved to construct BoW representations:

1. Find the vocabulary of words from the corpus (all the text data).
2. Measure the presence of words in the vocabulary for a specific text.

Figure 2.1 demonstrates BoW representations for a corpus of two simple documents. Each document is represented as a vector of all the words in the corpus. The word present in the document will have a value of one, and otherwise a value of zero. This is also known as one-hot encoding. The value in the one-hot encoded vectors is either 0 or 1, which can only tell whether the word appears in the document or not. However, other information such as the frequency of words that appeared in the document cannot be captured.

Other than one-hot encoding, there are other more sophisticated weighting schemes to measure the presence of words for BoW representation. The most effective and commonly adopted weighting scheme is called tf-idf weighting [20]. By considering both term frequency (tf) and inverse document frequency (idf), tf-idf weighting scheme reflects how important a word is to a document in a corpus. Formally, the tf-idf score for the word t in document d from the corpus D with a set of N documents is calculated as follows:

$$tf-idf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (2.1)$$

where $tf(t, d)$ and $idf(t, D)$ are normally defined as:

$$tf(t, d) = \log(1 + freq(t, d))$$

$$idf(t, D) = \log\left(\frac{N}{count(d \in D, t \in d)}\right)$$

Here, $freq(t, d)$ represents the frequency of word t in document d and $count(d \in D, t \in d)$ represents the number of documents in D which contains the word t .

	Document 1 I love machine learning so much						Document 2 I like natural language processing very much				
Term	I	language	learning	like	love	machine	much	natural	processing	so	very
Document 1	1	0	1	0	1	1	1	0	0	1	0
Document 2	1	1	0	1	0	0	1	1	1	0	1

FIGURE 2.1: Example of BoW representations for a corpus of two documents.

Due to simplicity and high efficiency, BoW is a commonly adopted text representation method for information retrieval and document classification [18, 19]. However, BoW suffers from the following shortcomings that hinder its performance:

- BoW simply looks at the surface form of words and ignores the semantic relatedness and conceptual information of words and phrases.
- BoW ignores the word order of a text, hence the grammar and context information of the text is lost.
- BoW suffers from high dimensionality and sparsity since it uses hard mapping of the terms in the whole vocabulary for representation.
- BoW focuses on single words without considering phrases. However, the phrase often deliver more semantic information than its constituent parts. Although we could consider n-grams as base terms in BoW model, it will further increase dimensionality and sparsity, and most of the n-grams are meaningless.

The above-mentioned limitations of BoW representation have led to our proposed Bag-of-Concepts (BoC) representation, which is presented in Chapter 3.

2.1.2 Topic Modeling

Topic modeling is a popular approach to address the high dimensionality and sparsity issues of BoW representation by discovering the latent semantics hidden behind the document. The idea behind topic modeling is that the semantics of documents are shaped by some latent variables (topics) that we are not observing. The goal of topic modeling is to uncover these topics from term–document matrix (i.e. BoW representation). Topic models are built based on the assumptions that each document consists of a mixture of topics, and each topic consists of a collection of words. Popular topic modeling approaches include Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA).

FIGURE 2.2: LSA using truncated Singular Value Decomposition.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) [42] is essentially a matrix decomposition technique that decomposes the term-document matrix into a separate term-topic matrix and a document-topic matrix. The matrix decomposition is achieved by a matrix factorization technique called truncated Singular Value Decomposition (SVD). As illustrated in Figure 2.2, the term-document matrix X is first factorized into the product of 3 separate matrices U , S , and V^T , where S is a diagonal matrix of the singular values of X . By selecting the t largest singular values and keeping only the first t columns of U and V , the matrix X can be approximated using low-rank matrices:

$$X \approx U_t \times S_t \times V_t^T \quad (2.2)$$

The t columns in U_t and V_t correspond to the most significant dimensions in the transformed space. In the case of topic modeling, t is the number of topics we want to find. U_t is a term-topic matrix with the rows representing term vectors expressed in terms of topics, and V_t is a document-topic matrix with the rows representing document vectors expressed in terms of topics.

LSA is a fast and efficient algorithm, but it has several drawbacks:

- The representation is not efficient enough since it is a distributional model based on the corpus statistics. Besides, a large corpus is needed to get an accurate result.
- Since LSA is a linear model, it is not the best solution to handle non-linear dependencies.
- The number of topics is decided based on heuristics and needs some expertise.
- The topics learned lack interpretability.

FIGURE 2.3: Graphical models of pLSA (left) and LDA (right).

Probabilistic Latent Semantic Analysis

Instead of matrix factorization, Probabilistic Latent Semantic Analysis (pLSA) [43] tackles the problem using a probabilistic approach. The idea of pLSA is to learn a probabilistic model with latent topics which is able to generate the observed documents in the corpus. Particularly, for any document d and word w , pLSA models the probability of the word appears in the document ($P(w|d)$), which corresponds to the entry in the document-term matrix.

The graphical model of pLSA is shown in Figure 2.3 (left). The topics c are first drawn from a document d , the word w is then drawn from the topics. Formally, $P(w|d)$ is calculated from Equation (2.3).

$$P(w|d) = P(d) \sum_c P(c|d)P(w|c) \quad (2.3)$$

where topic c is drawn from the document-topic distribution $P(c|d)$, and word w is drawn from word-topic distribution $P(w|c)$. $P(c|d)$ and $P(w|c)$ are the multinomial distributed probabilistic models, $P(d)$ is determined from the corpus. The parameters of the model can be trained using the expectation-maximization (EM) algorithm.

One of the main problems of pLSA is that it is difficult to assign probability $P(d)$ to a new document since no parameters are learned for new documents. Another problem is that the number of parameters for $P(c|d)$ grows linearly with the number of documents, thus the model is prone to overfitting.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [44] is a generalized version of pLSA, which adds Dirichlet priors on top of document-topic and word-topic distributions, as shown in Figure 2.3 (right). As a brief description, the generative process for each word in a document is: we first select a random sample from a Dirichlet distribution $Dir(\alpha)$, which represents the topic distribution (θ) of a particular document. From θ , we select a particular topic c based on the distribution. Next, we draw a random sample from another Dirichlet distribution $Dir(\beta)$, which represents the word distribution (ϕ) of the topic c . Finally, we can generate the word w from ϕ . Readers may refer to [44] for more detailed mathematical operations in LDA.

pLSA and LDA normally have better topic interpretability compared with LSA since they focus on the generative process of the documents. Compared with pLSA, LDA is able to generalize to new documents easily since the documents are sampled according to the Dirichlet distribution of document-topic distributions learned from training data. However, the representation of LDA may not yield good classification results since it is a generative instead of a discriminative model.

2.2 Word Embedding

The idea of word embedding comes from vector semantics, which is to represent a word as a point in multidimensional semantic space by learning representations of the meaning of words directly from their distributions in the text corpus. Word embedding is a distributed representation of words in vector space, where the semantic and syntactic properties of words can be captured in low-dimensional and dense space [21, 22]. As illustrated in Figure 2.4, compared with the one-hot word vectors in the BoW model, word embeddings are continuous vector representations in low-dimensional space, and each dimension of the word embedding represents certain latent features such as syntax or semantic meanings. The core merit of word embedding is that similar words have similar representations, and the semantic similarity between words can be conveniently evaluated by the distance between word embeddings in the vector space. The development of word embedding has led to a key breakthrough in using deep neural networks to tackle complex NLP

FIGURE 2.4: Comparison between one-hot word vector and word embedding [1].

FIGURE 2.5: Word embeddings are usually used as the input to deep neural networks for text representation learning.

problems. As a convention, word embeddings are usually used as the input to the first layer of neural networks, known as embedding layer, as shown in Figure 2.5.

2.2.1 Word2Vec and GloVe

Word embeddings can be obtained through unsupervised learning from large-scale text corpus. The two most famous models for learning word embeddings are Word2Vec and GloVe.

Word2Vec is a method for efficiently learning word embeddings from text corpus by training with a simple one-layer neural network [2, 23]. There are two learning models with different objectives to train the neural network, one is called Continuous Bag-of-Words model or CBOW; the other is called Skip-Gram model. As

FIGURE 2.6: Continuous Bag-of-Words and Skip-Gram models for Word2Vec training [2].

illustrated in Figure 2.6, the objective of CBOW is to predict the word given its context words, whereas the objective of Skip-gram is to predict context words of a given word. Both models focus on learning about words from their usage in a local context, where the local context is defined by a window of neighboring words. Two techniques including hierarchical softmax and negative sampling are also proposed to improve the training efficiency of the Word2Vec model. Readers may refer to [23] for details.

GloVe stands for Global Vectors for Word Representation, which is a global log-bilinear regression model for the unsupervised learning of word representations based on ratios of word co-occurrence probabilities [24]. Different from traditional matrix factorization techniques like LSA, GloVe pre-processes the word co-occurrence matrix by normalizing and log-smoothing the count based on the context of words. Different from Word2Vec that defines local context using a window, GloVe constructs an explicit word co-occurrence (or word-context) matrix using the statistics across the corpus. Hence, GloVe enables local context-based learning of the matrix factorization technique and results in better word embeddings.

The key benefit of Word2Vec and GloVe is that high-quality word embeddings can be learned efficiently from large corpus. The pre-trained word embeddings are publicly available¹ and can be easily applied to downstream NLP tasks. However,

¹Word2Vec and GloVe word embeddings can be downloaded from <https://code.google.com/archive/p/word2vec> and <https://nlp.stanford.edu/projects/glove> respectively.

one prominent issue of such word embedding is that polysemous words cannot be well-handled. Since each word is assigned to one word embedding regardless of any context surrounding it, the word embedding cannot distinguish different meanings of the word.

2.2.2 Contextualized Word Representations

Contextualized word representation offers an elegant solution to the polysemous problem by generating word embeddings dynamically based on the context of words. Different from Word2Vec and GloVe whose word embeddings are “static” no matter what the contexts of words are, the contextualized word representation changes dynamically with the changing of context scene. In other words, the word embedding is generated based on the current context that the word located in. Therefore, the polysemous issue can be resolved since different meanings of a word involve different contexts.

ELMo (Embedding from Language Models) was the first model for learning deep contextualized word representations [45]. The model is trained to gain language understanding by performing the Language Modeling task, which is to predict the next word in a sequence of words. The model architecture of ELMo is shown in Figure 2.7 (ELMo). It consists of two layers of bi-directional LSTM networks. Hence, ELMo is able to look at the entire sentence (the context before and after) before assigning an embedding to each word. After pre-training the model with large corpus, the hidden states of different LSTM layers can be then utilized as word embeddings for downstream tasks. The word embeddings generated from the lower layer capture more syntax information, while the upper layer captures more semantic information.

GPT (Generative Pre-Training) is another unsupervised language model developed by OpenAI [46]. The model architecture is shown in Figure 2.7 (OpenAI GPT). Following the similar idea of ELMo, GPT also performs the language modeling task on large corpus to gain language understanding. The differences are that: (1) Instead of using RNN as the feature extractor, GPT uses a multi-layer Transformer decoder [8], which has demonstrated better performance than RNN. (2) GPT performs one-way language modeling which only considers the context before the word

FIGURE 2.7: Model architectures of BERT, OpenAI GPT, and ELMo [3].

to predict the next word. (3) The usages of contextualized embeddings in downstream tasks are different: ELMo is a feature-based approach that the embeddings are fed into task-specific models as additional features, while GPT is a fine-tuning approach which uses the same base model for all end tasks.

BERT is an elegant model that combines the advantages of both ELMo and GPT for language understanding [3]. It performs a two-way language modeling task using deep bidirectional Transformers, as shown in Figure 2.7 (BERT). Instead of shallow concatenation of the hidden states as in ELMo, BERT uses a masked language model task to enable deep bidirectional representations by jointly conditioning on both the contexts before and after. In addition, the next sentence prediction task is also proposed for better understanding of the relationship between two sentences.

The pre-trained language models like BERT have achieved state-of-the-art performances for many NLP tasks, demonstrating their outstanding language understanding capability. However, the main drawback is that the models are quite heavy which normally contain hundreds of millions of parameters. Hence, sufficient training data and computational resources are needed. The training and inference speed of the models are also slow, which hinders their practical applications.

2.3 Deep Neural Networks

With the development of deep learning [25], deep neural networks with various forms have been widely applied in the field of NLP for text representation [47]. The advances in computational techniques and more data resources available for training have enabled deep neural networks to learn more semantics and better text representations. Deep neural network based approaches have demonstrated better performance than traditional feature-based and BoW-based approaches, especially

for complicated NLP tasks that require more semantic understandings, such as relation extraction and sentiment analysis.

A deep neural network is composed of interconnected nodes (neurons) that are able to perform linear and non-linear transformations of input values. In NLP, deep compositions over word embeddings are normally performed by neural networks to learn representations for sentences or documents. Depending on how neurons are connected in the neural network, various neural architectures can be involved in the deep composition processes. The most basic neural architecture, known as Feed-Forward Neural Network (FFNN), organizes the layers of neurons in a sequential manner, where each neuron receives inputs only from the neurons in the earlier layer. Besides FFNN, commonly adopted neural architectures in NLP also include CNN, RNN, attention mechanism, and Transformer. We review these neural architectures and their applications in NLP in this section.

2.3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is originally applied in the field of computer vision for image feature extraction [48, 49]. It is a special feed-forward neural network whose layers are formed by convolution operations followed by pooling operations, where the convolutional kernels or filters allow for the simultaneous analysis of different features in the data [50, 51].

In the field of NLP, CNNs were first applied to sentence classification task [26, 27]. These early works provide a general framework of how to use CNN for text representation. As illustrated in Figure 2.8, 1-d convolution operation is first performed on the embeddings of the input sentence using convolutional filters. Here, the convolution operation is illustrated using 6 filters with region/kernel sizes varying from 2 to 4. These filters are the main parameters to be trained on CNN. After the convolution, each filter will result in a feature map. Mathematically, given the input embedding matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$ ($\mathbf{x}_i \in \mathbb{R}^d$), and a convolutional filter $\mathbf{f} \in \mathbb{R}^{kd}$, where d is the dimensionality of word embedding, and k is the convolutional kernel size. The corresponding feature map $\mathbf{m} = [m_1, m_2, \dots, m_{n-k+1}]^T$ is generated as follows:

$$m_i = f(\mathbf{f}^T \cdot \text{Cat}(\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+k-1})) + b \quad (2.4)$$

FIGURE 2.8: How does the CNN work on text data [4]. 6 convolutional filters with different region sizes are used here for illustration.

where Cat means concatenation of vectors, f is a non-linear activation function, and b is a bias term.

After obtaining the feature map, max-pooling operation is conducted to reduce the dimensionality of the feature map and obtain the most salient features in the feature map. In the case of global max-pooling or 1-max pooling, the largest value is selected from each feature map, as shown in Equation (2.5).

$$h = \max(\mathbf{m}) = \max(m_1, m_2, \dots, m_{n-k+1}) \quad (2.5)$$

The max-pooling results from all the feature maps are then concatenated together to form the final sentence representation: $\mathbf{h} = [h_1, h_2, \dots, h_c]^T$, where c is the number of convolutional filters. Then, softmax classifier can be used to make classification based on the sentence representation $\mathbf{h}$.

Based on the above-mentioned framework, CNN has been extensively explored for

text representation, and different variants of CNN were proposed, such as character-level CNN [52], multi-level attention CNNs [53], densely connected CNNs [54], and Deep CNNs [55, 56]. The advantages of CNN are that it has a very good local feature extraction capability since the convolutional filters are able to capture the important n-gram features from text, and the computation is very efficient since the convolution operation can be easily set up for parallel computing. However, there are several limitations of CNN for learning text representations. Firstly, CNN is relatively weak in capturing long-distance dependencies of text, because the convolutional filters normally have small kernel size focusing only on local n-grams. Although we could apply dilated CNN [57] or construct deep CNNs with one layer stack on another to widen the convolution context to some extent, the performance gain is normally marginal with the cost of more data needed [58]. Secondly, some sequential information of text may be lost since the pooling operation ignores the position information. Thirdly, the convolutional filters learned by CNN may be redundant or irrelevant due to the poor quality of training data.

The above-mentioned limitations of conventional CNN have led to our proposed Knowledge-oriented CNN (Chapter 3) which is a knowledge-enhanced version of CNN, and also inspired us in coming up with novel neural architectures including Convolutional Transformer (Chapter 6) and Attentive Convolutional Transformer (Chapter 7).

2.3.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is essentially a fully connected neural network that contains a loop to refactor some of its layers, allowing information to persist from the first input to the last input [59, 60]. Intuitively, RNN may be the best suitable neural network for dealing with sequential data where the sequential information is important for reasoning, such as time-series data, speech signal, and text data [61–63].

As shown in Figure 2.9, RNN processes data in a sequential manner by memorizing the previous inputs while processing new ones. Specifically, RNN makes reasoning at the current time step t based not only on the current input $\mathbf{x}_t$, but also on previous hidden state $\mathbf{h}_{t-1}$. The current hidden state $\mathbf{h}_t$ is calculated following

FIGURE 2.9: Sequential processing in RNN [5].

Equation (2.6):

$$\mathbf{h}_t = f(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \quad (2.6)$$

where $\mathbf{W}_h$ and $\mathbf{W}_x$ denote linear transformation matrices for hidden state and input respectively, $\mathbf{b}$ is a bias vector, and f is a nonlinear activation function. After processing the sequential data, the final representation can be obtained from the last hidden state of RNN, or a weighted sum of all hidden states through attention mechanism, which will be introduced in Section 2.3.3.

Since text data is also a type of sequential data, and much of NLP is dependent on the order of words, phrases, or sentences, RNN is widely used in NLP for text representation learning with the aim of capturing such sequential information in the text. However, traditional RNN suffers from gradient vanishing or exploding problems during the Back-Propagation Through Time (BPTT) training [64]. Hence, it is not good at capturing long-term dependencies. In the case of text representation learning, the information from the words far away is not taken into consideration when learning the representation of the current word. Thankfully, the introduction of gates to the cell computations makes RNN capable of learning long-term dependencies by adaptively capturing dependencies from different time steps via the gates functions. The most popular variants of RNN are LSTM and GRU, which have been replacing traditional RNNs due to their better capabilities of handling long-term dependencies.

Long-short Term Memory Network

Long-short Term Memory (LSTM) network [65] introduces three gates to control the information flow among different time steps. Besides hidden state $\mathbf{h}_t$, there is an additional cell state $\mathbf{c}_t$ in LSTM that acts like a conveyor belt through which the information flows within and among cells. The three gates are:

- **Forget gate** (gate_f): controls what part of the previous cell state will be kept.
- **Input gate** (gate_i): controls what part of the new computed information will be added to the cell state.
- **Output gate** (gate_o): controls what part of the cell state will be exposed as the hidden state.

All these three gates are functions of current input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$, as shown in Equation (2.7). The difference is that different sets of parameters are used for different gates. With the three gates controlling what information will pass through, the updating rules for the cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ are shown in Equation (2.8).

$$\text{gate} = \sigma(\mathbf{W}_{gx}\mathbf{x}_t + \mathbf{W}_{gh}\mathbf{h}_{t-1} + \mathbf{b}_g) \quad (2.7)$$

$$\begin{aligned} \mathbf{c}_t &= \text{gate}_f \odot \mathbf{c}_{t-1} + \text{gate}_i \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{h}_t &= \text{gate}_o \odot \tanh(\mathbf{c}_t) \end{aligned} \quad (2.8)$$

where $\mathbf{W}$ denotes linear transformation matrices, $\mathbf{b}$ denotes bias vectors, σ is the sigmoid activation function, $\tanh$ is the hyperbolic tangent activation function, $\odot$ denotes the element-wise product.

Gated Recurrent Unit

Gated Recurrent Unit (GRU) [29], which is sometimes referred as Gated Recurrent Network, is a simplified version of LSTM that only has two gates:

- **Reset gate** (gate_r): controls what part of the previous hidden state will be kept for the calculation of the new proposal hidden state.
- **Update gate** (gate_u): controls what part of the previous hidden state and the proposal hidden state will be kept for the calculation of the current hidden state.

The calculation of the two gates is in the same way as in Equation (2.7). The hidden state $\mathbf{h}_t$ is updated following Equation (2.9):

$$\begin{aligned} \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_h(\text{gate}_r \odot \mathbf{h}_{t-1}) + \mathbf{b}) \\ \mathbf{h}_t &= (1 - \text{gate}_u) \odot \mathbf{h}_{t-1} + \text{gate}_u \odot \tilde{\mathbf{h}}_t \end{aligned} \quad (2.9)$$

FIGURE 2.10: Structures of bidirectional RNN (left) and deep bidirectional RNN (right).

where $\mathbf{W}$ denotes linear transformation matrices, $\mathbf{b}$ denotes bias vectors, $\tanh$ is the hyperbolic tangent activation function, and $\odot$ denotes the element-wise product.

Apparently, GRU works in a similar way as LSTM except that GRU has fewer model parameters. In many cases, GRU and LSTM produce equally good results. The gates introduced in RNN helps in overcoming the vanishing gradient problem since they are able to memorize the relevant information and pass it down to the subsequent time steps in the network.

Bi-directional and Deep RNN

Besides gated operations, different structures of RNN are also developed to overcome the shortcomings of the original RNN. The most popular structures include bidirectional RNN and deep RNN, as shown in Figure 2.10.

The idea behind bidirectional RNN is that correct processing of the current time step may not only depends on previous input, but also depends on feature time steps. For example, to predict a word in a sentence, we need to look at both the context before and the context after the word to make a reasonable prediction. Hence, it is beneficial to consider both directions to process a sequence. As shown in Equation (2.10), the hidden state of forward direction ($\vec{\mathbf{h}}_t$) depends on the current input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$; the hidden state of backward direction ($\overleftarrow{\mathbf{h}}_t$) depends on the current input $\mathbf{x}_t$ and future hidden state $\mathbf{h}_{t+1}$. The final hidden representation for the current time step is normally the concatenation

of the forward and backward outputs.

$$\begin{aligned}\vec{\mathbf{h}}_t &= RNN(\mathbf{x}_t, \mathbf{h}_{t-1}) \\ \overleftarrow{\mathbf{h}}_t &= RNN(\mathbf{x}_t, \mathbf{h}_{t+1}) \\ \mathbf{h}_t &= Cat(\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t)\end{aligned}\tag{2.10}$$

where RNN indicates the internal operations of standard RNN, LSTM or GRU, and Cat indicates concatenation of vectors.

Deep RNN is to stack multiple RNN layers together by feeding the outputs of the lower layer as the inputs to the upper layer. This multi-layer structure gives the model higher capacity, probably with the lower layer focusing more on lexical or syntactic information, and the upper layer focusing more on higher-level semantics. However, more training data is needed since there are much more model parameters in deep RNNs.

2.3.3 Attention Mechanism

Although LSTM and GRU are designed to overcome the gradient vanishing problem of RNN, they still have limited ability in capturing long-term dependencies from a particularly long sequence. The loss of information may still happen during the sequential processing. This is because of the recurrent nature of RNNs, the information flow is still from the beginning to the current element. Attention mechanism offers an elegant solution to this problem by simultaneously processing all the elements in a sequence, and focusing more on the important elements.

Inspired by the human brain that executes attention at many levels, for example, we often focus more on the keywords in an article, the salient parts of an image, and the specific scenes in a video clip, attention mechanism eliminates the overwhelming amount of background information and selects only the important information that is needed for the task at hand. Ever since the introduction of attention mechanism to neural networks [66, 67], it has become one of the most important components for neural networks to understand sequential data.

A typical attention mechanism is implemented as shown in Figure 2.11. It takes a sequence of elements $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ and a context vector (or query vector) $\mathbf{c}$ as input, and returns a vector $\mathbf{z}$ which is the summary of the elements focusing on the

FIGURE 2.11: A typical implementation of attention mechanism [6]

information related to context $\mathbf{c}$. Specifically, the relevance between each element and the context vector is calculated by Equation (2.11). The function f can be as simple as vector product of $\mathbf{y}_i$ and $\mathbf{c}$, or as complex as deep compositions of $\mathbf{y}_i$ and $\mathbf{c}$ using neural networks. The context vector $\mathbf{c}$ can be a randomly initialized vector, or one of the input elements (self-attention), or from other components of the network. Attention weights s_i are calculated through a softmax function (Equation (2.12)), and the final representation is obtained as the weighted arithmetic mean of all the elements, as shown in Equation (2.13).

$$m_i = f(\mathbf{y}_i, \mathbf{c}) \quad (2.11)$$

$$s_i = \frac{\exp(m_i)}{\sum_{j=1}^n \exp(m_j)} \quad (2.12)$$

$$\mathbf{z} = \sum_{i=1}^n s_i \mathbf{y}_i \quad (2.13)$$

In the early days, attention mechanism is normally applied on top of RNNs or CNNs as an augment, which can greatly improve the model's sequential processing capability, especially for text representation [7, 68–70]. One classic model is the hierarchical attention network for document representation proposed by Yang et al. [7]. The model architecture is shown in Figure 2.12. Two levels of attention mechanisms are used: the first level generates sentence representation from a sequence of words, and the second level generates document representation from a sequence of

FIGURE 2.12: Architecture of hierarchical attention network [7].

sentence representations. The basic encoder used in the network is bi-directional GRUs to obtain the hidden representation $\mathbf{h}_i$ of each element. The context vector $\mathbf{u}$ used in the attention mechanism is a randomly initialized vector, and relevance between the context vector $\mathbf{u}$ and each hidden representation $\mathbf{h}_i$ is calculated by Equation (2.14).

$$m_i = \mathbf{u}^T \cdot \tanh(\mathbf{W}\mathbf{h}_i + \mathbf{b}) \quad (2.14)$$

where $\mathbf{W}$ is a linear transformation matrix that transforms $\mathbf{h}_i$ to the same dimension as $\mathbf{u}$, $\mathbf{b}$ is a bias term. The final representation of the attention mechanism is calculated in the same way as in Equation (2.12) and 2.13.

Meanwhile, different variants of attention mechanisms were also proposed, such as intra-temporal [71], convolutional [72], gated [73], and self-attention [8]. All these variants have demonstrated the benefits of utilizing attention mechanisms in deep neural networks for NLP applications. Besides performance improvements, attention mechanism also brings better interpretability for the neural model. By visualizing the attention weights assigned to the elements, we can easily understand what the model learns and focuses on.

However, the majority of attention mechanisms in the literature are data-driven, which rely heavily on training data to learn their context vectors or calculate attention weights. To improve the attention mechanism, we propose a novel knowledge-attention mechanism (Chapter 5) which allows the network to pay attention to external knowledge. Besides, we also propose a sentiment-aware attention mechanism (Chapter 6) which incorporates additional sentiment knowledge into the attention mechanism when calculating attention weights.

2.3.4 Transformer

Besides the vanishing gradient problem, RNN also suffers from parallel-unfriendly problem due to its recurrent nature that relies on previous elements to process the current element. Hence, it requires more resources to train and run than attention-based models. Instead of augmenting existing RNNs using attention mechanism, Transformer uses a novel neural architecture which is fully based on attention mechanism [8]. It overcomes both vanishing gradient and parallel-unfriendly problems of RNN by relying entirely on self-attention to handle long-distance dependencies without recurrent computations.

Since the Transformer was originally proposed for machine translation, it has an encoder-decoder structure as shown in Figure 2.13. Each encoder and decoder has a multi-head self-attention followed by a fully connected feed-forward layer, and they can be stacked as multiple layers. Position encoding, residual connection, and layer normalization are also applied in Transformer. Readers may refer to [8] for more details. Here, we focus on introducing the core component of the Transformer, the multi-head attention. As shown in the bottom of Figure 2.13, the basic operation in multi-head attention is the scaled dot-product attention. It takes queries ($\mathbf{Q}$) and key-value pairs ($\mathbf{K}$, $\mathbf{V}$) as inputs, and output a weighted sum of values based on the attention weights computed from queries and keys. Mathematically,

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{QK}^T}{\sqrt{d_k}})\mathbf{V} \quad (2.15)$$

where d_k is a scaling factor to counteract the effect of the large magnitude of the dot product that leads the softmax function to the regions with small gradients. In Transformer, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ come from the same input sequence, and each element of the sequence is attending to all the elements to compute its representation.

FIGURE 2.13: Model architecture of Transformer [8]. Scaled dot-product attention is used as the basic operation in the multi-head attention of Transformer.

Therefore, The scaled dot-product attention is also called self-attention. Another novelty of Transformer is that instead of a single attention operation, it has a multi-head attention structure that jointly performs self-attention in different sub-spaces. As shown in Equation (2.16), for h -head attention, h different linear projections are first performed on $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ to transform their dimensionalities from d to d/h . Then, the scaled dot-product attention is performed in different sub-spaces simultaneously. Finally, the outputs from all the heads are concatenated together, and another linear transformation is conducted to obtain the final result.

$$\begin{aligned} MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= Cat(head_1, head_2, \dots, head_h) \mathbf{W}^O \\ \text{where } head_i &= Attention(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V) \end{aligned} \quad (2.16)$$

In Equation (2.16), $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V \in \mathbb{R}^{d \times (d/h)}$, and $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ are linear transformation matrices. *Cat* indicates concatenation. The multi-head structure gives Transformer higher capacity to capture important information from different positions in different sub-spaces.

Since the self-attention mechanism in Transformer is able to draw global dependencies between input and output without regard to their distances in the sequence, Transformer is more suitable for capturing long-distance dependencies and allows more parallelization compared with RNN. It has been demonstrated that the Transformer is more effective than RNN in a wide range of NLP tasks [74–78]. Especially, the emerging of pre-trained language models based on the Transformer has led to a series of breakthroughs and state-of-the-art performances in many NLP applications [3, 46, 79, 80]. Given so many advantages of Transformer over RNN, we believe that the Transformer is able to fully replace RNN for sequential processing and text representation.

Although the Transformer has excellent abilities in dealing with long-term dependencies, it is relatively weak in capturing local features since the self-attention mechanism focuses on global instead of local context. Besides, the heavy architecture of the Transformer often requires more training data, CPU/GPU memory, and computational power, especially for long texts. The knowledge-attention mechanism introduced in Chapter 5 is able to enhance the Transformer with external knowledge, and the proposed neural architectures including ConvTransformer and ACT presented in Chapter 6 and Chapter 7 respectively have better local feature extraction capability as well as computational efficiency.

Chapter 3

Incorporate Knowledge into Bag-of-Words

3.1 Introduction

Among various text analysis tasks, document classification is the most classical one, whose goal is to allocate documents into different categories. It has been widely applied in various applications such as spam detection [81], ontology mapping [82], document recommendation [83], topic labeling [84], and sentiment classification [85, 86]. In this chapter, we focus on text representation learning for document classification task.

A typical document classification system consists of three main processes, namely text preprocessing, document representation and classification. The system performance largely relies on the quality of document representation, whose goal is to convert raw documents into structured numerical vectors (feature vectors) to support various machine learning and data mining techniques. The quality of a document representation can be evaluated from two perspectives: semantic quality and statistical quality [87]. Semantic quality refers to the semantic meaning and interpretability of the feature vector, i.e. to what extent can each feature describes the content of the document. For example, Bag-of-Words (BoW) model represents document as a fixed-length vector of all terms (words or n-grams) occurred in the corpus. Each term is weighted with a numerical value such as term frequency or tf-idf weighting. Statistical quality refers to the discriminative power

of the feature vector in distinguishing documents from different classes. Recently, deep neural networks such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [7, 27, 30], especially the recent proposed pre-trained language models [3, 45, 46] have demonstrated excellent statistical quality for many NLP tasks. However, majority models focus only on the statistical quality and neglect the semantic quality of representations. The resulted representations are very hard to interpret since the value of each feature is computed through complex compositions of the neural network weights. However, many real text mining applications demand for interpretable representations in order to provide human deeper understandings of texts and clearer operation logic for reasoning and decision making. The focus of this work is to propose a good document representation that is able to capture the underlying characteristics and distinct features of documents in different classes while preserving its interpretability.

BoW model is commonly adopted for document classification due to its simplicity, acceptable accuracy and good interpretability. However, BoW model suffers from several shortcomings which hinder the performances of BoW-based models. Firstly, BoW simply looks at the surface form of words and ignores the semantic relatedness and conceptual information of words and phrases. For example, the two sentences “Ferrari delivers the best driving experience” and “Lamborghini has powerful engine” share no common words, and BoW will represent them as two totally different vectors. However, the two sentences are closely related because “Ferrari” and “Lamborghini” are both brands of sports car. Secondly, BoW does not consider word order, hence the context information of texts is lost. However, context information offers a lot of semantics and is very useful to distinguish polysemous words and recognize synonyms. For example, the word “apple” can be a kind of fruit or a company, by looking into its context we can distinguish the two senses: if the word comes along with “fruit”, “orange”, “lemon” etc, we can recognize it as a kind of fruit; if it comes along with “company”, “Google”, “Microsoft” etc, we can recognize it as a company. Thirdly, BoW suffers from high dimensionality and sparsity since it uses hard/crisp mapping of the terms in the whole vocabulary. Besides, the model considers single words only, however, sometimes phrases deliver more semantic meanings than single words. For example, “galaxy note” is actually a Samsung phablet, however the word “galaxy” and “note” themselves deliver no such meaning. Even though Bag-of-n-Grams considers n-grams as basic terms, it

will further increase dimensionality and sparsity severely. Besides, most of n-grams are meaningless.

To address the limitations of BoW model, we propose a novel Bag-of-Concepts (BoC) document representation scheme, which provides more semantic and conceptual information for document classification. The idea comes from human way of text understanding: we understand “Ferrari”, “Lamborghini” as sports car brands and “Google”, “Microsoft” as companies, because our mental world contains many concepts about such worldly facts. Besides acquiring information from the text we are reading, we simultaneously use the conceptual knowledge stored in our brain to help us understand text better. Inspired by this, the proposed BoC model conceptualizes words and phrases in the document into higher level semantics (i.e. concepts) based on the conceptual knowledge from external knowledge base, and finally represents document as a distributed vector in the learned concept space. The useful conceptual knowledge for a certain classification task is automatically acquired from a large scale probabilistic knowledge base called Probase, such conceptual knowledge serves as the background knowledge in human brain. Compared with BoW model, the proposed model essentially transforms a high dimensional and sparse term vector space into a low dimensional and more dense concept vector space in a non-linear way based on the conceptual information of texts.

The main contributions of this chapter are summarized as follows:

- * We propose an efficient way of automatically acquiring useful knowledge from large probabilistic knowledge base (Probase) that will benefit document classification task. The proposed knowledge acquisition scheme not only reduces the search space and querying time significantly, but also preserves the useful information for the certain classification task and reduces noises that are irrelevant at the meanwhile.
- * Based on the acquired knowledge, we propose a document conceptualization scheme which reveals the higher level semantics of documents, and constructs a Bag-of-Concepts (BoC) model for document representation using a novel concept score-inverse soft document frequency ($cs - idf^s$) weighting scheme. Compared with BoW representation, BoC has lower dimensionality and sparsity due to the probabilistic mapping of document in concept space. More importantly, BoC representation is able to capture the semantic relatedness

and conceptual information of texts, which will benefit document classification tasks.

- * We further propose concept clustering and entity sense disambiguation techniques, and construct a Bag-of-Concept-Clusters (BoCCl) model to improve the BoC representation. The concept clustering process clusters similar concepts together to further reduce concept space dimensionality and sparsity. The entity sense disambiguation process disambiguates vague entities according to the context information, making the conceptualization of entities more accurate. Compared with BoW representation which use sparse words or n-grams as indexing terms, BoCCl representation is more interpretable and provides much deeper understanding of the document.
- * Comprehensive experiments are conducted to evaluate both statistical and semantic qualities of the proposed BoC and BoCCl representations. Experimental results demonstrate that our proposed Bag-of-Concepts representation is able to improve document classification performance and also provide great interpretability for human understanding.

This chapter is organized as follows. In Section 3.2, the related works of document representation are reviewed. Our proposed Bag-of-Concepts (BoC) representation for document classification is presented in Section 3.3, and some refined approaches to improve BoC representation are described in Section 3.4. Experimental results and detailed analyses are presented in Section 3.5. Finally, our work is concluded in Section 3.6.

3.2 Related Work

In literature, document representation approaches fall into two categories: one is representation based on data only, where the representation is solely learned from training data; the other is incorporating knowledge from external knowledge bases into the representation. In this section, we give a comprehensive review of these two approaches.

3.2.1 Representation based on data

As mentioned in Section 3.1, the most established BoW model has limitations including high dimensionality and sparsity, and incapability of capturing contextual and conceptual information of texts. Many works are proposed to improve BoW representation. The most well-known models are latent semantic analysis (LSA) [42] and topic models including probabilistic latent semantic analysis (pLSA) [43] and latent dirichlet allocation (LDA) [44]. Based on the statistical analyses of term-document matrix, these models discover the latent semantic structures of the document and transform BoW representation into low dimension and dense vectors. In LSA, singular value decomposition (SVD) is applied on term-document matrix to represent documents in a new latent semantic space, where each latent semantic dimension is a linear combination of terms in the original BoW representation. In topic models, a document is represented as mixtures of topics that spit out words with certain probabilities. It is assumed that each topic is a distribution of words and each document is a mixture of topics. The topic models then try to backtrack from the documents to find a set of topics that are likely to have generated the documents. The term “Bag-of-Concepts” is originally proposed in [88]. However, the concepts are some synonym sets or latent dimensions of “BoW” instead of semantic hyponymies. All these models rely on word occurrence statistics to perform dimensionality reduction. However, the semantic relevance among words is ignored, and phrases cannot be captured since only single terms are considered. Besides, the derived latent dimensions are highly corpus-dependent and lack of semantic interpretation.

With the development of word embeddings, a distributed representation of words in vector space, semantic and syntactic properties of words can be captured in low-dimensional and dense space [21, 22]. The word embeddings can be obtained through unsupervised learning from large-scale text corpus [2, 23, 24]. Many document classification methods are proposed by utilizing pre-trained word embeddings. One simple approach is averaging the word embeddings of all words in a document [89], assuming that the semantic space is shared by words and documents. Le and Mikolov [90] proposed doc2vec model by extending the word2vec algorithm [2, 23], which utilizes contextual information of words and documents to learn document representations in a continuous vector space. Zhao and Mao [91] proposed fuzzy Bag-of-Words (FBoW) model which adopts fuzzy mapping of words based on word

embedding similarities to capture the semantic correlation among words. Kim et al. [92] also proposed a bag-of-concepts model for document representation. Different from our model, they treat each word cluster as a concept by performing clustering of word embeddings generated from Word2Vec. However, the performance of the model strongly depends on the quality of word embeddings. In many cases, Word2Vec model may not be able to correctly capture the semantic relationship among all the words since the training of word embeddings is based on the short context of words. For example, the cosine similarity between “good” and “bad” is 0.72, whereas the cosine similarity between “good” and “amazing” is only 0.48. Besides, word embeddings cannot deal with polysemous words such as “bank” and “apple”. Consequently, the clusters based on word embedding similarities may not be reliable, and the errors or outliers in the clusters will affect the performance of the model. Besides, since word embeddings are single-word based, the semantics of phrases cannot be captured.

Recently, with the prevalence of deep learning [25], many deep compositional models have been proposed for document classification. A deep compositional model contains multilayer neural networks with various forms, including convolution [26, 27], recurrent [7, 30], recursive [31, 32] neural networks as well as attention-based transformers [3, 8] to learn text representations by performing compositions over word embeddings. The advantages of deep compositional models are that word order and contextual information are preserved, and more complex latent semantics can be captured by training the models using large amount of labeled data. Even though high classification accuracy can be obtained from deep compositional models, the models require sufficient high-quality annotated training corpus, heavy computations and tricky hyper-parameter settings. Besides, the resulted representations lack of interpretability since the features are computed through complex composition of neural networks.

3.2.2 Representation leveraging knowledge bases

Solely data-based text representation learning requires sufficient human-annotated and high-quality training corpus, especially for deep compositional models. This is sometimes impractical due to the lexical, morphological and syntactic variations of natural language. As a result, the learned representations may not be reliable and

are normally domain specific. Many works are proposed to incorporate existing knowledge from external knowledge bases as prior knowledge for machine learning systems, in order to reduce the reliance on training data and provide additional useful information for text representation. In NLP, external knowledge bases can be categorized into lexical knowledge base and encyclopedic knowledge base.

The most widely used lexical knowledge bases include WordNet [37], SentiWordNet [39], VerbNet[93] and FrameNet [38]. Many works utilize the synonym sets and lexical categories of WordNet as concepts to improve text representation [94–96]. In [97], knowledge from WordNet is utilized to perform word sense disambiguation (WSD) by modeling the semantic space and semantic path using LSA and PageRank respectively. Some works use the lexicons in SentiWordNet to perform sentiment classification of reviews [98–100]. In [101], knowledge from WordNet and FrameNet is incorporated into CNN for causal relation extraction from texts. Despite the usefulness of lexical knowledge bases for text representation, these knowledge bases are normally manually constructed dictionaries which have limited coverage of worldly facts and conceptual knowledge. To capture more semantic and conceptual information of texts, many studies incorporate the knowledge from encyclopedic knowledge bases such as Wikipedia¹, DBpedia[102], Open Directory Project (ODP)², Yago [103], Probase [40] etc. In [104], both Wikipedia and WordNet are used to exploit useful features for short text representation. Some works use the units of knowledge from Wikipedia and DBpedia as concepts for text representation [105–107]. In [108] and [109], categorical information in ODP is utilized for text representation. In [110], word sense in BabelNet³ is embedded into document embeddings. In [111] and [112], the probabilistic knowledge of Probase is incorporated into topic models. Some works utilize Probase to learn concept embeddings which can be used for document representation [113, 114]. A few studies on short text understanding and classification use the conceptual knowledge in Probase to discover the semantics from short texts [115–117]. We also utilize Probase in our work because it is a state-of-art encyclopedic knowledge base which has a broad coverage of concepts and entities, and the associations between entities and concepts are measured using probabilities instead of Boolean variable, which makes the subsequent inferences more flexible.

¹<https://en.wikipedia.org>

²<http://curlie.org>

³<https://babelnet.org>

3.3 Bag-of-Concepts (BoC) representation

FIGURE 3.1: Overall architecture of BoC model.

In this section, the proposed BoC representation is presented. The overall architecture of the model is shown in Figure 3.1. The model consists of an offline knowledge acquisition phase including automatic knowledge acquisition and entity conceptualization, as well as an online representation learning phase including entity and concept recognition, document conceptualization and bag-of-concepts construction. The details of the two phases are described in Section 3.3.1 and Section 3.3.2 respectively.

3.3.1 Offline Knowledge Acquisition

3.3.1.1 Probase and typicality scores

Probase⁴ is a large scale encyclopedic knowledge base as well as a probabilistic semantic network. It contains over 5 million unique concepts, 12 million unique entities, and 87 million Is-A relations. The Is-A relationship is harvested from billions of web pages using Hearst patterns [118]. For example, from the sentence “Singapore University such as Nanyang Technological University...”, it extracts the concept-entity pair in which the entity “Nanyang Technological University” is an instance of the concept “Singapore University”.

⁴Available online at <https://concept.research.microsoft.com>.

One important characteristic of Probase is that the concepts and entities in the knowledge base are associated with each other in a probabilistic manner instead of Boolean association. This probabilistic manner is similar as our human mind. For example, given an entity “apple”, people will conceptualize it as “fruit” or “company” more likely than “movie” or “song”; given a concept “illness”, people are more likely to think of “flu” than “lupus”. In Probase, the probabilistic associations between concepts and entities are represented by typicality scores, including the probabilities of concepts for a certain entity ($P(c|e)$), as well as the probabilities of entities belonging to a certain concept ($P(e|c)$). For above examples, $P(\text{fruit}|\text{apple}) > P(\text{song}|\text{apple})$ and $P(\text{flu}|\text{illness}) > P(\text{lupus}|\text{illness})$. Formally, the typicality scores are derived from the co-occurrences of concepts and entities as follows:

$$P(c|e) = \frac{n(e, c)}{\sum_{e \in c_i} n(e, c_i)} \quad (3.1)$$

$$P(e|c) = \frac{n(e, c)}{\sum_{e_i \in c} n(e_i, c)} \quad (3.2)$$

where $n(e, c)$ is the co-occurrence of entity e and concept c in the Hearst patterns from web documents.

The typicality scores allow the knowledge representation to be more accurate and reliable, and also provide more flexibilities on querying and operating on the knowledge base. However, when conceptualizing an entity, the two typicality scores tend to give higher scores to “extreme” concepts [119]. $P(c|e)$ tends to assign higher scores to more general concepts as the score is proportional to the co-occurrence of c and e when e is given; and $P(e|c)$ tends to assign higher scores to more specific concepts as the score will be higher for the specific concepts that only contains e or majority of its entities are e . General concepts provide less discriminative powers, and specific concepts cover less entities, making the concept space even more sparse. To balance between general and specific concepts, and find the most suitable concepts for an entity, we use a modified $Rep(e, c)$ score proposed by Wang et al. [119] to conceptualize an entity, as shown in Equation 3.3.

$$Rep(e, c) = P(c|e) \cdot P(e|c)_{k-smooth} \quad (3.3)$$

where $P(e|c)_{k-smooth}$ is the smoothed typicality score to avoid the extreme values caused by the specific concepts that contain few entities:

$$P(e|c)_{k-smooth} = \frac{n(e, c) + k}{\sum_{e_i \in c} n(e_i, c) + kN_e} \quad (3.4)$$

where N_e is the total number of entities, and k is a small constant assuming that every concept-entity pair has a small co-occurrence whether we observe it or not.

3.3.1.2 Automatic knowledge acquisition from Probase

Even though Probase has a broad coverage of concepts and entities, utilizing the entire Probase knowledge for document classification is not a good idea. The reasons are as follows:

- As mentioned in Section 3.3.1.1, Probase contains large amount of concepts, entities, and Is-A relation pairs. Querying on the whole knowledge base is quite time-consuming due to the enormous search space. This will cause both offline and online learning processes to be very slow, hence hindering the computational efficiency of document classification.
- Using all the concepts in Probase will cause the dimensionality of concept space to be even larger than the vocabulary space, making the model suffer from curse of dimensionality as well as sparsity issue. Besides, many concepts and entities in Probase are not relevant to the specific classification task. These irrelevant concepts and entities will add noise to the model and affect the accuracy of document classification.

In fact, instead of acquiring all the knowledge from external resources, our human selectively acquire useful knowledge based on their needs for certain tasks. For document classification tasks, the classes or categories are normally pre-defined, and only the concepts and entities related to these classes are useful for document classification. Motivated by this, we propose an automatic knowledge acquisition scheme to select a subset of Probase for subsequent processes. Based on the class labels and class-related information, we automatically select as much as possible the concepts and entities from Probase that are relevant to the classes. The details are as follows.

Algorithm 1: Automatic knowledge acquisition

Step 1: Find the class labels that best describe each document class.
 $labels = \{l_1, l_2, \dots, l_c\}$

Step 2: Extend the class labels using the WordNet synsets of the class labels.
for $term$ in $labels$ **do**
 for $synset$ in WordNet synsets of $term$ **do**
 for $lemma$ in WordNet lemmas of $synset$ **do**
 if $lemma$ not in $labels$ **then**
 $labels = labels + lemma$
 end if
 end for
 end for
end for

Step 3: Concept and entity extraction from Probase using 3-level iterations.
for each term t in the extended class labels **do**
 1st-level iteration:
 Find top 20 concepts c according to $Rep(t, c)$.
 Find top 100 entities e according to $P(e|t)$.
 2nd-level iteration:
 For each concept c extracted from the 1st level, find all its entities.
 For each entity e extracted from the 1st level, find its top 20 concepts c according to $Rep(e, c)$; treat the entity e as concept and find all the entities belong to it.
 3rd-level iteration:
 For each concept extracted from the 2nd level, find all its entities.
 end for
 Aggregate all the concepts found during the 3-level iteration and form a concept set: $c_set = \{c_1, c_2, \dots, c_n\}$;
 Aggregate all the entities found during the 3-level iteration and form an entity set: $e_set = \{e_1, e_2, \dots, e_m\}$.
end for

We first find the most relevant words that can best describe each class, such as class labels. Then we extend the class labels using an external lexical resource called WordNet⁵. WordNet groups semantically similar English words into sets of synonyms (called synsets) to represent different meanings [37]. By searching the synsets of class labels, we can find all the synonyms of the class labels and hence extend the class labels to cover more relevant terms. Finally, we extract all the relevant concepts and entities from Probase based on the extended class labels. The detailed algorithm for automatic knowledge acquisition is shown in Algorithm 1. Figure 3.2 shows an example of concept and entity extraction from Probase using the proposed 3-level iterations based on the class label “hardware”. The

⁵Available online at <https://wordnet.princeton.edu>.

FIGURE 3.2: Example of concept and entity extraction from Probase using 3-level iterations based on the class label "hardware".

concepts extracted based on the first level entities are all the relevant concepts of the class label, as they contain common entities. The entities extracted based on the concepts of first level and second level concepts are the relevant entities of the class label, as they either belong to the same concept as class label, or belong to the relevant concepts of the class label. Besides, the entities found by treating the second level entities as concepts are the more specific entities related to the class label.

Through the proposed automatic knowledge acquisition algorithm, all the useful concepts and entities that are relevant to the document classification task can be extracted from Probase. This greatly reduces the search space and accelerates the subsequent processes. More importantly, by keeping only the relevant knowledge, the noises are removed and the dimensionality as well as sparsity of concept space are reduced significantly.

3.3.1.3 Entity conceptualization

After selecting relevant entities and concepts from Probase, we conceptualize the selected entities by mapping each entity to the selected concepts with probability scores. For each entity, we rank its concepts according to the $Rep(e, c)$ score described in Section 3.3.1.1. Then we keep only the top k_c concepts and transform

the $Rep(e, c)$ scores into probability scores according to Equation 3.5:

$$Rep(e, c_i)_p = \frac{Rep(e, c_i)}{\sum_{j=1}^{k_c} Rep(e, c_j)} \quad (3.5)$$

Finally, a mapping dictionary of the conceptualized entities is constructed, which map entities (words / phrases) into higher level semantics (i.e. concepts) with probability scores. The conceptualized entities can be directly used for online representation learning processes.

3.3.2 Online Representation Learning

3.3.2.1 Entity and concept recognition

We use the entities and concepts acquired from the automatic knowledge acquisition process as a mapping dictionary to recognize the entities and concepts in the document. The matching algorithm we used is called Backward Maximum Matching (BMM) [120]. BMM is originally proposed for Chinese word segmentation as most Chinese words contain many characters. Similarity, we found that BMM is also efficient to recognize English phrases which are composed of many words. The algorithm selects a string with *maxLen* words from right to left of the text, and use it to map the entities and concepts in Probase.

Before applying the matching algorithm, we first pre-process the document including sentence segmentation, word tokenization and part-of-speech (POS) tagging. Then we perform entity and concept recognition sentence by sentence, as shown in the flow chart in Figure 3.3. We use two mapping dictionaries: one for entities and the other for concepts, so that the algorithm can output the recognized entities and concepts separately. For a single word, we do not consider it as concept or entity candidate if the word is a stop word (similar like stop words removal), or the POS tag of the word is not noun. For example, in the sentence “watch star war movie”, the word “watch” is a verb and is not an entity or concept. However, if we do not consider the POS tag, the system may recognize it as an entity which belongs to concepts “timepiece”, “valuable” etc. The wrong recognition of entities or concepts will introduce noise and mislead the document classification system.

FIGURE 3.3: The flow chart of Backward Maximum Matching (BMM) for entity and concept recognition.

3.3.2.2 Document conceptualization

For each recognized entity in the document, we conceptualize it into higher level concepts by looking for the offline mapping dictionary of the conceptualized entities:

$$e_i \rightarrow \{c_1 : Rep(e_i, c_1)_p, c_2 : Rep(e_i, c_2)_p, \dots\}$$

where $Rep(e_i, c_j)_p$ is the concept probability score of concept c_j for entity e_i . For each recognized concept in the document, we set its concept probability score as 1:

$$c_i \rightarrow \{c_i : 1\}$$

After that, we conceptualize the sentences in the document by aggregating the concept probability scores of the recognized entities and concepts in the sentence.

$$S_i \rightarrow \{c_1 : Score(S_i, c_1), c_2 : Score(S_i, c_2), \dots\}$$

where $Score(S_i, c_j)$ is the concept score of c_j for sentence S_i :

$$Score(S_i, c_j) = \sum_{e_k \in S_i} Rep(e_k, c_j)_p + \sum_{c_j \in S_i} 1 \quad (3.6)$$

Similarly, the document is conceptualized by aggregating the concepts and their scores of all the sentences in the document.:

$$D_i \rightarrow \{c_1 : Score(D_i, c_1), c_2 : Score(D_i, c_2), \dots\}$$

where $Score(D_i, c_j)$ is the concept score of c_j for document D_i :

$$Score(D_i, c_j) = \sum_{S_k \in D_i} Score(S_k, c_j) \quad (3.7)$$

Eventually, each document is conceptualized into a set of concepts which are associated with concept scores reflecting the relevance of the concepts to the document.

3.3.2.3 Bag-of-Concepts construction

After conceptualizing all the documents in the collection, we find the concept vocabulary and their document frequencies. In BoW model, the document frequency of a term t is the total number of documents containing t , as the term t either appears or does not appear in a document (Boolean association). However, this is not applicable to our proposed BoC model, because each concept term is associated with the documents using concept scores (fuzzy association), and the concepts with very small concept scores in a document should not be counted as one occurrence. Therefore, we propose a soft document frequency df_i^s for concept c_i , as calculated in Equation 3.8.

$$df_i^s = \sum_{D_j \in \mathfrak{D}} Occ(D_j, c_i) \quad (3.8)$$

where $\mathfrak{D}$ is the set of all the documents, and $Occ(D_j, c_i)$ is the soft occurrence of concept c_i in document D_j :

$$Occ(D_j, c_i) = \begin{cases} Score(D_j, c_i) & \text{if } Score(D_j, c_i) < 1 \\ 1 & \text{if } Score(D_j, c_i) \geq 1 \end{cases} \quad (3.9)$$

We discard the concepts whose soft document frequency is less 1, since these concepts rarely appear in the documents and provide little information for document classification. Finally, we represent a document D_j as a distributed vector in the learned concept space: $d_j = (w_{1j}, w_{2j}, \dots, w_{lj})$, where l is the dimensionality of the concept space, and w_{ij} is the weight of concept c_i in document d_j . Similar to term frequency-inverse document frequency ($tf - idf$) weighting scheme, we propose a concept score-inverse soft document frequency ($cs - idf^s$) weighting scheme for calculating w_{ij} , as shown in Equation 3.10.

$$w_{ij} = Score(D_j, c_i) \times \log\left(\frac{N}{df_i^s}\right) \quad (3.10)$$

where N is total number of documents in the collection. The first term $Score(D_j, c_i)$ is the concept score which indicates the relevance of concept c_i in document D_j ; the second term $\log(\frac{N}{df_i^s})$ is the inverse soft document frequency to reduce the weights of frequent concepts since the concepts appear in majority documents are less informative than those that only appear in certain classes.

Compared with BoW representation, BoC has lower dimensionality due to the fact that concept space has much lower dimension than word or n-gram space. Besides, BoC representation is less sparse than BoW representation, because the conceptualization process in BoC model maps words and phrases into diverse concepts in a probabilistic way instead of the hard mapping of words as in BoW model. More importantly, BoC model is able to capture the semantic relatedness and conceptual information of words and phrases as well as higher-level semantics of documents, which will benefit the document classification tasks.

3.4 Refined Approaches

To further improve BoC representation, we propose some refined approaches including Bag-of-Concept-Clusters (BoCCL) model (Section 3.4.1), entity sense disambiguation (Section 3.4.2) and combination of BoC and BoW representations using attention mechanism (Section 3.4.3).

3.4.1 Bag-of-Concept-Clusters (BoCCL)

Among the concepts acquired from Probase, many of them are similar with each other, which contain a lot of common entities. For example, “company”, “firm”, “large company”, “big company”, “giant company”, and “large firm” are similar concepts representing “company”; “in room amenity”, “standard hotel amenity”, “room amenity”, “in-room amenity”, and “standard amenity” are similar concepts representing “in-room amenity”. These similar concepts are redundant features which will increase the dimensionality as well as sparsity of the concept space.

To further reduce dimensionality and sparsity of concept space, we perform concept clustering to group similar concepts together. Firstly, we represent each concept c_i as a vector in the entity space $\mathbb{R}^m$, where m is the total number of entities.

$$c_i \rightarrow \{P(e_1|c_i), P(e_2|c_i), \dots, P(e_m|c_i)\}$$

where $P(e_i|c_i)$ is the typicality score of entity e_i belonging to concept c_i as described in Section 3.3.1.1.

FIGURE 3.4: Illustration of clustering of similar concepts.

Secondly, we compute pair-wise distance between two concepts and obtain a distance matrix of all the concepts. The pair-wise distance is calculated as shown in Equation 3.11.

$$d(c_i, c_j) = 1 - \text{cosine}(c_i, c_j) \quad (3.11)$$

where $\text{cosine}(c_i, c_j)$ is the cosine similarity of concepts c_i and c_j in the entity space.

Thirdly, agglomerative clustering [121] is used to group similar concepts together based on the pair-wise distance matrix. Agglomerative clustering is a bottom-up hierarchical clustering method, which treats each concept as one cluster at the beginning, and successively merges close clusters together until we get the desired number of clusters or meet a certain distance threshold between clusters. Finally, the Probase concepts are grouped into clusters of similar concepts, as illustrated in Figure 3.4.

Based on the concept clusters, we propose Bag-of-Concept-Clusters (BoCCl) model, where an entity is conceptualized into concept clusters instead of concepts.

The entity conceptualization process is also done offline: for each entity, we rank its concept clusters according to the $Rep(e, ccl)$ score as defined in Equation 3.12.

$$Rep(e, ccl_i) = \sum_{c_j \in ccl_i} Rep(e, c_j) \quad (3.12)$$

Similar as in Section 3.3.1.3, we keep only the top k_c concept clusters and transform the $Rep(e, ccl_i)$ scores into probabilities $Rep(e, ccl_i)_p$ in the same way as Equation 3.5. The online document conceptualization and BoCCl construction processes are basically the same as BoC model, except that the basic term is concept clusters instead of concepts.

3.4.2 Conceptualization with entity sense disambiguation

In natural language, there often exists polysemous words or phrases which have several different meanings. If such polysemous word or phrase is an entity, then it will be conceptualized into different concepts reflecting different meanings. For example, the entity “apple” can be conceptualized into “fruit” or “company”, which are totally different concepts. Hence, entity sense disambiguation is important to achieve better representation of texts as well as better performance of NLP systems.

Inspired by Firth [122] who stated that “A word is characterized by the company it keeps”, we believe that context entities provide useful information for entity sense disambiguation. For example, if the context of “apple” contains “fruit”, “orange”, “banana” etc, we can conceptualize it as “fruit” with larger probability; if the context contains “company”, “Google”, “Microsoft” etc, we can conceptualize it as “company” with larger probability.

We first evaluate the ambiguity of an entity e_i using the entropy of e_i ’s concept clusters distribution, as shown in Equation 3.13.

$$H(e_i) = - \sum_{ccl_j \in CCl_{e_i}} P(ccl_j|e_i) \times \log_2 P(ccl_j|e_i) \quad (3.13)$$

$$P(ccl_j|e_i) = \sum_{c_k \in ccl_j, c_k \in C_{e_i}} P(c_k|e_i) \quad (3.14)$$

where CCl_{e_i} is the set of all the concept clusters of e_i , $P(ccl_j|e_i)$ is the probability of concept cluster ccl_j , and C_{e_i} is the set of all the concepts of e_i .

A higher entropy indicates higher uncertainty or ambiguity of the sense of an entity. We can set an entropy threshold h_t to determine whether an entity is ambiguous or not based on empirical studies. If an entity is ambiguous, then we can disambiguate the entity utilizing unambiguous context entities. To find the unambiguous context entities, we choose a context window of size 3 as an entity's context: the sentence which contains the entity, and the sentences before and after this sentence. Then we find all the unambiguous entities E_u in the context whose entropy is smaller than h_t . Finally, we re-weight the $Rep(e, ccl)$ scores of the concept clusters of the ambiguous entity according to Equation 3.15.

$$Rep(e, ccl_i)' = Rep(e, ccl_i) + \frac{\sum_{ccl_i \in CCl_{e_j}, e_j \in E_u} Rep(e_j, ccl_i)}{|E_u|} \quad (3.15)$$

The intuition behind Equation 3.15 is to increase the ambiguous entity's concept cluster score if the unambiguous context entities also belong to this concept cluster. Consequently, the probability scores of concept clusters ($Rep(e, ccl_i)_p$) for an ambiguous entity will be dynamically adjusted based on the context information, making the entity conceptualization process more accurate.

3.4.3 Combination of BoCCl and BoW Representations

As majority of entities in Probase are nouns or noun phrases, BoCCl (BoC) model focuses on the concepts of these words/phrases in the document. However, for some document classification tasks, other types of words such as verbs, adjectives and adverbs also play an important role in identifying the categories of the documents. Even though BoCCl provides higher level semantics of the document, only considering the concept-level information may not yield the best classification result. On the contrary, BoW model provides only word-level information but neglects concept-level information. To achieve the best performance for document classification, both concept-level and word-level information is needed. Therefore, we consider combining BoCCl and BoW representations to preserve both concept-level and word-level information for document representation.

To solve the high dimensionality and sparsity of BoW representation and allow the model to select concept-level and word-level information attentionally for different classification tasks, we propose an attention mechanism to effectively combine

FIGURE 3.5: Combination of BoCCI and BoW representations using attentional neural network.

BoCCI and BoW representations. Figure 3.5 shows the architecture of the proposed attention mechanism. We first feed BoCCI and BoW representations into a fully connected layer of standard neural network to get their hidden representations with same dimensionality. As in Equation 3.16, $\mathbf{W}$ is the weight matrix and b is a bias term of the fully connected layer. Then we measure the importance of two representations by comparing them with a task-specific context vector. As shown in Equation 3.17, dot product is used to evaluate the relatedness between the representation h_i and the context vector c , and softmax function is used to get a normalized attention weight α_i for each representation. Based on the attention weight, the two representations are combined as shown in Equation 3.18. The task specific context vector c is a high level representation reflecting which information from concept-level and word-level representations is important for the certain classification task. It is randomly initialized and adjusted by learning from training data.

$$h_i = \tanh(\mathbf{W}v_i + b) \quad (3.16)$$

$$\alpha_i = \frac{\exp(h_i^T c)}{\sum_i \exp(h_i^T c)} \quad (3.17)$$

$$r = \sum_i \alpha_i h_i \quad (3.18)$$

By using attention mechanism, the model is able to select word-level and concept-level information attentively and make good use of both of them to meet the classification needs.

3.5 Experiments

3.5.1 Datasets

Six real-life datasets for document classification are used to evaluate different document representation methods, especially our proposed BoC representation. To prove the robustness of our model, we select the datasets that are diverse in the aspect of size, type and number of categories. The statistics of the six datasets are shown in Table 3.1.

20Newsgroups (20NG) dataset is a collection of newsgroup posts which contains 20 different categories describing different topics. There are around 20,000 documents in the dataset and partitioned (nearly) evenly into the 20 categories. We adopted the version of 20NG sorted by date, and the duplicates as well as some headers are removed.⁶ Furthermore, we remove the “headers” and “footers” in each document to make the document classification task more realistic. Because our goal is to train a document classification system based on the documents themselves, not other information. Even though the information in the “headers” and “footers” of 20NG can improve the classification accuracy, the classifier can easily overfit to these information, resulting in bad generalization on other documents without “headers” and “footers”.

Reuters_8 (R8) and *Reuters_52* (R52) are both generated from Reuters-21578 dataset which contains 21578 articles on the Reuters newswire in 1987 and were manually classified into 90 categories by Reuters Ltd.⁷ As we only consider single-labeled documents, we eliminated documents with no topics or more than one

⁶The dataset can be downloaded from <http://qwone.com/~jason/20Newsgroups>

⁷The dataset can be downloaded from <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

Statistics	Document Classification Tasks					
	20NG	R8	R52	BBC	BBCSport	Yahoo
Vocab. Size	117160	19956	22248	21914	9637	883894
Class No.	20	8	52	5	5	10
Sample No.	18846	7674	9100	2225	737	1460k
Avg. Words	241	98	104	370	323	90
Train/Test	11314/7532	5485/2189	6532/2568	1112/1113	368/369	1400k/60k

TABLE 3.1: Statistics of six document classification datasets.

topic. Following the original train-test splitting, we kept the classes with at least one train and one test sample. Then we created two datasets R8 and R52, each containing 8 and 52 most frequent classes respectively.

BBC and *BBCSport* are originated from BBC News and BBC Sport websites respectively from 2004 to 2005 [123].⁸ *BBC* contains 2225 documents corresponding to stories in five topical areas: business, entertainment, politics, sport, and tech. *BBCSport* contains 737 documents corresponding to sports news articles in five topical areas: athletics, cricket, football, rugby, and tennis. Since train-test splitting is not specified for the two datasets, we splitted the datasets into training and test subsets equally.

Yahoo! Answers Topic (Yahoo) is a very large dataset which contains 1,460,000 samples obtained from ten largest categories of Yahoo! Answers Comprehensive Questions and Answers version 1.0, each category contains 140,000 training samples and 6,000 testing samples [52]. We combine the question title, question content and the best answer as a document to classify its category.

3.5.2 Experimental Settings

Our proposed BoC and BoCCl models are compared with the following benchmark document representation methods:

1. BoW: Bag-of-Words model, this serves as the baseline model in our experiment. Our implementation is based on *sklearn*⁹. Stop-words and punctuations are eliminated, and words with document frequency less than 2 are removed.

⁸The datasets can be downloaded from <http://mlg.ucd.ie/datasets/bbc.html>

⁹<https://scikit-learn.org/stable/>

2. LSA: Latent Semantic Analysis [42]. Our implementation is based on *sklearn*.
3. LDA: Latent Dirichlet Allocation [44]. Our implementation is based on *sklearn*. We use the default document topic distribution prior α and topic word distribution prior η because we found they are the optimum hyper-parameter settings.
4. AE: Document representation by taking the average of word embeddings of all words in the document [89]. Pretrained word embeddings *Word2Vec*¹⁰ from Google is used. The word embeddings have a dimensionality of 300 and were trained on part of Google News dataset containing about 100 billion words.
5. Doc2Vec: Also known as paragraph vector model which learns paragraph and document embeddings via distributed memory (DM) and distributed bag of words (DBoW) models [90]. Our implementation is based on *Gensim*¹¹. The dimensionality of embeddings is also set to 300, and DM model with negative sampling is used to learn document embeddings because neither DBoW nor the concatenation of DBoW and DM resulted in better performance for our document classification tasks.

For our proposed BoC and BoCCl models, concepts or concept clusters with soft document frequency less than 1 are removed. We use $k = 0.0001$ in the smoothed typicality score since it will affect the typicality score significantly if the value of k is too large. k_c is set to 20 to select top 20 concepts or concept clusters during entity conceptualization, and $maxLen$ is set to 3 for entity and concept recognition. For BoCCl model, the inter-cluster distance of the agglomerative clustering algorithm is evaluated using average linkage, which is the average of the distances between all observations of pairs of clusters. We choose a cluster size of 1000 for BBCSport dataset and 5000 for other datasets. For entity sense disambiguation, the entropy threshold h_t is set to be 2.5 based on empirical studies. We also investigate the combination of BoCCl and BoW representations. Two combination methods are studied: direct concatenation or combine using attention mechanism.

Linear support vector machine (SVM) [124] is used as document classifier based on the document representations learned from the aforementioned methods. When

¹⁰<https://code.google.com/archive/p/word2vec/>

¹¹<https://radimrehurek.com/gensim/index.html>

Datasets	No. of classes	No. of concepts	No. of entities
BBCSport	5	1294	92359
BBC	5	10901	461850
R8	8	14848	457931
Yahoo	10	25672	417683
20NG	20	21642	586078
R52	52	27515	600804

TABLE 3.2: Number of concepts and entities automatically acquired from Probase for six different document classification tasks.

training the SVM classifier, one-vs-rest training strategy is applied for multi-class case, and square of hinge loss as well as L2 penalty are used. For BoCCl+BoW model using attention mechanism, neural network and softmax classifier are used for learning the attentional combination of BoCCl and BoW representations. The dimensionality of context vector as well as hidden representations are set to 200. The model is trained by minimizing the categorical cross entropy loss function using mini-batch stochastic gradient descent (SGD) with the Adam update rule [125]. However, for fair comparison, we used the same SVM classifier for the final classification of documents based on the combined representations learned (before the softmax layer).

3.5.3 Results and Analyses

3.5.3.1 BoC knowledge acquisition results

We first evaluated the number of related concepts and entities automatically acquired from the original knowledge base which consists of over 5 million concepts and 12 million entities. Table 3.2 shows the number of concepts and entities acquired from Probase for different document classification tasks.

It is observed that with increasing number of document classes, the amount of concepts and entities automatically acquired from Probase also increases. This is because more conceptual knowledge is needed for document classification tasks with many categories. Besides, more concepts and entities are acquired for general categories than specific categories. This explains the reason why BBC acquired more concepts and entities than BBCSport, and Yahoo acquired more concepts than 20NG, since the categories in BBC and Yahoo are more general. Compared with the original size of the knowledge base, the amount of acquired concepts and

Datasets	BoW				LSA				LDA			
	tf	tf_n	$tf - idf$	$tf - idf_n$	tf	tf_n	$tf - idf$	$tf - idf_n$	tf	tf_n	$tf - idf$	$tf - idf_n$
20NG	69.97	78.24	66.99	80.32	69.62	78.16	67.98	80.24	74.58	58.63	75.25	56.81
R8	95.93	97.26	96.21	97.53	96.48	96.99	96.62	97.40	95.20	91.14	94.20	90.45
R52	92.72	94.74	92.48	95.33	92.68	94.74	92.87	95.29	90.30	81.43	88.16	82.32
BBC	96.32	96.68	96.05	96.86	96.32	96.77	96.14	97.21	95.15	91.28	95.06	92.09
BBCSport	97.56	97.56	97.83	98.37	98.64	97.83	98.37	98.37	92.41	72.09	90.79	81.84

TABLE 3.3: Classification accuracies (%) of BoW, LSA and LDA using different weighting schemes.

Datasets	cs	cs_n	$cs - idf^s$	$cs - idf_n^s$
20NG	66.75	72.29	65.49	74.35
R8	94.20	94.75	93.79	95.43
R52	88.98	90.81	89.21	91.94
BBC	94.98	95.27	96.23	96.51
BBCSport	88.86	88.10	91.24	93.86

TABLE 3.4: Classification accuracies (%) of BoC using different weighting schemes.

entities is significantly reduced, resulting in a much more smaller search space. Besides, the noises in the model are reduced since irrelevant concepts and entities are eliminated.

3.5.3.2 Weighting schemes evaluation

Besides determining the indexing terms or features for document representation, the term weighting scheme which assigns weight for each term also plays an important role in document representation. In this experiment, we evaluated various weighting schemes for BoW, LSA, LDA, as well as the proposed BoC model.

For LSA and LDA, the weighting schemes of the input matrix were studied, which is the term-document matrix for dimensionality reduction. The optimal number of latent dimensions/topics for each dataset was found by grid search, and the best classification accuracy was reported. Table 3.3 shows the performance of BoW, LSA and LDA using different weighting schemes including term frequency (tf), term frequency-inverse document frequency ($tf - idf$), as well as L2 normalized tf (tf_n) and tf-idf ($tf - idf_n$).¹² The results of L1 normalization are not reported because the performance of L1 normalization is much worse than L2 normalization. From Table 3.3, it can be concluded that for BoW and LSA, L2 normalization can improve the performance significantly, and tf-idf weighting with L2 normalization achieved

¹²Yahoo dataset is not used in this experiment because it requires significant amount of time and memory to obtain a single result.

the best performance for document classification; whereas for LDA, tf weighting achieved the best performance, and normalization of term-document matrix will degrade the model performance significantly since LDA depends on count-based term-document statistics to infer probabilities. Although other weighting schemes may achieve the best performance for some particular datasets due to the special characteristics of the dataset such as size and word distribution, the performance difference with the best weighting scheme is marginal.

Table 3.4 shows the performance of BoC using different weighting schemes including concept score (cs), concept score-inverse soft document frequency ($cs - idf^s$), as well as L2 normalized cs (cs_n) and $cs - idf^s$ ($cs - idf_n^s$). Similar as BoW and LSA, L2 normalization improves the performance of BoC significantly. Besides, incorporating inverse soft document frequency also improves the performance since it is able to reduce the weights of frequent but less informative concepts for document classification. Therefore, the L2 normalized $cs - idf^s$ is the best weighting scheme for BoC model.

3.5.3.3 Statistical quality evaluation

We compared our proposed BoC models with several other benchmark document representation methods as mentioned in Section 3.5.2. For BoW, LSA, LDA, BoC and BoCCL, the best weighting scheme of each model was used. Particularly, for LSA and LDA, the optimal number of latent dimensions were used and the best results are reported. For Doc2Vec, we tuned the negative sampling rate, number of training epochs, as well as initial and minimum learning rates, and the best results are reported. Table 3.5 shows the classification accuracies of all the above mentioned models on six real-life document classification datasets. We also perform pair-wise comparison of algorithms using Wilcoxon Signed-Rank Test [126] to test the statistical significance of our results, the test results are shown in Table 3.6.

We found that the classification accuracy of BoW (with L2 normalized tf-idf weighting) outperforms many other document representations, indicating that BoW is a very strong baseline for document classification tasks. The reason is that BoW can effectively capture word-level statistics of a document, and majority documents can be classified into correct categories based on these word-level statistics. Even though high dimensionality and sparsity are associated with BoW representations,

Datasets	BoW	LSA	LDA	AE	Doc2Vec	BoC	BoCCl	BoCCl+BoW (concat)	BoCCl+BoW (atten)
20NG	80.32	80.24	75.25	68.70	68.36	74.35	75.52	80.68	81.78
R8	97.53	97.40	95.20	96.98	94.95	95.43	96.69	97.62	97.85
R52	95.33	95.29	90.30	92.72	91.07	91.94	92.25	95.27	95.87
BBC	96.86	97.21	95.15	96.41	96.13	96.51	97.09	97.53	98.04
BBCSport	98.37	98.64	92.41	98.10	97.37	93.86	94.33	98.30	99.08
Yahoo	72.10	72.13	69.91	68.00	66.46	70.02	71.15	73.88	75.26

TABLE 3.5: Classification accuracies (%) of various models on six document classification datasets.

	BoW	LSA	LDA	AE	Doc2Vec	BoC	BoCCl	Concat	Atten
BoW		-0.31	-2.20	-2.20	-2.20	-2.20	-1.99	-1.57	-2.20
LSA	=		-2.20	-2.20	-2.20	-2.20	-2.20	-1.15	-2.20
LDA	-	-		-0.31	-0.10	-1.57	-2.20	-2.20	-2.20
AE	-	-	=		-2.20	-0.10	-0.52	-2.20	-2.20
Doc2Vec	-	-	=	-		-1.36	-1.36	-2.20	-2.20
BoC	-	-	=	=	=		-2.20	-2.20	-2.20
BoCCl	-	-	+	=	=	+		-2.20	-2.20
Concat	=	=	+	+	+	+	+		-2.20
Atten	+	+	+	+	+	+	+	+	

TABLE 3.6: Pair-wise comparison of algorithms using Wilcoxon Signed-Rank Test with $\alpha = 0.05$. Z statistics are shown above the diagonal, bold indicates significantly different between algorithms with $z < -1.96$. Comparisons of the algorithms in the row with the column are shown below the diagonal, + and - indicate significantly better or worse respectively; = indicates no statistical significant difference.

SVM classifier is very robust to the dimensionality and sparsity of feature vectors and is able to tolerate irrelevant information effectively [127]. However, unlike our proposed BoC model, BoW cannot capture the semantic relatedness and conceptual information of texts, and also loses word order and phrasal information. The conceptual information is very important for making document classification decisions and it is the key to further improve the document classification performance.

LSA and LDA are both dimensionality reduction techniques based on BoW representations. The performance of LSA is comparable with BoW, with the cost of searching the optimal number of latent dimensions. The performance of LDA is worse than BoW and LSA, because it is a generative model that focuses on the generative process of document collections instead of discrimination of document categories. The latent dimensions or topics of LSA and LDA are derived from word frequencies or co-occurrences, hence the semantic relatedness and conceptual information are also ignored. Besides, LSA and LDA are highly specific to the

training corpus, resulting in bad generalization capability. On the contrary, our proposed BoC model utilizes external probabilistic encyclopedic knowledge base to capture the semantic relatedness and conceptual information of texts. The higher level semantics (i.e. concepts) captured in BoC are more reliable and robust since the conceptualization is based on human knowledge.

AE and Doc2Vec are both document representation methods based on distributed representation of words or documents. Results show that AE cannot achieve comparable results as BoW model, because the averaging operation of the word embeddings is too simple to capture the complex semantics of documents. For Doc2Vec, complicated hyper-parameter tuning is needed to achieve better performance. However, the results of Doc2Vec are even worse than AE for our document classification tasks. The reason is that AE utilizes pre-trained word embeddings that are learned from large corpus, whereas Doc2Vec needs to learn embeddings from scratch based on our small and domain specific corpus, hence it is very hard for Doc2Vec to effectively capture the semantic information on document embeddings.

Comparing between our proposed BoC and BoCCl models, results show that BoCCl substantially outperforms BoC. The reason of the improvement is two-fold: one is that the concept clustering algorithm of BoCCl groups similar concepts into clusters. Hence, redundant concepts are reduced, and the dimensionality as well as sparsity of the feature space are reduced significantly, making the training on small datasets more efficient. The other is that BoCCl utilizes context information to perform entity sense disambiguation, making the conceptualization process more accurate. Even though the performances of BoCCl is better than LDA, and also comparable with AE, they are not as good as BoW representation. This is because that BoCCl focuses on the conceptual information of entities within the document, but ignores part of the word-level information which is also important for determining the document categories. On the contrary, BoW focuses only on the word-level information but ignore the concept-level information of texts. Hence, the two representations complement with other, and once combined properly, the best document classification performance can be achieved. As shown in Table 3.5, the combination of BoCCl and BoW representations achieved the best classification results. The direct concatenation of BoCCl and BoW results in little improvement, the reason may be that the two representations use different weighting schemes and the dimensionality of BoW is significantly larger than BoCCl. However, when we

Datasets	BoW	LSA	LDA	BoC	BoCCl
20NG	49241	8000	6000	12035	4168
R8	10553	700	600	5776	2997
R52	12029	5000	4000	8624	3172
BBC	12150	500	1000	4662	2891
BBCSport	5198	100	80	476	419
Yahoo	321210	6000	5000	23501	4939

TABLE 3.7: Dimensionality of different document representations learned from six datasets.

combine the two representations using attention mechanism, the final representation is able to select word-level and concept-level information attentionally for different classification tasks and the performance improved significantly.

3.5.3.4 Semantic quality evaluation

We first evaluated the dimensionalities of the document representations learned using the above mentioned approaches. High dimensional feature vector means using a large amount of features to represent a document, which will affect both semantic interpretability and statistical quality. Although SVM is robust to the dimensionality of feature vectors, this is not the case for other classifiers where feature selection is needed to reduce the feature space to manageable dimensions. Besides, it is computationally cheaper to operate the classifier in low dimensional spaces.

Among the above mentioned models, AE and Doc2Vec have the lowest dimensionality, which is equal to the dimensionality of word embeddings (300). However, the features are derived from distributed representation of words, and they are not interpretable. Hence, is it very hard for human to discover the semantic meanings from the representations. Table 3.7 shows the representation dimensionality of BoW, LSA, LDA, BoC and BoCCl learned from different document classification tasks. For LSA and LDA, the dimensionality is equal to the number of latent dimensions/topics, which is a pre-defined hyperparameter. We report the dimensionalities of LSA and LDA which achieved the best performance on each dataset. Results show that LSA and LDA have the lowest dimensionalities, however, the optimal dimensionality differs a lot for different datasets, hence careful hyper-parameter tuning is needed. The dimensionality of BoC is much lower than

BoW, since BoC is operated in the concept space which has much lower dimensionality than word space. BoCCI further reduces the dimensionality by clustering similar concepts together, resulting more appropriate dimensionalities.

In order to show the representation interpretability of different models, we discovered the top-weighted features from different representations. We took two documents from different categories as examples. The first document is from “sport” category and the second document is from “entertainment” category of BBC dataset, as shown below:

1. Aston Villa’s Carlton Cole could be out for six weeks with a knee injury. The striker, who is on a season-long loan from Chelsea, picked up the knock in an England Under-21 match against Holland earlier this month. “Carlton will be out of action for four to six weeks after a bad challenge,” said Villa boss David O’Leary. “I won’t be able to tell you whether he will need an operation until maybe next week. Whether he has an operation has got to be left to Chelsea.” Cole, who also struggled with an ankle problem earlier in the season, was unable to rest because O’Leary had a shortage of strikers. The return to fitness of Darius Vassell after four months out with a broken ankle and the emergence of Luke Moore has alleviated some of the Villa’s manager’s problems in that department.
2. The sixth and final Star Wars movie may not be suitable for young children, filmmaker George Lucas has said. He told US TV show 60 Minutes that *Revenge of the Sith* would be the darkest and most violent of the series. “I don’t think I would take a five or six-year-old to this,” he told the CBS programme, to be aired on Sunday. Lucas predicted the film would get a US rating advising parents some scenes may be unsuitable for under-13s. It opens in the UK and US on 19 May. He said he expected the film would be classified PG-13 - roughly equivalent to a British 12A rating. The five previous Star Wars films have all carried less restrictive PG - parental guidance - ratings in the US. In the UK, they have all been passed U - suitable for all - with the exception of *Attack of The Clones*, which got a PG rating in 2002. *Revenge of the Sith* - the third prequel to the original 1977 Star Wars film - chronicles the transformation of the heroic Anakin Skywalker into the evil Darth Vader as he travels to a Hell-like planet composed of erupting volcanoes and molten lava. “We’re going to watch him make a pact with the devil,” Lucas said. “The film is much more dark, more emotional. It’s much more of a tragedy.”

Table 3.8 shows the top 3 weighted features of different representations for the two documents. For BoW, the top-weighted words are shown directly. For LSA and LDA, the

Models	Document 1 (sport)
BoW	villa, cole, leary
LSA	l_{110} : ibm, patents, villa, fiat, students l_{109} : children, lee, england, kennedy, spider l_{111} : children, villa, israel, radio, award
LDA	t_{447} : ankle, villa, cole, injury, six t_{586} : villa, leary, us, southampton, three t_{768} : foot, robben, weeks, injury, six
BoC	club: manchester united, chelsea, arsenal, real madrid, liverpool operation: welding, sawing, copying, drilling, burning player: lebron james, kobe bryant, cristiano ronaldo, messi
BoCCl	club: manchester united, chelsea, arsenal, real madrid, liverpool operation: welding, sawing, copying, drilling, burning player: lebron james, kobe bryant, cristiano ronaldo, messi
Models	Document 2 (entertainment)
BoW	wars, lucas, rating
LSA	l_0 : said, mr, would, people, year l_{347} : green, galloway, hybrid, glazer, song l_4 : film, best, awards, award, economy
LDA	t_{482} : film, awards, said, star, nominations t_{618} : film, michael, said, life, different t_{327} : morrison, album, label, record, mark
BoC	film: star wars, matrix, panic room, sin city, harry potter movie: star wars, matrix, jurassic park, avatar, terminator star war character: darth vader, luke skywalker, han solo, r2d2
BoCCl	movie: star wars, matrix, jurassic park, harry potter, avatar star war character: darth vader, luke skywalker, han solo, r2d2 rating: default, star, high, low, medium

TABLE 3.8: Top 3 features of different representations for two documents from “sport” and “entertainment” categories respectively.

top-weighted latent dimensions/topics are recorded, and the meaning of each latent dimension/topic is shown using the top words it covers. For BoC and BoCCl, the name of the top-weighted concepts/concept clusters is recorded, and the meaning of each concept/concept cluster is shown using the top entities it contains. From the two examples, it is obvious that BoC and BoCCl provide much greater interpretability than other representations. BoW only provides sparse words from the document, which is very hard to discover the meaning of the document. LSA performs linear combination of words which will benefit statistical discrimination, however, semantic interpretation of each latent dimension is loss. LDA is a generative model that discovers document-topic and topic-word distributions based on word co-occurrences, and it provides better interpretability than LSA. However, the topics of LDA still lack intuitive interpretations because of the error-prone corpus-based inferences. Besides, these models only consider the surface form of words, hence the semantic and conceptual information of words and phrases cannot be captured.

Evaluator	BoW	LSA	LDA	BoC	BoCCl
No. 1	66	48	74	74	76
No. 2	46	42	58	70	70
No. 3	62	62	68	70	72
avg $\pm$ std	58.0 $\pm$ 10.6	50.7 $\pm$ 10.3	66.7 $\pm$ 8.1	71.3 $\pm$ 2.3	72.7 $\pm$ 3.1

TABLE 3.9: Classification accuracy (%) of 50 samples from BBC dataset based on human judgment of the top 5 weighted features of each algorithm. “avg” and “std” indicate average and standard deviation values of the three evaluators respectively.

Our proposed BoC and BoCCl conceptualize documents into concepts based on external knowledge base, the concepts reveal the higher-level semantic and conceptual meanings of texts and provide intuitive interpretations. For example, in Document 1, our model is able to recognize “Aston Villa” as “club” and “Carlton Cole” as “player”, which is impossible for other models. Contrary to LSA and LDA, the meaning of each concept is explicitly given, and the entities involved are reliable and understandable as they are inferred based on human knowledge. The difference between BoC and BoCCl is that semantic similar concepts are grouped together in BoCCl to further reduce the concept space dimensionality. For example, in Document 2, the concepts “film” and “movie” are grouped into a single concept cluster “movie”.

We further conduct quantitative evaluation to prove the better semantic interpretability of our model. We randomly sample 50 documents from BBC dataset for human evaluation. The experiment is conducted on three evaluators and only the top 5 weighted features of each sample are shown to the evaluators for them to determine the category of the document. For BoW, the top-weighted words are shown directly; for LSA and LDA, the meaning of each top-weighted latent dimensions/topics is shown using the top words it covers; for BoC and BoCCl, the top-weighted concepts/concept clusters are shown. Table 3.9 shows the classification accuracy of different algorithms based on human judgment of the top 5 weighted features. Evaluation results show that the proposed BoC and BoCCl are the most interpretable representations which achieve higher evaluation accuracy than other representations. Besides, the lower standard deviation of BoC and BoCCl also demonstrates that the understanding of the semantic meaning of features is more consistent across different evaluators. With the interpretable BoC and BoCCl representations, we are able to have a deeper understanding of documents and a clearer operation logic for further reasoning and decision making.

3.6 Conclusion

In this chapter, we address the problems of BoW representation and propose a novel Bag-of-Concepts (BoC) representation for document classification. The useful conceptual knowledge for a certain document classification task is automatically acquired from a large probabilistic knowledge base. Based on these conceptual knowledge, the entities (words and phrases) in the document are conceptualized into concepts in a probabilistic way, which capture their semantic relatedness and higher-level conceptual meanings. By aggregating the conceptualized entities within the document, a document is conceptualized into relevant concepts and eventually being represented as a distributed vector in the learned concept space using the proposed concept score-inverse soft document frequency ($cs - idf^s$) weighting scheme. We also propose Bag-of-Concept-Clusters (BoCCl) model to further improve BoC representation by grouping semantically similar concepts into clusters and performing entity sense disambiguation. Furthermore, we combine the BoCCl and BoW representations using an attention mechanism to effectively utilize both concept-level and word-level information to achieve optimal performance for document classification.

Both statistical and semantic qualities of the proposed BoC and BoCCl representations are evaluated and compared with other benchmark document representations. Experimental results show that BoC and BoCCl are able to effectively capture the concept-level information of documents and improve the performance of document classification. More importantly, BoC and BoCCl provide great interpretabilities which allow human to have a deeper understanding of the document and a clearer operation logic for further reasoning and decision making.

Chapter 4

Incorporate Knowledge into Convolutional Neural Network

4.1 Introduction

Relation extraction is a sub-field of Information Extraction (IE) that automatically extracts semantic relations between entities, such as component-whole, product-producer, employer-employee, cause-effect, and message-topic. In particular, the cause-effect relation plays an important part in human cognition due to its significant impact on reasoning and decision making [128, 129]. Causal relation extraction focuses on automatic detection of cause-effect relationship between entities in text. For example, from the sentence “the 2004 Indian Ocean earthquake triggered a series of devastating tsunamis”, we can infer causal relation between the cause “earthquake” and the effect “tsunamis”. Similarly, in “global warming resulted in higher sea levels”, the causal relation between the cause “global warming” and the effect “higher sea levels” can also be extracted. Effective extraction of causal relation from natural language texts becomes increasingly important for applications such as information retrieval [130, 131], question answering [132, 133], and event reasoning and predictions [134–137]. By building causal networks (e.g. causal chain / causal map), we can induce previously unknown knowledge and apply it in various domains such as biology [138], biomedicine [139], finance [140, 141] and environmental science [142].

Numerous efforts have been dedicated to extract causal relation from texts. The methods developed in literature fall into two categories: rule-based (non-statistical) method or machine-learning-based (statistical) method. For rule-based methods, extensive manual

work is required to construct handcrafted linguistic patterns including lexico-syntactic patterns and semantic patterns to infer causal relation from text based on pattern matching [130, 132, 143–145]. However, regardless of extensive human effort, we cannot exhaustively conclude all the linguistic patterns of causal relation due to the complexity of causal relation expressions in natural language, including lexical, morphological and syntactic variations. Besides, the polysemous patterns with exact pattern matching yield low precision and recall due to lexical ambiguities. For machine-learning-based methods, causal relations are automatically inferred from large amount of labeled data. In early years, people construct rich features (lexical, syntactic or semantic features) by sophisticated feature engineering, where the handcrafted features are carefully designed and adjusted for specific patterns and domains [146–149]. The performance of the system strongly depends on the quality of the designed features. Particularly, many features require external NLP toolkits such as POS tagger, dependency parser, named entity recognizer and sentence parser. Many NLP toolkits are imperfect and will cause error propagation to the the causal relation extraction system.

Recently, with the prevalence of deep learning [25], researchers started to construct models without complicated feature engineering and to minimize the reliance on NLP toolkits for feature acquisition. One of the most prominent deep learning models for relation extraction is Convolutional Neural Network (CNN) with pre-trained word embeddings, which achieved state-of-art performance for SemEval-2010 task 8 challenge [150–152]. The pre-trained word embeddings encode semantic and syntactic information of words into fixed-length vectors [2, 153] and CNN is able to extract meaningful n-grams from sentences [27]. Compared with rule-based methods and rich features-based methods, the CNN models with word embeddings are able to extract complex causal relations more effectively. However, these models rely on large amount of training data which should cover all the causal relation expressions in natural language. This is impractical due to the variety and ambiguity of words and sentences in natural language. With a large number of free parameters in deep learning models, the models can easily over-fit the limited and biased training data, and thus hinder the performance of deep learning models.

In this chapter, we introduce a novel deep learning model called Knowledge-oriented Convolutional Neural Network (K-CNN) which is specially tailored for causal relation extraction. The model combines human prior knowledge and the information learned from data in a complementary way to extract causal relation from natural language texts. The knowledge-oriented channel of K-CNN incorporates the linguistic knowledge of causal relationship from lexical knowledge bases including FrameNet and WordNet to capture the significant linguistic clues of causal relationship. In conventional CNN, the weights of convolutional filters are trained using training data via back-propagation, whereas the

convolutional filters of knowledge-oriented channel are automatically generated from lexical knowledge bases to represent keywords and cue phrases of causal relations, which are called word filters in our work. The data-oriented channel of K-CNN uses conventional CNN which gives the model enough capacity to adjust itself to learn other important features of causal relationship from training data. The weights of the word filters in knowledge-oriented channel are equal to the pre-trained word embeddings and are kept fixed, hence the number of free parameters of the model is significantly reduced. We also propose word filter selection and clustering techniques to remove non-discriminative and redundant features. Additionally, two types of semantic features including WordNet categorical features and FrameNet causal scores are created to capture useful semantic information hidden in the texts.

The main contributions of this chapter are summarized as follows:

- * We propose an effective way of combining human prior knowledge and information from data for convolutional neural network. The convolutional filters in knowledge-oriented channel of K-CNN are automatically generated from lexical knowledge bases. Compared with conventional CNN, the proposed word filters are able to represent keywords and cue phrases of causal relationship more accurately. The specially designed convolutional operations allow the model to extract these significant clues of causality from texts and output a similarity score for each word filter, this overcomes the problem of rigid pattern matching for rule-based methods. As word filters are static, the number of free parameters in K-CNN is significantly reduced compared with conventional CNN. Hence, overfitting can be alleviated and training efficiency can be improved. Besides, the word filters provide more interpretability than random initialized convolutional filters since each word filter has a specific semantic meaning.
- * The proposed word filter selection and clustering techniques for dimensionality reduction can further improve the performance of K-CNN. Firstly, the insignificant word filters are removed using hypothesis testing based on Analysis of Variance (ANOVA) F-ratio; Then the redundant features are removed by clustering of the remaining word filters based on their semantic similarities. The convolutional results for word filters within each cluster are grouped together by using either average-pooling or max-pooling, the performances of the two pooling methods are investigated.

- * Two types of semantic features including WordNet categorical features and FrameNet causal scores are proposed, these features are able to discover more semantic information of causal relationship hidden in the text, which allow our model to better deal with complex and implicitly expressed causal relations.
- * Comprehensive experiments are conducted on three datasets for causal relation extraction. The experimental results demonstrate that our proposed K-CNN is able to extract causal relations more effectively than conventional CNN models.

This chapter is organized as follows. In Section 4.2, some related works using machine-learning-based method are reviewed. Our proposed Knowledge-oriented CNN (K-CNN) for causal relation extraction is presented in Section 4.3. Experimental results and analyses are presented in Section 4.4. Finally, our work is summarized in Section 4.5.

4.2 Related Work

Due to the performance advantage of statistical methods over non-statistical methods for causal relation extraction in the literature, we concentrate on statistical methods in this section. To be more specific, we focus on supervised systems including basic machine-learning models as well as deep neural network models. The difference of the two types of models is that sophisticated feature engineering is normally required for basic machine-learning models such as Bayes classifier, Support Vector Machine (SVM) and Decision Tree; whereas for deep neural network models, features are automatically learned from training data.

4.2.1 Models with Sophisticated Feature Engineering

There are two main steps for supervised machine-learning-based approaches, including feature extraction/generation and pattern classification. Various methods were proposed for both steps in the literature.

Girju et al. [154] firstly generate syntactic patterns in the form of $\langle NP1 \text{ causal-verb } NP2 \rangle$ by searching a collection of texts on the Internet, where $NP1$ and $NP2$ are found from lexical knowledge base WordNet. Then, the extracted patterns are validated and ranked by imposing some constraints on $NP1$, $NP2$ and *causal-verb* through a WordNet-based coarse-grained process. Their work was modified in [132] where the semi-automatic pattern validation process is replaced by a supervised method using C4.5 decision tree. Even

through the causal relation is restricted to noun pairs with certain causal verbs as the connector, Girju and Moldovan’s work demonstrates that machine-learning framework is more effective than conventional rule-based framework. Instead of using large amount of causally-labeled corpus, which is time and effort consuming, Chang and Choi [155] generate ternary patterns $\langle \text{cause NP}, \text{cue phrase}, \text{effect NP} \rangle$ from dependency structure of sentences. The cue phrase probability and lexical pair probability of the ternary patterns are generated from unlabeled corpus by using EM (Expectation-Maximization) procedure. These probabilities are used as features and Naive Bayes classifier is used to classify the pattern as causal or non-causal. In [156], seven types of features are constructed, including the types of relator, left and right modifiers of the relator, semantic classes of cause verb (as well as effect verb), whether the cause verb (as well as effect verb) is potentially causal, and the tense of the cause and effect verbs. With these features, Bagging with C4.5 decision trees classifier is used to make binary classification, and a satisfying classification result is achieved. However, the model considers only causal relations with a fixed pattern $\langle \text{VerbPhrase Relator Cause} \rangle$ containing only four relator types: *after*, *as*, *because* and *since*, which is too simplified for causal relationship.

With the development of benchmarked dataset for semantic analysis tasks in NLP, such as SemEval-2007 task 4 and SemEval-2010 task 8 for relation classification, the semantic relations analyses including causal relation extraction have been improving and more novel methods have been developed. The winners of the two tasks ([157] for SemEval-2007 task 4 and [158] for SemEval-2010 task 8) both use a combination of lexical, syntactic and semantic features extracted from various knowledge resources (e.g. WordNet) and NLP toolkits (e.g Syntactic Parser), and Support Vector Machines (SVMs) as the classifier. Although the classification results for Cause-Effect relation are satisfying, most samples in the datasets are simple causal relations which are explicitly expressed with significant linguistic clues (such as “cause”, “result in”, “because”), the extraction of complex and implicitly expressed causal relations is still a challenging task.

Recently, many works are proposed to handle more complex causal relations. Rink et al. [159] proposed a novel graph pattern based framework. The lexical, syntactic, and semantic features of a sentence are automatically extracted and encoded into a graph representation which shows the dependency between features. The most representative graph patterns for causal relationship are discovered and the sentence is classified based on these pattern features using SVM. Sorgente et al. [147] combined both rules and machine-learning methods for causal relation extraction. The rules are defined based on lexico-syntactic patterns and dependency structure of the sentence to extract possible cause-effect pairs from a sentence, then a Bayesian classifier with Laplace smoothing is used to discard incorrect pairs. Yang and Mao [148] proposed a multi-level relation

extraction algorithm (MLRE) to extract all potential causal relations with any verb or preposition based on the linguistic structures of dependency grammar and constituent grammar. Informative features are found based on lexical knowledge bases (WordNet, VerbNet and FrameNet) and feature selection technique. Ensemble learning using restricted boosting and SVM with RBF kernel are used to make classification. Zhao et al. [160] proposed a Restricted Hidden Naive Bayes model based on causal connectives analysis.

Despite the significant improvements made by basic machine-learning models over rule-based models, there are still many problems in current systems. Firstly, sophisticated feature engineering is required as the model rely on a suitable feature set with good separation capability, which is labor and time consuming and hard to achieve. Secondly, many features are extracted based on existing NLP toolkits, such as POS tagger, dependency parser, named entity recognizer etc. Some NLP toolkits are imperfect and the errors caused by these toolkits will propagate to the causal relation extraction system. Thirdly, many systems are domain-dependent and the features need to be re-designed when adopted for other domains.

4.2.2 Deep Neural Network Models

The goal of constructing deep neural network models is to allow the model to learn and extract useful features automatically and more effectively than handcrafted features designed by human. In NLP, such models are primarily based on a distributed representation of words in vector space, called word embeddings, which is capable of capturing semantic and syntactic properties of words [2, 153]. Among various deep neural network models, two most widely used models for relation classification are recursive neural networks (RNNs) and convolutional neural networks (CNNs).

Socher et al. [31] presented a novel method for relation classification using recursive neural network (RNN) to learn compositional vector representations for phrases and sentences through the syntactic tree path that connects two nominals, and applied these representations to classify the semantic relationship between them. Hashimoto et al. [161] also used an RNN for relation classification, the novelty is that their method allows the explicit weighting of important phrases for the target task. However, RNN-based methods inherently require syntactic parse trees which may introduce error propagation.

Convolutional neural network (CNN) is a special feed-forward neural network whose layers are formed by convolution operations followed by pooling operations [49]. Recently,

CNN has demonstrated its advantage in capturing semantic and syntactic information of n-grams and inducing more abstract and discriminative representations of textual inputs [26, 27]. Based on the idea of modeling sentences using CNN, many works have been proposed for relation extraction and achieved start-of-art performance. Zeng et al. [150] first used position embeddings together with pre-trained word embeddings for relation classification; Nguyen and Grishman [151] improved their model by using convolutional filters with multiple window sizes and fine-tuning the pre-trained word embeddings; Instead of using CNN followed by a softmax classifier, Santos et al. [152] proposed a classification by ranking CNN (CR-CNN) that learns a distributed vector representation for each class by minimizing a novel pairwise ranking loss function. Lee et al. [162] further added entity type and part of speech information into the embedding and used rule-based post-processing to correct the relations detected by CNN, their system ranked first in SemEval-2017 task 10 for relation extraction in scientific articles.

Despite the better performance of deep neural network models, the models normally require large amount of labeled data to learn, and it is impractical to construct sufficient training data that covers all causal relation expressions in natural language. With a large number of free parameters in deep learning model and insufficient or biased training data, the model can easily over-fit the training data, resulting in bad generalization on test data. There are some works [163, 164] that learn distributed document embeddings based on unsupervised context modeling, called Paragraph Vector [90]. The models require less training data and able to capture word semantics and n-gram information from texts as well. The resulted document embeddings have the property that the documents containing semantically similar n-grams have closer embeddings in vector space. However, the embeddings learned in this way may not be suitable to be used for causal relation extraction directly, as sentences with semantically similar n-grams may or may not involve causal relation. For example, “the *tsunami* generated by *earthquake* killed hundreds of thousands of people” involves causal relation between “tsunami” and “earthquake”, whereas “the *tsunami* and *earthquake* killed hundreds of thousands of people” does not. Our work attempts to automatically construct the convolutional filters of CNN based on the linguistic knowledge of causal relation from lexical knowledge bases, which allows the model to extract significant clues of causality from natural language texts effectively and precisely, and this alleviates the overfitting issue of deep learning models.

4.3 Knowledge-oriented CNN (K-CNN)

In this section, our proposed Knowledge-oriented Convolutional Neural Network (K-CNN) is presented in detail. The overall architecture of our proposed K-CNN is shown in Figure 4.1. The model combines human prior knowledge and information from data to extract causal relationship. It consists of two CNN channels, one is called knowledge-oriented channel, which incorporates existing linguistic knowledge from lexical knowledge bases; the other is called data-oriented channel, which learns important features from data. The two channels complement with each other and extract useful features of causal relation from different perspective.

FIGURE 4.1: Overall architecture of K-CNN.

4.3.1 Knowledge-oriented Channel

The knowledge-oriented channel is the main part of K-CNN, which effectively extracts keywords and cue phrases of causal relationship from sentence. The convolutional filters in this channel are automatically generated based on the linguistic knowledge of causal relationship in lexical knowledge bases including WordNet¹ and FrameNet², we call them word filters. Compared with the convolutional filters in conventional CNN, word filters are able to represent linguistic clues of causal relationship more precisely. More importantly, the weights of word filters are embeddings of the words, which are pre-trained and directly used without any additional training. Hence, the number of free parameters of the model can be significantly reduced, alleviating the overfitting issue when the amount of training data is small. The detailed architecture of the knowledge-oriented channel of K-CNN is shown in Figure 4.2.

¹Available online at <https://wordnet.princeton.edu>.

²Available online at <https://framenet.icsi.berkeley.edu/fndrupal>.

FIGURE 4.2: Knowledge-oriented channel of K-CNN.

4.3.1.1 Sentence Representation

The input to K-CNN is a sentence marked with two target entities e_1 and e_2 for causal relation identification. It is observed that majority of keywords and cue phrases of causal relationship such as “cause”, “lead to”, and “result in” appear in the words between e_1 and e_2 , and causal relations can be identified by looking into these words. Keywords and cue phrases appear far away from two target entities may not be informative and will affect the classification result, as illustrated in the examples in Section 4.3.2.1. To remove noise and effectively extract these linguistic clues from the sentence, we only use words between two target entities as input to the knowledge-oriented channel. To reduce morphological variations of words and make them consistent with WordNet tokens, each word is transformed into its lower case and base form using WordNet lemmatizer. To capture the syntactic and semantic information of words, each word is represented by a vector $\mathbf{w} \in \mathbb{R}^e$ by looking up the word embedding table $\mathbf{W}^{\text{word}} \in \mathbb{R}^{e \times |V|}$ that is pre-trained using large corpus, where e is the dimensionality of word embedding vectors and $|V|$ is the vocabulary size. As CNN only works with fixed length inputs, the number of word tokens in a sentence is fixed at n_1 , which is the maximum number of words between e_1 and e_2 . The sentences with less than n_1 tokens are padded using a special padding

character with the zero embedding vector. Hence, the input $x_K = \{x_1, x_2, \dots, x_{n_1}\}$ is represented as a sequence of real-value vectors $\mathbf{emb}_K = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{n_1}\}$.

4.3.1.2 Automatic Word Filter Bank Generation

This process automatically generates convolutional filters of CNN for causal relation extraction without training the model using large amount of data. The filters are actually the embeddings of causation words, which are extracted from two publicly available lexical knowledge bases, WordNet and FrameNet.

WordNet is a large lexical database that groups English words into sets of synonyms (called synsets) to represent different meanings or concepts [37]. The synsets are connected in a hierarchical structure to reflect their lexical and semantic relations, and the meaning of each synset is explained by a gloss with some examples. The example below shows the WordNet elements of one of the synsets for the word “cause”.

- **Synset Members:** cause, do, make.
- **Gloss:** give rise to; cause to happen or occur, not always intentionally.
- **Example:** “cause a commotion”; “make a stir”; “cause an accident”.
- **Direct Troponym:** determine, initiate, effect, make, occasion, provoke, motivate...
- **Direct Hypernym:** make, create.
- **Derivationally Related Form:** causative, causation, cause.
- **Sentence Frame:** Somebody —s something; Something —s something.

FrameNet is a lexical resource built on the frame semantic theory, and it categorizes English words and sentences into higher level semantic frames describing different concepts [38]. Each frame is a conceptual structure describing a type of event, relation or object, which is provided with conceptual definition, elements participated in the frame (frame elements), words that often appear in the frame (lexical units) and relationship with other frames. The example below shows the FrameNet elements of the “causation” frame.

- **Definition:** A Cause causes an Effect. Alternatively, an Actor, a participant of a (implicit) Cause, may stand in for the Cause. The entity Affected by the Causation may stand in for the overall Effect situation or event.

- **Frame Elements (FEs):**
 - Core: Actor, Affected, Cause, Effect.
 - Non-core: Circumstances, Concessive, Explanation, Frequency, Manner, Means, Place, Time.
- **FE Core Sets:** {Actor, Cause}, {Affected, Effect}.
- **Examples:** “He MADE me angry.”, “If such a small earthquake CAUSES problems, just imagine a big one!”, “The strange mutations of the rumor mill in the end LED to it being said that he was actually a woman.”...
- **Lexical Units:** because of, cause, consequence, due to, for, force, lead (to), result (in), since...
- **Frame-frame Relations:**
 - Inherits from: Eventive_affecting
 - Is Inherited by: Cause_to_start
 - Perspective on: Causation_scenario
 - Is Perspectivized in: Means
 - Is Used by: Level_of_force_exertion

FrameNet contains over 1,200 semantic frames, 13,000 lexical units and 202,000 example sentences, some of the semantic frames can be used as the evidence for causal relation [165], we refer them as “causal frames”. In our work, we identify 40 causal frames from FrameNet including “Causation”, “Causation_Scenario”, “Triggering”, “Reason”, “Explaining_the_facts”, “Response” as well as 34 frames start with “Cause”. The lexical units involved in these causal frames are the significant clues and frequently occurring words that invoke causality in the sentence, hence these lexical units can be treated as keywords and cue phrases for causal relations. We further extend these lexical units using WordNet for a broader coverage of causal words and construct a bank of causal words automatically. The weights of the convolutional filters of CNN (word filters) are found based on these causal words and word embeddings. The details of the algorithm for automatically generating word filter bank are described in Algorithm 2.

The word filters constructed in this way have physical meanings that represent keywords and cue phrases of causal relationship. These word filters are constructed based on human prior knowledge of causal relationship, they are more precise than conventional convolutional filters that are learned from training data. Moreover, the weights of these

Algorithm 2: Automatic word filter bank generation

Step 1: Find all the lexical units of 40 causal semantic frames from FrameNet, group them according to No. of words (max: 3).

$$lu_1 = \{c_1, c_2, \dots, c_{n_1}\}$$

$$lu_2 = \{[c_{11}, c_{12}], [c_{21}, c_{22}], \dots, [c_{n_21}, c_{n_22}]\}$$

$$lu_3 = \{[c_{11}, c_{12}, c_{13}], [c_{21}, c_{22}, c_{23}], \dots, [c_{n_31}, c_{n_32}, c_{n_33}]\}$$

Step 2: Extend lexical units of lu_1 using WordNet.

for $word$ in lu_1 **do**

for $synset$ in WordNet synsets of $word$ **do**

if {“cause”, “effect”, “causal”, “causation”, “result”, “reason”, “because”, “responsible”} in WordNet gloss of $synset$ **then**

for $lemma$ in WordNet lemmas of $synset$ **do**

if length of $lemma$ == 1 **then**

$$lu_1 = lu_1 + lemma$$

else if length of $lemma$ == 2 **then**

$$lu_2 = lu_2 + lemma$$

else if length of $lemma$ == 3 **then**

$$lu_3 = lu_3 + lemma$$

end if

end for

end if

end for

end for

Step 3: Generate CNN convolutional filter weights.

for each lexical unit $[c_1, \dots, c_k]$ in $lu_k, (k = 1, 2, 3)$ **do**

 the corresponding filter weights is:

$$\mathbf{f} = [\mathbf{f}_1, \dots, \mathbf{f}_k]^T$$

 where $\mathbf{f}_i \in \mathbb{R}^e$ is the word embedding of c_i found by looking up the word embedding table $\mathbf{W}^{\text{word}} \in \mathbb{R}^{e \times |V|}$, and k is the convolutional window size.

end for

filters are static values instead of free parameters in the model. Hence, the number of free parameters in the model is significantly reduced, alleviating the overfitting issue when the amount of training data is small. Eventually, around 650 uni-gram, 140 bi-gram and 10 tri-gram word filters are generated from lexical knowledge bases.

Besides WordNet and FrameNet, there are a few other popular lexical knowledge bases such as PropBank [166], VerbNet [93] and OntoNotes [167], which can also be utilized to extract the linguistic knowledge of causal relation. Both PropBank and VerbNet can be used to find the keywords of causation. However, unlike FrameNet which is semantically motivated and contains lexical units with various part of speeches, PropBank and VerbNet are verb-oriented and focus more on syntactic level. Hence, many important

linguistic clues of causal relation such as “because”, “since”, “from”, “due to”, and “result in” cannot be extracted, and many verbs with no causal meaning may be extracted unexpectedly, producing more noises to the causal relation extraction system. OntoNotes sense groupings can be used to extend the causal words found from FrameNet. However, word senses in OntoNotes are more coarse-grained compared with WordNet, resulting in many irrelevant words.

4.3.1.3 Convolution and Pooling Operations

To capture the significant linguistic clues of causal relationship in the sentence, the word filters generated by Algorithm 2 are convolved with the n-grams in the sentence to produce a sequence of similarity scores. Compared with rule-based pattern matching, the proposed convolutional approach is able to capture semantically similar causal words other than the words in the word filter bank.

To be more specific, the word filters $\mathbf{f} = [\mathbf{f}_1, \dots, \mathbf{f}_k]^T$ generated by Algorithm 2 are convolved with the input matrix $\mathbf{emb}_K = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{n_1}\}$, where $k \in [1, 2, 3]$ is the convolutional window size which also represents the k-grams in the sentence. We modified the conventional convolutional operation of CNN to make each word filter generate a feature map $\mathbf{m} = [m_1, m_2, \dots, m_{n_1-k+1}]$, where m_i represents the similarity between the word filter $\mathbf{f}$ and the k-gram $\mathbf{w}_{kgram} = [\mathbf{w}_i, \dots, \mathbf{w}_{i+k-1}]^T$ in the sentence. The modified convolutional operation is expressed in Equation 4.1.

$$m_i = (\sum_{j=1}^k \mathbf{f}_j^T \mathbf{w}_{i+j-1} + b) / k \quad (4.1)$$

where b is a bias term. Unlike conventional CNN of applying a non-linear function to the convolution result, we divide the convolution result by the window size k instead. By restricting $\mathbf{f}_j$ and $\mathbf{w}_{i+j-1}$ (word embeddings) as unit vectors, the resulting value of m_i becomes cosine similarity between $\mathbf{f}$ and $\mathbf{w}_{kgram}$. The mathematical proof is shown in Appendix A. The reason why we produce cosine similarity in the feature map is that we want equal scale for all convolutional window sizes, hence achieving equal importance for word filters with different lengths. Whereas conventional convolution operation will result in higher value for longer window size.

We use max-pooling to further aggregate the convolution results for each filter and extract the most significant or relevant feature from the feature map. The max-pooling operation

for each feature map is expressed in Equation 4.2.

$$p = \max \{\mathbf{m}\} = \max \{m_1, m_2, \dots, m_{n_1-k+1}\} \quad (4.2)$$

The rationale for taking the maximum value from the feature map is that the largest cosine similarity indicates strong clue of the existence of causal keywords or cue phrases in the sentence.

4.3.1.4 Word Filter Selection and Clustering

There are nearly 800 word filters generated from Algorithm 2, which is considered as high dimensionality compared with limited training data, and many features generated from these word filters may be irrelevant or redundant for causal relation classification task. To boost the performance of the model, word filter selection and clustering techniques are proposed for dimensionality reduction.

Word Filter Selection

We first remove non-discriminative features produced from the word filters which do not provide enough class separability. Based on the training data and their labels, we use analysis of variance (ANOVA) F-ratio to evaluate the separability of each feature, which measures the degree of the difference among class means [168]. If the F-ratio is very small, then the means of the classes are almost the same and the feature is not helpful for classification; If the F-ratio is large, then at least one class mean is different from others, which means the feature is able to provide useful information for the classifier to distinguish this class from others. The F-ratio for each feature generated after max-pooling layer is calculated by taking the ratio of mean square between classes (MSB) and mean square within classes (MSE) as shown in Equation 4.3.

$$\begin{aligned} F &= \frac{MSB}{MSE} = \frac{SS_{condition}/df_n}{SS_{error}/df_d} \\ &= \frac{\left[\sum_{i=1}^c n_i (M_i - GM)^2 \right] / (c - 1)}{\left[\sum_{i=1}^c \sum_j (x_{ij} - M_i)^2 \right] / (N - c)} \end{aligned} \quad (4.3)$$

where SS is sum of squares, df is degree of freedom, c is the number of classes (conditions), N is the number of all samples, n_i is the number of samples in class i , GM (grand mean) is the mean of all samples, M_i is the mean of class i , and x_{ij} is the j -th sample in class i .

After calculating the F-ratio for each feature, hypothesis testing is performed based on the F-distribution of the F-ratio with degree of freedom $c - 1$ and $N - c$. The null hypothesis H_0 assumes that all class means are equal. We take a significance level of $\alpha = 5\%$ and the critical F-ratio F_α can be found from F-distribution. If $F > F_\alpha$, H_0 can be rejected and the corresponding filter is kept because it indeed provides separability of the class means; if $F \leq F_\alpha$, the corresponding filter is removed.

Clustering of Word Filters

After removing non-discriminative features, there are still many redundant features, which produce same or close values. This will increase the dimensionality of the model and may hurt the classification performance. These redundant features are produced by semantically similar word filters which have close word embeddings. Intuitively, we perform clustering of word filters based on their semantic similarities to solve the feature redundancy problem.

As there are only 10 tri-gram word filters generated, which is a very small number, we only cluster the uni-gram and bi-gram word filters. For bi-gram word filters, we concatenate two word embeddings to form a single vector representation. Then the K-means clustering algorithm is performed to find the clusters of uni-gram word filters and bi-gram word filters separately. Based on these clusters, a further pooling operation is performed on the features after the max-pooling layer of CNN. For features $\{p_{i1}, p_{i2}, \dots, p_{in}\}$ which correspond to the word filters within the i -th cluster, max-pooling or average-pooling is performed in the following way:

$$q_i = \begin{cases} \max \{p_{i1}, p_{i2}, \dots, p_{in}\} & \text{for max-pooling} \\ (\sum_{j=1}^n p_{ij})/n & \text{for average-pooling} \end{cases} \quad (4.4)$$

The max-pooling operation preserves the most significant linguistic clues in the sentence, whereas the average-pooling operation takes into consideration of all the word filters.

As the pooling operation aggregates the features within one cluster into a single feature, the dimensionality of the feature vector is reduced. The number of clusters of uni-gram and bi-gram word filters, h_1 and h_2 , which can be set by users, determines the dimensionality of the feature vector from knowledge-oriented channel, which is equal to $h = h_1 + h_2 + 10$.

4.3.2 Data-oriented Channel

The data-oriented channel of K-CNN uses conventional CNN to learn important features of causal relationship from the training data. Compared with knowledge-oriented channel, it captures longer dependencies in the whole sentence by using longer convolutional window sizes. By allowing the convolutional filters to adjust their weights based on the training data, we give the model enough capacity to learn such longer dependencies and important information that is neglected by the knowledge-oriented channel. Hence, data-oriented channel plays a complementary role to knowledge-oriented channel, and the combination of the two channels allows K-CNN to extract causal relation from sentence effectively.

4.3.2.1 Sentence Representation

To preserve the information in the words besides words between e_1 and e_2 , the input to the data-oriented channel contains all the words in the sentence. In this case, the maximum number of words in the sentence (n_2) can be very large compared with n_1 . However, CNN is not able to capture the position information of words in the sentence, and words that appear far away from e_1 and e_2 may not be informative. In the examples below, CNN is able to capture the cue phrase “result in” and correctly classify the relationship between e_1 and e_2 as causal relation in sentence 1. However, CNN may still classify the relationship between e_1 and e_2 in sentence 2 as causal by mistake once it captures the same cue phrase “result in” without any position information. Hence, position information is needed for CNN to deal with the semantic analysis of long sentences.

1. *The financial $\langle e_1 \rangle$ crisis $\langle /e_1 \rangle$ **resulted in** 12% $\langle e_2 \rangle$ unemployment $\langle /e_2 \rangle$ in this country.*
2. *In spite of great $\langle e_1 \rangle$ effort $\langle /e_1 \rangle$ put by the $\langle e_2 \rangle$ government $\langle /e_2 \rangle$ in providing more employment opportunities for low incomes, the financial crisis still **resulted in** 12% unemployment in this country.*

We adopt the word position embeddings proposed by Zeng et al. [150] to enable the CNN to keep track of how close the words are to e_1 and e_2 . For each word x_i in the input sentence, its relative distances to e_1 and e_2 ranging from $1 - n_2$ and $n_2 - 1$ are calculated first. For example, in sentence 1, the word “resulted” has relative distances of 1 and -3 to e_1 and e_2 respectively. Then the two relative distances are mapped into real-value vectors $\mathbf{p}_i^1, \mathbf{p}_i^2 \in \mathbb{R}^d$ by looking up the position embedding table $\mathbf{W}^{\text{psn}} \in \mathbb{R}^{d \times (2n_2 - 1)}$ which is

initialized randomly, where d is a hyper-parameter indicating the dimensionality of each position embedding, and $(2n_2 - 1)$ is the total number of possible relative distances. Finally, the word embedding of x_i , $\mathbf{w}_i$ is concatenated with $\mathbf{p}_i^1$ and $\mathbf{p}_i^2$ as $[\mathbf{w}_i, \mathbf{p}_i^1, \mathbf{p}_i^2]^T$. As a result, the input $x_D = \{x_1, x_2, \dots, x_{n_2}\}$ is represented as a sequence of real-value vectors $\mathbf{emb}_D = \left\{ [\mathbf{w}_1, \mathbf{p}_1^1, \mathbf{p}_1^2]^T, [\mathbf{w}_2, \mathbf{p}_2^1, \mathbf{p}_2^2]^T, \dots, [\mathbf{w}_{n_2}, \mathbf{p}_{n_2}^1, \mathbf{p}_{n_2}^2]^T \right\}$.

4.3.2.2 Convolution and Pooling Operation

We use conventional CNN in data-oriented channel. The convolutional filters $\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_k]^T$ are initialized randomly and trained through back-propagation, where $\mathbf{f}_i \in \mathbb{R}^{e+2d}$ and k is the convolutional window size. To capture longer dependencies in the sentence instead of causal keywords only, we use wider window sizes ($k = 3, 4$) compared with knowledge-oriented channel.

We simplify the representation of input sentence as $\mathbf{emb}_D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$. The convolution operation in this channel is shown in Equation 4.5.

$$m_i = \tanh\left(\sum_{j=1}^k \mathbf{f}_j^T \mathbf{x}_{i+j-1} + b\right) \quad (4.5)$$

where $\tanh$ is the hyperbolic tangent function, and b is a bias term. The resulting feature map $\mathbf{m} = [m_1, m_2, \dots, m_{n-k+1}]$ is passed to the same max-pooling layer as in knowledge-oriented channel to extract the most significant feature. The number of filters r , which is a hyper-parameter chosen by the user, determines the output dimensionality of data-oriented channel.

4.3.3 Additional Features for Causal Relation Extraction

For causal relations with specific linguistic clues such as “cause”, “because”, and “due to”, the proposed K-CNN model is able to capture such linguistic clues and identify the causal relationship effectively. However, there are many complicated causal relations that are implicitly expressed, for example:

1. The car $\langle e_1 \rangle$ accident $\langle /e_1 \rangle$ **made** she $\langle e_2 \rangle$ disabled $\langle /e_2 \rangle$.
2. The $\langle e_1 \rangle$ company $\langle /e_1 \rangle$ **made** a new $\langle e_2 \rangle$ product $\langle /e_2 \rangle$.

There is causal relationship between e_1 and e_2 in sentence 1, however there is no causal relationship in sentence 2. People understand such causal relationship by performing

mental operations based on their experiences or common senses, such knowledge may not be reflected on text itself, even though word embeddings are able to capture semantic meanings of words, the model requires large amount of data to learn. Therefore, more semantic features from external knowledge resources are needed to identify complicated causal relations. In this section, we propose two types of semantic features.

4.3.3.1 WordNet Categorical Features

Each word sense in WordNet hierarchy has a top level category, for example, there are 26 categories for nouns (such as location, event, state) and 15 categories for verbs (such as change, communication, emotion). Some categories such as “change”, “event”, “state” etc. have higher possibilities to be involved in a causal relationship, and some categories such as “animal”, “plant”, “communication” etc. have lower possibilities. Therefore, we find the WordNet top level categories of the two target entities (e_1 and e_2) and incorporate the categorical information into our model.

In the above example, the WordNet category of “accident” in sentence 1 is “event”, and the WordNet category of “company” in sentence 2 is “group”. By learning from data, the model is able to conclude that “event” category has much higher probability to be involved in causal relationship than “group” category, hence distinguish the two sentences. As majority of the target entities involved in causal relations are nouns and verbs, we only consider noun categories and verb categories of WordNet. The categorical information of e_1 and e_2 is encoded as categorical features using one-hot encoding, and these categorical features are concatenated with the feature vector from K-CNN and then passed to the classifier.

4.3.3.2 FrameNet Causal Scores

Based on the causal frames in FrameNet, we compute causal scores to reflect the extent to which a sentence involves causal relationship. Similar as word filter bank generation in Section 4.3.1.2, we find all the lexical units of causal frames which represent keywords and cue phrases of causal relationship. The difference is that we focus on the ambiguity of causal words, which means the word may indicate other semantic frames other than causal frames. For example, the word “make” invokes 10 semantic frames besides causal frame, such as “Building”, “Cooking_creation” and “Manufacturing” frames. Such ambiguity is captured by calculating the probability of the lexical unit w_i invoking causal frames based on the 170,000 annotated sentence exemplars in FrameNet corpus, as shown in

Equation 4.6.

$$p(c | w_i) = \begin{cases} \|\text{sent}_{cf}\| / \|\text{sent}\| & \text{if } w_i \in \text{causal frames} \\ 0 & \text{if } w_i \notin \text{causal frames} \end{cases} \quad (4.6)$$

where $\|\text{sent}_{cf}\|$ indicates number of causal frame exemplars invoked by w_i , and $\|\text{sent}\|$ indicates total number of exemplars invoked by w_i .

The probability $p(c | w_i)$ is used as the causal score of the lexical unit w_i , and the causal score for a sentence is calculated by adding the causal score of each lexical unit in the sentence.

$$s = \sum_i p(c | w_i) \quad (4.7)$$

Besides the causal score for the whole sentence, we also compute three additional causal scores for three different locations in the sentence respectively, including words before e_1 (including e_1), words after e_2 (including e_2) and words between e_1 and e_2 . The FrameNet causal score features are also concatenated with the feature vector from K-CNN and then passed to the classifier.

4.3.4 Regularization and Classification

The final feature vector $\mathbf{p} \in \mathbb{R}^{h+r+a}$ consists of the knowledge-oriented channel output, data-oriented channel output and additional features, which represents the high-level features of the input sentence extracted by our model. Before passing the feature vector to the classifier to make the final judgment of causality, we apply dropout regularization [169] to prevent co-adaptation of hidden units by randomly setting a proportion ρ of the feature vector to zero, as shown in Equation 4.8.

$$\mathbf{p}_d = \mathbf{p} \circ \mathbf{b} \quad (4.8)$$

where $\circ$ represents element-wise multiplication and $\mathbf{b} \in \mathbb{R}^{r+h+a}$ is a vector of Bernoulli random variables with probability ρ of being 0. The feature vector after dropout $\mathbf{p}_d$ ³ is then fed into a classifier to predict the class label. The classifier consists of a fully connected layer of standard neural network and a softmax layer to predict the class probabilities.

We train the model by minimizing the categorical cross entropy loss function. The free parameters for training include the position embedding matrix, data-oriented channel

³Dropout is only applied during training time, feature vector $\mathbf{p}$ (without dropout) is used at test time.

filter weights, and classifier weights including fully connected layer and softmax layer weights. Training is done using mini-batch stochastic gradient descent (SGD) with the Adadelta update rule [170].

4.4 Experiments

4.4.1 Datasets

Three datasets are used to evaluate our model, which are generated from SemEval-2010 task 8 dataset [171], Causal-TimeBank dataset [172] and Event StoryLine dataset [173].

SemEval-2010 task 8 dataset⁴ contains 10,717 annotated samples, each sample is a sentence annotated with a pair of entities (e_1 and e_2) and their relationship class. Besides *Cause-Effect* relation which is directed⁵, there are additional 8 directed relation classes and one undirected *Other* class. As we only interested in *Cause-Effect* relation, we annotate all other relations as *Other* class, resulting 478 *Cause-Effect*(e_1, e_2) samples, 853 *Cause-Effect*(e_2, e_1) samples and 9386 *Other* samples.

Causal-TimeBank dataset⁶ is a causally annotated dataset based on TempEval-3 TimeBank data. The causal relations and their directions are marked with C-SIGNALs (causal signals) and CLINKs (causal links) tags, we extract all the causal relations between two entities and annotate them in the same way as SemEval-2010 task 8. There are totally 137 *Cause-Effect*(e_1, e_2) samples and 161 *Cause-Effect*(e_2, e_1) samples generated from Causal-TimeBank dataset, and we randomly select 300 annotated entity pairs which are not causally related as *Other* class.

Event StoryLine dataset⁷ is a new benchmark dataset for temporal and causal relation detection, the corpus for annotation contains 258 documents concerning calamity events. We extract all the samples with causal relations by looking into the “CAUSES” and “CAUSED_BY” attributes in the “PLOT_LINK” tag and annotate them in the same format as SemEval-2010 task 8 dataset. There are totally 67 *Cause-Effect*(e_1, e_2) samples and 45 *Cause-Effect*(e_2, e_1) samples generated from Event StoryLine dataset, and we randomly select 220 sample sentences with no causal relationship as *Other* class.

⁴The dataset can be downloaded from https://docs.google.com/document/d/1Q0_CnmvNRnYwNWu1-QCAeR5ToQYkXUqFeAJbdEhsq7w/preview.

⁵*Cause-Effect*(e_1, e_2) means e_1 causes e_2 , *Cause-Effect*(e_2, e_1) means e_2 causes e_1 .

⁶The dataset can be requested for download on <https://hlt-nlp.fbk.eu/technologies/causal-timebank>

⁷The dataset can be downloaded from <https://github.com/cltl/EventStoryLine.git>.

The dataset generated from SemEval-2010 task 8 contains a large amount of samples, majority of causal relations in this dataset are simple causal relations which are explicitly expressed using linguistic clues such as “cause”, “result in”, “trigger”, “produce” etc. Besides, the data is unbalanced and more biased toward the *Other* class, hence it is suitable for evaluating explicit causal relation extraction. The data generated from Causal-TimeBank and Event StoryLine datasets contain more complex causal relations with no specific linguistic clues, which are complicated and hard to identify. Besides, the two datasets are much smaller than SemEval-2010 task 8 dataset, hence model overfitting issue is much more severe on these two datasets.

4.4.2 Experimental Settings

To investigate the performance of our proposed K-CNN model, we chose two conventional CNN models as competitors to make comparison, one is proposed by Zeng et al. [150] which uses filters with a single window size ($k = 3$), we name it *CNN_Single*; the other is proposed by Nguyen and Grishman [151] which uses filters with multiple window sizes ($k = 2, 3, 4, 5$), we name it *CNN_multiple*.

To make fair comparison, all models including ours use Dependency-Based Word Embeddings⁸ that are pre-trained using English Wikipedia, the word embeddings are unit vectors with a dimensionality of $e = 300$, and they are kept static during training; the dimensionality of position embeddings is $d = 20$; the dropout rate is $\rho = 0.4$; the mini-batch size for training is 20; the dimensionality of hidden layer before softmax layer is half of the feature vector dimension. For words not in the word embeddings, we initialize them as random unit vectors with same dimensionality as word embeddings. Besides the above mentioned hyper-parameters, the optimal number of filters for each model is found through grid search (as illustrated in Section 4.4.3.1 below). The evaluation metric is macro-averaged F1 score calculated from 10-fold cross-validation. To produce stable results, the 10-fold cross-validation is run 10 times for each experiment, and the data is shuffled with different random seeds before the cross-validation. We take the average value of the 10 macro-averaged F1 scores as the final result.

We used five variations of K-CNN model to study the effect of our proposed methods, including K-CNN, K-CNN_K, K-CNN_FS, K-CNN_FS+C_{max} and K-CNN_FS+C_{avg}. K-CNN is our base model (without word filter selection and clustering), K-CNN_K is the K-CNN model using knowledge-oriented channel only, FS means word filter selection,

⁸The pre-trained word embedding is available at <https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings>.

FIGURE 4.3: Effect of the number of filters in data-oriented channel. “SemEval”, “Event-SL”, “Causal-TB” indicate SemEval-2010 task 8, Event StoryLine and Causal-TimeBank datasets respectively.

$C_{\max}$ means clustering of word filters with max-pooling within each cluster, and C_{avg} means clustering of word filters with average-pooling within each cluster. For word filter selection process, the critical F-ratio is $F_{\alpha} = 2.9957$ by assuming the degree of freedom for the denominator is $+\infty$. For word filters clustering, we use half of the number of word filters as the number of clusters, hence reducing the dimensionality of the feature vector from knowledge-oriented channel by half.

4.4.3 Results and Analyses

4.4.3.1 Effect of Data-oriented Channel

To study the effect of data-oriented channel and find its optimal number of filters, we perform grid searching of the number of filters ranging from 0 to 100. We use the macro-averaged F1 score calculated from 10-fold cross-validation for evaluation. The grid search results for the three datasets are plotted in Figure 4.3.

From Figure 4.3, we can find the optimal number of filters for each window size in data-oriented channel is between 25 to 30. Further increasing the number of filters will cause the drop of F1 score and the model performance is not stable. This is because that the

Datasets		K-CNN_K	K-CNN
SemEval	P	92.74	93.12
	R	85.56	90.73
	F1	88.85	91.82
Causal-TB	P	71.84	76.91
	R	70.24	75.85
	F1	70.92	76.29
Event-SL	P	77.38	82.07
	R	74.11	80.58
	F1	75.60	81.25

TABLE 4.1: Macro-averaged precision (P), recall (R) and F1 scores of K-CNN without and with data-oriented channel. Scores are shown in percentage (%).

Models	SemEval	Causal-TB	Event-SL
CNN_single	90.15	72.67	75.72
CNN_multiple	90.84	73.37	76.45
K-CNN	91.49	75.83	80.64
K-CNN_FS	91.90	75.96	81.27
K-CNN_FS+Cmax	92.34	76.21	81.84
K-CNN_FS+Cavg	91.20	75.43	81.96

TABLE 4.2: Macro-averaged F1 score (%) of various model architectures for causal relation extraction task on three datasets.

model is prone to overfitting when the number of free parameters is large and the training data is limited. Compared with conventional CNNs which normally use hundreds of convolutional filters, the number of filters needed in the data-oriented channel of K-CNN is significantly reduced.

When the number of filters of data-oriented channel is 0, the model only uses knowledge-oriented channel of K-CNN (K-CNN_K). Table 4.1 shows a comparison of K-CNN_K and K-CNN with optimal number of filters in data-oriented channel. Experimental result shows that the data-oriented channel can improve both precision and recall of the model. The reason is that data-oriented channel is able to extract important features which are neglected by knowledge-oriented channel by learning from data. It involves the whole sentence and able to capture longer dependencies by using convolutional filters with longer window sizes. Therefore, it is necessary to have data-oriented channel in K-CNN which plays a supplementary role to knowledge-oriented channel and gives the model enough capacity to learn important information from data.

4.4.3.2 Effect of Knowledge-oriented Channel and Dimensionality Reduction

Table 4.2 shows the macro-averaged F1 scores of the above mentioned models for causal relation extraction on the three datasets. For conventional CNN models, we found that CNN_multiple performs slightly better than CNN_single, the reason is that CNN_multiple uses multiple convolutional window sizes, which is able to capture more information of

causality from different n-grams. Comparing our proposed K-CNN models with conventional CNN models, the improvements of the experimental results demonstrate that K-CNN models with knowledge-oriented word filters are able to extract causal relations more effectively. We attribute the performance advantage of K-CNN over conventional CNN models to the following three facts:

- The word filters constructed from lexical knowledge bases have more precise representation and broader coverage of the keywords and cue phrases of causal relationship than random initialized convolutional filters, which allow the model to extract the linguistic clues of causal relation from sentence more effectively.
- The combination of knowledge-oriented channel and data-oriented channel enables the model to capture significant features of causal relationship. The knowledge-oriented channel takes advantage of existing knowledge bases and captures significant linguistic clues of causal relationship; the data-oriented channel learns other important features such as features with long distance dependencies from the data. The two channels complement with each other, resulting a good model for causal relation extraction.
- As the word filters are static during training, majority of free parameters of K-CNN lie in data-oriented channel, which are the weights of convolutional filters. The number of free parameters in data-oriented channel depends on the number and the dimensionality of convolutional filters, which is equal to $(e+2d) \times (3r+4r) = (300 + 2 \times 20) \times 7 \times 25 = 59,500$, where r is the optimal number of filters for each window size, which is found to be 25 as shown in Section 4.4.3.1 (we ignore the bias term). Whereas for CNN_single and CNN_multiple, the numbers of free parameters are $(e+2d) \times 3r = 163,200$ and $(e+2d) \times (2r+3r+4r+5r) = 238,000$ respectively, where r is 160 for CNN_single and 50 for CNN_multiple, which are found by grid search. Therefore, the number of free parameters of K-CNN is reduced significantly compared with conventional CNN models. This alleviates overfitting issue when the amount of training data is small, such as the datasets generated from Causal TimeBank and Event StoryLine. That's why the performance improvements for the two small datasets are more significant than SemEval-2010 task 8 dataset which is relatively larger.

To further investigate the effectiveness of K-CNN in alleviating overfitting, for each fold in the 10-fold cross-validation, we train K-CNN as well as conventional CNN (we experiment on CNN_multiple) on the training set and test the models on both training set and validation set. Then we plot the learning curves which show the macro-averaged

FIGURE 4.4: Learning curves of K-CNN and CNN_multiple for SemEval-2010 task 8 (top), Causal-TimeBank (middle) and Event StoryLine (bottom) datasets. Squared dot indicates the highest score achieved on validation set.

F1 scores on training set and validation set with increasing number of training epochs. We observed similar trends of the learning curves for all the 10 folds, hence only one of the 10 folds is shown in Figure 4.4. It is observed that K-CNN consistently outperforms conventional CNN during training and testing stages. With increasing number of training epochs, the scores on training sets can eventually reach 100% for both models, however, the scores on validation sets cannot be further improved after certain training epochs. The gap between training scores and validation scores is actually due to overfitting of the models on training data. As shown in Figure 4.4, K-CNN achieves higher scores on validation sets compared with conventional CNN, this demonstrates the ability of K-CNN in alleviating overfitting issue. Besides, training is done much faster for K-CNN (within few epochs) compared with conventional CNN, and the difference of training efficiency is significant for the first epoch. The reason is that conventional CNN needs to train all the parameters from scratch, whereas only the data-oriented channel in K-CNN needs to be trained and the knowledge-oriented channel has already contained prior knowledge for extracting causal relationship before training.

Furthermore, the experimental results in Table 4.2 also show that the proposed word filter selection and clustering for dimensionality reduction can improve the performance of K-CNN. K-CNN with word filter selection (K-CNN_FS) using ANOVA F-ratio hypothesis testing performs better than K-CNN, this is because that the non-discriminative features produced by the insignificant word filters are considered as noises and will hinder the classification accuracy. K-CNN_FS removes such word filters and only keeps the word filters which produce discriminative features with large separability among class means. Further clustering of the features based on the similarities of word filters also improves the performance of K-CNN, as the redundant features produced by semantically similar word filters are removed and the feature dimensionality is reduced, resulting in a more compact and higher quality feature vector. The experimental results show that max-pooling within clusters ($C_{\max}$) generally outperforms average-pooling (C_{avg}), and the classification results of max-pooling are more stable. The reason is probably that the significant clues of causality are preserved by max-pooling, however they are affected by the outliers in the cluster for average-pooling.

4.4.3.3 Effect of Additional Semantic Features

We add the semantic features proposed in Section 4.3.3 to the best K-CNN model, K-CNN_FS+ $C_{\max}$, then the macro-averaged precision, recall and F1 score are evaluated. Table 4.3 shows the comparison results before and after adding the semantic features. Results show that the proposed semantic features are able to improve the F1 score for causal

Datasets		K-CNN w/o SF	K-CNN with SF
SemEval	P	93.37	94.61
	R	91.40	90.87
	F1	92.34	92.64
Causal-TB	P	76.94	78.71
	R	75.70	77.26
	F1	76.21	77.86
Event-SL	P	82.78	84.57
	R	81.03	82.15
	F1	81.84	83.31

TABLE 4.3: Macro-averaged precision (P), recall (R) and F1 scores of K-CNN without and with semantic features (SF). Scores are shown in percentage (%).

relation extraction. Furthermore, the performance improvements for Causal-TimeBank and Event StoryLine datasets are more significant than SemEval-2010 task 8 dataset, indicating that the proposed semantic features are especially useful for complex and implicitly expressed causal relations, as there are more such causal relations in Causal-TimeBank and Event StoryLine datasets compared with SemEval-2010 task 8 dataset. We observed that both precision and recall are improved for the two datasets by adding the semantic features. We attribute the performance improvements to the ability of the proposed semantic features for finding the useful semantic information hiding behind the text for causal relation extraction.

4.5 Conclusion

In this chapter, we propose a Knowledge-oriented Convolutional Neural Network (K-CNN) for causal relation extraction. We present an effective way of combining human prior knowledge from lexical knowledge bases and information from data for CNN to achieve better performance.

The proposed K-CNN contains two convolutional channels: knowledge-oriented channel and data-oriented channel. The convolutional filters (word filters) of knowledge-oriented channel are automatically generated based on the linguistic knowledge of causal relationship in lexical knowledge bases including WordNet and FrameNet. These word filters represent significant linguistic clues of causal relationship, allowing the model to extract these linguistic clues precisely and effectively. The convolutional filters in data-oriented channel are trained based on the training data, and the channel uses position embeddings and wider convolutional window size to capture longer dependencies and other useful features from the whole sentence.

The proposed K-CNN is fully automatic without sophisticated feature engineering. In addition, the model addresses the overfitting issue caused by the lack of training data by

reducing the number of free parameters of the model, yet achieves better performance for causal relation extraction compared with conventional CNNs. Word filter selection and clustering techniques are proposed to reduce dimensionality and further improve the performance of K-CNN model. We also propose additional semantic features to address more complex causal relations. The performance of our approaches has been experimentally verified using three datasets for causal relation extraction.

Chapter 5

Incorporate Knowledge into Transformer

5.1 Introduction

In this chapter, we are dealing with the relation extraction task with multiple relations besides causal relationship. Relation extraction aims to detect the semantic relationship between two entities in a sentence. For example, given the sentence: “*James Dobson* has resigned as chairman of *Focus On The Family*, which he founded thirty years ago.”, the goal is to recognize the organization-founder relation held between “*Focus On The Family*” and “*James Dobson*”. The various relations between entities extracted from large-scale unstructured texts can be used for ontology and knowledge base population [174, 175], as well as facilitating downstream tasks that requires relational understanding of texts such as question answering [176] and dialogue systems [177].

As mentioned in Chapter 4, traditional feature-based approaches require extensive feature engineering [158, 178, 179]. Deep neural networks such as CNNs and RNNs have the ability of exploring more complex semantics and extracting features automatically from raw texts for relation extraction tasks [162, 180, 181]. Recently, attention mechanisms have been introduced to deep neural networks to improve their performance [9, 53, 182]. Especially, the Transformer proposed by Vaswani et al. [8] is based solely on self-attention and has demonstrated better performance than traditional RNNs [11, 183]. However, deep neural networks normally require sufficient labeled data to train their numerous model parameters. The scarcity or low quality of training data will limit the model’s ability to recognize complex relations and also cause overfitting.

Our previous study on K-CNN described in Chapter 4 shows that incorporating prior knowledge from external lexical resources into deep neural network can reduce the reliance on training data and improve the performance of causal relation extraction. Motivated by this, we propose a novel knowledge-attention mechanism, which transforms texts from word semantic space into relational semantic space by attending to relation indicators that are useful in recognizing different relations. The relation indicators are automatically generated from lexical knowledge bases which represent keywords and cue phrases of different relation expressions. While the existing self-attention encoder learns internal semantic features by attending to the input texts themselves, the proposed knowledge-attention encoder captures the linguistic clues of different relations based on external knowledge. Since the two attention mechanisms complement each other, we integrate them into a single model to maximize the utilization of both knowledge and data, and achieve optimal performance for relation extraction.

In summary, the main contributions of this chapter are:

- * We propose a knowledge-attention encoder, a novel attention mechanism which incorporates prior knowledge from external lexical resources to effectively capture the informative linguistic clues for relation extraction.
- * To take the advantages of both knowledge-attention and self-attention, we propose three integration strategies: multi-channel attention, softmax interpolation, and knowledge-informed self-attention. Our final models are fully attention-based and can be easily set up for end-to-end training.
- * We present detailed analysis on the knowledge-attention encoder. Results show that it has complementary strengths with self-attention encoder, and the integrated models achieve start-of-the-art results for relation extraction.

This chapter is organized as follows: Section 5.2 reviews deep learning methods for relation extraction in the literature; Section 5.3 introduces the proposed knowledge-attention mechanism in detail; Section 5.5 presents the experimental results on relation extraction task, as well as detailed analyses on knowledge-attention encoder; Section 5.6 gives a brief summary of this chapter.

5.2 Related Work

We focus here on deep neural networks for relation extraction since they have demonstrated better performance than traditional feature-based and kernel-based approaches.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are the earliest and commonly used approaches for relation extraction. Zeng et al. [150] showed that CNN with position embeddings is effective for relation extraction. Similarly, CNN with multiple filter sizes [151], pairwise ranking loss function [184] and auxiliary embeddings [162] were proposed to improve performance. Zhang and Wang [185] proposed bi-directional RNN with max pooling to model the sequential relations. Instead of modeling the whole sentence, performing RNN on sub-dependency trees (e.g. shortest dependency path between two entities) has demonstrated to be effective in capturing long-distance relation patterns [180, 186]. Zhang et al. [10] proposed graph convolution over dependency trees and achieved state-of-the-art results on TACRED dataset.

Recently, attention mechanisms have been widely applied to CNNs [53, 187] and RNNs [9, 182, 188]. The improved performance demonstrated the effectiveness of attention mechanisms in deep neural networks. Particularly, Vaswani et al. [8] proposed a solely self-attention-based model called Transformer, which is more effective than RNNs in capturing long-distance features since it is able to draw global dependencies without regard to their distances in the sequences. Bilan and Roth [11] first applied self-attention encoder to relation extraction task and achieved competitive results on TACRED dataset. Verga et al. [183] used self-attention to encode long contexts spanning multiple sentences for biological relation extraction. However, more attention heads and layers are required for self-attention encoder to capture complex semantic and syntactic information since learning is solely based on training data. Hence, more high quality training data and computational power are needed. Our work utilizes the knowledge from external lexical resources to improve deep neural network’s ability of capturing informative linguistic clues.

External knowledge has shown to be effective in neural networks for many NLP tasks. Existing works focus on utilizing external knowledge to improve embedding representations [110, 189, 190], CNNs [101, 191, 192], and RNNs [193–196]. Our work is the first to incorporate knowledge into Transformer through a novel knowledge-attention mechanism to improve its performance on relation extraction task.

5.3 Knowledge-attention Encoder

We present the proposed knowledge-attention encoder in this section. Relation indicators are first generated from external lexical resources (Section 5.3.1); Then the input texts are transformed from word semantic space into relational semantic space by attending

FIGURE 5.1: Knowledge-attention process (left) and multi-head structure (right) of knowledge-attention encoder.

to the relation indicators using knowledge-attention mechanism (Section 5.3.2); Finally, position-aware attention is used to summarize the input sequence by taking both relation semantics and relative positions into consideration (Section 5.3.3).

5.3.1 Relation Indicators Generation

Relation indicators represent the keywords or cue phrases of various relation types, which are essential for knowledge-attention encoder to capture the linguistic clues of certain relation from texts. We utilize two publicly available lexical resources including FrameNet¹ and Thesaurus.com² to find such lexical units.

FrameNet is a large lexical knowledge base which categorizes English words and sentences into higher level semantic frames [38]. Each frame is a conceptual structure describing a type of event, object or relation. FrameNet contains over 1200 semantic frames, many of which represent various semantic relations. For each relation type in our relation extraction task, we first find all the relevant semantic frames by searching from FrameNet (refer Appendix B for detailed semantic frames used). Then we extract all the lexical units involved in these frames, which are exactly the keywords or phrases that often used to express such relation. Thesaurus.com is the largest online thesaurus which has over

¹<https://framenet.icsi.berkeley.edu/fndrupal>

²<https://www.thesaurus.com>

3 million synonyms and antonyms. It also has the flexibility to filter search results by relevance, POS tag, word length, and complexity. To broaden the coverage of relation indicators, we utilize the synonyms in Thesaurus.com to extend the lexical units extracted from FrameNet. To reduce noise, only the most relevant synonyms with the same POS tag are selected.

Relation indicators are generated based on the word embeddings and POS tags of lexical units. Formally, given a word in a lexical unit, we find its word embedding $\mathbf{w}_i \in \mathbb{R}^{d_w}$ and POS embedding $\mathbf{t}_i \in \mathbb{R}^{d_t}$ by looking up the word embedding matrix $\mathbf{W}^{wrd} \in \mathbb{R}^{d_w \times V^{wrd}}$ and POS embedding matrix $\mathbf{W}^{pos} \in \mathbb{R}^{d_t \times V^{pos}}$ respectively, where d_w and d_t are the dimensions of word and POS embeddings, V^{wrd} is vocabulary size³ and V^{pos} is total number of POS tags. The corresponding relation indicator is formed by concatenating word embedding and POS embedding, $\mathbf{k}_i = [\mathbf{w}_i, \mathbf{t}_i]$. If a lexical unit contains multiple words (i.e. phrase), the corresponding relation indicator is formed by averaging the embeddings of all words. Eventually, around 3000 relation indicators (including 2000 synonyms) are generated: $\mathbf{K} = \{\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_m\}$.

5.3.2 Knowledge Attention

5.3.2.1 Knowledge-attention process

In a typical attention mechanism, a query (q) is compared with the keys (K) in a set of key-value pairs and the corresponding attention weights are calculated. The attention output is weighted sum of values (V) using the attention weights. In our proposed knowledge-attention encoder, the queries are input texts and the key-value pairs are both relation indicators. The detailed process of knowledge-attention is shown in Figure 5.1 (left).

Formally, given text input $x = \{x_1, x_2, \dots, x_n\}$, the input embeddings $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$ are generated by concatenating each word's word embedding and POS embedding in the same way as relation indicator generation in Section 5.3.1. The hidden representations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}$ are obtained by attending to the knowledge indicators $\mathbf{K}$, as shown in Equation 5.1. The final knowledge-attention outputs are obtained by subtracting the hidden representations with the relation indicators mean, as shown in Equation 5.2.

$$\mathbf{H} = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (5.1)$$

³Same word embedding matrix is used for relation indicators and input texts, hence the vocabulary also includes all the words in the training corpus.

$$knwl(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{H} - \sum \mathbf{K}/m \quad (5.2)$$

where *knwl* indicates knowledge-attention process, m is the number of relation indicators, and d_k is dimension of key/query vectors which is a scaling factor same as in Vaswani et al. [8].

The subtraction of the mean of the relation indicators will result in small outputs for irrelevant words. More importantly, the output feature vector will be close to the related relation indicators and further apart from other relation indicators in relational semantic space. Therefore, the proposed knowledge-attention mechanism is effective in capturing the linguistic clues of relations represented by relation indicators in the relational semantic space.

5.3.2.2 Multi-head knowledge-attention

Inspired by the multi-head attention in Transformer [8], we also have multi-head knowledge-attention which first linearly transforms $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ h times, and then performs h knowledge-attentions simultaneously, as shown in Figure 5.1 (right).

Different from the Transformer encoder, we use the same linear transformation for $\mathbf{Q}$ and $\mathbf{K}$ in each head to keep the correspondence between queries and keys.

$$head_i = knwl(\mathbf{QW}_i^Q, \mathbf{KW}_i^Q, \mathbf{VW}_i^V) \quad (5.3)$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^V \in \mathbb{R}^{d_k \times (d_k/h)}$ and $i \in [1, 2, \dots, h]$. Besides, only one residual connection from input embeddings to outputs of position-wise feed forward networks is used. We also mask the outputs of padding tokens using zero vectors.

The multi-head structure in knowledge-attention allows the model to jointly attend inputs to different relational semantic subspaces with different contributions of relation indicators. This is beneficial in recognizing complex relations where various compositions of relation indicators are needed.

5.3.3 Position-aware Attention

It has been proven that the relative position information of each token with respective to the two target entities is beneficial for relation extraction task [150]. We modify the position-aware attention originally proposed by Zhang et al. [9] to incorporate such

relative position information and find the importance of each token to the final sentence representation.

Assume the relative position of token x_i to target entity is $\hat{p}_i$. We apply position binning function (Equation 5.4) to make it easier for the model to distinguish long and short relative distances.

$$p_i = \begin{cases} \hat{p}_i & |\hat{p}_i| \leq 2 \\ \frac{\hat{p}_i}{|\hat{p}_i|} \lceil \log_2 |\hat{p}_i| + 1 \rceil & |\hat{p}_i| > 2 \end{cases} \quad (5.4)$$

After getting the relative positions p_i^s and p_i^o to the two entities of interest (subject and object respectively), we map them to position embeddings based on a shared position embedding matrix $\mathbf{W}^p$. The two embeddings are concatenated to form the final position embedding for token x_i : $\mathbf{p}_i = [\mathbf{p}_i^s, \mathbf{p}_i^o]$.

Position-aware attention is performed on the outputs of knowledge-attention $\mathbf{O} \in \mathbb{R}^{n \times d_k}$, taking the corresponding relative position embeddings $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ into consideration:

$$\mathbf{f} = \mathbf{O}^T \text{softmax}(\tanh(\mathbf{O}\mathbf{W}_o + \mathbf{P}\mathbf{W}_p)\mathbf{c}) \quad (5.5)$$

where $\mathbf{W}_o \in \mathbb{R}^{d_k \times d_a}$, $\mathbf{W}_p \in \mathbb{R}^{d_p \times d_a}$, d_a is attention dimension, and $\mathbf{c} \in \mathbb{R}^{d_a}$ is a context vector learned by the neural network.

5.4 Integrate Knowledge-attention with Self-attention

The self-attention encoder proposed by Vaswani et al. [8] learns internal semantic features by modeling pair-wise interactions within the texts themselves, which is effective in capturing long-distance dependencies. Our proposed knowledge-attention encoder has complementary strengths of capturing the linguistic clues of relations precisely based on external knowledge. Therefore, it is beneficial to integrate the two models to maximize the utilization of both external knowledge and training data. In this section, we propose three integration approaches as shown in Figure 5.2, and each approach has its own advantages.

FIGURE 5.2: Three ways of integrating knowledge-attention with self-attention: multi-channel attention and softmax interpolation (top), as well as knowledge-informed self-attention (bottom).

5.4.1 Multi-channel Attention

In this approach, self-attention and knowledge-attention are treated as two separate channels to model sentence from different perspectives. After applying position-aware attention, two feature vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ are obtained from self-attention and knowledge-attention respectively. We apply another attention mechanism called multi-channel attention to integrate the feature vectors.

In multi-channel attention, feature vectors are first fed into a fully connected neural network to get their hidden representations $\mathbf{h}_i$. Then attention weights are calculated using a learnable context vector $\mathbf{c}$, which reflects the importance of each feature vector to final relation classification. Finally, the feature vectors are integrated based on attention weights, as shown in Equation 5.6.

$$\mathbf{r} = \sum_i softmax(\mathbf{h}_i^T \mathbf{c}) \mathbf{h}_i \quad (5.6)$$

After obtaining the integrated feature vector $\mathbf{r}$, we pass it to a softmax classifier to determine the relation class. The model is trained using stochastic gradient descent with momentum and learning rate decay to minimize the cross-entropy loss.

The main advantage of this approach is flexibility. Since the two channels process information independently, the input components are not necessary to be the same. Besides, we can add more features from other sources (e.g. subject and object categories) to multi-channel attention to make final decision based on all the information sources.

5.4.2 Softmax Interpolation

Similar as multi-channel attention, we also use two independent channels for self-attention and knowledge-attention in softmax interpolation. Instead of integrating the feature vectors, we make two independent predictions using two softmax classifiers based on the feature vectors from the two channels. The loss function is defined as total cross-entropy loss of the two classifiers. The final prediction is obtained using an interpolation function of the two softmax distributions:

$$\mathbf{p} = \beta \cdot \mathbf{p}_1 + (1 - \beta) \cdot \mathbf{p}_2 \quad (5.7)$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the softmax distributions obtained from self-attention and knowledge-attention respectively, and β is the priority weight assigned to self-attention.

Since knowledge-attention focuses on capturing the keywords and cue phrases of relations, the precision will be higher than self-attention while the recall is lower. The proposed softmax interpolation approach is able to take the advantages of both attention mechanisms and balance the precision and recall by adjusting the priority weight β .

5.4.3 Knowledge-informed Self-attention

Since knowledge-attention and self-attention share similar structures, it is also possible to integrate them into a single channel. We propose knowledge-informed self-attention encoder which incorporates knowledge-attention into every self-attention head to jointly model the semantic relations based on both knowledge and data.

The structure of knowledge-informed self-attention is shown in Figure 5.3. Formally, given texts input matrix $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$ and knowledge indicators $\mathbf{K} \in \mathbb{R}^{m \times d_k}$, the output of each attention head is calculated as follows:

$$head_i = knwl(\mathbf{QW}_i^Q, \mathbf{KW}_i^Q, \mathbf{KW}_i^V) + self(\mathbf{QW}_i^{Q_s}, \mathbf{QW}_i^{K_s}, \mathbf{QW}_i^{V_s}) \quad (5.8)$$

where *knwl* and *self* indicate knowledge-attention and self-attention respectively, and all the linear transformation weight matrices have the dimensionality of $\mathbf{W} \in \mathbb{R}^{d_k \times (d_k/h)}$.

Since each self-attention head is aided with prior knowledge in knowledge-attention, the knowledge-informed self-attention encoder is able to capture more lexical and semantic information than single attention encoder.

FIGURE 5.3: Knowledge-informed self-attention structure. Q , K represent input matrix and knowledge indicators respectively, h is the number of attention heads.

5.5 Experiments

5.5.1 Baseline Models

To study the performance of our proposed models, the following baseline models are used for comparison:

CNN-based models including: (1) CNN: the classical convolutional neural network for sentence classification [26]. (2) CNN-PE: CNN with position embeddings dedicated for relation classification [151]. (3) GCN: a graph convolutional network over the pruned dependency trees of the sentence [10].

RNN-based models including: (1) LSTM: long short-term memory network to sequentially model the texts. Classification is based on the last hidden output. (2) PA-LSTM: Similar position-aware attention mechanism as our work is used to summarize the LSTM outputs [9].

CNN-RNN hybrid model including contextualized GCN (C-GCN) where the input vectors are obtained using bi-directional LSTM network [10].

Self-attention-based model (Self-attn) which uses self-attention encoder to model the input sentence. Our implementation is based on Bilan and Roth [11] where several modifications are made on the original Transformer encoder, including the use of relative positional encodings instead of absolute sinusoidal encodings, as well as other configurations such as residual connection, activation function and normalization.

For our model, we evaluate both the proposed knowledge-attention encoder (Knwl-attn) as well as the integrated models with self-attention including multi-channel attention (MCA), softmax interpolation (SI) and knowledge-informed self-attention (KISA).

5.5.2 Experimental Settings

We conduct our main experiments on TACRED, a large-scale relation extraction dataset introduced by Zhang et al. [9]. TACRED contains over 106k sentences with hand-annotated subject and object entities as well as the relations between them. It is a very complex relation extraction dataset with 41 relation types and a *no_relation* class when no relation is held between entities. The dataset is suited for real-word relation extraction since it is unbalanced with 79.5% *no_relation* samples, and multiple relations between different entity pairs can exist in one sentence. Besides, the samples are normally long sentences with an average of 36.2 words.

Since the dataset is already partitioned into train (68124 samples), dev (22631 samples) and test (15509 samples) sets, we tune model hyper-parameters using dev set and evaluate model using test set. The evaluation metrics are micro-averaged precision, recall and F_1 score. For fair comparison, we select the model with median F_1 score on dev set from 5 independent runs, same as Zhang et al. [9]. The same “entity mask” strategy is used to replace subject (or object) entity with special $\langle \text{NER} \rangle$ -SUBJ (or $\langle \text{NER} \rangle$ -OBJ) tokens to avoid overfitting on specific entities and provide entity type information.

Besides TACRED, another dataset called SemEval2010-Task8 [171] is used to evaluate the generalization ability of our proposed model. The dataset is significantly smaller and simpler than TACRED, which has 8000 training samples and 2717 testing samples. It contains 9 directed relations and 1 other relation (19 relation classes in total). We use the official macro-averaged F_1 score as evaluation metric.

We use one layer encoder with 6 attention heads for both knowledge-attention and self-attention since further increasing the number of layers and attention heads will degrade the performance. For softmax interpolation, we choose $\beta = 0.8$ to balance precision and recall. Word embeddings are fine-tuned based on pre-trained GloVe [24] with dimensionality of 300. Dropout [197] is used during training to alleviate overfitting. Other model hyper-parameters and training details are described in Appendix C for reproduction purpose.

Model	P	R	F ₁
CNN [†]	72.1	50.3	59.2
CNN-PE [†]	68.2	55.4	61.1
GCN [‡]	69.8	59.0	64.0
LSTM [†]	61.4	61.7	61.5
PA-LSTM [†]	65.7	64.5	65.1
C-GCN [‡]	69.9	63.3	66.4
Self-attn ^{††}	64.6	68.6	66.5
Knwl-attn	70.0	63.1	66.4
Knwl+Self (MCA)	68.4	66.1	67.3*
Knwl+Self (SI)	67.1	68.4	67.8*
Know+Self (KISA)	69.4	66.0	67.7*

TABLE 5.1: Micro-averaged precision (P), recall (R) and F₁ score on TACRED dataset. †, ‡ and †† mark the results reported in [9], [10] and [11] respectively. * marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.

5.5.3 Results and Analyses

5.5.3.1 Results on TACRED dataset

Table 5.1 shows the results of baseline as well as our proposed models on TACRED dataset. It is observed that our proposed knowledge-attention encoder outperforms all CNN-based and RNN-based models by at least 1.3 F₁. Meanwhile, it achieves comparable results with C-GCN and self-attention encoder, which are the current state-of-the-art single-model systems.

Comparing with self-attention encoder, it is observed that knowledge-attention encoder results in higher precision but lower recall. This is reasonable since knowledge-attention encoder focuses on capturing the significant linguistic clues of relations based on external knowledge, it will result in high precision for the predicted relations similar to rule-based systems. Self-attention encoder is able to capture more long-distance dependency features by learning from data, resulting in better recall. By integrating self-attention and knowledge-attention using the proposed approaches, a more balanced precision and recall can be obtained, suggesting the complementary effects of self-attention and knowledge-attention mechanisms. The integrated models improve performance by at least 0.9 F₁ score and achieve new state-of-the-art results among all the single end-to-end models.

Comparing the three integrated models, softmax interpolation (SI) achieves the best performance. More interestingly, we found that the precision and recall can be controlled by adjusting the priority weight β . Figure 5.4 shows impact of β on precision, recall and

FIGURE 5.4: Change of precision, recall and F_1 score on dev set as the priority weight β in softmax interpolation changes.

F_1 score. As β increases, precision decreases and recall increases. Therefore, we can choose a small β for relation extraction system which requires high precision, and a large β for the system requiring better recall. F_1 score reaches the highest value when precision and recall are balanced ($\beta = 0.8$).

Knowledge-informed self-attention (KISA) has comparable performance with softmax interpolation, and without the need of hyper-parameter tuning since knowledge-attention and self-attention are integrated into a single channel. The performance gain over self-attention encoder is 1.2 F_1 with much improved precision, demonstrating the effectiveness of incorporating knowledge-attention into self-attention to jointly model the sentence based on both knowledge and data.

Performance gain is the lowest for multi-channel attention (MCA). However, the model is more flexible in the way that features from other information sources can be easily added to the model to further improve its performance. Table 5.2 shows the results of adding NER embeddings of each token to self-attention channel, and entity (subject and object) categorical embeddings to multi-channel attention as additional feature vectors. We use dimensionality of 30 and 60 for NER and entity categorical embeddings respectively, and the two embedding matrices are learned by the neural network. Results show that adding NER and entity categorical information to MCA integrated model improves F_1 score by

Model	P	R	F ₁
MCA	68.4	66.1	67.3
+ NER	68.7	66.2	67.5
+ Entity category	70.0	65.8	67.8
+ Both	70.1	66.0	67.9

TABLE 5.2: Results of adding NER embeddings and entity categorical embeddings to the multi-channel attention (MCA) integrated model.

Model	mask entity	keep entity
Self-attn	76.8	83.1
Knwl-attn	76.1	82.3
Knwl+Self (MCA)	77.4*	84.0*
Knwl+Self (SI)	78.0*	84.3*
Know+Self (KISA)	77.5*	84.0*

TABLE 5.3: Macro-averaged F₁ score on SemEval2010-Task8 dataset. * marks statistically significant improvements over Self-attn with $p < 0.01$ under one-tailed t-test.

0.2 and 0.5 respectively, and adding both improves precision significantly, resulting a new best F₁ score.

5.5.3.2 Results on SemEval2010-Task8 dataset

We use SemEval2010-Task8 dataset to evaluate the generalization ability of our proposed model. Experiments are conducted in two manners: mask or keep the entities of interest. Results in Table 5.3 show that the “entity mask” strategy degrades the performance, indicating that there exist strong correlations between entities of interest and relation classes in SemEval2010-Task8 dataset. Although the results of keeping the entities are better, the model tends to remember these entities instead of focusing on learning the linguistic clues of relations. This will result in bad generalization for sentences with unseen entities.

Regardless of whether the entity mask is used, by incorporating knowledge-attention mechanism, our model improves the performance of self-attention by a statistically significant margin, especially the softmax interpolation integrated model. The results on SemEval2010-Task8 are consistent with that of TACRED, demonstrating the effectiveness and robustness of our proposed method.

Model	Dev F ₁
Knwl-attn Encoder	66.5
1. – Multi-head structure	64.6
2. – Synonym relation indicators	64.7
3. – Relation indicators mean	65.0
4. – Output masking	65.8
5. – Entity masking	65.4
6. – Relative positions	63.0

TABLE 5.4: Ablation study on knowledge-attention encoder. Results are the median F₁ scores of 5 independent runs on dev set of TACRED.

5.5.3.3 Ablation study

To study the contributions of specific components of knowledge-attention encoder, we perform ablation experiments on the dev set of TACRED. The results of knowledge-attention encoder with and without certain components are shown in Table 5.4.

It is observed that: (1) The proposed multi-head knowledge-attention structure outperforms single-head significantly. This demonstrates the effectiveness of jointly attending texts to different relational semantic subspaces in the multi-head structure. (2) The synonyms improve the performance of knowledge-attention since they are able to broaden the coverage of relation indicators and form a robust relational semantic space. (3) The subtraction of relation indicators mean vector from attention hidden representations helps suppress the activation of irrelevant words and results in a better representation for each word to capture the linguistic clues of relations. (4) The two masking strategies are helpful for our model: the output masking eliminates the effects of the padding tokens and the entity masking avoids entity overfitting while providing entity type information. (5) The relative position embedding term in position-aware attention contributes a significant amount of F₁ score. This shows that positional information is particularly important for relation extraction task.

5.5.3.4 Attention visualization

To verify the complementary effects of knowledge-attention encoder and self-attention encoder, we compare the attention weights assigned to words from the two encoders. Table 5.5 presents the attention visualization results on sample sentences. For each sample sentence, attention weights from knowledge-attention encoder are visualized first, followed by self-attention encoder. It is observed that knowledge-attention encoder focuses more on the specific keywords or cue phrases of certain relations, such as “graduated”,

Sample Sentences	True Relation	Predict
SUBJ-PERSON graduated in 1992 from the OBJ-ORGANIZATION OBJ-ORGANIZATION OBJ-ORGANIZATION with a degree in computer science and had worked as a systems analyst at a Pittsburgh law firm since 1999 .	per:schools _attended	correct
SUBJ-PERSON graduated in 1992 from the OBJ-ORGANIZATION OBJ-ORGANIZATION OBJ-ORGANIZATION with a degree in computer science and had worked as a systems analyst at a Pittsburgh law firm since 1999 .		correct
OBJ-PERSON OBJ-PERSON , a public affairs and government relations strategist , was executive director of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION Policy Institute from 2005 to 2010 .	org:top_memb- ers/employees	correct
OBJ-PERSON OBJ-PERSON , a public affairs and government relations strategist , was executive director of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION Policy Institute from 2005 to 2010 .		wrong
Founded in 1992 in Schaumburg , Illinois , the SUBJ-ORGANIZATION is one of the largest Chinese - American associations of professionals in the OBJ-COUNTRY OBJ-COUNTRY .	org:country_of _headquarters	wrong
Founded in 1992 in Schaumburg , Illinois , the SUBJ-ORGANIZATION is one of the largest Chinese - American associations of professionals in the OBJ-COUNTRY OBJ-COUNTRY .		correct

TABLE 5.5: Attention visualization for knowledge-attention encoder (first) and self-attention encoder (second). Words are highlighted based on the attention weights assigned to them. Best viewed in color.

“executive director” and “founded”; while self-attention encoder attends to a wide range of words in the sentence and pays more attention to the surrounding words of target entities especially the words indicating the syntactic structure, such as “is”, “in” and “of”. Therefore, knowledge-attention encoder and self-attention encoder have complementary strengths that focus on different perspectives for relation extraction.

5.5.3.5 Error analysis

To investigate the limitations of our proposed model and provide insights for future research, we analyze the errors produced by the system on the test set of TACRED. For knowledge-attention encoder, 58% errors are false negatives (FNs) due to the limited ability in capturing long-distance dependencies and some unseen linguistic clues during training. For our integrated model⁴ that takes the benefits of both self-attention and knowledge-attention, FN is reduced by 10%. However, false positive (FP) is not improved due to overfitting that leads to wrong predictions. Many errors are caused by multiple entities with different relations co-occurred in one sentence. Our model may mistake irrelevant entities as a relation pair. We also observed that many FP errors are due to the confusions between related relations such as “city_of_death” and “city_of_residence”. More data or knowledge is needed to distinguish “death” and “residence”. Besides, some errors are caused by imperfect annotations.

⁴We observed similar error behaviors of the three proposed integrated models.

5.6 Conclusion

We introduce knowledge-attention encoder which effectively incorporates prior knowledge from external lexical resources for relation extraction. The proposed knowledge-attention mechanism is able to pay attention to the relation indicators constructed from external knowledge. It transforms texts from word space into relational semantic space and captures the informative linguistic clues of relations effectively. Furthermore, we show the complementary strengths of knowledge-attention and self-attention, and propose three different ways of integrating them to maximize the utilization of both knowledge and data. The proposed models are fully attention-based end-to-end systems and achieve state-of-the-art results on TACRED dataset, outperforming existing CNN, RNN, and self-attention based models.

Chapter 6

Incorporate Knowledge into Attention Mechanism

6.1 Introduction

Nowadays, people use Internet extensively to read or post articles, reviews and comments expressing opinions towards certain products, events or topics. Understanding the sentiment of customers is crucial for businesses and organizations to review their products, policies or business strategies. The overwhelming of such text data on the Internet has led to the revolution of automatically analyzing and extracting the sentiment from text. Sentiment analysis is a NLP technique that automatically identifies and categorizes opinions expressed in a piece of text, especially to determine its sentiment polarity (e.g. positive, negative, or neutral).

Early works on sentiment analysis focus on lexicon and rule-based approaches [198–201], which rely on a large set of handcrafted rules and extensive human effort to construct and maintain the rules. Traditional machine learning approaches classify sentiment polarity based on rich features (lexical, syntactic or semantic features) using classifiers such as Naive Bayes and SVM [202–205]. However, it requires sophisticated feature engineering and the performance strongly depends on the quality of the designed features. Recently, deep learning approaches become more and more popular since it is able to extract meaningful features automatically and more effectively by leveraging deep neural networks.

The commonly adopted deep neural network architectures for sentiment analysis include CNN, RNN, and Transformer, each architecture has its own advantages and limitations.

CNN is a special feed-forward neural network with convolutional layers interleaved with pooling layers. In NLP, 1d convolution is normally performed to compute the semantic relevance between the n-grams in text and convolutional filters/kernels learned by the network. With the help of pooling operations, the important n-grams that appear in the text can be captured. Hence, CNN is able to capture local features effectively and induce more abstract and informative representation for sentiment analysis [26, 52, 206]. However, conventional CNN is relatively weak in capturing sequential dependencies since the pooling operation ignores the position information. RNN is well-known for processing sequences of inputs while updating its internal states recurrently. Intuitively, it is well suited for sentiment analysis since text data is also a type of sequential data [62, 207, 208]. However, RNN suffers from gradient vanishing and parallel-unfriendly problems due to its recurrent nature. Many works attempt to alleviate the gradient vanishing problem by incorporating attention mechanisms to RNN [7, 68, 182]. The recently proposed Transformer [8] addresses the problems of RNN by relying entirely on self-attention for text representation. It has shown to be more effective than RNN for sentiment analysis [74, 75, 209] and many other NLP tasks [77, 210]. However, Transformer has limited ability in capturing local n-gram features which are crucial for sentiment analysis since self-attention focuses on global instead of local context. Besides, the heavy architecture of Transformer often requires more training data, computational power and memory, especially for long texts.

To address the limitations of above-mentioned neural architectures, we propose a multi-window Convolutional Transformer (ConvTransformer) which takes the advantages of both CNN and Transformer for efficient text representation. ConvTransformer has a simplified multi-head multi-layer structure compared with Transformer. In each head, instead of self-attention, convolution operation over the text represented in different subspaces is performed to better capture local n-gram features. To effectively capture the n-grams with various lengths, we use multiple convolution window sizes in different convolutional heads, allowing different heads to focus on different n-gram features. We also discard the pooling operation in order to preserve sequential information in the output of ConvTransformer. Moreover, we propose a sentiment-aware attention mechanism to summarize the sequential output of ConvTransformer and obtain the global text representation by incorporating external knowledge of words' sentiment from SentiWordNet. The sentiment intensity information used in sentiment-aware attention is beneficial for sentiment analysis and also reduces model's reliance on training data to learn everything from scratch. Meanwhile, position information is also considered when calculating attention weights.

The contributions of this chapter are summarized as follows:

- * We propose a novel neural architecture called Convolutional Transformer (ConvTransformer), which is able to capture n-grams features effectively while preserving sequential information.
- * We demonstrate an effective way of incorporating external knowledge into deep neural networks with attention mechanisms. We propose a sentiment-aware attention mechanism to incorporate sentiment intensity information of words from external knowledge base into the attention mechanism, which is able to take word's sentiment information into consideration when calculating attention weights.
- * Comprehensive experiments and analyses show that ConvTransformer outperforms CNN, RNN and attention-based baseline models, and the proposed sentiment-aware attention mechanism is able to improve the performance for sentiment analysis.

This chapter is organized as follows: Section 6.2 reviews the related work of sentiment analysis using deep learning approaches; Section 6.3 introduces the proposed ConvTransformer and sentiment-aware attention mechanism in detail; Section 6.4 presents experimental results on sentiment analysis task as well as detailed analyses on the proposed ConvTransformer and sentiment-aware attention; Section 6.5 gives a summarization of this chapter.

6.2 Related Work

We focus here on deep learning approaches for sentiment analysis since they have demonstrated better performance than rule-based and traditional machine learning approaches.

Convolutional Neural Network (CNN) is originally applied in the field of computer vision for image feature extraction [49]. With the development of distributed representations of words in NLP [2], many CNN-based models using word embeddings as input were proposed to learn the representation of texts for sentiment analysis. Kim [26] first used CNN for sentence classification. Dynamic CNN (DCNN) was also proposed for sentence modeling which uses a dynamic K-Max pooling as a non-linear sub-sampling function [27]. In [52], character-level CNN was proposed which uses character embeddings as input instead of word embeddings. In [206], Character to Sentence CNN (CharSCNN) was proposed which jointly uses character-level, word-level and sentence-level representations for sentiment analysis. Deep CNNs consisting of many convolutional and pooling layers also achieved good performance on sentiment analysis tasks [55, 56].

Besides CNN-based models, Recurrent Neural Networks (RNNs) are also widely used for sentiment analysis. RNNs are suitable for modeling sequential inputs like texts since the network is conditioned on all previously seen inputs and propagating the effects of words over the sentence recurrently. Practically, the improved variants of RNN such as Long Short-Term Memory Neural Network (LSTM) [62, 207, 208], Gated Recurrent Neural Network (GRU) [30], bi-directional LSTM (BiLSTM) [211] and Disconnected Recurrent Neural Network (DRNN) [212] are used for sentiment analysis to alleviate the long distance gradient vanishing problem. Some works combined CNN and RNN as hybrid models to utilize the benefits of different neural architectures [213, 214].

To solve the gradient vanishing problem and better capture long-distance dependencies, attention mechanisms are widely used in deep neural networks [215]. Many works focused on improving CNN and RNN with attention mechanisms. In [7], hierarchical attention network (HAN) was proposed for document-level sentiment analysis, which uses two levels of attention mechanisms to find the key structural information of the document. In [68], attention-based LSTM was proposed for aspect-level sentiment analysis, in which attention mechanism was used to concentrate on different parts of the sentence when different aspects are taken as input. Reference [69] introduced a matrix sentence embedding representation using BiLSTM with self-attention mechanism. In [70], three different attentions including attention vector, LSTM attention and attentive pooling were integrated with CNN model.

Recently, a novel neural architecture called Transformer was proposed which is based solely on the self-attention mechanism for text representation [8]. Since self-attention is able to draw global dependencies between input and output without regard to their distances in the sequence, Transformer is more suitable for capturing long-distance dependencies and allows more parallelization compared with RNN. The emerging of Transformer has led to a series of breakthroughs in a wide range of NLP tasks including sentiment analysis [74, 75, 209]. Especially, the pre-trained language models based on Transformer have achieved state-of-the-art performance on many benchmark datasets [3, 80].

Deep learning approaches normally require large amount of training data to achieve good performance. Data scarcity or low quality of training data will lead to bad generalization. Many works focused on improving text representation by incorporating external knowledge into deep neural networks [101, 110, 210]. For sentiment analysis, many emotional words and sentiment terms are critical to determine the whole sentiment of a sentence. Many works utilized such sentiment knowledge from external knowledge bases

such as SentiWordNet [100, 216], SenticNet [217] and ConceptNet [77] to improve the performance of sentiment analysis.

Different from existing approaches, our model combines CNN and Transformer to take advantage of both neural architectures for efficient text representation. We also incorporate external knowledge into our model in a novel and effective way to better capture words' sentiment intensity information.

FIGURE 6.1: Overall architecture of the proposed sentiment analysis system.

6.3 ConvTransformer with Sentiment-aware Attention

In this section, we present our methodologies in detail. The overall architecture of our model is shown in Fig. 6.1. The input text is encoded using the proposed Convolutional Transformer (ConvTransformer) to capture local n-gram features (Section 6.3.1); Meanwhile, sentiment embedding is generated for each word in the input text utilizing external knowledge base (Section 6.3.2); Then, sentiment-aware attention is used to summarize the output of ConvTransformer and obtain the final representation by incorporating sentiment intensity information and position information of each token (Section 6.3.3); Finally, sentiment polarity of the text is classified based on the final representation (Section 6.3.4).

6.3.1 Convolutional Transformer Encoder

ConvTransformer is the main component of our sentiment analysis system, which encodes text using multiple convolutional and linear operations. It is able to capture important local n-gram features for sentiment analysis while preserving sequential information. The architecture of the proposed ConvTransformer is shown in Fig. 6.2.

FIGURE 6.2: Convolutional Transformer (ConvTransformer).

6.3.1.1 N-gram Convolution Operation

Convolution is the fundamental operation of ConvTransformer which computes the semantic relevance between n-grams in the text and trainable convolutional filters. Specifically, given the input text $t = [t_1, t_2, \dots, t_l]$, each word t_i is represented as d_w -dimensional word embedding $\mathbf{x}_i$ by looking up the word embedding matrix $\mathbf{W}^{wrd} \in \mathbb{R}^{d_w \times V}$, where V is vocabulary size. Then, n-gram convolution over input embeddings $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_l]^T$ is performed using convolutional filters $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m]^T$, where $\mathbf{f}_i \in \mathbb{R}^{nd_w}$ and n is the convolution window size. Consequently, a feature map $\mathbf{M} \in \mathbb{R}^{l \times m}$ is generated as follows:

$$\mathbf{M} = \mathbf{X} \circledast \mathbf{F}^T \quad (6.1)$$

where $\circledast$ indicates the convolution operation of $\mathbf{f}_i$ over $\mathbf{X}$. Specifically, the value in the feature map is calculated as shown in Equation 6.2:

$$m_{ji} = f(\mathbf{f}_i^T \cdot \text{Cat}(x_j, x_{j+1}, \dots, x_{j+n-1}) + b) \quad (6.2)$$

where Cat means concatenation, f is a non-linear activation function and b is a bias term.

6.3.1.2 Multi-head Multi-layer Structure

Inspired by the structure of Transformer [8], ConvTransformer also has a multi-head multi-layer structure that allows the model to extract features in both parallel and sequential manners. As shown in Fig. 6.2, for an h -head ConvTransformer, the input embedding $\mathbf{X}$ is first transformed into h sub-spaces using different linear transforms. Then, different n -gram convolution operations are performed in different sub-spaces using multiple convolution window sizes. Finally, the outputs from all the convolutional heads are concatenated together and linearly transformed to the original input dimension, as shown in Equation 6.3.

$$\begin{aligned} \text{ConvTransformer}(\mathbf{X}) &= \text{Cat}(\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_h) \mathbf{W}^M \\ \text{where } \mathbf{M}_i &= \text{Conv}(\mathbf{X} \mathbf{W}_i^X) \end{aligned} \quad (6.3)$$

Here, Cat indicates column-wise concatenation, Conv indicates n -gram convolution operation, $\mathbf{W}_i^X \in \mathbb{R}^{d_w \times (d_w/h)}$ and $\mathbf{W}^M \in \mathbb{R}^{hm \times d_w}$ are the weight matrices of linear transformations. Moreover, we adopt the residual connection and layer norm as used in [8].

Compared with conventional CNN where the convolution operation is performed on words directly, ConvTransformer performs convolution in different sub-spaces of words simultaneously, where the words in different sub-spaces may represent different meanings. Such multi-head structure allows our model to capture more semantic or syntactic features from n -grams and gives our model more capabilities in dealing with complex NLP tasks like sentiment analysis. Besides, various convolution window sizes are used in different convolutional heads, allowing ConvTransformer to capture n -grams with various lengths effectively while each head focuses on specific n -grams. We also discard the pooling operation that is normally used in conventional CNNs, making ConvTransformer a sequence-to-sequence model. Therefore, the sequential information of texts is preserved.

Compared with Transformer which uses self-attention as its fundamental operation, our ConvTransformer uses convolution that focuses more on local instead of global context. This allows our model to capture local n -gram features more effectively than Transformer. These n -gram features are important keywords or phrases that are crucial for sentiment analysis. Besides, ConvTransformer has more simplified structure, hence it is more memory and computational efficient than Transformer, especially for long texts.

6.3.2 Sentiment Embedding Generation

In this section, we introduce our methodology of utilizing the external knowledge about words' sentiment and generating sentiment embedding that can be incorporated into our model effectively.

SentiWordNet¹ is a publicly available and frequently used lexical knowledge base for sentiment analysis and opinion mining [218, 219]. It is constructed based on WordNet [37], a large lexical database that groups English words into synonym sets (synsets) to represent different meanings or concepts. SentiWordNet assigns each synset a positivity score and a negativity score ranging from 0 to 1 to reflect its sentiment intensity. Table 6.1 shows some examples of SentiWordNet lexicons. Such knowledge of words' sentiment intensity can be utilized as prior knowledge in our model to improve sentiment analysis performance.

Specifically, for each word in the vocabulary, we search SentiWordNet for its positivity and negativity scores and then transform the scores into integers ranging from 1 to 10 to reflect its sentiment intensity, as shown in Equation 6.4.

$$I = \begin{cases} \lfloor 10 \times score \rfloor + 1 & \text{if } score \neq 1 \\ 10 & \text{if } score = 1 \end{cases} \quad (6.4)$$

where *score* represents positivity or negativity score ranging from 0 to 1. For unknown word that is not in SentiWordNet, we set both its positivity and negativity scores into 0 which means neutral or objective word.

Given the input text $t = [t_1, t_2, \dots, t_l]$, we are able to find the positive sentiment intensity I_i^p and negative sentiment intensity I_i^n of each word t_i . Then we transform the sentiment intensities into sentiment embeddings $\mathbf{s}_i^p$ and $\mathbf{s}_i^n$ by constructing two sentiment embedding matrices $\mathbf{W}^{pos}, \mathbf{W}^{neg} \in \mathbb{R}^{d_s \times 11}$, one for positive and the other for negative sentiment embedding. Here, d_s is the dimension of sentiment embedding, 11 means ten embeddings for the ten sentiment intensity levels and one embedding for padding tokens². The final sentiment embedding for t_i is obtained by concatenating the positive and negative sentiment embeddings: $\mathbf{s}_i = \text{Cat}(\mathbf{s}_i^p, \mathbf{s}_i^n)$.

¹Available for download from <https://github.com/aesuli/sentiwordnet>.

²We use fixed zero vector as the sentiment embedding of padding tokens.

Word	(POS, ID)	<i>PosScore</i>	<i>NegScore</i>
excellent	(a, 02343110)	1	0
miserable	(a, 01150205)	0	0.875
gentle	(a, 01455412)	0.375	0.25
employer	(n, 10054657)	0	0

TABLE 6.1: Example words and their sentiment intensities obtained from SentiWordNet. The (POS,ID) pair uniquely identifies a WordNet synset; *PosScore* and *NegScore* represent positivity score and negativity score respectively.

6.3.3 Sentiment-aware Attention

To effectively incorporate the sentiment intensity information of sentiment embedding and obtain the final text representation, we propose a sentiment-aware attention mechanism that summarizes the output of ConvTransformer while taking both sentiment and position information into consideration. As shown in Fig. 6.3, the attention weight of each token α_i is calculated based on its sentiment embedding $\mathbf{s}_i$ and position embedding $\mathbf{p}_i$ besides the ConvTransformer output $\mathbf{o}_i$. The position embedding $\mathbf{p}_i$ is obtained by mapping the token's absolute position to d_p -dimensional embeddings based on a trainable position embedding matrix $\mathbf{W}^p \in \mathbb{R}^{d_p \times P}$, where P is the total number of positions.

Formally, the attention weight α_i is calculated as follows:

$$u_i = \mathbf{c}^T \tanh(\mathbf{W}_o \mathbf{o}_i + \mathbf{W}_s \mathbf{s}_i + \mathbf{W}_p \mathbf{p}_i) \quad (6.5)$$

$$\alpha_i = \frac{\exp(u_i)}{\sum_{j=1}^l \exp(u_j)} \quad (6.6)$$

where $\mathbf{W}_o \in \mathbb{R}^{d_a \times d_w}$, $\mathbf{W}_s \in \mathbb{R}^{d_a \times 2d_s}$, $\mathbf{W}_p \in \mathbb{R}^{d_a \times d_p}$, d_a is attention dimension, and $\mathbf{c} \in \mathbb{R}^{d_a}$ is a context vector learned by the neural network. The attention weight α_i reflects the contribution of each token to the final text representation, and the final representation is computed as shown in Equation 6.7.

$$\mathbf{f} = \sum_{i=1}^l \alpha_i \mathbf{o}_i \quad (6.7)$$

The sentiment embedding used in our sentiment-aware attention mechanism contains prior knowledge of words' sentiment intensity, it is beneficial for sentiment analysis and also reduces the reliance on training data; The position embedding used in sentiment-aware attention provides positional information of each token, which is important for

FIGURE 6.3: Sentiment-aware attention mechanism. Attention weights α_i are calculated based on ConvTransformer output o_i , sentiment embedding s_i and position embedding p_i of each token.

sentiment analysis since words appear in different positions may have different affective meanings.

6.3.4 Classification and Optimization

After obtaining the final text representation $\mathbf{f}$, we pass it to a classifier to compute the probability for each sentiment class. The classifier consists of a fully connected layer of a standard neural network for dimension reduction and a softmax layer (fully connected layer with softmax activation function) for class probabilities calculation. For optimization, we train our model by minimizing categorical cross entropy loss and center loss [220] using mini-batch stochastic gradient descent (SGD) with momentum. We also apply dropout regularization [197] before feeding $\mathbf{f}$ into the classifier to prevent co-adaptation of hidden units.

Dataset	# of classes	Average length	Train samples	Test samples
Yelp P.	2	156	560k	38k
Yelp F.	5	158	650k	50k
IMDB	2	292	25k	25k

TABLE 6.2: Statistics of the three sentiment analysis datasets used in our experiments.

6.4 Experiments

6.4.1 Datasets

We use three commonly studied sentiment analysis datasets to evaluate our model, including Yelp Review Polarity (Yelp P.), Yelp Review Full (Yelp F.), and IMDB Movie Review. These datasets are diverse in the aspect of number of sentiment classes, average text length, and number of samples. The statistics of the three datasets are shown in Table 6.2.

Yelp P. and Yelp F. datasets are constructed from Yelp Dataset Challenge 2015 by Zhang et al. [52], which contain crowd-sourced reviews about businesses. Yelp P. is a binary sentiment classification dataset whose sentiment polarity is either positive or negative; Yelp F. contains more fine-grained sentiment classes ranging from rating 1 to rating 5. IMDB dataset is constructed from IMDB movie reviews by Maas et al. [221]. It is a binary sentiment classification dataset with even number of positive and negative reviews. Compared with Yelp reviews, IMDB movie reviews are much more longer, normally in document-level.

6.4.2 Experimental Settings

6.4.2.1 Baseline models

To study the performance of our proposed model, the following models are used as baseline models for comparison:

CNN-based models including conventional Word-level CNN and Char-level CNN [52], as well as a very deep CNN namely VDCNN for text classification [55].

RNN-based models including standard LSTM, discriminative LSTM (D-LSTM) implemented by Yogatama et al. [207], and Skim-LSTM which updates the hidden states dynamically [222].

Attentive models including label-embedding attentive model (LEAM) [223], Bi-directional LSTM with self-attention (BiLSTM+Self-attention) for sentence embedding representation [69], and Transformer encoder [8] for text classification.

For our proposed ConvTransformer, we also study its performance without external knowledge, i.e. without sentiment embedding in the sentiment-aware attention mechanism.

6.4.2.2 Hyper-parameters settings

We set aside 10% of training data as development set to tune model hyper-parameters and adjust learning rates. The evaluation metric is classification accuracy and we report the average accuracy on test set based on 5 independent trainings of the model.

The word embedding matrix $\mathbf{W}^{word}$ is constructed using 300-dimensional Glove word embeddings [24] and it is fixed in the model. The fully connected layer in the classifier has a dimensionality of 100. Dropout rate is set to 0.4. ReLU is used for all nonlinear activation functions except sentiment-aware attention which uses tanh. The attention dimension in sentiment-aware attention is set to 200, and the dimensions of sentiment embedding and position embedding are $d_s = 20$ and $d_p = 60$ respectively. For our proposed ConvTransformer, we use 3 layers with 6 convolutional heads in each layer and $m = 100$ convolutional filters are used in each head. We use three convolution window sizes $n = 2, 3, 4$ in ConvTransformer, each is applied in two heads. For Transformer encoder, we use the default implementation as in [8] with the same number of heads and layers as ConvTransformer.

For training, the initial learning rate is 0.01 with a momentum of 0.9, and it is decayed with a rate of 0.9 after 10 epochs if the classification accuracy on development set does not improve. The learning rate and weight for center loss are 0.1 and 0.001 respectively. Batch size is set to 100 and we train the model for 70 epochs.

6.4.3 Results and Analyses

Table 6.3 shows the test accuracy of our model as well as baseline models on the three sentiment analysis datasets. Results show that our model achieves the best performance

Model	Yelp P.	Yelp F.	IMDB
Word-level CNN [52]	95.40	59.84	89.08*
Char-level CNN [52]	94.75	61.6	–
VDCNN [55]	95.72	64.72	–
LSTM	95.37*	64.01*	89.14*
D-LSTM [207]	92.60	59.60	–
Skim-LSTM [222]	–	–	91.20
LEAM [223]	95.31	64.09	–
BiLSTM+Self-attention [69]	96.08*	65.79*	89.56*
Transformer Encoder [8]	96.13*	65.34*	88.05*
ConvTransformer (w/o sentiment embed)	97.14	67.60	92.00
ConvTransformer (with sentiment embed)	97.46	68.39	92.91

TABLE 6.3: Classification accuracy (%) on three sentiment analysis datasets. * means the results are obtained from our implementation. – means not reported. All other results are directly cited from the respective papers.

on all the three datasets, outperforming CNN-based, RNN-based and attentive baseline models.

For CNN-based models, results show that Character-level CNN is more effective for fine-grained sentiment analysis than binary classification, and deep CNN demonstrates performance improvement over shallow CNNs. Our proposed ConvTransformer (without sentiment embedding) shows significant improvement over conventional CNNs including deep CNN. This is due to the benefits of the multi-head structure with multi-window convolution that allows our model to capture more semantic and syntactic features from different n-grams, as well as the sequential output with attention mechanism that preserves important position information for final representation.

RNN-based models are very sensitive to parameter settings. Our implementation of LSTM performs better than D-LSTM. Skim-LSTM shows good performance on document-level sentiment analysis (IMDB dataset), probably due to its special treatment of unimportant inputs that alleviates gradient vanishing problem. The proposed ConvTransformer uses convolution operation that does not suffer from gradient vanishing problem.

Attentive models especially self-attention-based models outperform RNN-based models significantly for short texts (Yelp P. and Yelp F.). However, the performance degrades for long texts (IMDB). This is because that the model is affected by irrelevant words and not able to focus on important keywords since self-attention mechanism focuses on global context. Our proposed ConvTransformer focuses more on local context using n-gram convolution and is able to capture important keywords and phrases more effectively.

Model	Dev Acc (%)
ConvTransformer	93.11
1. – Multi-head	92.05
2. – Multi-layer	91.74
3. – Multiple window size	92.32
4. – Sentiment embed.	92.07
5. – Position embed.	91.94

TABLE 6.4: Ablation study on ConvTransformer. Classification accuracy is reported on the development set of IMDB.

Results also show that incorporating external knowledge of words’ sentiment in sentiment-aware attention mechanism further improves the performance of our model on all the datasets. This demonstrates the benefits of utilizing external knowledge as well as the effectiveness of the proposed sentiment-aware attention mechanism.

6.4.4 Ablation Study

To study the contributions of specific parts of our model, we perform ablation study on the development set of IMDB by comparing the performance of our model with and without the specific part. Experimental results are shown in Table 6.4.

- (1) We remove the multi-head structure by using only one head in each ConvTransformer layer. Result shows that the performance degrades by 1.14%, demonstrating the effectiveness of jointly performing n-gram convolutions in different sub-word spaces.
- (2) We remove the multi-layer structure by using single ConvTransformer layer. The performance degrading of 1.47% demonstrates that it is beneficial to use multi-layer ConvTransformer for text encoding. The upper layer of ConvTransformer is able to widen the convolution context and induce more abstract and discriminative representations of texts.
- (3) In all the convolutional heads, we use a single convolution window size of 3 instead of multiple window sizes, the performance drops by 0.85%. This proves the effectiveness of using different convolutional heads to focus on n-grams with different lengths.
- (4) After removing the sentiment embedding in sentiment-aware attention, the performance drop by 1.23%. This demonstrates the benefits of incorporating words’ sentiment knowledge from external knowledge base into the model using the proposed sentiment embedding construction and sentiment-aware attention methods.

(5) After removing the position embedding in sentiment-aware attention, the performance drop by 1.25%. This demonstrates the importance of preserving and capturing the positional or sequential information of texts.

6.4.5 Attention Visualization

#	Sample Sentences	Sentiment
1	Make sure you make this delightful comedy part of your holiday season ! If you admire Dennis Morgan or Barbara Stanwyck , this film is a fun one to watch . They really work well together as you would see in this movie . The whole cast was very entertaining . Since I 'm a Dennis Morgan fan , this film was a real treat ! But ... everyone can enjoy it ! Recommended !	positive
2	This movie really has nothing going for it . With the Reverend played by Phillip Seymour Hoffman complaining about his constipation and other toilet humor in a 2.5 hour movie , you know that they made no cuts at all and left the crap in , literally . It 's a waste of good talent , and a total embarrassment . Dreadful !	negative
3	In this Silly Symphony , a mouse from the country visits his cousin in the city . Most of the short is the two mice exploring the dinner table . The animation is fine , where this short suffers is in a lack of humor . Perhaps I 've just seen this `` dinner table adventure '' in one too many Tom and Jerry shorts . Even though this came first , I just did not find it that enjoyable .	negative

TABLE 6.5: Visualization of attention weights for sentiment-aware attention. Words are highlighted based on the attention weights assigned to them. Best viewed in color.

To study what our model focuses on, we conduct attention weights visualization for sentiment-aware attention. Table 6.5 shows the visualization results of three sample sentences from the test set of IMDB.

It is observed that our model pays more attention to words and phrases with high sentiment intensity, either positive sentiment such as “delightful”, “entertaining” and “Recommended” in the first sample sentence, or negative sentiment such as “crap”, “waste” and “embarrassment” in the second sample sentence. Besides, our model is able to capture the negation of a sentiment word effectively. For example, the negation of word “enjoyable” in the third sample sentence. Hence, the overall sentiment polarity of the text can be capture correctly.

6.5 Conclusion

We propose a novel neural architecture called ConvTransformer that combines the advantages of CNN and Transformer for text representation. ConvTransformer has a multi-head structure that jointly performs different n-gram convolutions in different sub-spaces.

It is more effective in capturing local n-gram features compared with CNN and Transformer and also able to preserve the sequential information of texts. Moreover, we incorporate external knowledge of words' sentiment intensity into our model by utilizing an external knowledge base, SentiWordNet. We propose a sentiment-aware attention mechanism that takes each word's sentiment intensity and position information into consideration while obtaining the final text representation. The proposed sentiment-aware attention mechanism provides a general framework of effectively incorporating external knowledge into attention mechanisms. Extensive experiments and analyses show that our model is effective for sentiment analysis, outperforming conventional CNN, RNN and attention-based models.

Chapter 7

Attentive Convolutional Transformer

7.1 Introduction

In Chapter 6, we have discussed the advantages and limitations of three commonly adopted neural network architectures for NLP, including CNN, RNN, and Transformer. To address the limitations of existing neural architectures, we proposed ConvTransformer in Chapter 6 which combines the convolution operation of CNN and the structure of Transformer. Although ConvTransformer preserves the sequential information of text by discarding the pooling operation of CNN and focusing more on local features, the global dependencies of text may not be well-captured. Besides, the convolutional filters learned by traditional convolution operation may be redundant since they do not contribute to the final representation directly.

Based on ConvTransformer, this chapter proposes an Attentive Convolutional Transformer (ACT) that further improves the performance of ConvTransformer. Similar as ConvTransformer, ACT also has a multi-head structure that jointly performs convolution operations in different semantic sub-spaces. However, instead of conventional convolution operation, a novel attentive convolution mechanism is performed in each attention head to better capture local n-gram features. Different from conventional convolution, the proposed attentive convolution utilizes the semantic meaning of convolutional filters attentively and transforms texts from complex word space to a more informative convolutional filter space. This not only simplifies the optimization of capturing important n-grams for classification, but also allows our model to learn meaningful convolutional

filters since all the filters contribute to the final representation directly. Compared with the self-attention in Transformer, the proposed attentive convolution focuses more on learning important n-gram features globally which are invariant to the specific inputs. These n-gram features are exactly the keywords and phrases that are crucial for text classification. While majority of existing works augment Transformer with conventional CNNs to improve locality modeling capability with the cost of introducing more parameters [224–227], our work is a more lightweight approach and it is the first to utilize the semantic meaning of convolutional filters with attention mechanism.

The proposed ACT is also sequence-to-sequence, with an additional global representation output by keeping the max-pooling functionality of CNN. Therefore, it is able to capture both local and global features while preserving sequential information. Furthermore, we propose a global attention mechanism to summarize the outputs of ACT and obtain the final representation by taking local representation, global representation, and position information into consideration. Experiments are conducted on six datasets covering sentiment analysis, topic categorization, and relation extraction tasks. We present detailed analyses on ACT, results show that ACT is a lightweight and effective universal text classifier, outperforming existing CNN, RNN, and attentive models including Transformer.

This chapter is organized as follows: Section 7.2 introduces the proposed ACT in detail; Section 7.3 shows experimental results on various text classification tasks, and Section 7.4 presents more in-depth analyses on the proposed ACT; finally, Section 7.5 gives a brief summary of this chapter.

7.2 Methodology

In this section, we present the proposed ACT in detail. The attentive convolution mechanism in ACT is introduced in Section 7.2.1; the multi-head multi-layer structure of ACT is described in Section 7.2.2; the final text representation for classification is obtained through a global attention mechanism, which is presented in Section 7.2.3.

7.2.1 Attentive Convolution Mechanism

Attentive convolution mechanism is the fundamental operation of ACT. It first performs n-gram convolution over text, then transforms text into convolutional filter space by combining the filters attentively. With different utilization of feature maps as attention

FIGURE 7.1: Left: attentive convolution mechanism. Outputs are obtained by combining convolutional filters attentively utilizing feature map as attention weights. Right: multi-head multi-layer structure of ACT. h and N indicate number of attentive convolution heads and layers respectively.

weights, attentive convolution mechanism is able to capture both local and global features of texts. The architecture of the proposed attentive convolution is shown in Figure 7.1 (left).

7.2.1.1 Local feature representation

Given a text input $t = [t_1, t_2, \dots, t_l]$, we first represent each word token t_i as word embedding $\mathbf{q}_i \in \mathbb{R}^{d_w}$ and obtain the input embeddings $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l]$ by looking up the word embedding matrix $\mathbf{W}^{wrd} \in \mathbb{R}^{d_w \times V}$, where d_w is the dimension of word embedding and V is vocabulary size. Then, n-gram convolution over input embeddings $\mathbf{Q}$ is performed using convolutional filters $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m]$, where $\mathbf{f}_i \in \mathbb{R}^{n \times d_w}$ and n is the convolution kernel size. A feature map matrix $\mathbf{M} \in \mathbb{R}^{m \times l}$ is generated as follows:

$$\mathbf{M} = \mathbf{Q} \circledast \mathbf{F} \quad (7.1)$$

where $\circledast$ indicates the convolution operation of $\mathbf{f}_i$ over $\mathbf{Q}$. Specifically, the value in the feature map is calculated as shown in Equation 7.2:

$$m_{ij} = f(\mathbf{f}_i^T \cdot \text{Cat}(q_j, q_{j+1}, \dots, q_{j+n-1}) + b) \quad (7.2)$$

where Cat means concatenation, f is a non-linear activation function and b is a bias term.

The values in the resulted feature map indicate semantic relevance between n-grams and convolutional filters. By treating the feature map values as attention weights and aggregating the semantic convolutional filters attentively, we transform each n-gram from complex word space to a more informative convolutional filter space while preserving the sequential information of texts. Formally, the attentive convolution for local feature representation is shown in Equation 7.3.

$$\mathbf{O} = \mathbf{F} \cdot \mathbf{M} = \mathbf{F} \cdot (\mathbf{Q} \circledast \mathbf{F}) \quad (7.3)$$

where $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_l] \in \mathbb{R}^{nd_w \times l}$ is the output obtained from attentive convolution.

Different from self-attention whose output space is still complex word space with varying components depending on the input, the output space in our proposed attentive convolution mechanism is formed by n-gram convolutional filters which are learned globally and invariant to the inputs. In such space, important n-grams will be close to the corresponding filters and irrelevant n-grams will have small values. Therefore, the important n-grams appear in texts can be captured effectively.

7.2.1.2 Global feature representation

Besides local features, attentive convolution mechanism can also capture global features of texts by applying max-pooling technique which is normally used in conventional CNNs. The max-pooling over each row of the feature map $\mathbf{M}$ finds the overall relevance of the texts to each convolutional filter. By aggregating the convolutional filters attentively using the max-pooling results, we can find the overall semantics of texts in the filter space. Formally, the attentive convolution for global feature representation is shown in Equation 7.4.

$$\mathbf{g} = \mathbf{F} \cdot \max(\mathbf{M}) \quad (7.4)$$

where $\mathbf{g} \in \mathbb{R}^{nd_w}$ and $\max$ means row-wise max-pooling.

7.2.1.3 Comparison with existing methods

Compared with conventional CNN whose outputs come from feature maps only, our proposed attentive convolution utilizes both feature maps and semantic meaning of convolutional filters for text representation. This allows our model to learn meaningful

convolutional filters effectively since all the filters contribute to the final representation directly. Moreover, the pooling operation in conventional CNN ignores the sequential information of texts, whereas the local feature representation in our method preserves the sequential information while capturing important n-gram features.

Compared with conventional attention mechanism whose attention weights are calculated from vector product of queries ($\mathbf{Q}$) and keys ($\mathbf{K}$), our proposed method calculates attention weights through convolution of queries ($\mathbf{Q}$) using the keys ($\mathbf{F}$), where the keys and values in our attention mechanism are convolutional filters learned during end-to-end training. The convolution operation involves wider context (n-grams) than vector product of single words, this allows our model to capture important n-gram features more effectively in attention mechanism. Besides, as mentioned in Section 7.2.1.1, the output space is more simplified and informative since it is formed by convolutional filters that are invariant to the inputs.

7.2.2 Multi-head Multi-layer Attentive Convolution

Inspired by Transformer [8], the proposed ACT also has multi-head and multi-layer structures as shown in Figure 7.1 (right).

For h -head ACT, we first linearly transform input embeddings $\mathbf{Q}$ h times and perform h attentive convolution simultaneously. Then the outputs from different attention heads are concatenated together and linearly transformed to the original input dimension, as shown in Equation 7.5.

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}) &= \mathbf{W}^O \text{Cat}(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_h) \\ \text{where } \mathbf{O}_i &= \text{AttenConv}(\mathbf{W}_i^Q \mathbf{Q}) \end{aligned} \quad (7.5)$$

Here, AttenConv indicates the proposed attentive convolution mechanism, $\mathbf{W}_i^Q \in \mathbb{R}^{(d_w/h) \times d_w}$ and $\mathbf{W}^O \in \mathbb{R}^{d_w \times nd_w}$ are the weight matrices of linear transformations. Furthermore, we adopt the residual connection and layer norm as used in Vaswani et al. [8]. For multi-layer ACT, we simply pass the local representations of lower-layer to the input of upper-layer to obtain the higher-level local representations. The global representation is obtained from the top ACT layer.

The multi-head structure of ACT allows our model to jointly capture important n-gram features in different sub-word spaces, where the n-grams in different spaces have different contributions to the final representation; The multi-layer structure allows our model to

capture higher level semantics effectively. Since upper-layer involves wider context for convolution, it is able to induce more abstract and discriminative representations.

FIGURE 7.2: Global attention mechanism. Attention weights α_i are calculated based on local representation $\mathbf{o}_i$, global representation $\mathbf{g}$, and position embedding $\mathbf{p}_i$ of each token.

7.2.3 Global Attention and Classification

To obtain the final representation of texts for classification, we propose a global attention mechanism which summarizes the sequential outputs of ACT. As shown in Figure 7.2, the attention weights are calculated by taking both local and global representations as well as position information of each token into consideration.

The local representation $\mathbf{O} \in \mathbb{R}^{d_w \times l}$ and global representation $\mathbf{g} \in \mathbb{R}^{d_w}$ are obtained from the top-layer of ACT. The position embedding $\mathbf{P} \in \mathbb{R}^{d_p \times l}$ is obtained by mapping each token's absolute position to d_p -dimensional embeddings based on a trainable position embedding matrix $\mathbf{W}^p \in \mathbb{R}^{d_p \times P}$, where P is the total number of positions. The final representation is obtained by Equation 7.6.

$$\mathbf{f} = \mathbf{O} \cdot \text{Softmax}(f(\mathbf{W}_o \mathbf{O} + \mathbf{W}_p \mathbf{P})^T \mathbf{c} + \frac{\mathbf{O}^T \mathbf{g}}{\sqrt{d_w}}) \quad (7.6)$$

where f is a non-linear activation function, $\mathbf{W}_o \in \mathbb{R}^{d_a \times d_w}$ $\mathbf{W}_p \in \mathbb{R}^{d_a \times d_p}$ are linear transformation weight matrices, d_a is attention dimension, $\mathbf{c} \in \mathbb{R}^{d_a}$ is a context vector learned by the neural network, $\sqrt{d_w}$ is a scaling factor depends on input dimension.

For classification, we pass the final representation to a classifier consisting of a fully connected layer and a softmax layer to predict class probabilities. Our model is trained by minimizing categorical cross entropy loss and center loss [220] using stochastic gradient descent (SGD) with momentum and learning rate decay.

7.3 Experiments

We evaluate our proposed ACT on three different text classification tasks, including sentiment analysis, topic categorization and relation extraction. Since relation extraction is slightly different from traditional text classification tasks where special considerations are needed for target entities, we conduct experiments on it separately.

Datasets	Types	Classes	Average lengths	Train samples	Test samples
Yelp Review Polarity (Yelp P.)	Sentiment	2	156	560,000	38,000
Yelp Review Full (Yelp F.)	Sentiment	5	158	650,000	50,000
AG’s News (AGNews)	Topic	4	44	120,000	7,600
DBPedia	Topic	14	55	560,000	70,000
TACRED	Relation	41	36	90,755	15,509
SemEval2010-task8 (SemEval)	Relation	19	19	8,000	2,717

TABLE 7.1: Statistics of the six text classification datasets used in our experiments.

7.3.1 Datasets

We use six widely-studied datasets to evaluate our model, two for each text classification task. These datasets are diverse in the aspects of type, size, number of classes and document length. Table 7.1 shows the statistics of the datasets.

For **sentiment analysis**, we use two datasets constructed by Zhang et al. [52] which are obtained from Yelp Dataset Challenge 2015. Yelp Review Polarity (Yelp P.) is a binary sentiment classification dataset whose class is either positive or negative; Yelp Review Full (Yelp F.) contains more fine-grained sentiment classes ranging from rating 1 to 5. For **topic categorization**, we use AG’s News (AGNews) and DBPedia datasets created

by Zhang et al. [52]. AGNews contains news articles from four categories: world, entertainment, sports, and business; DBPedia is an ontology classification dataset containing 14 non-overlapping categories picked from DBpedia 2014. For **relation extraction**, we use TACRED and SemEval2010-task8 (SemEval) datasets which contain hand-annotated subject and object entities as well as the relation type between the entities. TACRED is a large-scale and complex relation extraction dataset constructed by Zhang et al. [9] which has 41 relation types and a *no_relation* class; SemEval2010-task8 [171] is a relatively smaller relation extraction dataset which has 9 directed relations and 1 other relation.

7.3.2 Baseline Models

A variety of baseline models are used for comparison with our model. Different baseline models are used for relation extraction since the task normally requires dedicated models.

7.3.2.1 Text classification models

CNN-based models including Word-level and Char-level CNN [52], as well as deep CNNs including VDCNN [55] and DPCNN [56].

RNN-based models including standard LSTM [52], discriminative LSTM (D-LSTM) of Yogatama et al. [207], and Skim-LSTM which dynamically updates its hidden states [222].

Hybrid model ConvTransformer which is introduced in Chapter 6.

Attentive models including bi-directional block self-attention network (Bi-BloSAN) [228], label-embedding attentive model (LEAM) [223], and Transformer encoder [8] for text classification.

7.3.2.2 Relation extraction models

CNN-based models including the standard CNN for sentence classification [26], CNN with position embeddings (CNN-PE) [151], and graph convolutional network (GCN) over pruned dependency trees of the sentence [10].

RNN-based models including standard LSTM and LSTM with position-aware attention (PA-LSTM) [9].

Hybrid models including contextualized GCN (C-GCN) where the input vectors are obtained using bi-LSTM [10] and ConvTransformer introduced in Chapter 6.

Attentive models including Transformer encoder for relation extraction [11], as well as the knowledge-attention encoder (Knwl-attn) and knowledge-attention self-attention integrated model (Knwl+Self) introduced in Chapter 5.

7.3.3 Experimental Settings

For all the tasks, word embedding matrix $\mathbf{W}^{wrd}$ is initialized with 300-dimensional Glove word embeddings [24]. The fully connected layer before softmax has a dimension of 100. Dropout regularization [197] with a rate of 0.4 is applied during training. The weight and learning rate for center loss are 0.001 and 0.1 respectively. The models are trained using SGD with initial learning rate of 0.01 and momentum of 0.9. Learning rate is decayed with a rate of 0.9 after 10 epochs if the score on development set does not improve. Batch size is set to 100 and the model is trained for 70 epochs. The dimensions of global attention and position embedding are 200 and 60 respectively. We use GeLUs [229] for all the nonlinear activation functions.

The hyper-parameters of ACT are selected by grid-search (refer to Section 7.4.2 for details). Specifically, for sentiment analysis and topic categorization, we set aside 10% of training data as development set to tune model hyper-parameters. We report the average classification accuracy on test set based on 5 independent runs. For ACT, we use 3-layer encoder with 6 attentive convolution heads in each layer, and $m = 100$ convolutional filters with kernel size of 3 in the attentive convolution mechanism. For relation extraction, we use the same settings as Zhang et al. [9] for fair comparison with baseline models. Particularly, instead of using absolute positions in global attention, we use two relative positions for each token with respect to the two target entities. Each relative position embedding has a dimension of 30 and they are concatenated together as final position embedding. For ACT, we use one layer encoder with 6 attentive convolution heads in each layer, and $m = 40$ convolutional filters with kernel size of 3 in the attentive convolution mechanism.

7.3.4 Results and Analyses

Experimental results on the six text classification datasets are shown in Table 7.2. Left table shows the classification accuracy on sentiment analysis and topic categorization

Model	Yelp P.	Yelp F.	AGNews	DBPedia	Model	TACRED	SemEval
Word-level CNN	95.40	59.84	91.45	98.58	CNN	59.3*	70.0*
Char-level CNN	94.75	61.60	90.15	98.34	CNN-PE	61.4*	82.3*
VDCNN	95.72	64.72	91.33	98.71	GCN	64.0	/
DPCNN	97.36	69.42	93.13	99.12	LSTM	61.5*	80.9*
LSTM	94.74	58.17	86.06	98.55	PA-LSTM	65.1	82.7
D-LSTM	92.60	59.60	92.10	98.70	C-GCN	66.4	84.8
Skim-LSTM	/	/	93.60	/	Knwl-attn	66.4*	82.3*
Bi-BloSAN	94.56	62.13	93.32	98.77	Knwl+Self	67.8*	84.3*
LEAM	95.31	64.09	92.45	99.02	Transformer	66.5	83.1
Transformer	96.13*	65.34*	93.89*	98.98*	ConvTransformer	66.6*	83.9*
ConvTransformer	97.14*	67.60*	94.03*	99.15*	ACT	68.0	84.5
ACT	97.41	68.16	94.25	99.19			

TABLE 7.2: Left: classification accuracy (%) on sentiment analysis and topic categorization tasks. Right: F_1 scores on relation extraction task, official micro-averaged and macro-averaged F_1 scores are used for TACRED and SemEval2010-task8 datasets respectively. * means the results are obtained from our implementation. / means not reported. All other results are directly cited from the respective papers mentioned in Section 7.3.2.

tasks; right table shows the F_1 score on relation extraction task. It is observed that the proposed ACT achieves the best performance among all the baseline models for majority of datasets. For Yelp F. and SemEval datasets, ACT ranks the 2nd best and has comparable performance with sophisticated models such as DPCNN and C-GCN.

Compared with CNN-based models, ACT performs better than shallow CNN (word/character-level), graph convolution network (GCN) and simple deep CNN (VDCNN) with a significant margin. The reason is that ACT is able to capture both local n-gram features and global dependencies effectively while preserving sequential information. DPCNN has the most comparable performance with ACT, however, it is much more complex which has 15 layers and an additional region embedding. Our ACT is lightweight, fast and efficient for text classification.

Compared with RNN-based models, ACT consistently outperforms standard LSTM and improved variants of LSTM (D-LSTM, Skim-LSTM, and PA-LSTM) for all the tasks. This is credit to the attentive convolution mechanism for better capturing n-gram features, as well as the multi-head multi-layer structure that does not suffer from gradient vanishing problem when capturing long-distance dependencies. The contextualized GCN (C-GCN) using bi-LSTM performs slightly better than ACT on SemEval dataset, probably due to the benefits of dependency trees. Our ACT does not require any dependency parsing of the sentences.

The experimental results also show that the ConvTransformer introduced in Chapter 6 performs better than CNNs and Transformer for various text classification tasks besides

sentiment classification. This proves the effectiveness of incorporating convolution operation into the architecture of Transformer. ACT further improves the performance of ConvTransformer by replacing the traditional convolution operation with the proposed attentive convolution. In ACT, the learning of convolutional filters is more efficient since the semantic meanings of the filters are utilized for text representation, and the global features can be captured more effectively in the global representation of ACT.

It is observed that attentive models generally outperform RNN-based models. This is due to the better ability of attention mechanisms in capturing long-distance dependencies, especially the self-attention used in Transformer. The proposed ACT outperforms all the attentive models including Transformer encoder. The reason is that ACT has better local n-gram feature extraction capability by using attentive convolution mechanism. However, important n-gram features may not be captured effectively by Transformer because each token will attend to the whole sequence instead of n-grams, the output may be affected by irrelevant tokens. Besides, ACT also simplifies the optimization because it transforms text representation from complex word space to a more informative filter space, leading to more stable training and better keywords extraction capability. The knowledge-attention and self-attention integrated model introduced in Chapter 5 performs as well as ACT on relation extraction task, with the aid of external lexical resources to better capture the keywords of relations. The proposed ACT is able to capture such keywords effectively without the need of external resources, and achieves a new state-of-the-art result on TACRED, a complex relation extraction dataset.

7.4 Discussions

We present more in-depth analyses and discussions on ACT in this section. Two relatively different datasets are used to conduct our experiments: one is Yelp F., a large dataset for sentiment analysis; the other is TACRED, a relation extraction dataset which is much more smaller.

7.4.1 Ablation Study

We perform ablation study on ACT to investigate the contributions of specific components of ACT. Results are shown in Table 7.3.

(1) We replace the proposed attentive convolution mechanism with conventional CNN where feature maps are used for text representation directly, the performance drop by

FIGURE 7.3: Hyper-parameter study on ACT. X-coordinate indicates the hyper-parameters studied, Y-coordinate indicates classification accuracy for Yelp F. dataset and micro-averaged F_1 score for TACRED dataset.

1.8-1.9%. This demonstrates the advantage of utilizing the semantic meaning of convolutional filters attentively for text representation. (2) The proposed multi-head structure outperforms single-head significantly, showing the effectiveness of jointly capturing n-gram features in different sub-word spaces in the multi-head structure. (3) Removing the global representation in global attention degrades the performance by 1%. This demonstrates that incorporating global representation into attention mechanism yields better attention weights for local representations. (4) After removing the position embeddings in global attention, the performance drops by 1.3% for Yelp F. and 6.3% for TACRED. This shows that position information is important for text classification, especially for relation extraction task.

7.4.2 Hyper-parameter Study

In this section, we study the influence of some important hyper-parameters to the performance of ACT, including number of layers, number of attentive convolution heads, kernel size, and number of filters in attentive convolution. Experimental results are shown in Figure 7.3.

It is observed that the number of ACT layers affects the performance significantly. For small dataset like TACRED, single-layer ACT achieves the best performance. For large

Model	Yelp F.	TACRED
ACT	68.3	67.8
1. – Attentive Conv.	67.1	66.5
2. – Multi-head	67.0	65.9
3. – Global rep.	67.6	67.1
4. – Position embed.	67.4	63.5

TABLE 7.3: Ablation study on ACT. Accuracy (%) and micro-averaged F_1 score are reported on the development sets of Yelp F. and TACRED respectively.

Sample Sentences	True Class	Prediction
		no relation
	spouse	spouse
		5 star
	3 star	3 star

TABLE 7.4: Attention visualization for Transformer and ACT. For each sample, the visualization of Transformer is presented first, followed by our proposed ACT. Words are highlighted based on the attention weights assigned to them. Best viewed in color.

datasets like Yelp F., the optimal number of layers is 3. Further increasing the number of layers will increase model complexity and cause performance drop due to overfitting. Besides, multiple attentive convolution heads are beneficial for ACT, the optimal number of heads is 6. For kernel size, results show that 3-gram convolution is most effective for ACT.¹ It is also observed that ACT is not very sensitive to number of convolutional filters. However, larger dataset (Yelp F.) requires more filters than smaller dataset (TACRED) to achieve the best performance.

7.4.3 Attention Visualization

To investigate what ACT focuses on, as well as its difference from Transformer, we conduct visualization of attention weights assigned to words. We sample sentences from the development sets of Yelp F. and TACRED. The visualization results are shown in Table 7.4. More visualizations are presented in Appendix D.

The visualization results show that the proposed ACT can capture the keywords and cue phrases more effectively than Transformer. It is observed that Transformer attends to a wide range of words in the sentence, including stop words and punctuations which may be irrelevant for the classification task. On the contrary, ACT pays more attention to the important n-grams such as “married” for “spouse” relation and “is n’t” for fine-grained

¹We also tried using multiple kernel sizes simultaneously, results show no improvements over single kernel size.

Model	# param.	Inf. time
Transformer	3.38M	0.19s
ACT	1.49M	0.07s

TABLE 7.5: Comparison of model parameters and inference time per batch on Yelp F. dataset.

sentiment classification. These n-grams are the keywords and cue phrases of certain class which are crucial for classification task.

7.4.4 Model Size and Inference Speed

In this section, we investigate two practical aspects of our model for real-world applications: model size and inference speed. For model size, we report the number of model parameters. For inference speed, we report the average time needed to compute a single batch (batch size of 100) of Yelp F. dataset using NVIDIA Tesla P40 GPU with Intel Xeon E5-2667 CPU. We also compare our model with Transformer under the same hyper-parameter settings as described in Section 7.3.3, results are shown in Table 7.5.

The proposed ACT is much more smaller and faster compared with Transformer. It has 56% less parameters and 2.7 times faster inference speed. Therefore, ACT is a lightweight and efficient model for text classification, and it is more practical for real-world applications. Although the large pre-trained language models based on Transformer such as BERT [3] and XLNet [80] have achieved start-of-the-art performance in many NLP tasks, the memory and speed constraints will become obstacles for practical applications.

7.5 Conclusion

We introduce an Attentive Convolutional Transformer (ACT) for efficient text classification. By taking advantage of both Transformer and CNN, ACT is able to capture both local and global dependencies effectively while preserving sequential information of texts. Particularly, a novel attentive convolution mechanism is proposed to better capture n-gram features in convolutional filter space. We also propose a global attention mechanism to obtain the final representation by taking local, global, and position information into consideration. Detailed analyses show that ACT is a lightweight and effective universal text classifier which achieves consistently good results over different text classification tasks, outperforming CNN, RNN, and attentive models including Transformer.

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

This thesis demonstrates the advantages of leveraging both knowledge and data in machine learning systems. We have provided general frameworks and valuable insights on how external knowledge can be utilized and incorporated into various machine learning algorithms to improve their performance on NLP tasks.

In Chapter 3, we demonstrate how external knowledge can be incorporated into BoW representation. A novel Bag-of-Concepts (BoC) model is proposed for document representation which addresses the limitations of BoW. We propose a text conceptualization scheme by utilizing the conceptual knowledge automatically acquired from a probabilistic knowledge base. The proposed BoC representation provides not only higher-level semantics of text but also better interpretability.

In Chapter 4, we demonstrate how external knowledge can be incorporated into CNN. A knowledge-oriented CNN (K-CNN) is proposed whose convolutional filters are not only learned from data, but also automatically generated from external lexical knowledge bases. The proposed word filters in K-CNN are able to enhance CNN’s capability of capturing important linguistic clues, especially when lack of training data. Besides, the word filters are more interpretable than random initialized convolutional filters. The advantages of K-CNN over traditional CNN are demonstrated through experimenting on causal relation extraction task.

In Chapter 5, we demonstrate how external knowledge can be incorporated into the Transformer. We propose a knowledge-attention encoder that is able to attend to the

knowledge indicators generated from external lexical resources. The proposed knowledge-attention mechanism has complementary strengths with self-attention which is able to capture the keywords effectively. We present three ways of integrating the proposed knowledge-attention encoder with the Transformer to maximize the utilization of both knowledge and data. The integrated models achieved state-of-the-art performance for relation extraction task.

In Chapter 6, we demonstrate how external knowledge can be incorporated into traditional attention mechanisms. We take the sentiment analysis task as an example and generate sentiment embeddings from external knowledge base which contains word’s sentiment intensity information. We propose a sentiment-aware attention mechanism that incorporates the sentiment embeddings and able to take external sentiment knowledge into consideration when calculating attention weights. In addition, a novel neural architecture, Convolutional Transformer (ConvTransformer) is introduced which combines the advantages of both convolution operation and Transformer architecture.

In Chapter 7, we extend the work of ConvTransformer and propose an efficient universal text classifier, Attentive Convolutional Transformer (ACT). Instead of representing text in complex word space as in Transformer, or in feature map space as in conventional CNN, we propose a novel attentive convolution operation that represents text in convolutional filter space unitizing the feature maps as attention weights. ACT is able to capture both local and global dependencies while preserving sequential information of texts and achieves good performances in various text classification tasks.

8.2 Recommendations for Future Research

Our work has demonstrated great advantages of incorporating external knowledge into machine learning systems. Especially for NLP, the abundant publicly available knowledge resources of natural language make it feasible and convenient for knowledge acquisition and utilization. The successes of the models introduced in this thesis inspire us to conduct more in-depth research in this area. In this section, we provide some directions and recommendations for future research.

Firstly, based on the works in this thesis, we could develop more refined approaches to further improve our models, and also explore more potential applications of our model in a wide range of NLP tasks. Specifically,

- For BoC/BoCCl model, since it is proposed for document representation, effectively utilizing BoC/BoCCl for sentence-level representation can be studied. Particularly, we could incorporate conceptual knowledge into deep neural networks for better semantic understandings of natural language while preserving model interpretability.
- For K-CNN model, more effective combinations of the knowledge-oriented channel and data-oriented channel can be investigated, for instance, through attention mechanism. Besides, the automatic selection of the target entities for causal relation identification and the more effective extraction of complex causal relations are to be investigated. Furthermore, we can also try to apply K-CNN to other domain-specific tasks besides causal relation extraction.
- For knowledge-attention encoder, besides focusing on lexical features, we can enrich it with more semantic features by incorporating other types of knowledge besides lexical knowledge. For example, we can incorporate conceptual knowledge from encyclopedic knowledge bases to capture the high-level semantics of texts. We can also apply the proposed knowledge-attention mechanism in other NLP tasks such as text classification, sentiment analysis, and question answering.
- For sentiment-aware attention mechanism, since the knowledge is incorporated at word level, effectively incorporate knowledge into attention mechanism at sentence level or document level can be further investigated. In addition, more efficient ways of generating sentiment embeddings can be explored. Besides sentiment information, other types of knowledge can also be incorporated into attention mechanisms for various applications.
- For the proposed neural architectures including ConvTransformer and ACT, we can use them as basic building blocks for text representation, given the advantages of the proposed ConvTransformer and ACT over CNN and Transformer. We could investigate the possible replacement of CNN and Transformer in a wide range of NLP tasks besides text classification, such as machine translation, text summarization, and language model pre-training.

Secondly, to save labor and cost, our works focus on automatic knowledge acquisition instead of manually constructing knowledge. However, since different tasks require different knowledge, we still need to manually decide what kind of knowledge is useful before the knowledge acquisition process. To achieve true AI-complete, the knowledge acquisition process shall be fully automatic, i.e., the system can automatically decide what kind of knowledge is useful for a certain task, and able to look for the knowledge resources to

perform knowledge acquisition precisely without human assistance. More research topics can be discovered towards this direction.

Last but not least, we can explore the potential applications of our methodologies beyond the area of NLP. The methodologies introduced in this thesis can be served as general frameworks of incorporating knowledge into various machine learning algorithms. As long as external knowledge is available for the task, one can utilize the frameworks and ideas proposed in this thesis for other domains such as computer vision, speech recognition, and signal processing.

List of Author's Publications

Journal Articles

- * **Pengfei Li**, Kezhi Mao, Yuecong Xu, Jiaheng Zhang and Qi Li. “Bag-of-Concepts Representation for Document Classification based on Automatic Knowledge Acquisition from Probabilistic Knowledge Base.” *Knowledge-based Systems*, vol. 193, p. 105436. 2020.
- * **Pengfei Li** and Kezhi Mao. “Knowledge-oriented Convolutional Neural Network for Causal Relation Extraction from Natural Language Texts.” *Expert Systems with Applications*, vol. 115, pp.512-523. 2019.

Conference Proceedings

- * **Pengfei Li**, Peixiang Zhong, Kezhi Mao, Dongzhe Wang, Xuefeng Yang, Yunfeng Liu, Jian-xiong Yin and Simon SEE. “ACT: an Attentive Convolutional Transformer for Efficient Text Classification.” Submitted to *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*. 2021.
- * **Pengfei Li**, Peixiang Zhong, Jiaheng Zhang and Kezhi Mao. “Convolutional Transformer with Sentiment-aware Attention for Sentiment Analysis.” In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8. 2020.
- * **Pengfei Li**, Kezhi Mao, Xuefeng Yang and Qi Li. “Improving Relation Extraction with Knowledge-attention.” 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)*, pp. 229-239. 2019.

Bibliography

- [1] Francois Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018. [ix](#), [15](#)
- [2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. [ix](#), [2](#), [15](#), [16](#), [35](#), [66](#), [70](#), [113](#)
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. [ix](#), [2](#), [18](#), [30](#), [32](#), [36](#), [114](#), [140](#)
- [4] Marc Moreno Lopez and Jugal Kalita. Deep learning applied to nlp. *arXiv preprint arXiv:1703.03091*, 2017. [ix](#), [20](#)
- [5] Understanding lstm networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs>. Accessed: 2020-06-01. [ix](#), [22](#)
- [6] Memory, attention, sequences. <https://towardsdatascience.com/memory-attention-sequences-37456d271992>. Accessed: 2020-06-01. [ix](#), [26](#)
- [7] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016. [ix](#), [26](#), [27](#), [32](#), [36](#), [112](#), [114](#)
- [8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. [ix](#), [2](#), [4](#), [17](#), [27](#), [28](#), [29](#), [36](#), [93](#), [95](#), [98](#), [99](#), [112](#), [114](#), [117](#), [122](#), [123](#), [131](#), [134](#)
- [9] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, 2017. [xi](#), [93](#), [95](#), [98](#), [102](#), [103](#), [104](#), [134](#), [135](#)
- [10] 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, 2018. [xi](#), [95](#), [102](#), [104](#), [134](#), [135](#)
- [11] Ivan Bilan and Benjamin Roth. Position-aware self-attention with relative positional encodings for slot filling. *arXiv preprint arXiv:1807.03052*, 2018. [xi](#), [93](#), [95](#), [102](#), [104](#), [135](#)
 - [12] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971. [1](#)
 - [13] Naomi Sager. Natural language information formatting: The automatic conversion of texts to. *Advances in computers*, page 89, 1979. [1](#)
 - [14] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999. [1](#)
 - [15] Mark Johnson. How the statistical revolution changes (computational) linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 3–11, 2009. [1](#)
 - [16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. [1](#)
 - [17] Rui Zhao. *Representation learning for sentences and documents*. PhD thesis, Nanyang Technological University, 2017. [1](#)
 - [18] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. [2](#), [11](#)
 - [19] Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):721–735, 2008. [11](#)
 - [20] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37, 2008. [2](#), [10](#)
 - [21] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003. [2](#), [14](#), [35](#)
 - [22] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011. [14](#), [35](#)
 - [23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. [15](#), [16](#), [35](#)

- [24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. [2](#), [16](#), [35](#), [103](#), [122](#), [135](#)
- [25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. [2](#), [18](#), [36](#), [66](#)
- [26] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014. [2](#), [19](#), [36](#), [71](#), [102](#), [112](#), [113](#), [134](#)
- [27] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, 2014. [2](#), [19](#), [32](#), [36](#), [66](#), [71](#), [113](#)
- [28] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013. [2](#)
- [29] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014. [23](#)
- [30] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015. [2](#), [32](#), [36](#), [114](#)
- [31] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211. Association for Computational Linguistics, 2012. [2](#), [36](#), [70](#)
- [32] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. [2](#), [36](#)
- [33] Peter Jarvis. *Towards a comprehensive theory of human learning*, volume 1. Psychology Press, 2006. [3](#)
- [34] Robert E Schapire, Marie Rochery, Mazin Rahim, and Narendra Gupta. Incorporating prior knowledge into boosting. In *ICML*, volume 2, pages 538–545, 2002. [3](#)
- [35] Fabien Lauer and Gérard Bloch. Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing*, 71(7-9):1578–1594, 2008.

- [36] Ting Yu, Simeon Simoff, and Tony Jan. Vqsvm: a case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing*, 73(13-15):2614–2623, 2010. [3](#)
- [37] Christiane Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010. [4](#), [37](#), [41](#), [74](#), [118](#)
- [38] Josef Ruppenhofer, Michael Ellsworth, Miriam RL Petruck, Christopher R Johnson, and Jan Scheffczyk. Framenet ii: Extended theory and practice. berkeley. CA: *International Computer Science Institute*, 2006. [4](#), [37](#), [74](#), [96](#)
- [39] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: a high-coverage lexical resource for opinion mining. *Evaluation*, 17:1–26, 2007. [4](#), [37](#)
- [40] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM, 2012. [4](#), [37](#)
- [41] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954. [10](#)
- [42] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990. [12](#), [35](#), [54](#)
- [43] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196, 2001. [13](#), [35](#)
- [44] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. [14](#), [35](#), [54](#)
- [45] 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)*, volume 1, pages 2227–2237, 2018. [17](#), [32](#)
- [46] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Technical report, OpenAI*, 2018. [17](#), [30](#), [32](#)
- [47] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. [18](#)
- [48] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. [19](#)
- [49] 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. [19](#), [70](#), [113](#)

- [50] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. [19](#)
- [51] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014. [19](#)
- [52] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015. [21](#), [53](#), [112](#), [113](#), [121](#), [123](#), [133](#), [134](#)
- [53] Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, 2016. [21](#), [93](#), [95](#)
- [54] Shiyao Wang, Minlie Huang, and Zhidong Deng. Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI*, pages 4468–4474, 2018. [21](#)
- [55] Alexis Conneau, Holger Schwenk, Yann LeCun, and Lc Barrault. Very deep convolutional networks for text classification. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 1107–1116. Association for Computational Linguistics (ACL), 2017. [21](#), [113](#), [121](#), [123](#), [134](#)
- [56] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570, 2017. [21](#), [113](#), [134](#)
- [57] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. [21](#)
- [58] Hoa T Le, Christophe Cerisara, and Alexandre Denis. Do convolutional networks need to be deep for text classification? In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [21](#)
- [59] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. [21](#)
- [60] Danilo P Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001. [21](#)
- [61] Wenyu Wei and Pengfei Li. Multi-channel lstm with different time scales for foreign exchange rate prediction. In *Proceedings of the International Conference on Advanced Information Science and System*, pages 1–7, 2019. [21](#)
- [62] Jakub Nowak, Ahmet Taspinar, and Rafa Scherer. Lstm recurrent neural networks for short text and sentiment classification. In *International Conference on Artificial Intelligence and Soft Computing*, pages 553–562. Springer, 2017. [112](#), [114](#)

- [63] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. [21](#)
- [64] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 (2):157–166, 1994. [22](#)
- [65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. [22](#)
- [66] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015. [25](#)
- [67] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. [25](#)
- [68] 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, 2016. [26](#), [112](#), [114](#)
- [69] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017. [114](#), [122](#), [123](#)
- [70] Zufan Zhang, Yang Zou, and Chenquan Gan. Textual sentiment analysis via three different attention convolutional neural networks and cross-modality consistent regression. *Neurocomputing*, 275:1407–1415, 2018. [26](#), [114](#)
- [71] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017. [27](#)
- [72] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016. [27](#)
- [73] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017. [27](#)
- [74] Xingyi Cheng, Weidi Xu, Taifeng Wang, Wei Chu, Weipeng Huang, Kunlong Chen, and Junfeng Hu. Variational semi-supervised aspect-term sentiment analysis via transformer. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 961–969, 2019. [30](#), [112](#), [114](#)
- [75] Ming Jiang, Junlei Wu, Xiangrong Shi, and Min Zhang. Transformer based memory network for sentiment analysis of web comments. *IEEE Access*, 7:179942–179953, 2019. [112](#), [114](#)

- [76] Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, 2018.
- [77] Peixiang Zhong, Di Wang, and Chunyan Miao. Knowledge-enriched transformer for emotion detection in textual conversations. 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 165–176, 2019. [112](#), [115](#)
- [78] Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 681–689, 2019. [30](#)
- [79] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. [30](#)
- [80] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. [30](#), [114](#), [140](#)
- [81] Tiago A Almeida, Tiago P Silva, Igor Santos, and José M Gómez Hidalgo. Text normalization and semantic indexing to enhance instant messaging and sms spam filtering. *Knowledge-Based Systems*, 108:25–32, 2016. [31](#)
- [82] Jlangbo Dang, Murat Kalender, Candemir Toklu, and Kenneth Hampel. Semantic search tool for document tagging, indexing and search, June 20 2017. US Patent 9,684,683. [31](#)
- [83] Yanshan Xiao, Bo Liu, Jie Yin, and Zhifeng Hao. A multiple-instance stream learning framework for adaptive document categorization. *Knowledge-Based Systems*, 120:198–210, 2017. [31](#)
- [84] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research*, 47:521–573, 2013. [31](#)
- [85] Gichang Lee, Jaeyun Jeong, Seungwan Seo, CzangYeob Kim, and Pilsung Kang. Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network. *Knowledge-Based Systems*, 152:70–82, 2018. [31](#)
- [86] Yijing Li, Haixiang Guo, Qingpeng Zhang, Mingyun Gu, and Jianying Yang. Imbalanced text sentiment classification using universal and domain-specific knowledge. *Knowledge-Based Systems*, 160:1–15, 2018. [31](#)
- [87] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 38(3): 2758–2765, 2011. [31](#)

- [88] Magnus Sahlgren and Rickard Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 487. Association for Computational Linguistics, 2004. [35](#)
- [89] Chao Xing, Dong Wang, Xuwei Zhang, and Chao Liu. Document classification with distributions of word vectors. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5. IEEE, 2014. [35](#), [54](#)
- [90] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014. [35](#), [54](#), [71](#)
- [91] Rui Zhao and Kezhi Mao. Fuzzy bag-of-words model for document representation. *IEEE Transactions on Fuzzy Systems*, 26(2):794–804, 2018. [35](#)
- [92] Han Kyul Kim, Hyunjoong Kim, and Sungzoon Cho. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*, 266:336–352, 2017. [36](#)
- [93] Karin Kipper Schuler. *VerbNet : A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005. [37](#), [76](#)
- [94] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Third IEEE international conference on data mining*, pages 541–544. IEEE, 2003. [37](#)
- [95] Qiming Luo, Enhong Chen, and Hui Xiong. A semantic term weighting scheme for text categorization. *Expert Systems with Applications*, 38(10):12708–12716, 2011.
- [96] Ming Liu, Gholamreza Haffari, Wray Buntine, and Michelle Ananda-Rajah. Leveraging linguistic resources for improving neural text classification. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 34–42, 2017. [37](#)
- [97] Yinglin Wang, Ming Wang, and Hamido Fujita. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, 2019. doi: <https://doi.org/10.1016/j.knosys.2019.105030>. [37](#)
- [98] Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*, 36(6):823–848, 2010. [37](#)
- [99] Chihli Hung and Shiuan-Jeng Chen. Word sense disambiguation based sentiment lexicons for sentiment classification. *Knowledge-Based Systems*, 110:224–232, 2016.
- [100] Farhan Hassan Khan, Usman Qamar, and Saba Bashir. A semi-supervised approach to sentiment analysis using revised sentiment strength based on sentiwordnet. *Knowledge and information Systems*, 51(3):851–872, 2017. [37](#), [115](#)

- [101] Pengfei Li and Kezhi Mao. Knowledge-oriented convolutional neural network for causal relation extraction from natural language texts. *Expert Systems with Applications*, 115:512–523, 2019. [37](#), [95](#), [114](#)
- [102] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. [37](#)
- [103] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007. [37](#)
- [104] Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 919–928. ACM, 2009. [37](#)
- [105] Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, volume 6, pages 1301–1306, 2006. [37](#)
- [106] Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19(3):265–281, 2009.
- [107] Alaa Alahmadi, Arash Joorabchi, and Abdhussain E Mahdi. A new text representation scheme combining bag-of-words and bag-of-concepts approaches for automatic text classification. In *GCC Conference and Exhibition (GCC), 2013 7th IEEE*, pages 108–113. IEEE, 2013. [37](#)
- [108] Jung-Hyun Lee, Jongwoo Ha, Jin-Yong Jung, and Sangkeun Lee. Semantic contextual advertising based on the open directory project. *ACM Transactions on the Web (TWEB)*, 7(4):24, 2013. [37](#)
- [109] HaeYong Shin, GeunJae Lee, Woo-Jong Ryu, and SangKeun Lee. Utilizing wikipedia knowledge in open directory project-based text classification. In *Proceedings of the Symposium on Applied Computing*, pages 309–314. ACM, 2017. [37](#)
- [110] Roberta A Sinoara, Jose Camacho-Collados, Rafael G Rossi, Roberto Navigli, and Solange O Rezende. Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, 163:955–971, 2019. [37](#), [95](#), [114](#)
- [111] Liang Yao, Yin Zhang, Baogang Wei, Hongze Qian, and Yibing Wang. Incorporating probabilistic knowledge into topic models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 586–597. Springer, 2015. [37](#)
- [112] Yi-Kun Tang, Xian-Ling Mao, Heyan Huang, Xuewen Shi, and Guihua Wen. Conceptualization topic modeling. *Multimedia Tools and Applications*, 77(3):3455–3471, 2018. [37](#)
- [113] Yiming Li, Baogang Wei, Yonghuai Liu, Liang Yao, Hui Chen, Jifang Yu, and Wenhao Zhu. Incorporating knowledge into neural network for text representation. *Expert Systems with Applications*, 96:103–114, 2018. [37](#)

- [114] Walid Shalaby, Wlodek Zadrozny, and Hongxia Jin. Beyond word embeddings: learning entity and concept representations from large scale knowledge bases. *Information Retrieval Journal*, pages 1–18, 2018. [37](#)
- [115] Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the twenty-second international joint conference on artificial intelligence-volume three*, pages 2330–2336. AAAI Press, 2011. [37](#)
- [116] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1069–1078. ACM, 2014.
- [117] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. Short text understanding through lexical-semantic analysis. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 495–506. IEEE, 2015. [37](#)
- [118] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992. [38](#)
- [119] Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, and Yanghua Xiao. An inference approach to basic level of categorization. In *Proceedings of the 24th acm international on conference on information and knowledge management*, pages 653–662. ACM, 2015. [39](#)
- [120] Ming-hua CHEN, Jing-hua YIN, Chang SHU, and Ming-jiang WANG. A chinese word segmentation system design based on forward-backward maximum matching algorithm. *Information Technology*, 6:042, 2009. [43](#)
- [121] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005. [48](#)
- [122] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957. [49](#)
- [123] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pages 377–384. ACM, 2006. [53](#)
- [124] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008. [54](#)
- [125] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [55](#)
- [126] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006. [57](#)

- [127] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998. [58](#)
- [128] Eugenia Goldvarg and Philip N Johnson-Laird. Naive causality: a mental model theory of causal meaning and reasoning¹. *Cognitive science*, 25(4):565–610, 2001. [65](#)
- [129] Alejandro Peña, Humberto Sossa, and Agustín Gutiérrez. Causal knowledge and reasoning by cognitive maps: Pursuing a holistic approach. *Expert Systems with Applications*, 35(1-2):2–18, 2008. [65](#)
- [130] Christopher SG Khoo, Jaklin Kornfilt, Robert N Oddy, and Sung Hyon Myaeng. Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, 13(4):177–186, 1998. [65](#), [66](#)
- [131] Lars Juhl Jensen, Jasmin Saric, and Peer Bork. Literature mining for the biologist: from information retrieval to biological discovery. *Nature reviews genetics*, 7(2): 119, 2006. [65](#)
- [132] Roxana Girju. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering- Volume 12*, pages 76–83. Association for Computational Linguistics, 2003. [65](#), [66](#), [68](#)
- [133] Ryuichiro Higashinaka and Hideki Isozaki. Automatically acquiring causal expression patterns from relation-annotated corpora to improve question answering for why-questions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 7(2):6, 2008. [65](#)
- [134] Eduardo Jacobo Miranda Ackerman. Extracting a causal network of news topics. In *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*, pages 33–42. Springer, 2012. [65](#)
- [135] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918. ACM, 2012.
- [136] Francesco Mele and Antonio Sorgente. Ontotimefl—a formalism for temporal annotation and reasoning for natural language text. In *New Challenges in Distributed Information Filtering and Retrieval*, pages 151–170. Springer, 2013.
- [137] Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 987–997, 2014. [65](#)

- [138] Hyunmin Cheong and LH Shu. Automatic extraction of causally related functions from natural-language text for biomimetic design. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 373–382. American Society of Mechanical Engineers, 2012. [65](#)
- [139] Jin-Dong Kim, Tomoko Ohta, and Jun’ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10, 2008. [65](#)
- [140] Shanshan Wang, Zhang Zhe, Ye Kang, Huaiqing Wang, and Xiaojian Chen. An ontology for causal relationships between news and financial instruments. *Expert Systems with Applications*, 35(3):569–580, 2008. [65](#)
- [141] Kun Chang Lee and Sangjae Lee. A causal knowledge-based expert system for planning an internet-based stock trading system. *Expert Systems with Applications*, 39(10):8626–8635, 2012. [65](#)
- [142] Pilar León Araúz and Pamela Faber. Causality in the specialized domain of the environment. In *Semantic Relations-II. Enhancing Resources and Applications Workshop Programme*. Citeseer, page 10. Citeseer, 2012. [65](#)
- [143] Ki Chan and Wai Lam. Extracting causation knowledge from natural language texts. *International Journal of Intelligent Systems*, 20(3):327–358, 2005. [66](#)
- [144] Takashi Inui, Kentaro Inui, and Yuji Matsumoto. Acquiring causal knowledge from text using the connective marker tame. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(4):435–474, 2005.
- [145] Ashwin Ittoo and Gosse Bouma. Extracting explicit and implicit causal relations from sparse, domain-specific texts. In *International Conference on Application of Natural Language to Information Systems*, pages 52–63. Springer, 2011. [66](#)
- [146] Steven Bethard and James H Martin. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 177–180. Association for Computational Linguistics, 2008. [66](#)
- [147] Antonio Sorgente, Giuseppe Vettigli, and Francesco Mele. Automatic extraction of cause-effect relations in natural language text. In *DART@AI*IA*, volume 1109 of *CEUR Workshop Proceedings*, pages 37–48, 2013. [69](#)
- [148] Xuefeng Yang and Kezhi Mao. Multi level causal relation identification using extended features. *Expert Systems with Applications*, 41(16):7171–7181, 2014. [69](#)
- [149] Partha Pakray and Alexander Gelbukh. An open-domain cause-effect relation detection from paired nominals. In *Mexican International Conference on Artificial Intelligence*, pages 263–271. Springer, 2014. [66](#)
- [150] 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, 2014. [66](#), [71](#), [80](#), [85](#), [95](#), [98](#)

- [151] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, 2015. [71](#), [85](#), [95](#), [102](#), [134](#)
- [152] Cicero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, 2015. [66](#), [71](#)
- [153] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010. [66](#), [70](#)
- [154] Roxana Girju, Dan I Moldovan, et al. Text mining for causal relations. In *FLAIRS conference*, pages 360–364, 2002. [68](#)
- [155] Du-Seong Chang and Key-Sun Choi. Causal relation extraction using cue phrase and lexical pair probabilities. In *International Conference on Natural Language Processing*, pages 61–70. Springer, 2004. [69](#)
- [156] Eduardo Blanco, Nuria Castell, and Dan I Moldovan. Causal relation extraction. In *Lrec*, 2008. [69](#)
- [157] Roxana Girju, Brandon Beamer, Alla Rozovskaya, Andrew Fister, and Suma Bhat. A knowledge-rich approach to identifying semantic relations between nominals. *Information processing & management*, 46(5):589–610, 2010. [69](#)
- [158] Bryan Rink and Sanda Harabagiu. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics, 2010. [69](#), [93](#)
- [159] Bryan Rink, Cosmin Adrian Bejan, and Sanda M Harabagiu. Learning textual graph patterns to detect causal event relations. In *FLAIRS Conference*, 2010. [69](#)
- [160] Sendong Zhao, Ting Liu, Sicheng Zhao, Yiheng Chen, and Jian-Yun Nie. Event causality extraction based on connectives analysis. *Neurocomputing*, 173:1943–1950, 2016. [70](#)
- [161] Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376, 2013. [70](#)
- [162] Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. Mit at semeval-2017 task 10: Relation extraction with convolutional neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 978–984. Association for Computational Linguistics, 2017. [71](#), [93](#), [95](#)

- [163] Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, Ildar Batyrshin, David Pinto, and Liliana Chanona-Hernández. Application of the distributed document representation in the authorship attribution task for small corpora. *Soft Computing*, 21(3):627–639, 2017. 71
- [164] Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and David Pinto. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*, pages 1–16, 2018. 71
- [165] Mehwish Riaz and Roxana Girju. Recognizing causality in verb-noun pairs via noun and verb semantics. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 48–57, 2014. 75
- [166] Paul Kingsbury and Martha Palmer. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, volume 3. Citeseer, 2003. 76
- [167] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics, 2006. 76
- [168] M. Lane David. Online statistics education: An interactive multimedia course of study @ONLINE, April 2018. URL <http://onlinestatbook.com>. 78
- [169] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. 83
- [170] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *Computing Research Repository (CoRR)*, abs/1212.5701, 2012. Retrieved from <http://arxiv.org/abs/1212.5701>. 84
- [171] 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 Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics, 2009. 84, 103, 134
- [172] Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. Annotating causality in the tempeval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19, 2014. 84
- [173] Tommaso Caselli and Piek Vossen. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86, 2017. 84
- [174] Muhao Chen, Yingtao Tian, Xuelu Chen, Zijun Xue, and Carlo Zaniolo. On2vec: Embedding-based relation prediction for ontology population. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 315–323. SIAM, 2018. 93

- [175] Marco Fossati, Emilio Dorigatti, and Claudio Giuliano. N-ary relation extraction for simultaneous t-box and a-box knowledge base augmentation. *Semantic Web*, 9(4):413–439, 2018. [93](#)
- [176] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*, 2017. [93](#)
- [177] Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [93](#)
- [178] Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717. ACM, 2006. [93](#)
- [179] Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics, 2008. [93](#)
- [180] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1461–1470, 2016. [93](#), [95](#)
- [181] Ngoc Thang Vu, Heike Adel, Pankaj Gupta, et al. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of NAACL-HLT*, pages 534–539, 2016. [93](#)
- [182] 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)*, volume 2, pages 207–212, 2016. [93](#), [95](#), [112](#)
- [183] Patrick Verga, Emma Strubell, and Andrew McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. 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 872–884, 2018. [93](#), [95](#)
- [184] Cicero dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, 2015. [95](#)
- [185] Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*, 2015. [95](#)

- [186] 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)*, volume 1, pages 1105–1116, 2016. [95](#)
- [187] Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245, 2018. [95](#)
- [188] Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. Multi-level structured self-attentions for distantly supervised relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2216–2225, 2018. [95](#)
- [189] Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. Revisiting word embedding for contrasting meaning. 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 106–115, 2015. [95](#)
- [190] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. Learning semantic word embeddings based on ordinal knowledge constraints. 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 1501–1511, 2015. [95](#)
- [191] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, 2015. [95](#)
- [192] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017. [95](#)
- [193] Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*, 2016. [95](#)
- [194] Yun-Nung Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. Knowledge as a teacher: Knowledge-guided structural attention networks. *arXiv preprint arXiv:1609.03286*, 2016.
- [195] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417, 2018.
- [196] Ying Shen, Yang Deng, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. Knowledge-aware attentive neural network for ranking question answer pairs. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 901–904. ACM, 2018. [95](#)

- [197] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. [103](#), [120](#), [135](#)
- [198] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002. [111](#)
- [199] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM, 2003.
- [200] Yao Lu, Xiangfei Kong, Xiaojun Quan, Wenyin Liu, and Yinlong Xu. Exploring the sentiment strength of user reviews. In *International Conference on Web-Age Information Management*, pages 471–482. Springer, 2010.
- [201] Magdalini Eirinaki, Shamita Pital, and Japinder Singh. Feature-based opinion mining and ranking. *Journal of Computer and System Sciences*, 78(4):1175–1184, 2012. [111](#)
- [202] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002. [111](#)
- [203] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 412–418, 2004.
- [204] M Rushdi Saleh, Maria Teresa Martín-Valdivia, Arturo Montejo-Ráez, and LA Ureña-López. Experiments with svm to classify opinions in different domains. *Expert Systems with Applications*, 38(12):14799–14804, 2011.
- [205] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57: 77–93, 2014. [111](#)
- [206] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014. [112](#), [113](#)
- [207] D Yogatama, C Dyer, W Ling, and P Blunsom. Generative and discriminative text classification with recurrent neural networks. In *Thirty-fourth International Conference on Machine Learning (ICML 2017)*. International Machine Learning Society, 2017. [112](#), [114](#), [122](#), [123](#), [134](#)
- [208] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *ICLR*, 2018. [112](#), [114](#)

- [209] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. Exploiting bert for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, 2019. [112](#), [114](#)
- [210] Pengfei Li, Kezhi Mao, Xuefeng Yang, and Qi Li. Improving relation extraction with knowledge-attention. 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 229–239, 2019. [112](#), [114](#)
- [211] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, 2016. [114](#)
- [212] Baoxin Wang. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2311–2320, 2018. [114](#)
- [213] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015. [114](#)
- [214] 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, 2016. [114](#)
- [215] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. [114](#)
- [216] Yanliu Wang and Pengfei Li. Knowledge-oriented hierarchical neural network for sentiment classification. In *IOP Conference Series: Materials Science and Engineering*, volume 646, page 012023. IOP Publishing, 2019. [115](#)
- [217] Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [115](#)
- [218] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, volume 6, pages 417–422. Citeseer, 2006. [118](#)
- [219] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204, 2010. [118](#)
- [220] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016. [120](#), [133](#)

- [221] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011. [121](#)
- [222] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *International Conference on Learning Representations*, 2018. [122](#), [123](#), [134](#)
- [223] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331, 2018. [122](#), [123](#), [134](#)
- [224] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*, 2018. [128](#)
- [225] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019.
- [226] Baosong Yang, Longyue Wang, Derek F Wong, Lidia S Chao, and Zhaopeng Tu. Convolutional self-attention 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 4040–4045, 2019.
- [227] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020. [128](#)
- [228] T Shen, T Zhou, G Long, J Jiang, and C Zhang. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *International Conference on Representation Learning*, 2018. [134](#)
- [229] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. [135](#)

Appendices

A Proof of the Cosine Similarity Value in K-CNN Feature Map

K-CNN feature map:

Given a word filter $\mathbf{f} = [\mathbf{f}_1, \dots, \mathbf{f}_k]^T$ and a k-gram from text $\mathbf{w}_{kgram} = [\mathbf{w}_i, \dots, \mathbf{w}_{i+k-1}]^T$, the resulted feature map value is:

$$m_i = (\sum_{j=1}^k \mathbf{f}_j^T \mathbf{w}_{i+j-1} + b)/k$$

where $\mathbf{f}_j \in \mathbf{f}$, $\mathbf{w}_{i+j-1} \in \mathbf{w}_{kgram}$, k is the convolutional window size, and b is a bias term.

Prove:

$$m_i = \cos(\theta) = \frac{\mathbf{f} \cdot \mathbf{w}_{kgram}}{\|\mathbf{f}\| \|\mathbf{w}_{kgram}\|}$$

Assumptions and constraints:

1. $\mathbf{f}_j$ and $\mathbf{w}_{i+j-1}$ are unit vectors obtained from pre-trained word embeddings.
2. The bias term is ignored, i.e. $b = 0$.

Proof:

$$\begin{aligned} \cos(\theta) &= \frac{\mathbf{f} \cdot \mathbf{w}_{kgram}}{\|\mathbf{f}\| \|\mathbf{w}_{kgram}\|} \\ &= \frac{\sum_{j=1}^k \mathbf{f}_j^T \mathbf{w}_{i+j-1}}{\sqrt{\sum_{j=1}^k \|\mathbf{f}_j\|^2} \sqrt{\sum_{j=1}^k \|\mathbf{w}_{i+j-1}\|^2}} \\ &= \frac{\sum_{j=1}^k \mathbf{f}_j^T \mathbf{w}_{i+j-1}}{\sqrt{k} \sqrt{k}} \\ &= (\sum_{j=1}^k \mathbf{f}_j^T \mathbf{w}_{i+j-1})/k \\ &= m_i \end{aligned}$$

B FramNet Frames Used for Relation Extraction on TACRED Dataset

Relation Types	FrameNet Frames
org:alternate_names, per:alternate_names	Being_named, Name_conferral, Namesake, Referring_by_name, Simple_naming
org:city_of_headquarters, org:country_of_headquarters, org:stateorprovince_of_headquarters	Being_located, Locale, Locale_by_characteristic_entity, Locale_by_collocation, Locale_by_event, Locale_by_ownership, Locale_by_use, Locale_closure, Locating, Locative_relation, Spatial_co-location
org:founded, org:founded_by	Intentionally_create
org:dissolved	Location_in_time, Relative_time, Time_vector, Timespan, Temporal_collocation, Temporal_subregion
org:member_of, org:members	Becoming_a_member, Membership
org:political/religious_affiliation, per:religion	People_by_religion, Religious_belief, Political_locales
org:subsidiaries, org:parents	Part_whole, Partitive, Inclusion
org:shareholders	Capital_stock
org:top_members/employees, org:number_of_employees/members, per:employee_of	Leadership, Working_a_post, Employing, People_by_vocation, Cardinal_numbers
per:age	Age
per:cause_of_death, per:date_of_death, per:city_of_death, per:country_of_death, per:stateorprovince_of_death	Death, Cause_harm
per:charges	Notification_of_charges, Committing_crime, Criminal_investigation
per:children, per:parents, per:other_family, per:siblings	Kinship
per:cities_of_residence, per:countries_of_residence, per:stateorprovinces_of_residence	Expected_location_of_person, Residence
per:date_of_birth, per:city_of_birth, per:country_of_birth, per:stateorprovince_of_birth	Being_born, Giving_birth
per:origin	Origin, People_by_origin
per:schools_attended	Education_teaching
per:spouse	Personal_relationship, Forming_relationships
per:title	Performers_and_roles

C Hyper-parameter Settings and Training Details of Knowledge-attention Encoder

The dimension of POS and position embeddings are both 30. The inner layer dimension in position-wise feed-forward network is 130. The dimension of the relative positional encoding within knowledge-attention and self-attention is 50. The attention dimension is 200 in position-aware attention and 100 in multi-channel attention. The fully connected network before softmax has a dimensionality of 100. We use ReLU for all the nonlinear activation functions. Dropout rate is 0.4 for input embeddings, attention outputs and position-wise feed-forward outputs, and 0.1 for attention weights dropout.

The models are trained using Stochastic Gradient Descent with learning rate of 0.1 and momentum of 0.9. The learning rate is decayed with a rate of 0.9 after 15 epochs if F_1 score on dev set does not improve. The batch size is set to 100 and we train the model for 70 epochs.

D Attention Visualization

The table below shows more attention visualization results for Transformer and ACT. Words are highlighted based on the attention weights assigned to them. For each sample sentence, the visualization of Transformer is presented first, followed by our proposed ACT. Best viewed in color.

Sample Sentences	True Class	Prediction
<p>A federal appeals court on Wednesday ruled against the Securities and Exchange Commission in its effort to get wiretaps from the criminal prosecution of OBJ-PERSON OBJ-PERSON , the founder of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION hedge fund .</p> <p>A federal appeals court on Wednesday ruled against the Securities and Exchange Commission in its effort to get wiretaps from the criminal prosecution of OBJ-PERSON OBJ-PERSON , the founder of the SUBJ-ORGANIZATION SUBJ-ORGANIZATION hedge fund .</p>	founded by founded by	founded by founded by
<p>OBJ-PERSON returned to Buffalo in 1955 and was a part of a group of black intellectuals who included philosopher and poet SUBJ-PERSON SUBJ-PERSON , whom she married in 1958 .</p> <p>OBJ-PERSON returned to Buffalo in 1955 and was a part of a group of black intellectuals who included philosopher and poet SUBJ-PERSON SUBJ-PERSON , whom she married in 1958 .</p>	spouse spouse	no relation spouse
<p>Grand Imam of Al-Azhar Tantawi dies of heart attack : MENA SUBJ-PERSON , 81 , was in OBJ-CITY to attend the King Faisal awards ceremony , it said .</p> <p>Grand Imam of Al-Azhar Tantawi dies of heart attack : MENA SUBJ-PERSON , 81 , was in OBJ-CITY to attend the King Faisal awards ceremony , it said .</p>	city of death	cities of residence city of death
<p>This Giant Eagle maybe the worst supermarket I've ever been to . There are thugs hanging out by the entrance and in the parking lot . The interior is dark and dank and has an outdated `` made in the 70s '' feel . With options like this I do n't have a problem taking the extra time to drive to Trader Joe 's .</p> <p>This Giant Eagle maybe the worst supermarket I've ever been to . There are thugs hanging out by the entrance and in the parking lot . The interior is dark and dank and has an outdated `` made in the 70s '' feel . With options like this I do n't have a problem taking the extra time to drive to Trader Joe 's .</p>	1 star 1 star	1 star 1 star
<p>When I worked at the Renaissance tower , I 'd come here when I was too lazy to walk down the street for something better . Because , honestly , their pizza just is n't that great . Or good , really . But I 've had the breakfast muffin twice and both times it was beyond awesome ! Just the right amount of grease to let you know it 's good . And super cheap !</p> <p>When I worked at the Renaissance tower , I 'd come here when I was too lazy to walk down the street for something better . Because , honestly , their pizza just is n't that great . Or good , really . But I 've had the breakfast muffin twice and both times it was beyond awesome ! Just the right amount of grease to let you know it 's good . And super cheap !</p>	5 star 3 star	5 star 3 star
<p>Heard good things , but , did not even measure up . Very bland ... really man ? It 's Thai how can it be flavorless ? Do n't waste your time !</p> <p>Heard good things , but , did not even measure up . Very bland ... really man ? It 's Thai how can it be flavorless ? Do n't waste your time !</p>	1 star	2 star 1 star