Semantic Representation Learning for Natural Language Understanding

Yong Zhang

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

2018
Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

........................................  ........................................
Date                                      Yong Zhang
Acknowledgements

This thesis would not have been possible without the support of many people. Firstly, I wish to express my greatest gratitude to my advisors, Professor Meng Joo Er and Professor Guangbin Huang, for their professional guidance and encouragement throughout my PhD study. Professor Er has provided a great independent research environment and patiently revised many drafts of papers for me. I am sincerely grateful that Professor Huang helped me out from the most difficult time of my PhD. Secondly, I also wish to give my sincere gratitude to Professor Mahardhika Pratama, for his unconditional help in reviewing my papers. He has taught me a great many skills in how to write high-level research papers.

I also want to thank my many cool friends in the Machine Learning and Internet of Things Labs who make my stay in NTU memorable and joyful. Special thanks go to Rui Zhao for advising me to explore the area of natural language processing and for all those insightful discussions. I am also thankful for the support and love from all the brothers and sisters in the church, making the life in Singapore like being at home.

Last but never the least, I must thank my beloved family. Mother and father, thank you for your unconditional love and support. Mother and father-in-law, thank you for taking care of Xiaoxiao during the period she was pregnant and after the baby was born. The greatest thanks go to my wife. Thank you for appearing in my life, thank you for being my wife, thank you for bring our beautiful daughter to the world, thank you for all the sacrifices made for our family. Love you, my girls.
“One, remember to look up at the stars and not down at your feet. Two, never give up work. Work gives you meaning and purpose and life is empty without it. Three, if you are lucky enough to find love, remember it is there and don’t throw it away.”

—Stephen Hawking

To my beloved family.
# Contents

Acknowledgements .......................... i

Abstract ..................................... ix

List of Figures ............................... x

List of Tables ............................... xv

Abbreviations ............................... xvii

1 Introduction .............................. 1
   1.1 Overview .............................. 1
   1.2 Contributions and Outline of The Thesis .... 4

2 Background Review ...................... 9
   2.1 Natural Language Understanding ............ 9
   2.2 Deep Learning .......................... 13
      2.2.1 Neural Networks ...................... 14
      2.2.2 Backpropagation Algorithm .......... 17
      2.2.3 Stochastic Gradient Descent ......... 18
   2.3 Word Embedding ........................ 19

3 Document Summarization ............ 23
   3.1 Window-based Sentence Representation .... 25
      3.1.1 Introduction ......................... 25
      3.1.2 Extreme Learning Machine .......... 26
      3.1.3 The Proposed Model .................. 28
      3.1.4 Performance Evaluation ............. 31
      3.1.5 Summary ............................. 34
   3.2 Multi-view Convolutional Neural Networks .. 36
      3.2.1 Introduction ......................... 36
      3.2.2 Related Works ......................... 38
      3.2.3 The Proposed Model .................. 40
      3.2.4 Experimental Studies and Discussions .. 48
3.2.5 Summary ................................................. 57

4 Sentiment Analysis ........................................ 61
   4.1 Introduction ........................................... 61
   4.2 Overview of Related Works ............................. 63
      4.2.1 Convolutional Recurrent Neural Networks ...... 63
      4.2.2 Deep Neural Networks for Sentiment Classification ... 64
   4.3 The Proposed Model .................................... 65
      4.3.1 Standard RNN ..................................... 65
      4.3.2 LSTM ............................................. 67
      4.3.3 GRU ............................................. 68
      4.3.4 Comprehensive Attention Recurrent Neural Networks ... 69
   4.4 Experimental Studies and Discussions .................. 72
      4.4.1 Dataset ......................................... 72
      4.4.2 Parameter Settings ................................ 73
      4.4.3 Experiment Results ............................... 74
   4.5 Summary ............................................... 76

5 Sentence Modeling .......................................... 79
   5.1 Introduction ........................................... 79
   5.2 Related Works ......................................... 82
   5.3 The Proposed Model .................................... 83
      5.3.1 Word Embedding .................................. 84
      5.3.2 Convolution Layer ................................ 84
      5.3.3 Attention Pooling ................................. 85
      5.3.4 Parallel CNNs .................................... 87
      5.3.5 Softmax Classifier ................................ 88
   5.4 Experimental Results and Discussions ................. 89
      5.4.1 Datasets ......................................... 91
      5.4.2 Comparison Systems ............................... 91
      5.4.3 Parameter Settings ............................... 93
      5.4.4 Comparison of Classification Accuracy ............ 94
      5.4.5 Sensitivity Analysis .............................. 96
      5.4.6 Comparison of Pooling Strategies ................ 98
      5.4.7 Visualization of Sentence Representation Space .... 99
   5.5 Summary ............................................... 100

6 Demography Prediction ..................................... 103
   6.1 Introduction ........................................... 103
   6.2 Related Works ......................................... 106
   6.3 Methodology ........................................... 107
      6.3.1 Problem Formulation ............................... 108
      6.3.2 Website Representation ............................ 108
      6.3.3 Aggregation ....................................... 111
6.3.4 Classification ................................. 112
6.4 Datasets ........................................... 112
  6.4.1 Demographic Attributes ...................... 112
  6.4.2 Data Preparation .............................. 112
  6.4.3 Web Crawling .................................. 114
6.5 Experiment Analysis ............................... 115
  6.5.1 Experiment Setup .............................. 115
  6.5.2 Performance of Different Tf-idf Weighting Schemes .... 116
  6.5.3 Performance of Different Aggregation Methods ......... 118
  6.5.4 Performance of Different Classification Models ........ 118
  6.5.5 Comparison of Website Representation Methods ........ 119
  6.5.6 Study on Overall Framework .................. 121
6.6 Summary ............................................. 121

7 Conclusions and Future Work ......................... 123
  7.1 Conclusions .................................... 123
  7.2 Future Work ..................................... 127

A Backpropagation Procedure ......................... 131

Author’s Publications ............................... 133

Bibliography ........................................... 135
Abstract

With the explosive growth of Internet and computing technology, human beings are confronted by a great amount of unstructured text data. The need to extract useful knowledge from the data also grows. Researchers in the natural language processing community have delivered many marvelous technologies for various applications, such as information retrieval, machine translation, sentiment analysis, etc. Traditional methods usually rely on rigid language assumptions and require great efforts and time to be devoted to manual feature engineering. The research goal of this thesis is to develop machine learning models that can automatically learn semantic representations from texts with few or no human interventions. The models proposed in this thesis can induce effective representations for sentences or documents which are used to solve high-level language understanding tasks. The models are shown in four main chapters in this thesis according to the tasks they are addressing. The first task is document summarization which is addressed by two new approaches; after that, another two innovative algorithms are proposed for sentiment analysis and sentence modeling respectively; at last, one model is developed for human demography prediction. However, the models are never limited to these applications but can easily generalized to diverse natural language understanding tasks. The core of all the models lies in learning good semantic representations.

Document summarization is aimed at generating a brief summary for a long document or a set of documents. In this thesis, the task is transformed into a regression problem which ranks sentences by saliency scores. Methods are explored to represent sentences as vectors so as to obtain scores of sentences by a regressor. The first model leverages on word embedding to represent sentences so as to avoid the intensive labor of feature engineering. A new technique, termed window-based sentence representation, is proposed and achieves satisfactory summarization performance compared with baseline methods. However, the representation power is still weak because of its simple structure. To improve the representation capability,
we employ deep learning algorithms and develop an innovative variant of the convolutional neural network, namely multi-view convolutional neural network which can obtain the features of sentences and rank sentences jointly. The performance of the new model is evaluated on five benchmark datasets and demonstrates better performance than the state-of-the-art approaches.

The second natural language understanding task addressed in this thesis is sentiment analysis which has been applied to recommender systems, business intelligence and automated trading, etc. A new architecture termed comprehensive attention recurrent model is developed to access comprehensive information contained in sentences. The model employs the recurrent neural network to capture the past and future context information and the convolutional neural network to access local information of words in a sentence. Empirical results on large-scale datasets demonstrate that the new architecture effectively improves the prediction performance compared with standard recurrent methods.

The sentence modeling problem is at the core of many natural language processing tasks whose main objective is to learn good representations for sentences. Actually the objective of the thesis is to learn good semantic representations for texts. Therefore, this task lies at the core and is the foundation of the other three tasks addressed in this thesis. One innovative model combining the bidirectional long-term short memory and convolutional structures is developed for the problem. A new pooling scheme for the convolutional neural networks, which better retains significant information than the popular max pooling method, is proposed by leveraging on attention mechanism. The model achieves state-of-the-art performance on seven benchmark datasets for text classification.

At last, a simple but effective document representation approach is designed for predicting demographic attributes of web users based on their browsing history. I put this task at the last position because it is a practical application of natural language understanding. The new representation approach exploits word embedding and term frequency-inverse document frequency weighting scheme and owns both the power of word embedding capturing semantic and syntactic information and the statistical nature of term frequency-inverse document frequency. Experimental results demonstrate that the new representation method is more powerful than other feature representation methods including sophisticated deep learning models for this task.
List of Figures

1.1 An example of constructing dependency tree over a sentence. The abbreviations in color boxes are part-of-speech tags, e.g., NN for normal noun and VBZ for third person singular verb. The abbreviations connecting two words denote dependency relations between the two words, e.g., nsubj for nominal subject. 3

1.2 Illustration of differences in learning framework between models proposed in this thesis and traditional natural language processing models. Models in this thesis skip manual feature engineering module and require no rigid assumptions compared with traditional methods. 3

2.1 Illustration of the relationship between natural language processing and natural language understanding. The figure is adapted from the tutorial given by Bill MacCarteney in ACM SIGAI Bay Area Chapter Meeting in 2014. 10

2.2 Illustration of the basic form of neural networks with inputs and only one output. 15

2.3 Deep neural network with \( L \) hidden layers. The bias nodes have been ignored for clarity. 16

2.4 Graphical illustration of the function \( y = x^2 \). The pink arrow is the negative gradient of the function at \( x = 1 \). 17

2.5 A toy example of the space proximity of word vectors in a 2-D reduced space. 21

3.1 Network structure of extreme learning machine. 26

3.2 CNN Architecture. 41

3.3 MV-CNN framework. 44

3.4 Rouge-2 scores (%) on DUC2006 by effect of the number of Convolutional Layers. 54

3.5 Rouge-1 scores (%) comparison by impact of complementary and consensus principles. 55

3.6 Rouge-2 scores (%) comparison by impact of complementary and consensus principles. 55

3.7 Rouge-SU4 scores (%) comparison by impact of complementary and consensus principles. 56
3.8 Rouge scores (%) comparison by impact of sentence position embedding. From top to bottom are Rouge-1, Rouge-2 and Rouge-SU4 successively. ................................. 57

4.1 Illustration of a standard RNN model (left) and a BRNN model (right). The weights of the BRNN are left out for clarity. ............... 66

4.2 Illustration of (A) the LSTM unit and (B) the GRU unit. Weight matrices are left out for clarity. ............................. 68

4.3 The architecture for the Comprehensive Attention Recurrent Neural Network. .................................................. 70

4.4 Illustration of ‘same’ convolution graph. The window size of the filter is 3 and weights are shared across different windows. The arrows with the same colors represent the same weight values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this thesis.) ............. 71

4.5 F1-score (%) comparison by impact of initialized word embedding. From top to bottom are GRU, LSTM and RNN successively. ....... 76

5.1 Architecture of APCNN. The BLSTM is incorporated in the pooling stage to obtain attention weights which are used to combine local representations (only one local representation is shown in the figure for conciseness) into the sentence representation. The dashed line indicates that the intermediate sentence representation will also be used as input to the softmax layer at the testing phase. .......... 83

5.2 Illustration of parallel convolution procedure. It involves $H$ parallel convolution layers with varying convolution window sizes and each layer has $k$ filters. Each convolution layer is followed by an attention pooling layer to generate a distinct sentence representation. The $H$ distinct representations are then concatenated to form the final representation vector. ........................................... 88

5.3 Sensitivity analysis of (A) filter window size, (B) number of feature maps, (C) dropout rate, and (D) $l_2$ norm constraint. The plots depict the percent changes of accuracy compared with basic configurations for the four parameters respectively on seven datasets. The basic configurations are set as (3, 100, 0.5, 3). In (C), the zero value of dropout rate means that no dropout is used. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.) ............... 97

5.4 Accuracy comparison by the impact of pooling strategies. Five pooling strategies are compared on six datasets. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.) .......... 99
5.5 Class distribution visualization of the sentence representation space of SST-2 dataset. The green dots and red triangles represent the negative and positive classes respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.) .......................... 100

5.6 Enlarged views of the visualization of the sentence representation space of SST-2 dataset. (a) Enlarged view from green area of Figure 5.5. (b) Enlarged view from red area of Figure 5.5. The original dots and triangles are replaced by the represented sentences. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.) ....... 101

6.1 General view of predicting user’s demographics based on browsing history ................................................................. 104

6.2 The flow chart that explains the general methodology .................. 108

6.3 RMSE results by tag combinations on website demography tendency prediction. The terms h, p, a, i, v are used to represent headlines, paragraphs, link words, image descriptions, and visible texts respectively for brevity. Tag combination xy means content associated with the corresponding tags x and y are used. For instance, hp stands for using both headlines and paragraph content. .................. 115

6.4 Accuracy comparison results on gender prediction using various tf-idf weighting scheme on Demo20 and Demo100 datasets. The first letter in the scheme notation denotes the name of the variant of term frequency weight. The second letter accounts for the variant of inverse document frequency weight. .......................... 116

6.5 Accuracy comparison results using various aggregation methods. The orange (LA), turquoise (WA), and navy (SA) bars denote log, weighted, and simple average methods respectively. The upper two sub-figures show results on Demo20 dataset while the bottom figures on Demo100. The left two sub-figures are results on gender attribute prediction while the right two on age. .................. 117

6.6 Accuracy comparison results using aggregation-classification and regression-aggregation framework. The upper two are age and gender prediction on Demo100 dataset, while the bottom two are prediction results on Demo10. .......................... 121
List of Tables

3.1 Characteristics of Data Sets ................................................. 32
3.2 System Comparison Results on DUC2006 .......................... 33
3.3 System Comparison Results on DUC2007 .......................... 33
3.4 Characteristics of Data Sets ................................................. 49
3.5 Rouge-1 score comparison results (%) .......................... 51
3.6 Rouge-2 score comparison results (%) .......................... 51
3.7 Rouge-SU4 score comparison results (%) .......................... 51
3.8 One manual summary and an automatic summary generated by the proposed model on the topic “An Air France Concorde crash” ................................................. 58
4.1 Classification accuracy (%) comparison results on three sentiment tasks ................................................. 75
4.2 Performance comparison results (%) on IMDB dataset ............ 76
5.1 Descriptions of related pooling strategy .......................... 82
5.2 Characteristics of datasets ................................................. 90
5.3 Characteristics of the comparison systems ........................ 93
5.4 Parameter settings of different datasets .......................... 94
5.5 Classification accuracy results of APCNN against other approaches on benchmark datasets ................................................. 95
6.1 Variants of tf weight ...................................................... 110
6.2 Variants of idf weight ...................................................... 110
6.3 Data distribution over gender and age of Demo20 ................. 113
6.4 Data distribution over gender and age of Demo100 ............... 113
6.5 Gender prediction accuracy of SVM against other baseline classifiers (%) ................................................. 119
6.6 Comparison of the percentage (%) accuracy of different websites representation methods ................................................. 120
7.1 Comparison of all models in this thesis: Composition, Classifier/Regressor, and Task addressed. The “RNN” in the third algorithm can be replaced by “LSTM” and “GRU”. ................................................. 125
Abbreviations

DT  Part-of-speech: Determiner
IN  Part-of-speech: Preposition/Subordinating Conjunction
NN  Part-of-speech: Common Noun
NNS Part-of-speech: Plural Noun
RB  Part-of-speech: Adverb
VBZ Part-of-speech: Verb (Third Person Singular)
advmod Dependency Relation: Adverb Modifier
case Dependency Relation: Case-marking, Prepositions, Possessive
compound Dependency Relation: Compounding
det Dependency Relation: Determiner
nmod Dependency Relation: Nominal Modifier
nsubj Dependency Relation: Nominal Subject

AI  Artificial Intelligence
CPU Central Processing Unit
DUC Document Understanding Conferences
GPU Graphics Processing Unit
MDS Multi-Document Summarization
NLG Natural Language Generation
NLP Natural Language Processing
NLU Natural Language Understanding

CR  Customer Review Dataset
IMDB IMDB Movie Review Dataset
MPQA Multi-Perspective Question Answering Dataset
MR  Movie Review Dataset
SST Stanford Sentiment Treebank Dataset
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJ</td>
<td>Subjectivity Dataset</td>
</tr>
<tr>
<td>TREC</td>
<td>Text Retrieval Conference Dataset</td>
</tr>
<tr>
<td>BoW</td>
<td>Bag of Words</td>
</tr>
<tr>
<td>BRNN</td>
<td>Bidirectional Recurrent Neural Network</td>
</tr>
<tr>
<td>CBoW</td>
<td>Continuous Bag of Words</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CRF</td>
<td>Conditional Random Field</td>
</tr>
<tr>
<td>DCNN</td>
<td>Dynamic Convolutional Neural Network</td>
</tr>
<tr>
<td>ELM</td>
<td>Extreme Learning Machine</td>
</tr>
<tr>
<td>GD</td>
<td>Gradient Descent</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Programming</td>
</tr>
<tr>
<td>LSA</td>
<td>Latent Semantic Analysis</td>
</tr>
<tr>
<td>LSE</td>
<td>Least Squares Estimate</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-term Memory</td>
</tr>
<tr>
<td>MMR</td>
<td>Maximal Marginal Relevance</td>
</tr>
<tr>
<td>MNB</td>
<td>Multinomial Naive Bayes</td>
</tr>
<tr>
<td>MRW</td>
<td>Markov Random Walk</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>Matrix-Vector Recursive Neural Network</td>
</tr>
<tr>
<td>NB</td>
<td>Naive Bayes</td>
</tr>
<tr>
<td>one-hot CNN</td>
<td>One-hot Vector Convolutional Neural Network</td>
</tr>
<tr>
<td>P.V.</td>
<td>Paragraph Vector</td>
</tr>
<tr>
<td>RAE</td>
<td>Recursive Auto-encoder</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>RBM</td>
<td>Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>RecursiveNN</td>
<td>Recursive Neural Network</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>RNTN</td>
<td>Recursive Neural Tensor Network</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic gradient descent</td>
</tr>
<tr>
<td>SLFN</td>
<td>Single-hidden-layer Feedforward Neural Network</td>
</tr>
<tr>
<td>SSR</td>
<td>Summation Sentence Representation</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>Term Frequency-Inverse Document Frequency</td>
</tr>
<tr>
<td>AP-CNN</td>
<td>Attention Pooling-based Convolutional Neural Network</td>
</tr>
<tr>
<td>CA-LSTM</td>
<td>Comprehensive Attention Long Short-term Memory</td>
</tr>
<tr>
<td>CA-RNN</td>
<td>Comprehensive Attention Recurrent Neural Network</td>
</tr>
<tr>
<td>CA-GRU</td>
<td>Comprehensive Attention Gated Recurrent Unit</td>
</tr>
<tr>
<td>MV-CNN</td>
<td>Multi-View Convolutional Neural Network</td>
</tr>
<tr>
<td>TF-IDF_word2vec</td>
<td>Word Embedding with TF-IDF Weighting Scheme</td>
</tr>
<tr>
<td>WSR</td>
<td>Window-based Sentence Representation</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

With the explosive growth of Internet and computing technology, human beings are confronted by a great amount of unstructured text data. The need to extract useful knowledge from the data also grows. Natural language processing plays a significant role in various areas such as information retrieval, machine translation, sentiment analysis, etc. In recent years, people have witnessed a great many interesting and powerful applications driven by natural language processing, like Siri, Google translation, and IBM’s Watson, etc. With similar significance as computer vision, natural language processing is a key engine to achieve real artificial intelligence.

Researchers in the natural language processing community have delivered many marvelous technologies to extract knowledge from unstructured text data. Traditional methods usually rely on rigid language assumptions and require great efforts and time to be devoted to manual feature engineering [1]. The two shortcomings have hampered the power of natural language processing techniques.

- Rigid Language Assumptions: In natural language processing community, researchers usually make some assumptions before processing texts. Most methods assume that the texts they are addressing follow rigid grammatical structures. The assumption is reasonable for most formal texts, however texts appearing in social media like Facebook and Twitter are usually casual and do not follow rigid grammars. Another limit of the assumption lies in spoken
language, which is confronted by voice-driven assistants like Siri and Google Now. As personalized services become one of the most promising directions of artificial intelligence, it is necessary to cast off the limit of rigid language assumptions when attempting to understand language. Furthermore, distinct languages follow distinct grammatical structures. The methods developed based on rigid language assumptions can hardly be adapted to other languages. The models proposed in this thesis do not abide by certain grammatical assumptions and understand language at the semantic level. They may be easily adapted to other languages.

- Feature Engineering: It is widely accepted that the success of machine learning models depends crucially on feature representations because they can entangle the explanatory factors of variation behind the data [2]. Traditional machine learning methods rely heavily on syntactic features like part-of-speech tags, parse tree features and dependency relations. Figure 1.1 gives an example of constructing dependency tree over a sentence. The dependency tree demonstrates the grammatical structure of a sentence and extracts part-of-speech features (e.g., (success, NN), (depends, VBZ)) and dependency relation features (e.g., nsubj(depends, success), nmod(success, models)). Syntactic information is significant for understanding natural language, however it is never a trivial task to parse sentences due to the ambiguity of natural language. Other popular features used by natural language processing models include name entities (e.g., person, location or organization) and a lexical database such as WordNet. The development of all the features requires a long time and great efforts and demands expert knowledge in linguistics. Even more efforts have to be done to select features because many designed features are useless or may even detriment the learning performance. Furthermore, the power of a learning algorithm integrating these features has to depend on the reliability of the selected parsing method, which severely limits the potential of the algorithm.

Many researchers have attempted to address the two limitations. Following their efforts, the research goal of this thesis is to develop machine learning models that can automatically learn semantic representations from texts with few or no human.

\footnote{The Stanford CoreNLP Natural Language Processing Toolkit [3] is used to build the parse tree.}
Chapter 1. Introduction

Figure 1.1: An example of constructing dependency tree over a sentence. The abbreviations in color boxes are part-of-speech tags, e.g., NN for normal noun and VBZ for third person singular verb. The abbreviations connecting two words denote dependency relations between the two words, e.g., nsubj for nominal subject.

Figure 1.2: Illustration of differences in learning framework between models proposed in this thesis and traditional natural language processing models. Models in this thesis skip manual feature engineering module and require no rigid assumptions compared with traditional methods.

interventions. The models proposed in this thesis can overcome the two shortcomings. Rigid language assumptions are no longer necessary, and few or no manual efforts are needed in feature engineering. The models can take in raw texts directly and generate semantic representations of texts and output results for various language understanding tasks. The differences in learning framework between new models and traditional natural language processing models are depicted in Figure 1.2. The ideas of deep learning and word embedding are taken advantage of to break the limits. Deep learning is able to overcome the second shortcoming because it can automatically learn features from raw data. Word embedding is a useful tool to tackle the first challenge because it can capture both semantic and syntactic information without complicated grammatical analysis. The basic concepts of deep learning and word embedding will be introduced in Chapter 2.

The text representations induced by my models are used to solve high-level language understanding tasks. Natural language understanding is a subfield of natural language processing, and tackles the most challenging task in natural language processing, namely enabling machines to understand the semantic meaning contained in language. It is a difficult task to handle unstructured inputs because they are governed by poorly defined and flexible rules. Natural language understanding aims at converting unstructured data into a structured form that a machine can understand and act upon. With an explosion of interest in building machines which
are capable of understanding human language and the emergence of powerful machine learning tools, natural language understanding has stepped into the center of the stage of natural language processing community. Natural language understanding plays a key role in many widely used applications, such as voice-driven assistants (i.e., Siri and Google Now), question answering (i.e., Google search and IBM’s Watson), and even automated trading.

In this thesis, I address four natural language understanding tasks and propose several new machine learning models to achieve the state-of-the-art learning performance in the tasks. The models are shown in four main chapters according to the tasks they are addressing. The first task is document summarization which is addressed by two new approaches; after that, another two innovative algorithms have been proposed for sentiment analysis and sentence modeling respectively; at last, one model has been developed for human demography prediction. However, the models are never limited to these applications but can be easily generalized to diverse natural language understanding tasks. The core of all the models lies in learning good semantic representations.

All the new models in this thesis exploit the idea of word embedding. Word embedding is also called distributed word vector or word vector which represents a word as a vector capturing the meaning of that word. Then, distinct machine learning models, mainly neural network-based models, are employed to obtain representation vectors of sentences or documents from word vectors, which conforms to the principle of compositionality that states the meaning of a longer expression (e.g. a sentence or a document) comes from the meaning of its constituents and the rules used to combine them [4]. The models do not demand the structures of texts in any rigid forms but are capable of discovering structures directly from training data. The biggest advantage of the models is that they can learn semantic representations automatically from texts with few or no manual feature design and selection efforts.

1.2 Contributions and Outline of The Thesis

In this thesis, I propose two new approaches for document summarization task and two new models for sentiment analysis and sentence modeling respectively. The
four methods are developed based on the word embedding technique and neural network-based models. I also design an innovative method for human demography prediction which also takes advantage of the word embedding technique. The models can induce effective feature representations for texts from raw input without any manual feature engineering. They are capable of discovering structures and semantics purely from training data. The models need no external interventions and linguistic expertise and thus are easy to implement. The new models achieve state-of-the-art performance in the tasks they are addressing. The idea of new models can be easily generalized to other natural language understanding tasks.

Chapter 2 reviews the background knowledge in this thesis from three perspectives, namely natural language understanding, deep learning, and word embedding. In particular, I introduce the history and difficulties of natural language understanding research, basic forms and optimization methods of deep learning models, and power and wide usage of the word embedding technique. The next four chapters demonstrate the new natural language understanding approaches according to the tasks they are addressing. At last, a conclusion Chapter 7 summarizes the thesis and envisions the future work.

Summary Chapter 3: Document Summarization

Document summarization is aimed at generating a brief summary for a long document or a set of documents. In this thesis, the task is transformed into a regression problem which ranks sentences by saliency scores. Methods are explored to represent sentences as vectors so as to obtain scores of sentences by a regressor. The first model leverages on word embedding to represent sentences so as to avoid the intensive labor of feature engineering. A new technique, termed window-based sentence representation [5], is proposed and achieves satisfactory summarization performance compared with baseline methods with significantly faster learning speed. However, the representation power of the first method is still weak because of its simple structure. To improve the representation capability, I employ a powerful deep learning model named convolutional neural network and develop an innovative variant termed multi-view convolutional neural network [6] which can obtain the features of sentences and rank sentences jointly. The new model is evaluated on five
benchmarks and demonstrates better performance than the state-of-the-art approaches. The contributions of the second model are shown as follows:

- Multi-view learning is applied to the convolutional neural network to enhance the learning capability of the standard convolutional neural network. The complementary and consensus principles of multi-view learning are fully exploited to improve the model learning performance. To the best of my knowledge, this is the first time that the combination of multi-view learning and convolutional neural network is used for document summarization.

- With the help of pre-trained word embedding, human feature engineering and selection are no longer needed for document summarization task. This makes our model much easier to be implemented.

- An innovative technique named sentence position embedding is developed to advance the learning capacity of the proposed model to an even higher level.

This chapter is based on the following papers: *Multi-document extractive summarization using window-based sentence representation* [5] and *Multi-view Convolutional Neural Networks for Multi-document Extractive Summarization* [6], each being the basis for one section.

**Summary Chapter 4: Sentiment Analysis**

The second natural language understanding task addressed in this thesis is sentiment analysis which has been applied to recommender systems, business intelligence and automated trading, etc. Three new models termed *comprehensive attention recurrent models* [7] are developed to access comprehensive information contained in sentences. The models employ recurrent neural networks to capture the past and future context information and convolutional neural networks to access local information of words in a sentence. Empirical results on large-scale datasets demonstrate that the new architecture effectively improves the prediction performance compared with standard recurrent methods. The contributions of this chapter are as follows:
Chapter 1. Introduction

• The models propose new architectures that can access comprehensive information, namely history, future and local context information of any position in a sequence, thus improves the representation capability.

• Our model can be trained end-to-end by using pre-trained word embedding and human feature engineering is no longer needed.

This chapter is based on the paper: Sentiment classification using Comprehensive Attention Recurrent models [7].

Summary Chapter 5: Sentence Modeling

The sentence modeling problem is at the core of many natural language processing tasks whose main objective is to learn good representations for sentences. Actually the objective of the thesis is to learn good semantic representations for texts. Therefore, this task lies at the core and is the foundation of the other three tasks addressed in this thesis. One innovative model termed attention pooling-based convolutional neural network [8] is developed for the problem by combining the bidirectional long short-term memory and convolutional structures. A new pooling scheme for the convolutional neural networks, which better retains significant information than the popular max pooling method, is proposed by leveraging on attention mechanism. The model achieves state-of-the-art performance on seven benchmark datasets for text classification. The contributions of this model are summarized as follows:

• A new pooling scheme termed attention pooling is proposed to retrieve the most significant information at the pooling stage.

• The combination of the bidirectional long short-term memory and convolutional structures enables the model to extract comprehensive information in sentences. This further improves the learning capacity because comprehensive context information is very significant for extracting semantics in sentences.

• The model needs no external interventions and requires no feature design and feature selection.
Summary Chapter 6: Demography Prediction

In this chapter, a simple but effective document representation approach is designed for predicting demographic attributes of web users based on their browsing history. The new representation approach exploits word embedding and term frequency-inverse document frequency weighting schemes and proves to be more powerful than other feature representation methods for this task. Deep learning neural networks are not used because I find they are not appropriate for the characteristics of the task. Actually, my approach performs much better than deep learning models in this application. The task is a practical application solved by natural language understanding techniques. The contributions of the new approach are listed as follows:

- The new representation method is unsupervised and requires no human annotations. The idea of employing the word embedding technique provides a new thought of solution for the task of predicting demographic attributes of web audience.
- Exhaustive experiments have been done to determine the web crawling protocol for browsing history analysis.
- Two benchmark datasets are created for demographic prediction task.

This chapter is based on the paper: *Targeted Advertising Based on Browsing History* [9], which is still under review.
Chapter 2

Background Review

Natural language understanding (NLU) stands in the center of stage in natural language processing (NLP) community nowadays because it is the key engine of many machine comprehension applications, such as Siri and IBM’s Watson. NLU has to learn from a great amount of unstructured data and extracts the underlying semantics. Traditional methods require designing features manually before employing learning models to make final predictions. The manual feature engineering procedure is usually time-consuming and tedious, and cannot guarantee to obtain the best feature representations. With the help of deep learning and word embedding, we can jointly learn good features and make final predictions. The two procedures are put together in a complete pipeline.

In this chapter, I review the history and difficulties of natural language understanding research, and give a brief introduction on how deep learning and word embedding work in extracting knowledge from unstructured text data. Basic forms of deep learning models and their optimization methods are shown. I also demonstrate why word embedding has almost become an indispensable tool in NLU.

2.1 Natural Language Understanding

NLU is a subtopic of NLP that endows comprehension power to machines so that machines can understand the semantic meaning in the language like the human being. The relationship between NLU and NLP is shown in Figure 2.1. We can
see that NLU focuses more on tasks that extract semantic information from language, such as sentiment analysis, relationship extraction, semantic parsing, etc., while NLP deals with a much larger scope. I put machine translation and question answering systems on the boundary of NLU because they not only involve understanding language but also have to produce meaningful phrases or sentences which is called natural language generation (NLG). It is usually believed that NLU is a much harder task than NLG. Actually, NLU and NLG are put together and addressed in one single system as sequence-to-sequence models [10–12] appear recently. NLU tasks range from understanding short commands issued to a search engine, to extremely complex endeavors like comprehending an entire novel.

Various attempts have been done to build NLU systems in academic community and industry throughout the years. The history of NLU can trace back to 1960s when Daniel Bobrow developed the first computer program (i.e., STUDENT) aiming at understanding natural language [13]. Another precursor in NLU field is Joesph Weizenbaum, one of the fathers of modern artificial intelligence, who wrote ELIZA which is an interactive program studying natural language communication between men and machines [14]. The early works employed the idea of pattern-matching with small rule-sets and could only tackle very easy tasks. In the 1970s and 1980s, researchers built more grounded NLU systems that are more linguistically rich and logic-driven [15–18]. However, these systems could only be applied to restricted applications. After the artificial intelligence (AI) winter of the late 1980s and early 1990s, the focus of NLP research community turned to statistics and NLU saw a decreasing interest in both academia and industry. Until recent years, NLU has become a hot research topic again with the popularity of machine
learning and wide discussion in AI. Researchers and scientists are taking advantage of machine learning to build very complicated NLU systems that tackle various applications with deep understanding although there are still a lot of difficulties to be addressed.

NLU is regarded as an AI-hard problem because natural language is very ambiguous due to its extremely rich forms and structures. The first difficulty lies in word-level ambiguity. We can hardly know whether the word “bank” refers to an institution where people keep their money or an area of ground along a river without context information. A sentence can also be ambiguous because it may be parsed in different ways. For example, a machine may have difficulty to understand the sentence “Michal is eating fish with chopsticks” because it may mean Michal is using chopsticks to eat fish or Michal is eating fish that has chopsticks when parsed differently. Another difficulty in understanding natural language is referential ambiguity. In the sentence “Jim is chatting with Tom. He is happy”, we don’t know exactly who is happy. There are even more difficulties in NLU, such as synonyms, hidden assumptions, sarcasm, etc. The very bottom questions like What is language, How is language generated, and Where are generation rules stored are out of the scope of this thesis. We leave those questions to professional linguists. In this thesis, machine learning methods are proposed to teach machines to understand natural language.

The core of machine learning is function approximation. In [19], Kyunghyun Cho regarded the problem of language understanding as a mathematical function approximation procedure. The NLU task boils down to figuring out the internal working mechanism of a function whose name is language. In machine learning, we are given a dataset of input and output pairs (e.g., movie reviews and their rating scores) and fit a function to well predict the output given an input. It must be emphasized that what matters is not how well the approximated function fits the training data, but how well the function fits the unseen data. This is the intrinsic nature of machine learning: learn from existing knowledge and generalize to new knowledge.

Let us suppose that we have a pair of input and output data, \( x \in \mathbb{I}^d \) and \( y \in \mathbb{O}^k \), where \( \mathbb{I} \) and \( \mathbb{O} \) are all possible input and output values respectively. Our goal is to learn a function \( f : \mathbb{I}^d \rightarrow \mathbb{O}^k \) so that we can get the output \( y \) given its corresponding input \( x \). The function \( f \) is usually defined as a parametric function
specified with a set of parameters $\theta$. As the function $f$ is constrained, the output of the function given input $x$ is usually not exactly $y$. Therefore, we must define a way to determine how well the function fits the input and output data. Let us denote $\hat{y}$ as the prediction output of the function $f$ given input $x$. We have:

$$\hat{y} = f(x; \theta)$$

(2.1)

Then we use the difference between the predicted output and real output, denoted as $D(y, \hat{y})$, to measure how well the function $f$ approximates the underlying mechanism. One simple form of the difference is the Euclidean distance between $y$ and $\hat{y}$:

$$D(y, \hat{y}) = \frac{1}{2} \|y - \hat{y}\|_2^2$$

(2.2)

However, the difference $D(y, \hat{y})$ can appear in many other forms. To find how well the function fits over the whole distribution, we want to minimize $D(y, \hat{y})$ for every input and output pair in the space $I \times O$. The data distribution $p(x, y)$ is used to define how likely that $x$ and $y$ are a pair. It can be used as the weight of the difference between $y$ and $\hat{y}$ that contributes to the overall difference. Then, the goal becomes finding a set of parameters $\theta$ that minimize the following loss function or cost function:

$$\mathbb{L}(\theta) = \mathbb{E}_{(x,y) \sim p_{data}} D(y, \hat{y})$$

$$= \begin{cases} \int_x \int_y p(x, y)D(y, \hat{y})dx dy, & x \text{ and } y \text{ continous} \\ \sum_x \sum_y p(x, y)D(y, \hat{y}), & x \text{ and } y \text{ discrete} \end{cases}$$

(2.3)

The above loss function is called expected loss function. The problem of minimizing expected loss function is that $\mathbb{L}(\theta)$ cannot be exactly computed because we can hardly know the data distribution $p_{data}$. In practice, we only have a finite set of data randomly drawn from the data distribution. The set of data is called training data: $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. Then, the expected loss function can be approximated with empirical loss function $\mathbb{\tilde{L}}(\theta)$ using Monte Carlo method:

$$\mathbb{L}(\theta) \approx \mathbb{\tilde{L}}(\theta) = \frac{1}{N} \sum_{n=1}^{N} D(y_n, \hat{y}_n)$$

(2.4)

The problem of understanding language has become finding the best set of parameters $\theta$ that minimize the empirical loss function. The parameter learning procedure
is called optimization in mathematic perspective. Researchers have proposed various optimization methods. In some cases like linear regression, we can find the optimal parameters in a closed-form equation. The approximation function of linear regression is given as: \( f(\mathbf{x}; \theta) = \mathbf{Wx} \), in which \( \theta \) contains only one weight matrix \( \mathbf{W} \). The difference equation is governed by Equation 2.2. The we can find the optimal \( \mathbf{W} \) as follows:

\[
\mathbf{W} = \mathbf{YX}^T(\mathbf{XX}^T)^{-1}
\]  

(2.5)

where

\[
\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N], \quad \mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]
\]

However, there is no closed-form solutions in most cases. In such scenarios, it is typical to use iterative optimization algorithms, among which gradient descent (GD) algorithm is the most widely used and one of the foundations of deep learning models. The GD algorithm will be introduced in next section.

\section{Deep Learning}

I introduce deep learning in this section because three out of five models proposed in this thesis are developed based on deep learning models. The other two do not directly employ deep learning models but exploit the word embedding technique which is inseparable from deep learning. Deep learning has almost become the synonym of machine learning nowadays and made real AI possible again. Deep learning is actually not a very new idea but another name of deep neural networks which have been around for many decades [20–23]. However, deep neural network-based models performed worse than shallow architectures until Hinton \textit{et al.} used restricted Boltzmann machines to pre-train neural networks [24] and proposed a fast learning algorithm for deep belief neural networks [25]. Later in [26], Vincent \textit{et al.} used autoencoders to initialize weights of neural networks by learning useful representations of original data which are further used to reconstruct the data. Convolutional neural network [27] is another significant tool that drive the development of deep learning. They learn multiple levels of representations of images, which is similar as human brains. The models use bottom layers to learn minuscule details like edges, and more layers to learn primitive shapes, and top layers to combine low-level representations to form objects. In NLP community,
neural language models [28–30] have outperformed traditional language models and greatly advanced the development of the area. In 2012, Hinton and his group achieved great successes in speech recognition [31] and computer vision [32] using deep neural networks. The successes started a new wave of enthusiasm in deep neural networks in AI community. In 2015, AlphaGo developed by Google’s Deep-Mind became the first computer program to beat human professional Go players. This ignited unprecedented interest in deep learning and AI, not only in academia and industry but also in general public.

The most commonly accepted reasons why the old deep neural network structures obtain state-of-the-art performance are much larger datasets and greater computing power. In the era of “Big data”, researchers can easily and cheaply collect datasets that are necessary to train complex models. The powerful multi-core central processing units (CPU) and graphics processing units (GPU) together with mature parallelized computing technologies greatly cut down the time and cost required by the large number of computations in neural networks. The other reasons include the proposal of many innovative optimization techniques, such as various stochastic gradient descent (SGD) methods, dropout [33], batch normalization [34], and so on. I will introduce the basics of neural networks in the following section.

2.2.1 Neural Networks

In this section, I demonstrate the basics of neural networks with the simplest form of neural network. The structure in Figure 2.2 fully connects the inputs \( x \) and outputs \( y \) directly without any hidden layers. Let us assume the \( x \) and \( y \) are in real number space \( \mathbb{R}^d \) and \( \mathbb{R}^k \) respectively. Only one output neuron is shown in the figure for clarity and the connections between inputs and other output neurons are similar. The neural network approximates the following function:

\[
y_i = f(W^T_i x + b_i) \tag{2.6}
\]

where \( W \in \mathbb{R}^{d \times k} \) is the connection weight matrix, \( b_i \) is a bias value, and \( f(\cdot) \) is a nonlinear function. Therefore, we can see that the approximation function is the weighted sum of the inputs adding a bias (the green neuron in Figure 2.2), followed by an activation function (the cyan neuron in Figure 2.2) which limits the amplitude of the output of the neuron. The activation function can be as simple as
identity function $f(x) = x$, but in most cases nonlinear ones like sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

or hyperbolic tangent function:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The sigmoid function squashes values to the $[0, 1]$ interval and the hyperbolic tangent function to the $[-1, 1]$ interval. Other popular activation functions include rectified linear unit $f(x) = max(0, x)$ which addresses the vanishing gradient problem and hard tanh function which is faster than tanh to compute.

When we consider all the output neurons, we get the approximation function that maps input space to output space:

$$y = f(W^T x + b)$$

where $b \in \mathbb{R}^k$ is a bias vector.

The above structure does not have any hidden layers. We can add hidden layers between input nodes and output nodes to form deep neural networks as shown in Figure 2.3. Each hidden layer forms a feature extraction layer that extracts the interactions among previous layer. The output $y$ of the deep neural networks given
The input $\mathbf{x}$ is calculated as follows:

$$
\mathbf{h}^{(1)} = f(\mathbf{W}^{(1)^T} \mathbf{x} + \mathbf{b}^{(1)}) \\
\cdots \\
\mathbf{h}^{(L)} = f(\mathbf{W}^{(L-1)^T} \mathbf{h}^{(1)} + \mathbf{b}^{(L-1)}) \\
\cdots \\
\mathbf{y} = f(\mathbf{W}^{(L)^T} \mathbf{h}^{(L)} + \mathbf{b}^{(L)})
$$

There are totally $L$ hidden layers. We use a superscript to denote the number of layer. The term $\mathbf{h}^{(l)}$ denotes the hidden representations of the $l$th layer, $\mathbf{W}^{(l)}$ denotes the weight matrix connecting the $(l-1)$th and $l$th layer, and $\mathbf{b}^{(l)}$ denotes the bias vector of $l$th layer. The input is regarded as the 0th layer and the output as the $(L+1)$th layer.

The above equation describes the feed-forward process of deep neural networks. We define a set of parameters $\mathbf{\theta} = [\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \cdots, \mathbf{W}^{(L)}, \mathbf{b}^{(L)}]$. If an optimal set of parameters exists, we can obtain the real corresponding output using the model given any input. However, it is almost impossible to find the optimal parameters in most cases. We can only attempt to find the solution that is as good as possible. The procedure of finding the optimal parameter set is called optimization in mathematics. As mentioned in Section 2.1, most problems have no closed-form optimization solutions like Equation (2.5). In such situations, the backpropagation algorithm can be used.
Chapter 2. Background Review

2.2.2 Backpropagation Algorithm

The backpropagation algorithm [21] is a technique following the inverse directions of the feed-forward process to calculate loss gradients for parameters in the network. It uses chain rule of differentiation and efficiently reuse the same parts of gradient computation. Before exploring the backpropagation algorithm, I demonstrate the reason why we need to calculate gradients. The gradient descent (GD) is a first-order iterative optimization algorithm that finds a local minimum of a function by moving the parameters in the opposite direction of the gradient of the function at the current point. This is based on the observation that the function decreases fastest in the direction of the negative gradient. It is illustrated in Figure 2.4.

One important issue in GD is the step size of moving the parameters. If the step size is too large, we may overshoot and miss the minimum point at $x = 0$. If too small a step is taken, more steps and time have to be taken to reach the minimum point. The step size, called learning rate, is a most significant hyperparameter of GD algorithm. The (almost) optimal parameters of the network can be found by minimizing the loss function using GD:

$$\theta \leftarrow \theta - \eta \nabla L(\theta) \quad (2.11)$$

where $\nabla L(\theta)$ is the gradient of loss function and $\eta$ the learning rate.

After introducing the GD algorithm, let us come back to the backpropagation algorithm. In order to illustrate the working mechanism of the backpropagation algorithm, I derive backpropagation procedure for the toy neural network depicted
in Figure 2.2. Its approximation function is Equation (2.9). We make a further simplification assuming that the output contains only one perceptron, a.k.a, \( k = 1 \). The detailed backpropagation procedure is in Appendix A.

The above calculation procedure can be easily applied to neural networks with multiple neurons and deep layers. When calculating the gradient with regard to \( W_{ij}^{(l)} \), the weight connecting the \( i \)th neuron in \((l-1)\)th layer and \( j \)th neuron in \( l \)th layer, we just need to know the product of local error signal at the \( j \)th neuron in \( l \)th layer (\( \delta_j^{(l)} \)) and the local input signal at the \( i \)th neuron in \((l-1)\)th (\( h_i^{(l-1)} \)). The local error signal \( \delta_j^{(l)} \) is essentially the error propagating backward from the \( j \)th neuron in the \( l \)th layer. When calculating the derivatives for lower layers, the already calculated derivatives for higher layers can be reused, making the backpropagation algorithm very efficient.

### 2.2.3 Stochastic Gradient Descent

The GD algorithm is the fundamental basis of many advanced optimization algorithms. However, it comes across with a problem when the dataset size becomes large. We see from Equation (2.4) that the expected loss (cost) \( L \) is the sum of per-sample costs of all data in the training set. Therefore, the computation of \( L \) becomes very expensive with the increasing size of the training set. A similar issue happens to the computation of its gradient \( \nabla L \) as well. In the era of “Big Data”, the dataset size can easily go up to millions or even billions.

Stochastic gradient descent (SGD) algorithm \([35, 36]\) is an efficient method addressing the above issue. Recall from Equation (2.4) that we use empirical cost \( \widetilde{L} \) to approximate the expected cost \( L \) based on the assumption that the training dataset is randomly drawn from the actual data distribution \( p_{\text{data}} \). It is natural to use fewer number of samples to approximate the expected cost as long as the assumption holds:

\[
L(\theta) \approx \widetilde{L}_{\mathcal{D}}(\theta) = \frac{1}{M} \sum_{m \in \mathcal{D}} D(y_m, \hat{y}_m) \tag{2.12}
\]

where \( \mathcal{D} \) is a much smaller subset of the training set with size \( M \) and \( M << N \). The small subset is called minibatch.
The gradient of the loss function is also calculated using the minibatch:

$$\nabla L(\theta) \approx \nabla \tilde{L}_D(\theta) = \frac{1}{M} \sum_{m \in D} \nabla D(y_m, \hat{y}_m)$$  \hspace{1cm} (2.13)

By computing $\tilde{L}_D$ and $\nabla \tilde{L}_D$, we do not have to worry about the growing size of training data because they are independent of the size of training set. It becomes possible to train models with as many steps as we want. The SGD algorithm has been proven to help reach even better solutions compared with the GD algorithm [37]. Furthermore, the SGD algorithm fits parallel computing well which can further improve learning efficiency. In recent years, various forms of SGD algorithms have appeared to improve the optimization performance and efficiency. Some widely used algorithms include RMSprop [38], Adagrad [39], Adadelta [40], Adam [41], and so on.

### 2.3 Word Embedding

In this section, I give a brief introduction over the word embedding technique. All of the models proposed in this thesis employ the technique. It has drawn great attention in recent years because it can capture both semantic and syntactic information. It has been applied to many NLP applications such as named entity recognition [42], word sense disambiguation [43], parsing [44], tagging [45], sentiment analysis [46] and machine translation [47]. Word embedding is also called word vector or distributed word representation. I will use word embedding, word vector, word representation interchangeably in this thesis.

The majority of traditional NLP algorithms are based on counts over words. Each word is represented using a “one-hot” vector of the size of the vocabulary where the position that the word appears is 1 while the other positions are zeros. The representation method is called bag-of-words (BoW). The vector is extremely sparse resulting in the models easily overfit to the training data. As there are millions of words in one language, say English, the vocabulary size is so large that the model suffers from the problem of the curse of dimension. Furthermore, the models can hardly generalized to words out of vocabulary. The discrete word representations cannot capture the similarity between words (e.g., “football” and “soccer” usually mean the same thing), let alone the semantic relationships among words.
2.3. Word Embedding

The underlying principle of word embedding can find its shadow in the famous quote “You shall know a word by the company it keeps.” by Firth [48]. Each word is represented based on the context it appears. Many modern NLP methods employed this idea, such as the widely used topic modeling algorithms latent semantic analysis (LSA) [49] and latent Dirichlet allocation (LDA) [50]. The idea of distributed word representations was first proposed in [21, 22] and other early works including [51, 52]. In [28], Bengio et al. used neural language models to predict how likely a word is to occur and learn its representation vector based on its preceding word vectors. Since Bengio et al.’s work, a lot of researchers have explored the salient features of vector representation of words using neural networks [29, 30, 42, 43, 53, 54]. The most popular architectures of neural networks language models were proposed by Milkov et al. [29, 30]. They developed their methods in the framework of recurrent neural networks (RNN) and proposed two efficient word representation estimation models, namely continuous bag-of-words (CBoW) and Skip-gram. When predicting a word, the CBoW model not only considers the preceding words like Bengio et al.’s model but also takes the subsequent words into consideration. The CBoW model predicts the middle word given 2n surrounding words (n words to the left and n words to the right) and exhibits an interesting property which is that the learned word vector representations reflect the underlying structures of words. The property makes distributed word representations substantially attractive among NLP researchers. A similar property is found in the Skip-gram model as well. Contrary to the CBoW model, the Skip-gram model attempts to predict the 2n surrounding words given the middle word. For further details about the word embedding, please refer to the references [28–30, 42, 43, 53, 54].

The vector representations obtained by the word embedding technique are low-dimensional dense vectors which effectively solve the problem of the curse of dimension and sparsity of conventional BoW method. In the vector space, words with similar semantics lie close to each other. This is because similar words appear in similar contexts. For example, the two sentences “Cats have tails” and “Dog have tails” indicate that “cats” and “dogs” belong to the same family named animal. Let us have a look at Figure 2.5 which demonstrates the spatial proximity of word vectors in a 2-D reduced space. We can find that the words indicating dates lie close to each other while the words for locations form clusters. The vector representations of words can even preserve the semantic relationship. Figure 2.5
demonstrate the following relationship:

\[ \text{vec}(\text{France}) - \text{vec}(\text{Paris}) = \text{vec}(\text{China}) - \text{vec}(\text{Beijing}) \]  \hspace{1cm} (2.14)

This suggests that vector representations capture the fact that Paris and Beijing are capitals of France and China respectively.

With advances in deep learning techniques, distributed vector representations have become a common practice for representing words. The word embeddings can be trained in both supervised and unsupervised manner from scratch. After training word embedding matrices, a word can be represented by a dense vector as follows:

\[ x = Lw \]  \hspace{1cm} (2.15)

where \( w \in \mathbb{R}^V \) is the one-hot vector used in the BoW model, \( L \in \mathbb{R}^{d \times V} \) is a word-representation matrix, in which the \( i \)th column is the vector representation of the \( i \)th word in the vocabulary, and \( V \) is the vocabulary size.

We can also easily adopt off-the-shelf word embedding matrices as initial matrices to make better use of semantic and grammatical associations of words. Word2vec [30] and GloVe [54] are two most widely used pre-trained word embedding matrices. Previous research results demonstrate that different performance may result from using either matrix on different tasks, but with slight differences. In this thesis,
we use word2vec\(^1\) embeddings for all the models. The word2vec embeddings are obtained by training on 100 billion words from the Google News by using the Skip-gram method and maximizing the average log probability of all the words as follows:

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t)
\]  

(2.16)

\[
p(w_O|w_I) = \frac{\exp(v'^T_w v_{w_I})}{\sum_{w=1}^{V} \exp(v'^T_w v_{w_I})}
\]  

(2.17)

where \(c\) is the context window size, \(w_t\) is the center word, \(v_w\) and \(v'_w\) are the “input” and “output” vector representations of \(w\). More details can be found in [30]. I will show the power of word embeddings compared with conventional word-counting methods in the subsequent four chapters.

\(^1\)https://code.google.com/p/word2vec
Chapter 3

Document Summarization

Introduction

Automatic text summarization has been widely researched in recent years with the explosive growth of accessible information due to the rapid development of Internet and computing technology. It can mainly be divided into two categories, namely abstractive summarization and extractive summarization. The abstractive summarization methods involve sentence compression and reformulation which resemble human summarizers, but the linguistic processing procedure is so complicated that it is not easy to be implemented [55–57]. However, the extractive methods directly extract the most informative sentences in a document to form the final summary. Due to its simplicity, most works done in this area fall in the extraction-based category.

Extractive document summarization methods roughly fall into three main categories [58, 59]: 1) Methods based on sentence positions and article structure, 2) Unsupervised methods, 3) Supervised methods. For the first category, important sentences, such as those in introductory or concluding part, will be selected to fit into the summary. One famous method (LEAD) is proposed in [60] which simply extracts the leading sentences to summarize a document. However, such methods can only be applied to strictly structured documents like newswire articles. On the other hand, unsupervised methods rank sentences by salience scores which are estimated based on statistical and linguistic features and extract the top ones to constitute the summary. Maximal marginal relevance (MMR) [61] is a well-known
unsupervised method which obtains a trade-off between relevance and redundancy of sentences. The MMR framework is cast as an integer linear programming (ILP) in [62] to find the global optimal solution. Graph-based ranking methods play a significant role in unsupervised document summarization in recent years, such as the LexRank [63] and the TextRank [64]. They first build a graph of sentence similarities and then calculate the importance of a sentence by inspecting its links to all other sentences in the graph recursively. Some other unsupervised extraction-based document summarization methods include the Latent semantic analysis (LSA) [65], the Markov random walk (MRW) [66] and the submodularity-based methods [67]. In contrast to unsupervised methods, supervised approaches utilize a set of training documents together with their corresponding hand-crafted summaries to train a binary classifier which is used to predict whether a sentence should be included in the summary or not. In [68], a probabilistic approach termed conditional random field (CRF) is proposed to rank full sentences while researchers in [69, 70] train their models based on n-gram regression. Recently, some researchers [71, 72] measure the salience of both sentences and n-grams. The author of [71] takes advantage of support vector regression (SVR) and the approach in [72] is developed based on the recursive neural networks (RecursiveNN). The two new methods proposed in this chapter belong to the supervised category. The existing methods have some obvious drawbacks which can be summarized as:

- Most of the existing methods depend on hand-crafted features to represent sentences which result in intensive human labor.
- They can hardly capture semantic and syntactic information contained in documents simultaneously.
- Their summarization capabilities are not good enough.

This chapter aims to solve the problem of designing laborious hand-crafted features and enhance summarization capabilities. I leverage on the word embedding technique to represent sentences and propose easy-to-implement systems. We are among the earliest research groups using word embedding to reduce the labor of feature engineering in the research area of document summarization. This chapter is based on the following papers: Multi-document extractive summarization using window-based sentence representation [5] and Multi-view Convolutional Neural Networks for Multi-document Extractive Summarization [6], each being the basis
for one section. Before going to the two new document summarization methods, let us have a look at the problem formulation.

**Problem Formulation**

Extractive summarization is defined as the selection process of salient sentences in a corpus of documents to generate a brief summary which can best describe the subject. We denote the document corpus as $C = \{D_1, D_2, \ldots\}$, in which $D_i$ is the $i$th document in the corpus. Each document consists of a set of sentences. We include all the sentences in the corpus to form the candidate set $CS = \{s_1, s_2, \ldots, s_N\}$, where $s_n$ is the vector representation of the $n$th sentence in the corpus. The term $N$ is the number of distinct sentences in the corpus. The determination of the sentence representation has never been a trivial task. Most conventional methods require hand-crafted features which result in intensive human labor. Our proposed method leverages on the word embedding technique to obtain sentence representations. Next, neural network-based models are employed to assign salience scores to sentences in the candidate set. After assignment of salience scores, the sentences can be ranked so that the sentences in the candidate set with high scores will be selected as summary sentences. The selected sentences form the summary set $SS = \{s_1^*, s_2^*, \ldots, s_K^*\}$. Note that $K << N$ and $SS \subset CS$.

### 3.1 Window-based Sentence Representation

#### 3.1.1 Introduction

In Chapter 2.3, we have demonstrated that the word embedding technique represents words as low-dimensional dense vectors that can effectively capture the underlying structures of words. In practice, we care more about the phrase-level, sentence-level, and document-level representations because the meanings of these longer expressions are more meaningful. The simplest method to represent a sentence based on word embedding may be summing up all the word vectors in a sentence. However, this method will lose the word order information just like conventional bag-of-words (BoW) method. In this section, we propose a new technique
termed \textit{window-based sentence representation} (WSR) to represent sentences which can be implemented as easily as the summing method while preserving partial word order information. We employ the extreme learning machine (ELM) of \cite{huang2006extreme} to train our model. The training speed is hundreds or thousands of times faster compared with conventional machine learning models. We evaluate our method on two summarization benchmark datasets DUC2006 and DUC2007. The proposed method achieves significantly superior performance compared with baseline document summarization algorithms within a much shorter time. Before giving the details of the new WSR method, I will give a brief introduction of the extreme learning machine.

\subsection*{3.1.2 Extreme Learning Machine}

The extreme learning machine (ELM) and its variants have been widely used due to their salient features such as tuning free and extreme fast learning speed. The ELM was first proposed by Huang \textit{et al.} \cite{huang2006extreme} for a single-hidden-layer feedforward neural network (SLFN) with randomly chosen input weights and hidden nodes and analytically determined output weights. The ELM follows the idea of Frank Rosenblatts \cite{rosenblatt1961perceptron} and Ali Rahimi’s \cite{rahimi2007random} works. The relationship and differences between the ELM and other related works can be found in a recently published work \cite{zhou2016extreme}. The structure of the original ELM is shown in Figure 3.1.

Let us suppose that there are $K$ observations $(x_k, t_k)_{k=1}^K$ with $x \in \mathbb{R}^n$ and $t \in \mathbb{R}^m$. If a SLFN with $L$ hidden nodes can approximate the data with zero error, the
following equation holds:

$$\sum_{i=1}^{L} \beta_i G(x_i; a_l, b_l) = t_k, \quad k = 1, \ldots, K \quad (3.1)$$

where $\beta_l$ are the output weights and $G(x; a, b)$ is the activation function where $a, b$ are the parameters of hidden nodes. The term $G(x_k; a_l, b_l)$ represents the output of the $lth$ hidden node with respect to the $kth$ input data. The activation function can be both additive function and radial basis function (RBF). For the additive function, the term $a_l$ is the weight vector connecting the input layer to the $lth$ hidden node and the term $b_l$ is the bias of the $lth$ hidden node. For the RBF function, the terms $a_l$ and $b_l$ are the center and impact factor of the $lth$ hidden node respectively. Equation (3.1) can be written in the following compact form:

$$H\beta = T \quad (3.2)$$

where

$$H = \begin{bmatrix}
G(x_1, a_1, b_1) & \cdots & G(x_1, a_L, b_L) \\
\vdots & \ddots & \vdots \\
G(x_K, a_1, b_1) & \cdots & G(x_K, a_L, b_L)
\end{bmatrix}$$

$$\beta = \begin{bmatrix}
\beta_1 \\
\vdots \\
\beta_L
\end{bmatrix}^T \text{ and } T = \begin{bmatrix}
t_1 \\
\vdots \\
t_K
\end{bmatrix}^T$$

The term $H$ is called the hidden layer output matrix of the network, $T$ is the target matrix and $\beta$ is the output weight matrix. The parameters of the hidden nodes $(a_l, b_l)$ are randomly generated, and the only parameter needed to be calculated is the output matrix $\beta$, which can be analytically determined using the least squares estimate (LSE) as follows:

$$\beta = H^T T \quad (3.3)$$

where $H^T$ is the Moore-Penrose generalized inverse of the matrix $H$ which can be calculated through the orthogonal projection method, orthogonalization method and singular value decomposition (SVD) [77]. We can find that $\beta$ is obtained using a closed-form solution similar to Equation (2.5).

It should be highlighted that the number of hidden neurons is the only parameter to be specified in the ELM as opposed to other learning algorithms which usually have a lot of parameters to be tuned [78]. Recently, authors of [79, 80] demonstrate
an adaptive selection method of the number of hidden neurons based on a solid theoretical ground. Another remark of the ELM is that the activation function for additive nodes can be any bounded non-constant piece-wise continuous function and the activation function for RBF nodes can be any integral piece-wise continuous function. Since the original ELM, many variants [81, 82] have been proposed and applied to all kinds of applications. The recent developments and future trend of ELM algorithm can be found in [83]

3.1.3 The Proposed Model

Preprocessing

Our proposed method is a supervised approach. However, ready-made salience scores are not available for the training data. Therefore, we first pre-process the documents to obtain a salience score for each sentence. The document sets are given together with their reference summaries. We adopt the widely-accepted automatic summarization evaluation metric, ROUGE, to measure the salience. Rouge assesses the quality of an automatic summary by counting the overlapping units such as n-gram, common word pairs and longest common sub-sequences between automatic summary and a set of reference summaries. The n-gram ROUGE metric (ROUGE-N) can be computed as follows:

\[
ROUGE - N = \frac{\sum_{S \in \{RefSum\}} \sum_{n\text{-gram} \in S} Count_{match}(n\text{-gram})}{\sum_{S \in \{RefSum\}} \sum_{n\text{-gram} \in S} Count(n\text{-gram})}
\] (3.4)

where \( n \) stands for the length of the n-gram, \( Count(n\text{-gram}) \) is the number of n-grams in the set of reference summaries and \( Count_{match}(n\text{-gram}) \) refers to the number of n-grams co-occurring in a system-generated summary and the set of reference summaries. As ROUGE-1 \( (R_1) \) and ROUGE-2 \( (R_2) \) most agree with human judgment among all the ROUGE scores [84], we calculate each sentence \( s \)'s score as follows:

\[
y(s) = \alpha R_1(s) + (1 - \alpha) R_2(s)
\] (3.5)
Chapter 3. Document Summarization

We set the coefficient $\alpha = 0.5$ for equal weighting of the two scores. The scores $R_1$ and $R_2$ are obtained by comparing each sentence in multiple documents with corresponding reference summary.

Training

The focal point of the proposed method is to obtain sentence representations which are input features of the ELM model. With the aid of pre-trained word vectors, we can denote a sentence as $s = \{w_1, w_2, ..., w_l\}$, in which $w_i \in \mathbb{R}^d$ is the word embedding of the $i$th word in the sentence and $l$ is the length of the sentence. An intuitive and simple method to represent the sentence is to sum up all the word vectors as follows:

$$s = \sum_{i=1}^{l} w_i \quad (3.6)$$

However, this method completely loses the word order information in the sentence. In order to address the problem, we propose a window-based sentence representation (WSR) method which is able to partially preserve the order information. We represent a word taking into its context words instead of simply using its own word vector. The new word representation is denoted as the concatenation of word vectors within a context window. Supposing the window size is $m$, the new word representation, $v_i$, is given by

$$v_i = [w_{i-m+1}^T, ..., w_i^T, ..., w_{i+m-1}^T]^T \in \mathbb{R}^{md} \quad (3.7)$$

where the term $m$ should be assigned an odd number. For those words at the first or last positions, zero vectors are padded to guarantee the context window size. With the new representations of words, we can obtain the representation of a sentence as follows:

$$s = \sum_{i=1}^{l} v_i \quad (3.8)$$

This method is still very simple and easy to implement, but it is able to preserve partial order information compared with the summation method. In order to demonstrate the effectiveness of the context window, we compare our method with the summation method in the experiment section. The summation method is termed summation sentence representation (SSR) in this section.
After obtaining sentence representations, we can start to train the ELM model. We use RBF kernel in the ELM model, and the centers and impact factors of the kernel are all randomly generated. The only parameter needed to be tuned is the number of hidden neurons. It has been proved that the single-hidden-layer neural network is a universal function approximator as long as the number of hidden neurons is big enough. As the input feature number is \( md \), which is not a very large number, the number of hidden layer nodes can be selected slightly larger than \( md \). The detail training procedure can be found in Section 3.1.2. The cost function of the model is formulated as follows \[85\]:

\[
\sum_{i=1}^{N} \| y(s_i) - h(s_i)\beta \|^2_2 + \lambda \| \beta \|^2_2 \tag{3.9}
\]

where \( y(s_i) \) is the true score of the \( i \)th sentence obtained by Equation (3.5), \( h(s_i) \) is the corresponding hidden-layer output and \( \beta \) is the output weight matrix. The second term in the above equation is a regularization term in case of overfitting. The term \( \lambda \) controls the trade-off between the training error and overfitting. If \( \lambda \) is too large, the training error may be large. On the other hand, the model may overfit on testing data if \( \lambda \) is too small. Equation (3.9) can also be written in a compact form:

\[
\| Y - H\beta \|^2_2 + \lambda \| \beta \|^2_2 \tag{3.10}
\]

where \( H = [h(s_1), \cdots, h(s_N)]^T \) is the hidden layer output matrix for all the sentences and \( Y = [y(s_1), \cdots, y(s_N)]^T \) is the output score vector.

This equation minimizes both training error and norm of output weights because both small training error and small norm of weights contribute to the improvement of generalization performance according to Bartlett’s theory \[86\]. The output matrix \( \beta \) can be analytically determined using the following equation \[85\]:

\[
\beta = (H^TH + \lambda I)^{-1}H^TY \tag{3.11}
\]

As there are few parameters to tune, the training process is very fast. This is a strong advantage over other conventional machine learning models.
Testing

For the testing documents, features of all the sentences are calculated just like in the training documents. Next, they will be used as the input to the trained ELM model to obtain their salience scores. Afterwards, sentences in each multi-document cluster can be ranked according to their salience scores. Since a good summary should be not only informative but also non-redundant, we employ a sentence selection method in [87] to select from the ranked sentences. The selection method queries the sentence with the highest salience score and adds it to the summary if the similarity of the sentence with all the sentences already existing in the summary does not exceed a threshold. This sentence selection procedure is especially necessary for multi-document summarization because sentences extracted from different documents in one topic can be very similar. The selection process repeats until the length limit of the final summary is met.

3.1.4 Performance Evaluation

Datasets

The benchmark datasets from the Document Understanding Conferences\(^1\) (DUC) are used to evaluate our proposed document summarization method. The datasets are English news articles. In this section, we evaluate the performance of multi-document summarization on DUC 2006 and 2007 datasets. When evaluating on the DUC 2006 dataset, we train our model on DUC 2004 and 2007 datasets. And DUC 2004 and 2006 datasets are used to train the model for evaluation on the DUC 2007 dataset. The characteristics of the three-year datasets are given in Table 3.1. The table also contains the length limit of automatic summary for each dataset. The reference summaries for each set of documents are given together with the documents.

\(^1\)http://duc.nist.gov
Table 3.1: Characteristics of Data Sets

<table>
<thead>
<tr>
<th>Year</th>
<th>Clusters</th>
<th>Documents/Cluster</th>
<th>Length Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>50</td>
<td>10</td>
<td>250 words</td>
</tr>
<tr>
<td>2006</td>
<td>50</td>
<td>25</td>
<td>250 words</td>
</tr>
<tr>
<td>2007</td>
<td>45</td>
<td>25</td>
<td>250 words</td>
</tr>
</tbody>
</table>

**Evaluation Metrics**

For the evaluation of summarization performance, we employ the widely used ROUGE\(^2\) toolkit [88]. ROUGE has become the standard evaluation metric for DUC since 2004. ROUGE-1 (unigram) and ROUGE-2 (bigram) are used as measure metrics because they most agree with human judgment [89]. Rouge-SU4 is also a very popular metric because it conveys the readability of a candidate summary. Rouge-S is a co-occurrence statistic measuring the overlap of skip-bigrams between a candidate summary and a set of reference summaries. Skip-bigram is any pair of words in their sentence order. Rouge-SU is an extension of Rouge-S which adds unigram as counting unit as well to decrease the chances of zero scores where there is no skip-bigram overlap. Rouge-SU4 limits that word pairs at most 4 words apart can form skip-bigrams. The calculation formula of Rouge-SU4 can be easily obtained by replacing the n-gram of Equation (3.4) by skip-bigram. Therefore, ROUGE-1, ROUGE-2 and Rouge-SU4 scores are reported in our evaluation study. We set the length parameter “-l 250”. Rouge generates precision, recall and F-measure metrics. We employ F-measure, which is a combination of precision and recall metrics, in our experiment to compare the performances of different algorithms.

**Parameter Settings**

The embeddings of words used in our experiments are initialized using Skip-gram neural networks method proposed by Mikolov* et al. [29]. The dimension of each word vector is 300. The word vectors are already available and can be directly downloaded from the word2vec tool website\(^3\). Therefore, it is not necessary to spend a great amount of time to train and obtain word vectors. For the setting of ELM, the only parameter needed to be determined is the number of hidden

\(^2\)ROUGE-1.5.5 with options: -n 2 -x -m -2 -d -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d

\(^3\)https://code.google.com/p/word2vec
neurons, which we set as 1000 in this section. The python package on the ELM website\(^4\) is employed for our experiment. The threshold value of sentence selection for multi-document summarization is set as 0.4 and the context window size is set as 3. These parameters are determined using cross-validation and considering both learning performance and efficiency. The experiments are done on a 2.0GHz PC computer.

### Comparative Studies

We compare our proposed method with several baseline document summarization techniques which are briefly described as follows:

**Random**: Selects sentences randomly from the candidate set.

**Lead** [60]: First sorts all the documents chronologically and then extracts the lead sentence from each document to form the summary.

**LSA** [65]: Applies the singular value decomposition (SVD) on the term frequency matrix and then selects sentences with highest eigenvalues.

**DSDR** [90]: Uses sparse coding to represent each sentence as a non-negative linear combination of the summary sentences.

\(^4\)http://www.ntu.edu.sg/home/egbhuang/
Table 3.2 and Table 3.3 are the overall performance comparison results of our proposed method WSR against other baseline algorithms. The results of baseline methods are directly retrieved because the datasets and evaluation metrics are standard. It is obvious that the proposed method WSR outperforms all the other algorithms significantly. Among all the algorithms, the LSA gives the poorest performance on both datasets. The LSA exploits SVD on term-frequency matrix and supposes that sentences with highest eigenvalues are most informative sentences in a document corpus. The experiment results suggest that such hypothesis may not be true for human understanding. The table also shows that the Lead performs a little better than the Random method, which may be because that article writers like to put summary sentences at the beginning of the documents. The DSDR achieves the best performance on ROUGE-1 score except for the two word embedding-based methods. It proves that sparse coding helps to find informative sentences out of documents. The two word-embedding-based methods, SSR and WSR, show the best performance, proving that word embedding is effective in capturing semantic and syntactic information. Comparing the WSR with the SSR, we can conclude that the incorporation of a context window indeed helps document summarization. This is because the WSR preserves word order information compared with the SSR.

In addition, our method is very efficient with a very fast learning speed. The average running time of the DSDR on one document set is more than 4000s while the time of the WSR is only about 8s. Our algorithm is more than 500 times faster than the DSDR. This results from the ELM’s salient tuning-free features. Besides, the biggest advantage of our method is that it does not need hand-crafted features. The pre-trained word embedding has saved us an enormous amount of time and efforts, enabling us to avoid the intensive human labor of feature engineering.

3.1.5 Summary

In this section, a new technique termed window-based sentence representation has been successfully developed to obtain the features of sentences based on pre-trained word vectors. The use of word embedding enables us to avoid the intensive human labor of feature engineering. We employ a context window to preserve partial
word order information in sentences. The extreme learning machine is finally employed to assign scores to sentences. Our proposed framework does not require any prior knowledge and thus can be applied to various document summarization tasks with different languages, written styles and so on. We evaluate our proposed method on the DUC 2006 and 2007 datasets. The proposed method achieves superior performance compared with baseline document summarization algorithms with significantly faster learning speed.
3.2 Multi-view Convolutional Neural Networks

3.2.1 Introduction

The *window-based sentence representation* achieves satisfactory performance in summarizing documents. However, its representation power is still weak because it can only keep partial order information of sentences. To improve the representation capability, we employ deep learning algorithms and develop an innovative variant of the convolutional neural network to extract more representative features from sentences in this section.

In recent years, deep learning methods together with pre-trained word embedding have achieved remarkable results for various natural language processing (NLP) tasks. Deep learning models reduce the intensive labor of feature engineering because they can learn features from data automatically. Deep learning takes advantage of the increase in the amount of available computation and data and produces extremely marvelous results in speech recognition, visual object recognition, object detection and many other complicated tasks [31, 32]. More detailed information and recent developments in deep learning can be found in the review paper published in *Nature* [91].

Convolutional Neural Networks (CNN) was originally proposed for computer vision by LeCun et al. in [27]. It produces excellent results for computer vision tasks [92–94] and recently has been proven to be effective for various NLP tasks as well, such as POS tagging [95], sentence modeling [96], semantic embedding [97] and sentence classification [46], to name a few. As CNN has demonstrated great power in latent feature presentation [98, 99], we propose an enhanced CNN model termed *Multi-view Convolutional Neural Networks* (MV-CNN), to obtain the features of sentences and rank sentences jointly, and to build a novel summarization system. Our new approach is developed based on the basic CNN but incorporates the idea of multiview learning to enhance the learning capability. The new model leverages on pre-trained word embeddings to refrain people from intensive feature engineering labor. Furthermore, the word embeddings can efficiently help to improve the learning capability of the proposed model.
The basic CNN structure in this section adopts two consecutive convolution layers followed by a max-pooling layer to express sentence vectors. One problem of representing sentences is that users’ opinions towards sentences may vary tremendously. People tend to summarize documents very differently due to their perceptions and biases as well as different background knowledge and intentions. In order to address the issue, we incorporate the idea of multi-view learning. Multi-view learning is a paradigm designed for problems where data come from diverse sources or different feature subsets, a.k.a., multiple views. It employs one function to model a particular view and jointly optimizes all the functions to exploit the redundant views of the same input data and improve the learning performance [100]. Multi-view learning follows two significant principles, namely complementary and consensus principles. The complementary principle employs complementary information underlying multi-view data to comprehensively and accurately describe the data. The consensus principle aims at maximizing the agreement of distinct learners trained on multiple views. The new model follows a two-level complementary principle and the consensus principle. Our model theoretically and experimentally proves that multi-view is a feasible direction to improve CNN learning capability.

In the proposed model, multiple convolution filters with varying window sizes are used to complete the sentence representation and sentence ranking tasks. This technique has been demonstrated to be very useful for computer vision and natural language processing tasks. We find the technique conforms to the spirit of multi-view learning to some extent. The CNNs with different filter window sizes can be regarded as different summarizers. The saliency scores of sentences assigned by different CNNs will be combined appropriately to obtain the final scores, just like different summarizers negotiate with each other in order to determine the final summaries after getting their summaries independently. Furthermore, we not only combine the final scores rated by each distinct-filter-size CNN, but also concatenate sentence representations acquired by each CNN before calculating the saliency scores of sentences and use the new sentence representations to analyze the sentences. This is analogous to the collaborative process that different summarizers exchange one another’s opinions first and perform the summarization task together. The above procedure forms a two-stage complementary principle. The consensus principle is also used to maximumly exploiting the power of multi-view learning. Consensus principle can be regarded as that different summarizers discuss and minimize the discrepancy of their opinions while the complementary
3.2. Multi-view Convolutional Neural Networks

principle can be thought of as overcoming the problem that each summarizer cannot take everything into consideration. Another innovative component, namely sentence position embedding is also used to improve learning ability. We evaluate our new model on five DUC generic summarization benchmark datasets. The proposed method achieves superior performance on all the measure metrics on the five datasets compared with state-of-the-art methods. MV-CNN obtains 1.28% higher Rouge-1 score, 4.35% higher Rouge-2 score, and 3.68% higher Rouge-SU4 score relatively compared with the best existing summarization system on the DUC2006 dataset. The main contributions of this work are summarized as follows:

- Multi-view learning is applied to CNN to enhance the learning capability of the original CNN. The two-level complementary and consensus principles are fully exploited to improve the model’s learning performance. To the best of our knowledge, this is the first time that the combination of multi-view learning and CNN is used for multi-document summarization.

- With the help of pre-trained word embedding, human feature engineering is not needed. This makes our model much easier to be implemented.

- Sentence position embedding is incorporated into the model to advance the learning capacity of the proposed model to a higher level.

3.2.2 Related Works

The proposed model applies deep learning to extractive multi-document summarization (MDS) and exploits multi-view learning to enhance the learning performance. The two most related methods, namely deep learning in MDS and multi-view learning, are briefly introduced in this section.

Deep Learning in Multi-document Summarization

Many recent works have already been done using deep learning methods to summarize documents. The restricted Boltzmann machine (RBM) was incorporated to generate generic summary in [101]. However, it only extracted four linguistic features per sentence as the input to the RBM and its performance was not very satisfactory. The authors of [72] ranked the sentences with recursive neural networks,
obtaining much better performance compared with the traditional methods. However, it used hand-crafted features to represent the input of the model. Besides, the recursive neural networks are built via a tree structure and its performance heavily depends on the construction of the textual tree. The tree construction can be very time-consuming. One simple method termed paragraph vector [102] was developed to encode sentences into vector representations. The model was developed based on the architecture of CBoW and Skip-gram, but proved to perform sub-optimally on some tasks. The authors of [103] attempted to summarize documents with hierarchical CNNs. They employed CNNs to extract sentence features from words and another CNN on top of the learned sentence features to obtain document features. Deconvolutional networks were used to train their models but a lot of hyper-parameters had to be tuned [104]. Our new model is most similar to the model proposed by Cao et al. in [105]. The authors also concatenated CNNs with distinct window sizes to obtain sentence features and rank sentences. Several hand-crafted features were introduced and concatenated with sentence features for sentence regression. The main difference between our model and Cao et al.’s model lies in the incorporation of the idea of multi-view learning which enables our model to achieve superior performance. Actually, their model’s concatenation of CNNs with distinct window size forms the first-level complementary principle of our model, but our model is developed based on two-level complementary principle and consensus principle. Recently, researchers find that multi-task learning framework can be used to improve summarization performance. Authors of [106, 107] leverage on document classification to address the problem of lacking training data in document summarization task. An even more promising method [108] is proposed to achieve abstractive document summarization without linguistic processing.

Multi-view Learning

Multi-view learning learns multiple views of the same input data and improves the learning performance [100]. Multi-view learning has been widely researched in recent years and can be mainly divided into three categories: (1) Co-training algorithms which alternately train two separate learners using features from distinct views to maximize the mutual agreement on the prediction of the two classifiers on the labeled dataset as well as to minimize the disagreement on the prediction on the unlabeled dataset [109–111]. These approaches combine multiple views at
3.2. Multi-view Convolutional Neural Networks

a later stage after training the base learners. (2) Multiple kernel learning algo-
rithms which exploit separate kernels that naturally correspond to different views
and combine kernels to integrate multiple information sources [112–114]. Views
(kernels) are combined at an intermediate stage before or during the training of
learners. (3) Subspace learning-based approaches which aim at obtaining an ap-
propriate subspace shared by multiple views by assuming that input views are
generated from a latent subspace [115–117]. These approaches can be regarded as
prior combinations of multiple views.

Recently, there is an attempt to combine multi-view learning and CNN in the re-
search area of computer vision. The authors in [118] employ multiple CNNs to com-
pile information from a collection of 2D views of 3D objects into a compact shape
descriptor of the object which offers better recognition performance compared with
single-view methods. This method regards different 2D shapes extracted from 3D
objects as distinctive views. However, this method used multi-view learning at a
very superficial level and did not consider the inner connections of different views
of the observed objects. The authors in [119] employed CNN to deal with multiple
sources of micro-blog data in order to detect psychological stress. However, they
handled the data separately and did not consider their inner connections as well.
In contrast, our model aims at digging underlying distinct views of sentences by
using different CNNs. It is like a multiple-kernel-learning method where each CNN
can be seen as one kernel.

3.2.3 The Proposed Model

This section will firstly introduce the structure of the basic CNN of our model and
give detailed information of the proposed MV-CNN model. The entire summariza-
tion procedure using the new approach is given.

Convolutional Neural Networks

In this section, we first give the structure of basic CNN used in our model. In [46],
it was shown that a simple CNN model with one convolution layer followed by one
max-pooling layer is able to perform extremely well for sentence classification. The
CNN approach demonstrates excellent efficiency in latent feature presentation and
can naturally address variable-length sentences. We add one more convolution layer before pooling so as to extract deeper and more abstract information contained in the sentences. The previous classification model is adapted to solve a regression task in order to rank sentences. As shown in Figure 3.2, this specific convolutional architecture only consists of two convolution layers followed by a max-pooling layer. Finally, the sentence representation vector obtained by the CNN, concatenated with sentence position embedding, is fed to the fully connected layer to calculate the regression score of the input sentence.

**Convolutional Layer:** Convolutional layers play critical roles in the success of the CNN because they can encode significant information about input data with significantly fewer parameters than other deep learning architectures. Empirical experiences in the area of computer vision suggest that deep architectures with multiple convolutional layers are necessary to achieve good performance [91]. However, only one convolutional layer was used to achieve satisfactory performance in the sentence classification task in [46]. This may be due to the fact that the datasets used in the paper are not very large. Our experiment results show that two convolutional layers are needed to achieve sufficient model capacity.

We only conduct convolution operation in the sentence length dimension cause it does not make sense convolving in the word vector dimension. The first convolutional layer is done between k filters $W^{(1)} \in \mathbb{R}^{md \times k}$ and a concatenation vector $x_{i:i+m-1}$ which represents a window of $m$ words starting from the $i$th word, obtaining features for the window of words in the corresponding feature maps. The term
Multi-view Convolutional Neural Networks

d is the dimension of word embedding. Multiple filters with different initialized weights are used to improve the model’s learning capacity. The number of filters \( k \) is determined using cross-validation. The convolution operation is governed by

\[
c^{(1)}_i = g(W^{(1)T} x_{i:i+m-1} + b^{(1)}) \in \mathbb{R}^k
\]  

(3.12)

where \( x_i \in \mathbb{R}^d \). The term \( b^{(1)} \) is a bias vector and \( g(\cdot) \) is a nonlinear activation function such as sigmoid, hyperbolic tangent or rectified linear units (ReLU). The ReLU has become a standard nonlinear activation function of CNN recently because it can improve the learning dynamics of the networks and significantly reduce the number of iterations required for convergence in deep networks. We employ a special version of the ReLU called LeakyReLU [120] that allows a small gradient when the unit is not active. It helps further improve the learning performance compared with ReLU.

Suppose the length of a sentence is \( n \). As the word window slides, the feature maps of first convolutional layer can be represented as follows:

\[
c^{(1)} = \left[ c^{(1)}_1, c^{(1)}_2, \ldots, c^{(1)}_{n-m+1} \right]
\]

(3.13)

We set the window size and number of feature maps of the second convolutional layer the same as the first convolutional layer for easy understanding because we distinguish the distinct CNNs by their window sizes. Therefore, the output of the second convolutional layer is given by:

\[
c^{(2)}_j = g(W^{(2)T} c^{(1)}_{j:j+m-1} + b^{(2)}) \in \mathbb{R}^k
\]

(3.14)

where \( W^{(2)} \in \mathbb{R}^{mk \times k} \) and \( g(\cdot) \) is also the nonlinear activation function LeakyReLU.

As the window slides on the previous convolutional layer, the feature maps of the second convolutional layer are given as follows:

\[
c^{(2)} = \left[ c^{(2)}_1, c^{(2)}_2, \ldots, c^{(2)}_{n-(m-1) \times 2} \right]
\]

(3.15)

Max-pooling Layer: Max-pooling layers are useful in reducing the number of parameters in the network by reducing the spatial size of the vector representation. We conduct a max-pooling operation: \( h = max(c^{(2)}) \in \mathbb{R}^k \) to obtain features corresponding to the filters. The idea behind the max-pooling operation is that it can
reduce feature dimensionality and lead to a fixed-length feature vector regardless of variable sentence lengths.

**Fully-connected Layer:** After the max-pooling layer, we obtain the penultimate layer \( h = [h_1, \cdots, h_k]^T \), \( k \) is the number of filters, which is the vector representation of the input sentence. Besides the sentence representation vector, we also feed sentence position embedding into the fully connected layer. The location where a sentence occurred in one document can be critical for determining how important the sentence is for the document. We use position embedding instead of a position integer so that the position feature will not be drowned by the long sentence vector. In [99], word position embedding was proposed to facilitate the relation classification task. Inspired by their ideas, we incorporate sentence position embedding into our proposed CNN model. Generally speaking, sentences appearing in the beginning part or conclusion section of one document may contain the most useful information. Therefore, we denote the position of a sentence \( s_i \) according to the following equation:

\[
p(s_i) = \begin{cases} 
0 & s_i \in S_{1:3} \\
1 & s_i \in S_{-3:-1} \\
2 & \text{otherwise}
\end{cases}
\]  

(3.16)

where \( S_{1:3} \) denotes the set containing the first three sentences of a document while \( S_{-3:-1} \) the set constituted by the last three sentences. Equation (3.16) holds for our experiment because all the documents used have more than six sentences. The three integers are mapped to a vector space \( h_{sp} \in \mathbb{R}^{k'} \). The mapping procedure is done like that of word embedding by Equation (2.15), except that the position embedding matrices are initialized randomly. The concatenation of sentence representation vector \( h \) and sentence position embedding \( h_{sp} \) formulates the sentence embedding \( h_s = [h, h_{sp}] \in \mathbb{R}^{k+k'} \) which is used as the input of the fully-connected layer.

To avoid overfitting, dropout [33] with a masking probability \( p \) is applied on the penultimate layer. The key idea of dropout is to randomly drop units (along with their connections) from the neural network during the training phase. Thus, the significance of the sentence is calculated through a regression process governed by:

\[
\hat{y} = \sigma(w_r(h_s \otimes r) + b_r)
\]

(3.17)

where \( \sigma(\cdot) \) is a sigmoid function, \( \otimes \) is an element-wise multiplication operator and \( r \) is the masking vector with \( p = 0.5 \). In addition, a \( L2 \)–norm constraint of the filter
weights $w_r$ is imposed during training as well. The model parameters including word vectors and sentence position embeddings are all fine-tuned via stochastic gradient descent using the Adadelta [40] update rule, which has been shown as an effective and efficient backpropagation algorithm.

### Multi-view Convolutional Neural Networks for Multi-document Summarization

The proposed CNN is able to efficiently extract features of sentences and perform sentence ranking jointly. However, we believe that using a single CNN can hardly capture all the information contained in sentences sufficiently. In practical cases, it is common that human beings may hold significantly different opinions towards the same sentences in summarizing documents. This is because that they own unique perceptions and biases as well as different background knowledge and intentions. Therefore, we leverage on the idea of multi-view learning to consider different perspectives towards the same set of documents. The framework of the newly proposed MV-CNN model is depicted in Figure 3.3.

The success of multi-view learning approaches heavily depends on two significant principles, namely complementary and consensus principles. The former principle demonstrates that complementary information underlying multi-view data can be exploited to comprehensively and accurately describe the data and thus improve
the learning performance while the latter aims at maximizing the agreement of distinct learners trained on multiple views.

**Complementary Principle:** As foreshadowed in Section 3.2.2, our new MV-CNN model can be regarded as a multiple-kernel-learning method where each CNN functions like a kernel. Separate kernels naturally correspond to various views, assessing data from distinct perspectives. The complementary information underlying various views of data help comprehensively evaluate the data. As shown in Figure 3.3, our proposed MV-CNN uses multiple CNNs to represent an input sentence with a vector. The CNN in Figure 3.3 is a little bit different from the basic CNN structure depicted in Figure 3.2. It does not include the fully-connected layer because CNNs are exploited to obtain vector representation of sentences. The fully-connected layers are separately shown as classifiers. The CNNs have distinct window-size filters convolving with the input data, extracting different latent information. We are not concerned with which CNN has the best performance. Instead, we combine the results of distinct CNNs appropriately using the complementary principle. In addition, we also concatenate sentence representations acquired by each CNN into *multi-view sentence representations* and use an additional classifier to analyze the new representations. Therefore, complementary principles are applied to our new model at two levels jointly, namely final stage combining distinct outputs and the intermediate stage combining distinct sentence representations. The final-stage combination is like different summarizers negotiate and compromise to determine the final summaries after getting their own summaries. And the intermediate-stage combination can be thought of as different summarizers exchange one another’s opinions first and perform the summarization task together. Therefore, the learning process of the MV-CNN is imitating the human summarizers’ behavior.

We formulate a theorem in order to prove that the complementary principle is helpful in improving the learning performance.

**Theorem 1.** We denote the input as $x$, real output as $y$ and the predicted output by a single CNN as $f(x)$. The MV-CNN predicted output is the weighted sum of the distinct CNNs’ outputs. Then the expected mean squared error of a single CNN is equal or greater than that of MV-CNN.
Proof. As all the CNNs are from the same distribution (denoted as $\chi$), the multi-view CNN predicted output is given by:

$$f_{MV}(x) = E_{f \sim \chi}(f(x))$$  \hspace{1cm} (3.18)

where the expression $f \sim \chi$ means $f$ conforms to the distribution $\chi$. The expected error of the MV-CNN is given by:

$$e_{MV} = E_x(y - f_{MV}(x))^2$$  \hspace{1cm} (3.19)

The expected error of a single CNN is calculated as follows:

$$e = E_x(y - f(x))^2$$  \hspace{1cm} (3.20)

Now, we prove that the MV-CNN can obtain better learning performance than a single CNN if $e \geq e_{MV}$ holds. We have

$$e = E_x(y - f(x))^2$$  
$$= E_{f \sim \chi}E_x(y - f(x))^2$$  
$$= E_{f \sim \chi}E_x(y^2 - 2yf(x) + f^2(x))$$  
$$= E_x(y^2 - 2yE_{f \sim \chi}(f(x)) + E_{f \sim \chi}(f^2(x)))$$  \hspace{1cm} (3.21)  
$$\geq E_x(y^2 - 2yf_{MV}(x) + E_{f \sim \chi}^2(f(x)))$$  
$$= E_x(y^2 - 2yf_{MV}(x) + f_{MV}^2(x))$$  
$$= E_x(y - f_{MV}(x))^2 = e_{MV}$$

Thus the usefulness of the complementary principle is established. \hfill \Box

Consensus Principle: The consensus principle is another key to the success of multi-view learning. It ensures the similarity of the predicted results of multiple learners on the same data. The authors of [121] demonstrated that the probability of two independent hypotheses that disagree on two views imposes an upper bound on the error rate of either hypothesis. This relationship can be described by the following inequality:

$$P(f_1 \neq f_2) \geq \max(P(error(f_1)), P(error(f_2)))$$  \hspace{1cm} (3.22)
Therefore, the error rate of each hypothesis decreases if the consensus between two hypotheses is enhanced. This means that we can increase each CNN’s learning accuracy thus increase the final learning capability if the disagreement of distinct CNNs’ predictions is minimized. In the framework of the MV-CNN as shown in Figure 3.3, we not only try to minimize the discrepancy between the multi-view prediction output and real target but also minimize the disagreement of distinct CNNs’ predictions. The predicted saliency score of the input sentence by each CNN is denoted as \( f_i(x) \). The consensus principle is governed by:

\[
min \sum_x \sum_{i \neq j} (f_i(x) - f_j(x))^2 
\]

(3.23)

As complementary and consensus principles have been introduced, the entire summarization procedure is described below.

**Training:** As ready-made salience scores are not available for the training data, we first pre-process the documents to obtain a salience score for each sentence like in Section 3.1.3. After the preprocessing procedure, we use the training documents to train our MV-CNN model. The single CNN structure has been demonstrated in Section 3.2.3. Besides the real CNNs, the score generated by multi-view sentence representation is regarded of being assigned by a pseudo CNN. The objective function in terms of minimizing the cross-entropy is used:

\[
L = -\sum_x y \ln(\sum_i u_i f_i(x)) + (1 - y) \ln(1 - \sum_i u_i f_i(x)) 
\]

(3.24)

where \( u_i \) is the weight of each saliency score assigned by the corresponding CNN. The score weights are updated during training.

Besides the cross-entropy loss function, we also apply the consensus principle to minimize disagreement between every two classifiers. Thus, by combining Equation (3.23) and Equation (3.24), we formulate the final objective function as follows:

\[
min \lambda \sum_x \sum_{i \neq j} (f_i(x) - f_j(x))^2 + L 
\]

(3.25)

where the term \( \lambda \) is the parameter regulating the two components, which is chosen using cross-validation. In the experiments, we only use three CNNs due to
Algorithm 1 Pseudo-code for MV-CNN

1. Pre-process the documents and reference summaries, obtaining the saliency score for each sentence;
2. Construct word embedding table using pre-trained word vectors;
3. Generate sentence position embedding $h_{sp}$ for each sentence using Equation (3.16);
4. for i in [1, K] do
   5. For the $ith$ CNN with window size $m_i$, apply two consecutive convolution operation between $k$ filters and the input sentences using Equations (3.12-3.15);
   6. Apply max-pooling operation to get sentence representations $h_i$;
   7. Concatenate $h_i$ and $h_{sp}$ and apply fully-connected sigmoid classifier to obtain regression scores $f_i$ using Equation (3.17);
end for
9. Concatenate $K$ sentence representations $h_i (i = 1, \cdots, K)$ and $h_{sp}$ and apply fully-connected sigmoid classifier to obtain regression scores $f_{MV}$;
10. Update parameters of the model using the loss function Equation (3.25) with the Adadelta.

the computation and time constraints. The learning performance is expected to improve if we use more CNNs.

Testing: For the testing documents, all the sentences will be used as the input to the trained CNN model to obtain their saliency scores. Next, sentences in each document set can be ranked according to their salience scores. Since a good summary should be not only informative but also non-redundant, we also employ the sentence selection method of [87] to select from the ranked sentences like in Section 3.1.3. The selection method queries the sentence with highest salience score and adds it to the summary if the similarity of the sentence with all the sentences already in the summary does not exceed a threshold. The selection process repeats until the length limit of the final summary is met.

The entire learning algorithm of MV-CNN is summarized as Algorithm 1.

3.2.4 Experimental Studies and Discussions

Datasets

The benchmark datasets from the Document Understanding Conferences (DUC) are used to evaluate our proposed multi-document summarization system. The datasets are English news articles. In this section, DUC2001/2002/2004/2006/2007
datasets are used for evaluating generic multi-document summarization tasks. The characteristics of the datasets are given in Table 3.4. The table gives the number of document clusters, total document size, and the upper limit of summary length. When evaluating on DUC2001/2002/2004, we use two years of data as training data and the other year of data as test data. When evaluating on DUC2006/2007, we use one year of data as training data and the other year’s data as test data. The reference summaries for each set of documents are given together with the documents.

### Parameter Settings

Same as the window-based sentence representation method, the F-measures of ROUGE-1, ROUGE-2 and Rouge-SU4 scores are reported in our evaluation study to compare the performances of different algorithms, see Section 3.1.4. We set the length parameter “-l 100” for DUC2001/2002, “-b 665” for DUC2004, and “-l 250” for DUC2006/2007. The setting of word embeddings is also the same as that in the window-based sentence representation method as in Section 3.1.4. The number of filters $k$ and the dimension of sentence position embedding $k'$ are both set to 100. The further increase of the number of filters does not enhance the learning performance obviously but increases the complexity of the model. The threshold value of sentence selection for multi-document summarization is set to 0.5. These parameters are determined using cross-validation. Only three CNNs are used in our experiment because of the computation and time constraints. We expect to obtain better learning performance when using more CNNs. In order to have a fair comparison with the baseline CNN, the filter window sizes for the three CNNs are chosen as 3, 4 and 5. Other hyper-parameters are fine-tuned via the stochastic gradient descent method Adadelta. All the simulation studies are conducted using

<table>
<thead>
<tr>
<th>Year</th>
<th>Clusters</th>
<th>Documents</th>
<th>Length Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>30</td>
<td>303</td>
<td>100 words</td>
</tr>
<tr>
<td>2002</td>
<td>59</td>
<td>533</td>
<td>100 words</td>
</tr>
<tr>
<td>2004</td>
<td>50</td>
<td>500</td>
<td>665 bytes</td>
</tr>
<tr>
<td>2006</td>
<td>50</td>
<td>1250</td>
<td>250 words</td>
</tr>
<tr>
<td>2007</td>
<td>45</td>
<td>1125</td>
<td>250 words</td>
</tr>
</tbody>
</table>
an NVIDIA Tesla K20c GPU on a Windows Server with 2.0 GHz CPU and 256 GB RAM.

**Comparison Results**

In order to demonstrate the summarization performance of our model, we compare it with several state-of-the-art extractive summarization techniques. The most baseline method is Random which selects sentences randomly from the candidate set. As mentioned in the related work section, extractive summarization methods are roughly classified into three categories and our method belongs to the supervised category. We compare with methods from the other two categories. Lead is a representative of the methods based on sentence positions and article structure while LSA belongs to the unsupervised category. Two most recently published results, namely MDS-Sparse [122] and RA-MDS [123] are also included for comparison. Both methods are developed based on sparse coding and belong to the unsupervised category. We directly retrieve the results of LSA, MDS-Sparse and RA-MDS in [122, 123] because the datasets and evaluation metrics are standard. As the two papers only did experiments on DUC2006/2007, the results for the three methods on DUC2001/2002/2004 are not given in our experiments. These comparison systems are basic baselines.

To demonstrate the effectiveness of incorporating multi-view learning into CNN, we compare our method with two other CNN baselines, i.e., BasicCNN and PriorSum. BasicCNN is the basic CNN used in our model with a fixed window size. PriorSum is the model proposed by Cao et al. [105]. In order to have a fair comparison, we only concatenate position features to the sentence features obtained by PriorSum excluding the other handcrafted features. We experiment on BasicCNN using three window sizes ($m = 3, 4, 5$) and show the best result. The other settings of the two CNN models are the same with the MV-CNN. The three CNN models fall into the category of supervised methods.

Table 3.5, Table 3.6 and Table 3.7 are the overall performance comparison results. It is clear that the MV-CNN outperforms all the other algorithms except the RA-MDS. Among all the algorithms, the LSA gives the poorest performance

---

5 RA-MDS is the best existing summarization system on DUC2006 and DUC2007 datasets. It is only included as a reference but not taken into consideration for result comparison because it exploits more input information compared with other methods.
Table 3.5: Rouge-1 score comparison results (%)

<table>
<thead>
<tr>
<th>System</th>
<th>DUC01</th>
<th>DUC02</th>
<th>DUC04</th>
<th>DUC06</th>
<th>DUC07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>28.25</td>
<td>27.78</td>
<td>31.47</td>
<td>28.05</td>
<td>30.20</td>
</tr>
<tr>
<td>Lead</td>
<td>29.43</td>
<td>28.68</td>
<td>32.10</td>
<td>30.76</td>
<td>31.19</td>
</tr>
<tr>
<td>LSA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>24.42</td>
<td>25.98</td>
</tr>
<tr>
<td>MDS-Sparse</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>34.44</td>
<td>35.40</td>
</tr>
<tr>
<td>RA-MDS⁹</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>39.1</td>
<td>40.3</td>
</tr>
<tr>
<td>BasicCNN</td>
<td>35.38</td>
<td>36.07</td>
<td>38.24</td>
<td>37.98</td>
<td>40.16</td>
</tr>
<tr>
<td>PriorSum</td>
<td>35.76</td>
<td>36.24</td>
<td>38.58</td>
<td>38.16</td>
<td>40.48</td>
</tr>
<tr>
<td>MV-CNN</td>
<td>35.99</td>
<td>36.71</td>
<td>39.07</td>
<td>38.65</td>
<td>40.92</td>
</tr>
</tbody>
</table>

Table 3.6: Rouge-2 score comparison results (%)

<table>
<thead>
<tr>
<th>System</th>
<th>DUC01</th>
<th>DUC02</th>
<th>DUC04</th>
<th>DUC06</th>
<th>DUC07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>4.23</td>
<td>4.76</td>
<td>4.97</td>
<td>4.61</td>
<td>4.62</td>
</tr>
<tr>
<td>Lead</td>
<td>4.03</td>
<td>5.28</td>
<td>6.38</td>
<td>4.84</td>
<td>5.76</td>
</tr>
<tr>
<td>LSA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.02</td>
<td>4.06</td>
</tr>
<tr>
<td>MDS-Sparse</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.12</td>
<td>6.45</td>
</tr>
<tr>
<td>RA-MDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8.1</td>
<td>9.2</td>
</tr>
<tr>
<td>BasicCNN</td>
<td>7.69</td>
<td>8.73</td>
<td>9.80</td>
<td>7.55</td>
<td>8.60</td>
</tr>
<tr>
<td>PriorSum</td>
<td>7.79</td>
<td>8.85</td>
<td>9.86</td>
<td>7.58</td>
<td>8.77</td>
</tr>
<tr>
<td>MV-CNN</td>
<td>7.91</td>
<td>9.02</td>
<td>10.06</td>
<td>7.91</td>
<td>9.11</td>
</tr>
</tbody>
</table>

Table 3.7: Rouge-SU4 score comparison results (%)

<table>
<thead>
<tr>
<th>System</th>
<th>DUC01</th>
<th>DUC02</th>
<th>DUC04</th>
<th>DUC06</th>
<th>DUC07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>8.46</td>
<td>8.61</td>
<td>8.82</td>
<td>8.79</td>
<td>8.76</td>
</tr>
<tr>
<td>Lead</td>
<td>8.88</td>
<td>9.53</td>
<td>10.23</td>
<td>8.65</td>
<td>10.20</td>
</tr>
<tr>
<td>LSA</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.10</td>
<td>8.34</td>
</tr>
<tr>
<td>MDS-Sparse</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.72</td>
<td>11.67</td>
</tr>
<tr>
<td>RA-MDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13.6</td>
<td>14.6</td>
</tr>
<tr>
<td>PriorSum</td>
<td>12.84</td>
<td>14.37</td>
<td>13.05</td>
<td>13.59</td>
<td>14.97</td>
</tr>
<tr>
<td>MV-CNN</td>
<td>13.16</td>
<td>14.92</td>
<td>13.67</td>
<td>14.09</td>
<td>15.34</td>
</tr>
</tbody>
</table>

on DUC2006/2007. The LSA exploits SVD on term frequency matrix and assumes that sentences with highest eigenvalues are most informative sentences in a document corpus. Our experiment results suggest that such hypothesis may not be true for human understanding. The table also shows that the Lead performs slightly better than the Random method. One reason could be because the authors like to put summary sentences at the beginning of the documents. The two sparse coding-based methods achieve state-of-the-art performance on DUC2006/2007 proving that sparse coding helps to find informative sentences.
3.2. Multi-view Convolutional Neural Networks

out of documents. The RA-MDS achieves the best performance for Rouge-1 and Rouge-2 on DUC2006. However, it must be highlighted that the RA-MDS exploits a set of user comments associated with the documents to help with the summarization task. This means that it has more input information which is pretty unfair for the other compared methods. Irrespective of this, the MV-CNN shows higher Rouge-SU4 score on DUC2006 and Rouge-1 and Rouge-SU4 scores on DUC2007 than RA-MDS. This effectively demonstrates the outstanding summarization ability of the MV-CNN. Besides, one of the biggest advantages of our method is that it does not need hand-crafted features. The pre-trained word embedding has saved us an enormous amount of time and effort, enabling us to avoid the intensive human labor of feature engineering. The MV-CNN outperforms the other two CNN baselines on all metrics on all the datasets. This proves the effectiveness of the incorporation of multi-view learning idea. It should be noted that MV-CNN obtains much better Rouge-SU4 scores than all the compared methods. This indicates that the MV-CNN has good readability which may because that multi-view learning excludes the biases of single CNNs and assigns the highest score to the most appropriate sentence from a similar sentence list.

Some researchers may argue that deep learning methods demand prohibitive computational cost and memory because there are so many parameters to tune and store in the memory. However, the problem is no longer a limitation with the progress in hardware, software, algorithm parallelization and the efficient use of graphics processing units (GPUs). It takes less than two hours to run 30 iterations for the MV-CNN in our experiment. It must be highlighted that conventional machine learning methods require good feature extractors before applying learning models. If we take the entire pipeline into consideration, deep learning methods should use much less time and efforts. On the other hand, the complexity of MV-CNN increases compared with the standard CNN because two convolution layers and multiple CNNs are used. But we believe it is worthy of increasing a little complexity to enhance the learning performance.

Statistical Test

In order to verify that the performance improvement of the MV-CNN over other approaches is statistically significant, we perform some statistical tests. We use
paired comparison t-tests because we believe the results of distinct approaches on the same dataset have correlations to some extent.

The paired t-test is a statistical technique that is used to compare population means of two approaches. It can be used for comparison of two different methods of measurement when the measurements are applied to the same subject. For each approach, we conducted the experiment ten times for statistical comparison (the accuracy is supposed to be the same for ten times if the results are extracted from the original papers).

Firstly, we set up the null hypothesis as the mean difference between the two paired methods is zero. Next, we calculate the differences of each time for each pair of the methods. Next, the mean difference $\bar{d}$ and standard deviation of differences $s_d$ are calculated in order to obtain the t-statistic as follows:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}}$$

where $n$ is the number of pairs of observation, which is 10 in our case. Under the null hypothesis, the t-statistic follows a t-distribution with $n - 1$ degrees of freedom. Comparing the t-statistic with the t-distribution table gives the p-value for the paired t-test. The p-value is used to indicate whether the differences of measurements of the two methods are statistically significant.

We find the associated p-values for each pair of approaches on all the datasets are all smaller than 0.001. As the precision of p-values from t-distribution table can only reach 0.001, the complete p-value results are not shown. Therefore, we can conclude that our proposed method obtains superior performance compared with the other methods at nearly 100% confidence.

**Effect of Convolutional Layer Number**

The basic CNN used in our framework uses two convolutional layers. Empirical results are given in this section to justify the statement. We show the Rouge-2 scores experimented on DUC2006 dataset over the number of convolutional layers in Figure 3.4. The figure demonstrates that two convolutional layers achieve better learning performance than one convolutional layer model. However, the performance only increases marginally and even decrease because of over-fitting with
the increasing number of convolutional layers. Similar trends are detected on other metrics experimented on other datasets. Besides, the computation complexity increases fast if using more layers. Therefore, two convolutional layers are used in our model. The author of [79] proposed an adaptive method to select the number of neurons and layers in neural network for superior generalization performance based on real-time recurrent learning, which is a theoretically proved alternative of the empirical method for finding layer number.

Discussions

The comparison results demonstrate that multi-view learning idea is highly effective in improving learning capability. As foreshadowed in Section 3.2.3, complementary and consensus principles are the keys to the success of multi-view learning. The complementary principle shows that multiple views describing the sentences from distinct perspectives can help comprehensively extract underlying information. The consensus principle leads to the different summarizers reaching a maximal consensus. In this section, we analyze the impacts of the two principles using experiments on DUC2006/2007. In addition, our model adopts a novel component, i.e., sentence position embedding. The impact of sentence position embedding will be demonstrated in this section as well.

Impact of the Complementary and Consensus Principles: The MV-CNN applies complementary principles at two levels, namely final stage combining distinct outputs and the intermediate stage combining distinct sentence representations. We analyze the impact of each level complementary setting by excluding
the final-stage combination and intermediate combination respectively from the MV-CNN model. Similarly, the impact of the consensus principle is demonstrated by comparing the MV-CNN with and without consensus setting.

The comparison results are shown in Figures 3.5-3.7. In the figures, MVCNN +CompI+CompF+Cons denotes the complete MV-CNN model, MVCNN+CompI+CompF denotes the MV-CNN model without consensus setting but with a full complementary setting, and MVCNN-CompI+CompF and MVCNN +CompI-CompF denote the MV-CNN without intermediate complementary and final complementary setting respectively. Furthermore, MVCNN-CompI-CompF denotes CNN
3.2. Multi-view Convolutional Neural Networks

![Figure 3.7: Rouge-SU4 scores (%) comparison by impact of complementary and consensus principles.](image)

without any complementary and consensus settings, which is actually the BasicCNN. We ignore the consensus principle when analyzing the impact of the complementary principle. It can be seen from the figures that both complementary and consensus principles are useful in performance improvement. Comparing the green and blue bars, we can conclude that the intermediate complementary setting is more important than the final-stage complementary. Employing both stage complementary further enhances the model’s learning capability (blue bar). These experiments demonstrate the effectiveness and efficiency of multi-view learning empirically.

**Impact of the Sentence Position Embedding:** The position a sentence occurred in one document can be critical for determining how important the sentence is for the document. A position embedding instead of a position integer is used because the effect of an integer feature may almost be ignored when it is concatenated with the long sentence vector.

Comparison results are shown in Figure 3.8. In the figure, MVCNN+pos denotes the complete MV-CNN model and MVCNN-pos denotes MV-CNN model without using sentence position embedding. It can be easily concluded from the figure that the existence of sentence position embedding helps enhance the performance on all the three metrics on both DUC2006 and DUC2007 datasets. This is because position information of a sentence in one document can be significant to determine whether it should be included in the final summary.
Chapter 3. Document Summarization

3.2.5 Summary

In this section, an enhanced convolutional neural network termed multi-view convolutional neural network (MV-CNN) has been successfully developed to obtain the features of sentences and rank sentences jointly. We leverage on pre-trained word embedding to represent sentences so as to avoid the intensive labor of feature engineering. This makes our model much easier to be implemented. The biggest innovation of the new model is the incorporation of multi-view learning into standard CNN. The learning process of the MV-CNN imitates the human summarizers’ behavior. The two-level complementary and consensus principles of multi-view

---

6Only part of the reference and automatic summaries are shown for brevity.
### Table 3.8: One manual summary and an automatic summary generated by the proposed model on the topic “An Air France Concorde crash”.

<table>
<thead>
<tr>
<th>Reference Summary</th>
<th>An Air France Concorde en route to New York crashes outside Paris shortly after takeoff, killing all 109 people on board and four people on the ground. Most of the crash victims were Germans flying to New York to join a luxury cruise through the Caribbean. After the crash, Air France agreed to temporarily ground its five remaining Concorde, and British Airways grounded its two remaining flights for Tuesday night. French investigators have said they believe that a thin strip of metal fell off the Continental plane, which had taken off on the same runway as the Concorde, and set off the events that caused the Air France jet to burst into flames and crash within minutes after its takeoff. One month after an Air France Concorde crashed outside Paris, victims’ relatives gathered Saturday to remember their loved ones near the site where the supersonic jet went down. The sleek, needle-nosed aircraft could cross the Atlantic at an altitude of 60,000 feet and at 1,350 mph, completing the trip from London to New York in less than four hours – half the time of regular jets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The only supersonic passenger plane—the Concorde—crashed on July 25, 2000, killing all 109 on board and 4 on the ground. The plane had 100 seats and flew 1,300 mph at 60,000 feet. It took less that 4 hours to cross the Atlantic carrying the rich, the famous, and business people. The crash was caused by a piece of metal falling off of a Continental DC10 that used the runway before the Concorde. The metal cut the Concorde’s tire and flying rubber damaged the engines causing a fuel leak that burst into flame, too late to abort. Almost all of the passengers were Germans going to New York. Air France paid each family $20,000 to cover immediate expenses. Memorial services were held in Paris, in Cologne, Germany, and later at the crash site. Air France immediately grounded its fleet and British Airways stopped flights on August 15.</td>
<td></td>
</tr>
</tbody>
</table>

---

*6 Reference Summary from [source](source)*
learning are fully exploited to improve the model’s learning performance. To the best of our knowledge, this is the first time that the combination of multi-view learning and CNN is used for multi-document summarization. Novel sentence position embedding is introduced to help retrieve information in the documents, advancing the learning ability of proposed model to a higher level. Experiment results demonstrate that our model performs remarkably well, achieving better performance compared with state-of-the-art approaches.
Chapter 4

Sentiment Analysis

4.1 Introduction

The problem of sentiment classification is a hot topic in natural language processing and has received a lot of attention lately. It has been applied to recommender systems, business intelligence and other areas where it is significant to understand the sentiment of user-generated feedback. Machine learning techniques have been widely used in sentiment analysis since Pang et al. [124] used methods such as Naive Bayes classification and maximum entropy to study the sentiment contained in sentences. The traditional machine learning methods focus on designing hand-crafted features [125–127] and use classifiers such as support vector machines to accomplish the task. Feature representation is a key component of many machine learning systems because the performance of a machine learner heavily depends on it [91].

A widely used sentence representation model is the bag-of-words (BoW) model which uses one-hot vectors to represent words, where each dimension of the vector corresponds to a distinct word. The model achieves very good performance on a variety of tasks but faces the problem of losing word order which is critical for semantic analysis. For example, “Jimmy loves Kate” and “Kate loves Jimmy” have exactly the same representation using the BoW model while the two sentences have different meanings. The bag-of-n-grams model is proposed to consider the
word order in short context, but it suffers from high dimensionality and data sparsity. Both BoW and bag-of-n-grams models have difficulty capturing the semantic information.

Deep learning models together with word embeddings have been effective in tackling the feature representation problem because they can learn features from data automatically. Convolutional neural networks [27] and recurrent neural networks [128] have been proven to be powerful semantic composition models for sentiment classification [46, 96, 129]. They leverage on randomly initialized word vectors or pre-trained word vectors to represent words and employ the deep neural networks to learn vector representations for sentences of variable length. We adopt off-the-shelf word vectors to make better use of semantic and grammatical associations of words in our new method. The sentence presentations can well capture the semantics of the sentences and are used as features to classify the sentiment of the sentences. This conforms to the principle of compositionality that states the meaning of a longer expression (e.g. a sentence or a document) comes from the meanings of its constituents and the rules used to combine them [4].

One shortcoming of the standard recurrent neural networks is that only historical context can be exploited. Schuster et al. proposed the bidirectional recurrent neural networks to access both the preceding and succeeding contexts by combining a forward hidden layer and backward hidden layer [130]. However, the bidirectional recurrent neural networks still miss the local context information for a position in one sequence. In this chapter, we leverage on the standard recurrent neural networks to develop a new architecture termed comprehensive attention recurrent neural networks which can store comprehensive information, namely preceding, succeeding and local contexts for any position in a sequence. The bidirectional recurrent neural network is used to access the past and future information while a convolutional layer is employed to capture local information. Then, the preceding, succeeding and local context representations are effectively combined and fed into another composition layer to obtain the sentence representations. Finally, a softmax classifier is used for sentiment labeling.

Another significant limitation of the standard recurrent neural networks results from “exploding gradient” or “vanishing gradient” effect [131], i.e., the gradients may grow or decay exponentially over a long-range temporal interval. This highly limits the capability of accessing long history. A lot of efforts have been done to
address the issue, among which long short-term memory [10, 131] and gated recurrent unit [11, 132] achieve excellent empirical results. In this chapter, we replace the standard recurrent neural network with the long short-term memory and gated recurrent unit to propose another two new models to enhance the effectiveness of the new architecture.

Our new models can be trained end-to-end without any human intervention and it is very easy to be implemented. We conduct experiments on several sentiment analysis datasets. The experiment results demonstrate that capturing comprehensive contextual information can enhance the classification accuracy compared with the standard recurrent neural networks and the new models can achieve competitive performance compared with the state-of-the-art approaches. The main contributions of this work are summarized as follows:

- We represent a new deep neural network architecture which can access comprehensive information, namely preceding, succeeding and local contexts of any position in a sequence for sentiment classification.
- Our model can be trained end-to-end by using pre-trained word embedding and human feature engineering is no longer needed.
- Empirical results on large-scale datasets demonstrate that the new architecture effectively improves the classification performance compared with standard recurrent methods.

This chapter is based on the paper: Sentiment classification using Comprehensive Attention Recurrent models [7].

### 4.2 Overview of Related Works

#### 4.2.1 Convolutional Recurrent Neural Networks

The convolutional neural network (CNN) was originally proposed for computer vision by LeCun et al. in [27]. It produces excellent results for computer vision tasks and recently has been proven to be effective for various natural language processing (NLP) tasks as well, such as part-of-speech tagging [95], sentence modeling [96], and
sentence classification [46], to name a few. The CNN has demonstrated excellent performance in latent feature presentation [98, 99]. Another popular deep learning model, recurrent neural network (RNN) [128], is able to retain the memory of all the previous text in a sequence of hidden states. Therefore, it can better capture relations between words and semantics of long texts. RNN has been successfully applied to tasks like text generation [133], machine translation [134], etc.

Recently, great efforts have been devoted to combining the CNN and the RNN because they are complementary in their modeling capabilities. Pinheiro et al. [135] proposed a recurrent convolutional neural network method for scene parsing and similar algorithms have emerged for image representation [136], visual description [137, 138] and object recognition [139]. The ensemble methods have also achieved excellent results in natural language understanding tasks. In [140], Kalchbrenner et al. introduced a sentence model based on a convolutional architecture and a discourse model based on a novel use of the RNN to achieve state-of-the-art results in a dialogue act classification experiment without manual feature engineering. The authors of [141] used the CNNs to reduce the spectral variation of the input features, and passed the outputs of CNNs to long short-term memory (LSTM) layers to reduce temporal variations, and finally passed to deep neural networks (DNN) layers. This model can produce feature representations that are more easily separable. Lai et al. [142] proposed a model capturing contextual information with the recurrent structure and constructing the representation of text using a CNN. All the aforementioned attempts combining the CNN and the RNN are done in a hierarchical or vertical way while our new method embeds the CNN into the RNN and done in a horizontal manner.

4.2.2 Deep Neural Networks for Sentiment Classification

The pipeline of traditional sentiment classification tasks is mainly divided into three components, namely feature design, feature selection and applying machine learning algorithms. Every component of the pipeline is never a trivial task and can take much effort and time. Deep neural networks and representation learning have provided new means for sentiment classification problem where no manual feature engineering is needed and the entire model is trained end-to-end. Kim [46] used a one-layer CNN with pre-trained word embeddings to achieve very good
performance on sentence classification problem. The authors of [96] and [129] also applied the CNN to model sentences but used different pooling methods. The recurrent convolutional neural network structure was used for text classification in [142], in which a recurrent structure was used to capture contextual information and a max-pooling layer was employed to capture key components in texts. The authors of [143] proposed a gated recurrent neural network for sentiment classification recently. They first learned sentence representation using the CNN and encoded document representation with a novel recurrent structure similar to the gated recurrent unit (GRU). The existing convolutional or recurrent models can only access part of the contextual information. In comparison, our new comprehensive attention recurrent models can capture historical, future and local information at the same time. Recently, deep recursive neural networks inspired by Socher’s work [144] have also been proposed to classify sentences. Their performance heavily depends on the performance of the textual tree construction which is too time-consuming and recursive models are not suitable for modeling long sentences. Our model is much easier to be implemented by leveraging on recurrent neural networks. Recently, many methods have been proposed to identify the sentiments contained in twitter [145], among which BB\_twr [146] ranked first in many subtasks. The above methods all use word embedding technique to represent words and sentences. An alternative to word embedding is concept and bag of concept-based approaches proposed in [147, 148].

4.3 The Proposed Model

We introduce our new recurrent architecture in this section. Before that, brief introductions on the basic components of the new model namely standard recurrent neural network (RNN), long short-term memory (LSTM) and gated recurrent unit (GRU) are given.

4.3.1 Standard RNN

The standard RNN is a kind of feedforward neural network that has a recurrent hidden state whose activation at certain time is dependent on the previous states. Therefore, the RNN can handle variable-length sequences and model contextual
Figure 4.1: Illustration of a standard RNN model (left) and a BRNN model (right). The weights of the BRNN are left out for clarity.

information dynamically. Given an input sequence \( \{x_1, x_2, \ldots, x_T\} \), the standard RNN computes the output vector \( y_t \) corresponding to each \( x_t \) with the following equations:

\[
\begin{align*}
  h_t &= g(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\
  y_t &= W_{hy}h_t + b_y
\end{align*}
\] (4.1) (4.2)

where \( W \) denotes the weight matrix connecting two layers, in which \( W_{hh} \) denotes the self-updating weights of the hidden layer. The term \( b \) denotes bias vector, while \( g(\cdot) \) is an element-wise activation function such as sigmoid or hyperbolic tangent. The graphical illustration of the standard RNN can be found in the left-hand side of Figure 4.1.

The standard RNN can only exploit the historical context. Therefore, the bidirectional recurrent neural network (BRNN) has been proposed to access both the preceding and succeeding contexts by combining a forward hidden layer \( \overset{\rightarrow}{h} \) and a backward hidden layer \( \overset{\leftarrow}{h} \) as depicted in the right part of Figure 4.1. The process
can be expressed as follows:

\[
\begin{align*}
\vec{h}_t &= g(\mathbf{W}_{\vec{x}} \vec{x}_t + \mathbf{W}_{\vec{h}} \vec{h}_{t-1} + \mathbf{b}_{\vec{h}}) \\
\vec{h}_t &= g(\mathbf{W}_{\vec{x}} \vec{x}_t + \mathbf{W}_{\vec{h}} \vec{h}_{t-1} + \mathbf{b}_{\vec{h}}) \\
y_t &= \mathbf{W}_{\vec{y}} \vec{h}_t + \mathbf{W}_{\vec{h}} \vec{h}_t + \mathbf{b}_y
\end{align*}
\]

(4.3)  (4.4)  (4.5)

The equations of both the RNN and the BRNN iterate from \( t = 1 \) to \( T \). The standard RNN is denoted as RNN in the sequel. Each hidden state can store all the input historical information theoretically. However, they face the problem of “vanishing gradient” or “exploding gradient” which states that the gradients tend to either vanish or explode exponentially, making the gradient-based optimization methods struggle. A lot of efforts have been made to address the issue. The LSTM and the GRU are two popular solutions.

### 4.3.2 LSTM

The LSTM addresses the problem of “vanishing gradient” by replacing the self-connected hidden units with memory blocks as illustrated in Figure 4.2a. The memory units enable the network to be aware of when to learn new information and when to forget old information. We follow the implementation of the LSTM unit described in [149] which is a slight simplification of the one described in [150]. The hidden state \( \mathbf{h}_t \) given input \( \mathbf{x}_t \) is computed as follows:

\[
\begin{align*}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi} \mathbf{x}_t + \mathbf{W}_{hi} \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf} \mathbf{x}_t + \mathbf{W}_{hf} \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo} \mathbf{x}_t + \mathbf{W}_{ho} \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\bar{\mathbf{c}}_t &= \tanh(\mathbf{W}_{xc} \mathbf{x}_t + \mathbf{W}_{hc} \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \bar{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{align*}
\]

(4.6)  (4.7)  (4.8)  (4.9)  (4.10)  (4.11)

where \( \mathbf{i} \) is an input gate modulating how much new memory content is added to the memory, \( \mathbf{f} \) is a forget gate modulating how much existing memory is forgotten, and \( \mathbf{o} \) is an output gate modulating the amount of memory content exposure. The memory cell \( \mathbf{c}_t \) consists of two components, namely partially forgotten previous memory \( \mathbf{c}_{t-1} \) and modulated new memory \( \bar{\mathbf{c}}_t \). \( \sigma(\cdot) \) and \( \tanh(\cdot) \) are sigmoid and
4.3. The Proposed Model

The proposed model utilizes long short-term memory (LSTM) units for handling complex temporal dependencies in the data. The LSTM unit consists of an input gate, forget gate, output gate, and cell gate. Each gate controls the flow of information into and out of the cell state.

\[
\begin{align*}
& z_t = \sigma(W_{xz} x_t + W_{zh} h_{t-1} + b_z) \\
& r_t = \sigma(W_{xr} x_t + W_{hr} h_{t-1} + b_r) \\
& \tilde{h}_t = tanh(W_{xh} x_t + W_{hh} (r_t \odot h_{t-1}) + b_h) \\
& h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t
\end{align*}
\]

Figure 4.2: Illustration of (A) the LSTM unit and (B) the GRU unit. Weight matrices are left out for clarity.

The gated recurrent unit (GRU) is similar to the LSTM unit but without a separate memory cell. The GRU also exploits gates to modulate the existing memory and new memory. Figure 4.2b gives an illustration of the GRU. The GRU exposes its full content because it does not have an output gate. It controls the information flow from previous activation when computing new candidate activation by using a reset gate. The amount of previous activation and new candidate activation into new activation are tied by an update gate. The hidden state \( h_t \) given input \( x_t \) is computed as follows:

\[
\begin{align*}
& z_t = \sigma(W_{xz} x_t + W_{zh} h_{t-1} + b_z) \\
& r_t = \sigma(W_{xr} x_t + W_{hr} h_{t-1} + b_r) \\
& \tilde{h}_t = tanh(W_{xh} x_t + W_{hh} (r_t \odot h_{t-1}) + b_h) \\
& h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t
\end{align*}
\]

4.3.3 GRU

The gated recurrent unit (GRU) is very similar to the LSTM unit but without a separate memory cell. The GRU also exploits gates to modulate the existing memory and new memory. Figure 4.2b gives an illustration of the GRU. The GRU exposes its full content because it does not have a output gate. It controls the information flow from previous activation when computing new candidate activation by using a reset gate. The amount of previous activation and new candidate activation into new activation are tied by an update gate. The hidden state \( h_t \) given input \( x_t \) is computed as follows:

\[
\begin{align*}
& z_t = \sigma(W_{xz} x_t + W_{zh} h_{t-1} + b_z) \\
& r_t = \sigma(W_{xr} x_t + W_{hr} h_{t-1} + b_r) \\
& \tilde{h}_t = tanh(W_{xh} x_t + W_{hh} (r_t \odot h_{t-1}) + b_h) \\
& h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t
\end{align*}
\]
where $z$ is the update gate and $r$ is the reset gate. The gate values also lie within the range $[0, 1]$. Similar to the LSTM, the GRU also demonstrates strong capability of modeling long-term sequences. We can also obtain bidirectional GRU (BGRU) by replacing the hidden states in BRNN with the GRU. Besides, the depth of the standard RNN/BRNN, the LSTM/BLSTM and the GRU/BGRU can be added by stacking one layer on top of another.

### 4.3.4 Comprehensive Attention Recurrent Neural Networks

Although the bidirectional recurrent models can access preceding and succeeding contexts of a position in a sequence, they still miss the local context information for the position. In this chapter, we incorporate a convolutional structure into the recurrent models to capture the local information contained in the position of a sentence. The local context representations of sentences are combined with the historical context representations captured by the forward recurrent model and future context representations captured by the backward recurrent model. The comprehensive context representations are fed into another recurrent layer to obtain the sentence representations. Finally, a softmax classifier is used for sentiment labeling. We propose three new models in this chapter, namely comprehensive attention recurrent neural network (CA-RNN), comprehensive attention long short-term memory (CA-LSTM) and comprehensive attention gated recurrent unit (CA-GRU). The model of the CA-RNN is depicted in Figure 4.3. Replacing the RNN by the LSTM and the GRU gives the models the CA-LSTM and the CA-GRU respectively.

**Context Representation**

The historical and future contextual information can be accessed by the bidirectional recurrent models introduced earlier in this section. Here, we will use a convolutional structure to capture the local context.

Convolutional layers can encode significant information about input data with significantly fewer parameters than other deep learning architectures. Empirical experiences in the area of computer vision suggest that deep architectures with multiple convolutional layers are necessary to achieve good performance [91]. However, only
one convolutional layer was used to achieve satisfactory performance in the sentence classification task by Kim in [46]. We adopt Kim’s implementation in this chapter. The convolution operation in our model is conducted in one dimension. In the convolutional layer, $k$ filters $\mathbf{W}_{lc} \in \mathbb{R}^{md \times k}$ convolved with a window of $m$ word vectors $\mathbf{x}_{i:i+m-1}$, obtaining features for the window of words in the corresponding feature maps. We use multiple filters because it has been demonstrated that filters with differently initialized weights are able to improve the model’s learning capability. The number of filters $k$ is determined using cross-validation and the convolution operation is governed by:

$$\mathbf{Lc}_i = g(\mathbf{W}_{lc}^T \mathbf{x}_{i:i+m-1} + \mathbf{b}_{lc}) \in \mathbb{R}^k$$

(4.16)

where $\mathbf{x}_i \in \mathbb{R}^d$, $d$ is the word vector dimension. The term $\mathbf{LC}_i$ is the local context representation at the $ith$ position, $\mathbf{b}_{lc}$ is a bias vector and $g(\cdot)$ is a nonlinear activation function. We also employ the special version of the ReLU named LeakyReLU [120] in this model.

Suppose the length of a sentence is $n$. We set the border mode of convolution as
Figure 4.4: Illustration of ‘same’ convolution graph. The window size of the filter is 3 and weights are shared across different windows. The arrows with the same colors represent the same weight values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this thesis.)

“same” so that the output length of convolution layer is the same as that of the input. This kind of convolution is shown in Figure 4.4. The border needs zero-paddings to guarantee the same length. As the word window slides, the feature maps of the convolutional layer can be represented as follows:

\[ \text{Lc} = [L_{c1}, L_{c2}, \cdots, L_{cn}] \]  

(4.17)

We use \( \text{Hc} = [H_{c1}, H_{c2}, \cdots, H_{cn}] \) and \( \text{Fc} = [F_{c1}, F_{c2}, \cdots, F_{cn}] \) to represent the historical and future context representations obtained by forward and backward recurrent models respectively. Then, the comprehensive attention (comprehensive context) representations of a sentence can be calculated as follows:

\[ s = W_h \text{Hc} + W_f \text{Fc} + W_l \text{Lc} \]  

(4.18)

where \( W_h \), \( W_f \) and \( W_l \) are the weights of historical, future and local context representations respectively. These weights can map the different context representations with different dimensions into the same attention space. The comprehensive attention representations are fed into another recurrent layer to obtain the sentence representations. We use the LSTM as the additional recurrent layer in our model.
4.4. Experimental Studies and Discussions

**Softmax Classifier**

The sentence representations $s$ are naturally regarded as features of sentences for sentiment classification. A linear transformation layer and a softmax layer are added on top of the model to produce the conditional probabilities over the class space. To avoid overfitting, dropout with a masking probability $p$ is applied on the penultimate layer. This is calculated as follows:

$$y_c = W_{sc}(s \odot q) + b_c \quad (4.19)$$

$$P_c = \frac{\exp(y_c)}{\sum_{c' \in C} \exp(y_{c'})} \quad (4.20)$$

where $\odot$ is an element-wise multiplication operator, $q$ is the masking vector with $p = 0.5$, and $C$ is the class number.

As our model is a supervised method, each sentence has its gold sentiment label $P^g_c$. The objective function in terms of minimizing the cross-entropy is used:

$$L = - \sum_x \sum_{c=1}^C P^g_c(x) \log(P_c(x)) \quad (4.21)$$

where $P^g_c$ has a 1-of-K coding scheme, where the dimension corresponding to the true class is 1 with all others being 0. The model parameters including word embedding are all fine-tuned via stochastic gradient descent using the RMSprop [38] update rule, which has been shown as an effective and efficient backpropagation algorithm.

The entire learning algorithms of comprehensive attention recurrent models are summarized as Algorithm 2.

4.4 Experimental Studies and Discussions

4.4.1 Dataset

We evaluate the performance of our proposed models on two benchmark sentiment analysis datasets: the IMDB Large Movie Review Dataset (IMDB) dataset [127]
Algorithm 2 Pseudo-code for Comprehensive Attention Recurrent Models

1: Construct word embedding table using pre-trained word vectors with Equation (2.15);
2: Employ the forward RNN/LSTM/GRU to obtain the historical context representations \( H_c = [H_{c1}, H_{c2}, \cdots, H_{cn}] \) using Equations (4.1, 4.11, and 4.15);
3: Employ the backward RNN/LSTM/GRU to obtain the future context representations \( F_c = [F_{c1}, F_{c2}, \cdots, F_{cn}] \), using variants of Equations (4.1, 4.11, and 4.15) with respect to Equation (4.4);
4: Exploit CNN to obtain the local context representations \( L_c = [L_{c1}, L_{c2}, \cdots, L_{cn}] \), using Equation (4.16);
5: Combine the historical, future, and local context representations with weighted sum method to obtain the comprehensive context representations using Equation (4.18);
6: Obtain the final sentence representations using another LSTM;
7: Feed the sentence representations into a softmax classifier to get the class labels;
8: Update parameters of the model using the loss function Equation (4.21) with the RMSprop method.

and Stanford Sentiment Treebank (SST) dataset [144]. The IMDB dataset consists of 50,000 binary labeled reviews from IMDB. These reviews are split 50:50 into training and testing sets. The average length of the IMDB reviews is 268 with a standard deviation of 199 tokens. We preprocess the dataset following the implementation of [152]. The SST dataset consists of 11,855 reviews from Rotten Tomatoes. The train, dev and test sets have 8,544, 1,101, and 2,210 reviews respectively. The average review length is 20 tokens with a standard deviation of 9.3. The SST reviews are labeled on a 5 point scale corresponding to very negative, negative, neutral, positive, and very positive. The SST dataset can be used for both multiple classification and binary classification. We denote this dataset for fine-grained classification task as SST1 and binary classification SST2. For SST2, neutral reviews are removed.

4.4.2 Parameter Settings

The word embeddings are initialized with both random vectors and off-the-shelf \textit{word2vec} vectors. The dimension of each word vector is 300. The number of hidden layer nodes of recurrent layers and the number of filters of convolution layers are set as 100 for all the tasks. The training batch size for IMDB is set as 100 while 1http://deeplearning.net/tutorial/lstm.html
that for SST is 50. The learning rate, decay factor, and fuzzy factor of RMSprop are set as 0.001, 0.9, and $1e^{-6}$. All the simulation studies are conducted using a GeForce 510 GPU on a Windows PC with 2.0 GHz CPU and 4 GB RAM.

4.4.3 Experiment Results

The performance of the new method is compared with several basic baseline methods and state-of-the-art approaches. The comparison systems are described as follows:

**SVM-unigram** [153]: is the bag-of-words model using the SVM as the classifier.

**SVM-bigram**: is the same as SVM-unigram except using bigram features.

**Averaged Word Vector**: averages the vectors for words contained in a text to obtain the sentence representations.

**Recursive Neural Network** [144]: constitutes the sentence representations from words in a bottom-up fashion on tree structures. It is also a deep learning method using pre-trained word vectors.

**Paragraph Vector** [102]: learns representations for sentences or paragraphs in the same way as learning word vectors using CBoW and Skip-grams.

The comparison results are demonstrated in Table 4.1. The accuracies of comparison systems are retrieved from their original papers. We can see that the bag-of-words models (both unigram and bigram) and the Averaged Word Vector method perform very well on the IMDB dataset. These three models all lose order information but achieve satisfactory performance. This may be because the order information is not very significant for long reviews like the IMDB dataset. In comparison, the complicated Recursive Neural Networks method does not generalize very well on the IMDB dataset. This is because tree construction for a long sentence or document can be very hard while the tree construction process is very important for the final performance of Recursive Neural Networks. Although our comprehensive recurrent models under-perform the current state-of-the-art Paragraph Vector method, they outperform the three order-loss methods and Recursive Neural Networks.

The experimental results on the Stanford Sentiment Treebank datasets (both fine-grained and binary classification) demonstrate that the proposed models can achieve competitive performance compared with the Paragraph Vector method. The new
Table 4.1: Classification accuracy (%) comparison results on three sentiment tasks.

<table>
<thead>
<tr>
<th>Methods</th>
<th>IMDB</th>
<th>SST1</th>
<th>SST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-unigram</td>
<td>86.9</td>
<td>79.4</td>
<td>40.7</td>
</tr>
<tr>
<td>SVM-bigram</td>
<td>89.2</td>
<td>83.1</td>
<td>41.9</td>
</tr>
<tr>
<td>Averaged Word Vector</td>
<td>88.3</td>
<td>78.6</td>
<td>40.6</td>
</tr>
<tr>
<td>Recursive Neural Networks</td>
<td>87.0</td>
<td>82.4</td>
<td>43.2</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td>92.7</td>
<td>87.8</td>
<td>48.7</td>
</tr>
<tr>
<td>CA-RNN</td>
<td>89.0</td>
<td>86.5</td>
<td>48.1</td>
</tr>
<tr>
<td>CA-LSTM</td>
<td>90.1</td>
<td>87.7</td>
<td>48.5</td>
</tr>
<tr>
<td>CA-GRU</td>
<td>90.1</td>
<td>87.2</td>
<td>48.9</td>
</tr>
</tbody>
</table>

Comprehensive recurrent models perform much better than the bag-of-words models and the other two word embedding-based methods (the Averaged Word Vector and the Recursive Neural Networks). The empirical performance results prove the effectiveness of new models.

In order to explore the effect of comprehensive context representation, we do another experiment on IMDB dataset to compare the new model with the standard recurrent models and the bidirectional recurrent models. All the settings are the same except for the contextual information extraction layer. While the standard recurrent models only have forward recurrent layer, the bidirectional recurrent models have both forward and backward recurrent layers. In comparison, the new models contain another convolutional layer (See Figure 4.3).

Performance comparison results are given in Table 4.2. We use precision, recall and f1-score as measure metrics. We can see from the table that the LSTM and the GRU achieve similar performance results. This is because the two models are actually very similar. The table also shows that the LSTM and the GRU outperform the standard RNN. The introduction of gate mechanism helps the LSTM and the GRU to handle more complex and longer temporal dynamics. When comparing the three architectures, we find that the bidirectional architecture performs better than the standard architecture for all the three recurrent methods. It turns out that the comprehensive attention architecture achieves the best performance for all the three metrics. The experiment results demonstrate that capturing comprehensive contextual information enhances the classification accuracy.

We also compare the empirical results initialized with random word embedding and
Table 4.2: Performance comparison results (%) on IMDB dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Recall</th>
<th>Precision</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>87.58</td>
<td>87.83</td>
<td>87.75</td>
</tr>
<tr>
<td>BRNN</td>
<td>88.85</td>
<td>89.01</td>
<td>88.93</td>
</tr>
<tr>
<td>CA-RNN</td>
<td>89.32</td>
<td>89.44</td>
<td>89.38</td>
</tr>
<tr>
<td>LSTM</td>
<td>87.86</td>
<td>88.24</td>
<td>88.05</td>
</tr>
<tr>
<td>BLSTM</td>
<td>89.56</td>
<td>89.24</td>
<td>89.40</td>
</tr>
<tr>
<td>CA-LSTM</td>
<td>90.04</td>
<td>90.14</td>
<td>90.09</td>
</tr>
<tr>
<td>GRU</td>
<td>88.06</td>
<td>88.29</td>
<td>88.17</td>
</tr>
<tr>
<td>BGRU</td>
<td>89.36</td>
<td>89.38</td>
<td>89.37</td>
</tr>
<tr>
<td>CA-GRU</td>
<td>89.97</td>
<td>90.24</td>
<td>90.11</td>
</tr>
</tbody>
</table>

Figure 4.5: F1-score (%) comparison by impact of initialized word embedding. From top to bottom are GRU, LSTM and RNN successively.

off-the-shelf word embedding. The F1-score comparison results of the three recurrent architectures are depicted in Figure 4.5. It is clear that superior performance is obtained through the use of pre-trained word embedding. The results suggest that the pre-trained word vectors are good universal feature extractors and can be utilized across datasets.

4.5 Summary

In this chapter, three novel recurrent models (CA-RNN, CA-LSTM and CA-GRU) are developed by incorporating a convolutional structure into the bidirectional recurrent models. The new architecture exploits bidirectional models to access the
past and future contextual information and a convolutional layer to capture local contextual information. The combined preceding, succeeding and local context representations are encoded by another long short-term memory layer to form the sentence representations, which are used as features of the sentences and inputs of a softmax classifier. The new model can store comprehensive information for any position in a sequence. Our new model can be trained end-to-end without any human intervention. The experiment results on the sentiment analysis datasets demonstrate that the new architecture is superior to the standard recurrent models and the bidirectional recurrent models in terms of classification accuracy.
Chapter 5

Sentence Modeling

5.1 Introduction

The sentence modeling problem is at the core of the field of natural language processing and has received a great deal of attention recently. The main objective of sentence modeling is to learn representations of sentences which are inputs of tasks like sentiment analysis, document summarization, machine translation, discourse analysis, etc. Feature representation is a key component of many machine learning systems because the performance of a machine learner depends heavily on it [91]. Sentence features are usually extracted by performing composition over features of words or n-grams.

With advances in deep learning techniques, distributed vectorial representation has become a common practice for word representation. The simplest sentence modeling method based on word embedding may be the continuous bag-of-words model which employs max-pooling or average-pooling over representations of all words in a sentence to compose the representation of that sentence. The model achieves very good performance on a variety of tasks but faces the problem of losing word order which is critical for semantic analysis. A number of other neural sentence models have been proposed to capture the word order. They leverage on neural networks to construct non-linear interactions between words so as to learn sentence representations which can well capture the semantics of sentences.
The recursive neural networks of [44, 144, 154] rely on parse trees to compose word vectors into sentence vectors. The composition procedure is recursively applied to child nodes in the parse tree in a bottom-up manner to generate hidden representations of parent nodes until reaching the root of the tree, whose representation is the sentence representation. However, the learning performance of recursive neural networks depends heavily on the construction of the textual tree while the tree construction can be very time-consuming.

The recurrent neural networks of [128, 130] are special cases of the recursive neural networks which can only compose word vectors from one end to the other and therefore can be regarded as formed through time. The recurrent neural networks can handle variable-length sequences and model contextual information dynamically. However, they face the problem of “vanishing gradient” or “exploding gradient” [131] which states that the gradients tend to either vanish or explode exponentially, making the gradient-based optimization method inefficient and ineffective.

Another method termed Paragraph Vector [102] is also very powerful. The method learns representations for sentences or paragraphs in the same way as learning word vectors using CBoW or Skip-grams [29, 30]. It is an unsupervised algorithm that learns fixed-length feature representations from variable-length texts. It achieves amazingly good performance on some tasks. However, other researchers find that it performs sub-optimally on other tasks.

The convolutional neural network, first proposed for computer vision tasks [27], has been proven to be a powerful semantic composition model for modeling sentences [46, 96, 129, 155]. A standard convolutional neural network is usually constituted by several convolutional and pooling layers at the bottom of a linear or non-linear classifier. A pooling function is applied to the feature map obtained by each convolutional filter to reduce the spatial size of the vector representation and induce a fixed length vector. Next, the feature vectors for all the filters are concatenated to form a single feature vector which is used as an input to the classifier.

All the existing pooling strategies discard information contained in the context to some extent, which may affect the semantic extraction procedure. In this chapter, a new convolutional neural network model termed Attention Pooling-based Convolutional Neural Network is developed to address the problem. A new pooling
scheme termed *Attention Pooling* is proposed to retain the most significant information at the pooling stage. The bidirectional long short-term memory model is employed to enhance the information extraction capability of the pooling layer. The bidirectional long short-term memory model is also combined with the convolutional structure to extract comprehensive information, namely historical, future and local context information, of any position in a sequence. Therefore, the new model retains more information contained in sentences and is able to generate more representative feature vectors for sentences.

Our new model can be trained end-to-end with limited hyper-parameters and it is very easy to implement. We conduct experiments on several sentence classification tasks. Experiment results demonstrate that the new model outperforms state-of-the-art approaches on seven benchmark datasets. It should be highlighted that the absolute classification accuracy on Stanford Treebank Datasets is even significantly improved by over two percent. The new attention pooling scheme is shown to be more effective than existing pooling strategies. The proposed sentence model can even implicitly separate sentences from different classes in the semantic space which is a very powerful ability for classification. In summary, the main contributions of this work are as follows:

- A new convolutional neural network model is developed and a new pooling scheme termed *Attention Pooling* is proposed to retrieve the most significant information at the pooling stage. The proposed model does not need external modules and can be trained end-to-end.

- The combination of the bidirectional long short-term memory model and convolutional structure enables the model to extract comprehensive information in sentences. This further improves the learning capacity because comprehensive context information is very significant for extracting semantics in sentences.

- Empirical results on text classification datasets demonstrate that the proposed model can achieve higher accuracy compared with state-of-the-art approaches.

This chapter is based on the paper: *Attention pooling-based convolutional neural network for sentence modelling* [8].
### 5.2. Related Works

Pooling is a significant component of the convolutional neural network (CNN). The pooling function can reduce the number of parameters in the model and thus alleviate the problem of over-fitting. Max pooling is the most widely used pooling operator which returns the maximum value of a set of values. The 1-max pooling is applied over the entire feature map, inducing a feature vector of length 1 for each convolutional filter. By only keeping the biggest value, max pooling can capture the most relevant feature. However, the max pooling has some disadvantages as well. It forgets position information of the features because only the maximum value is used in the pooling stage. The intensity information of the same feature is also lost because it cannot distinguish whether a feature occurs once or in multiple times.

Local max pooling [156] is applied over small local regions of the feature map, producing a number for each local region. Next, the numbers are concatenated to form the representation vector for the feature map. In [96], Kalchbrenner et al. proposed an effective pooling strategy termed k-max pooling which extracts the k maximum values of the feature map and preserves the relative order of these values. Both local max pooling and k-max pooling are able to preserve some position information and intensity information of features. It can be easily seen that 1-max pooling is a special case of either local max pooling or k-max pooling. Another popular pooling function is average pooling [157] which returns the average value of the feature maps. It was demonstrated in [158] that max pooling outperforms average pooling in most applications. The descriptions of related pooling strategies are summarized in Table 5.1. All these pooling strategies discard information contained in the context to some extent, which may affect the semantic extraction procedure.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>methods</th>
<th>dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-max</td>
<td>( \max([c_1, c_2, \cdots, c_T]) )</td>
<td>1</td>
</tr>
<tr>
<td>local-max</td>
<td>( \max([c_1, c_T]), \max([c_{T+1}, c_{T+1}]), \cdots, \max([c_{T_n-1}, c_{T_n}]) )</td>
<td>n</td>
</tr>
<tr>
<td>k-max</td>
<td>( \text{k-max}([c_1, c_2, \cdots, c_T]) )</td>
<td>k</td>
</tr>
<tr>
<td>average</td>
<td>( \text{average}([c_1, c_2, \cdots, c_T]) )</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 5. Sentence Modeling

5.3 The Proposed Model

We describe the proposed model in details in this section. Figure 5.1 depicts the structure of the new model. Convolutional filters perform convolutions on the input sentence matrix and generate local representations. The convolution operation can independently capture local information contained in every position of a sentence. An attention pooling layer is used to integrate local representations into the final sentence representation with attention weights. These weights are obtained by comparing local representations position by position with an intermediate sentence representation generated by the bidirectional long short-term memory (BLSTM) [131, 151] and optimized during the training phase. At last, sentence representations of all distinct convolutional filters are concatenated into the final
feature vector which is fed into a top-level softmax classifier. The intermediate sentence representation generated by the BLSTM will also be used as an input to the softmax classifier in the testing phase, which is indicated by the dashed-lines in Figure 5.1.

The two salient contributions, namely the new pooling scheme and the combination of the BLSTM model with the convolutional structure, of the proposed model are described in detail in Section 5.3.3. The two novel components enable the model to extract comprehensive semantic information which is very important for good classification performance. The other necessary components of our method, namely word embedding (Section 5.3.1), convolution methods (Section 5.3.2), parallel CNNs (Section 5.3.4) and softmax classifier (Section 5.3.5) are also introduced so that the description is complete and comprehensive.

5.3.1 Word Embedding

The input of the algorithm is \( N \) variable-length sentences. Each sentence \( S \) is constituted by words which are represented by vectors. Traditional word representations, such as one-hot vectors, achieve good learning performance in the task of document classification [129]. However, one-hot vectors may face the problem of the curse of dimensionality when used to classify short sentences because of sparsity. Recent research results have demonstrated that continuous word representations are more powerful. We use distributed word vectors to represent words. The values of word vectors are included in the parameters which are optimized during the training procedure. Each sentence is represented as a matrix after representing each word within it as a dense vector using Equation (2.15).

5.3.2 Convolution Layer

Convolutional layers play critical roles in the success of the CNN because they can encode significant information contained in input data with significantly fewer parameters than other deep learning architectures. Empirical experiences in the area of computer vision suggest that deep architectures with multiple convolutional layers are necessary to achieve good performance [91]. However, only one convolutional layer was used to achieve state-of-the-art or comparable performance on
several datasets of the sentence classification task in [46]. In some cases, the performance only increases marginally or even decreases because of over-fitting with increasing number of convolutional layers. Furthermore, the computation complexity increases quickly if more layers are used. In this chapter, we also use only one convolution layer and carry out experiments on the same datasets as those of [46].

The convolution operation in our model is also conducted in one dimension between \( k \) filters \( W_c \in \mathbb{R}^{md \times k} \) and a concatenation vector \( x_{i:i+m-1} \) which represents a window of \( m \) words starting from the \( i \)th word, obtaining features for the window of words in the corresponding feature maps. The parameters of each filter are shared across all the windows. The number of filters \( k \) is determined using cross-validation and the convolution operation is governed by

\[
c_i = g(W_c^T x_{i:i+m-1} + b_c) \in \mathbb{R}^k
\]

where \( x_i \in \mathbb{R}^d \), \( d \) is the dimension of word vectors. The term \( b_c \) is a bias vector and \( g(\cdot) \) is a nonlinear activation function. Like in Chapter 3 and Chapter 4, we also employ the special version the ReLU called LeakyReLU [120] in this model.

Suppose the length of a sentence is \( T \). We set the border mode of convolution as ‘same’ so that the output length of the convolution layer is the same as that of the input. This kind of convolution is shown in Figure 4.4. The border needs zero-padding to guarantee the same length. As the word window slides, the feature maps of the convolutional layer can be represented as follows:

\[
c = [c_1, c_2, \cdots, c_T] \in \mathbb{R}^{k \times T}
\]

The output of the convolutional layer represents local representations of the sentence and each element \( c_i \) is a local representation of the corresponding position.

### 5.3.3 Attention Pooling

Most existing convolutional neural networks take advantage of max pooling to reduce parameters and induce fixed length vector. However, max pooling loses position and intensity information of features which may deteriorate the learning performance. In this chapter, an innovative pooling strategy termed Attention Pooling is proposed.
First, as shown in Figure 5.1, an intermediate sentence representation is needed. The intermediate representation is generated by the BLSTM [131, 151]. The BLSTM is a variant of the recurrent neural network which can learn both historical and future information contained in a sequence. It is also able to address the problem of “vanishing gradient” by replacing the hidden state of the recurrent neural network with a gated memory unit. With regard to the LSTM unit, we follow the implementation described in [149]. The working mechanism of the LSTM can be found in Chapter 4.3.2. We denote the intermediate sentence representation as \( \tilde{s} \).

Once the intermediate sentence representation is generated, we can compare local representations generated by the convolutional layer with it to calculate the attention weights. In order to compare the two representations, we should map both local representation and intermediate sentence representation to the space of the same dimension. This can be achieved by controlling the output dimension of the BLSTM same as the number of convolutional filters \( k \). The higher the similarity between the intermediate sentence representation and each local representation, the bigger attention weight is assigned to that local representation. The attention weights are calculated as follows:

\[
\alpha_i = \frac{\exp(e_i)}{\sum_{i=1}^{T} \exp(e_i)}
\]  

(5.3)

where

\[
e_i = \text{sim}(c_i, \tilde{s})
\]  

(5.4)

The term \( \alpha_i \) is a scalar and the function \( \text{sim}(\cdot) \) is used to measure the similarity between its two inputs. Cosine similarity is used in our model. After the attention weights are obtained, the final sentence representation is given by:

\[
s = \sum_{i=1}^{T} \alpha_i c_i \in \mathbb{R}^k
\]  

(5.5)

The BLSTM model is jointly trained with all the other components of the model. The gradients of the cost function back-propagate through the intermediate sentence representation so that it is optimized during the training phase. As such, no external modules are needed in our model and the model can be trained end-to-end. The intermediate sentence representation obtained by the BLSTM model will
be concatenated with the sentence representation obtained by the convolutional structure to form the input of the top classifier in the testing phase.

The attention pooling can be regarded as taking a weighted sum of all the word annotations to compute the sentence annotation. The weight of each word measures how much the word contributes to the meaning of the entire sentence. This method borrows the idea of one very effective mechanism namely ‘attention’ used in recent years in various tasks, like machine translation [12], object recognition [159], and image captioning [160], etc. Intuitively, the model can decide which features to pay attention to. Compared with the max pooling methods, the attention pooling method is able to reserve more information contained in the sentence. Compared with the average pooling method, the new pooling strategy has an obvious advantage by assigning bigger weights to more significant features.

The other important component of the new model is the combination of the BLSTM and the convolutional structure. The intermediate sentence representation generated by the BLSTM should already be a sufficiently satisfactory input representation for the top classifier. By comparing local representations with this intermediate sentence representation, the obtained attention weights encode richer information of the sentence. On the other hand, the local context extraction capacity of convolutional structure improves the information retrieval ability compared with just using the BLSTM. The proposed model is able to access comprehensive information, namely historical, future and local context of any position in a sequence. In summary, the position and intensity information of features are completely preserved with the help of the proposed innovative attention mechanism. This is what we aim to achieve at the pooling stage so as to overcome disadvantages of existing pooling strategies.

\subsection*{5.3.4 Parallel CNNs}

The CNN architecture described so far is very simple with only one convolutional layer and one attention pooling layer. Following the approach in [46, 129], we also use filters with varying convolution window sizes to form parallel CNNs so that they can learn multiple types of embedding of local regions so as to complement each other to improve model accuracy. The convolutional layers with distinct window sizes are depicted in Figure 5.2. Sentence representations produced by all
5.3. The Proposed Model

Figure 5.2: Illustration of parallel convolution procedure. It involves $H$ parallel convolution layers with varying convolution window sizes and each layer has $k$ filters. Each convolution layer is followed by an attention pooling layer to generate a distinct sentence representation. The $H$ distinct representations are then concatenated to form the final representation vector.

the distinct CNNs are concatenated to form the final feature vector as an input to the top softmax classifier.

5.3.5 Softmax Classifier

The sentence representation $s$ is naturally regarded as an input to the top classifier during the training phase while $[s, \bar{s}]$ is used at the testing phase. A linear transformation layer and a softmax layer are added at the top of the model to produce conditional probabilities over the class space. To avoid overfitting, dropout with a masking probability $p$ is applied to the penultimate layer. This output layer is calculated as follows:

$$y = \begin{cases} W_s(s \odot q) + b_s & \text{training phase} \\ W_s([s, \bar{s}]) + b_s & \text{testing phase} \end{cases}$$ \hspace{1cm} (5.6)

$$P_c = \frac{\exp(y_c)}{\sum_{c' \in C} \exp(y_{c'})}$$ \hspace{1cm} (5.7)
where $\odot$ is an element-wise multiplication operator, $q$ is the masking vector with dropout rate $p$ which is the probability of dropping a unit during training, and $C$ is the class number. In addition, a $l_2$ norm constraint of the output weights $W_s$ is imposed during training as well.

As our model is a supervised method, each sentence $S$ has its golden label $P_g^c$. The following objective function in terms of minimizing the categorical cross-entropy is used:

$$L = - \sum_{i=1}^{N} \sum_{c=1}^{C} P_g^c(S_i) log(P_c(S_i))$$ (5.8)

where $P_g^c$ has a 1-of-K coding scheme whose dimension corresponding to the true class is 1 while all others being 0. The parameters to be determined by the model include all the weights and bias terms in the convolutional filters, the BLSTM and the softmax classifier. The attention weights will be updated during the training phase. Word embeddings are fine-tuned as well. Optimization is performed using the Adadelta update rule of [40], which has been shown as an effective and efficient back-propagation algorithm.

The entire learning algorithm of APCNN is summarized as Algorithm 3.

## 5.4 Experimental Results and Discussions

In this section, we evaluate the performance of the proposed model on seven benchmark datasets for text classification and compare it with state-of-the-art approaches. The performance of the proposed model is evaluated by comparing it with other pooling strategies in Section 5.4.4. Statistical tests are carried to demonstrate that the improvement of our method over other approaches is statistically significant. A sensitivity analysis of four key parameters of the model is done in Section 5.4.5. The effectiveness of the Attention Pooling mechanism is verified in Section 5.4.6. We also visualize the sentence representation space in Section 5.4.7 in order to have a better understanding of the sentence distribution produced by our model.

We test our model on seven benchmark datasets for sentence classification tasks which are the same as those used in [46]. Statistics of the datasets are listed
Algorithm 3 Pseudo-code for Attention Pooling-based Convolutional Neural Network

1: The input and output of the proposed algorithm are $N$ variable-length sentences and their corresponding labels $P^g$. The labels have been represented by 1-of-K coding scheme.
2: Considering one sentence, construct the sentence matrix using pre-trained word vectors with Equation (2.15);
3: for $i$ in $[1, H]$ do
4:   For the $i$th convolutional neural network with window size $m_i$, exploit one convolutional layer to obtain the local representations $c = [c_1, c_2, \ldots, c_T]$ using Equation (5.1);
5:   Employ the bidirectional long-short term memory to obtain the intermediate sentence representation $\tilde{s}$;
6:   Calculate the attention weights using Equations (5.3-5.4);
7:   Obtain the sentence representation using attention pooling with Equation (5.5);
8: end for
9: Concatenate the $H$ sentence representations to form the final sentence feature vector;
10: Feed the final sentence representation into a softmax classifier to predict the class label;
11: With all the training sentences and labels, update parameters of the model using the loss function Equation (5.8) with the Adadelta update rule.
12: At testing phase, the intermediate sentence representation is concatenated with the sentence representation obtained by the convolutional and pooling layers to form the input of the softmax classifier.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of classes</th>
<th>Average sentence length</th>
<th>Maximum sentence length</th>
<th>Dataset size</th>
<th>Test-set size$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>2</td>
<td>20</td>
<td>56</td>
<td>10662</td>
<td>CV</td>
</tr>
<tr>
<td>SUBJ</td>
<td>2</td>
<td>23</td>
<td>120</td>
<td>10000</td>
<td>CV</td>
</tr>
<tr>
<td>CR</td>
<td>2</td>
<td>19</td>
<td>105</td>
<td>3784</td>
<td>CV</td>
</tr>
<tr>
<td>MPQA</td>
<td>2</td>
<td>3</td>
<td>36</td>
<td>10606</td>
<td>CV</td>
</tr>
<tr>
<td>SST-1</td>
<td>5</td>
<td>18</td>
<td>53</td>
<td>11855</td>
<td>2210</td>
</tr>
<tr>
<td>SST-2</td>
<td>2</td>
<td>19</td>
<td>53</td>
<td>9618</td>
<td>1821</td>
</tr>
<tr>
<td>TREC</td>
<td>6</td>
<td>10</td>
<td>37</td>
<td>5952</td>
<td>500</td>
</tr>
</tbody>
</table>

in Table 5.2. To facilitate the following discussion, the datasets are now briefly described:

$^1$CV means there is no standard train/test split and thus 10-fold CV is used
5.4.1 Datasets

- **MR [161]**: This is the dataset for movie reviews with one sentence per review. The objective is to classify each review into either positive or negative by its overall sentiment polarity. The class distribution of this dataset is 5331/5331.²

- **SUBJ [153]**: This is the subjectivity dataset where the goal is to classify a sentence as being subjective or objective. The class distribution is 5000/5000.

- **CR [162]**: This dataset gives customer review of various products (MP3s, cameras, etc.) where the objective is to classify each review into positive or negative class. The class distribution is 2411/1373.³

- **MPQA [163]**: This is the opinion polarity detection subtask of the MPQA dataset. The class distribution is 3310/7296.⁴

- **SST-1 [144]**: This is the Stanford Sentiment Treebank dataset, an extension of MR dataset but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative). The class distribution is 1837/3118/2237/3147/1516.⁵

- **SST-2**: This dataset is derived from SST-1, which only has binary labels by excluding neutral reviews. The class distribution is 4955/4663.

- **TREC [164]**: This is the question classification dataset where the objective is to classify a question into 6 question types (whether the question is about a person, location, numeric information, etc.). The class distribution is 1288/916/1009/1344/95/1300.⁶

5.4.2 Comparison Systems

The performance of the proposed method is compared with different basic baseline and state-of-the-art neural sentence models reviewed in the introduction section.

²https://www.cs.cornell.edu/people/pabo/movie-review-data/
³http://www.cs.uic.edu/iub/FBS/sentiment-analysis.html
⁴http://www.cs.pitt.edu/mpqa/
⁵http://nlp.stanford.edu/sentiment/ We train the model on both phrases and sentences but only score on sentences at test time, as in [46, 96, 102, 144]. Thus the training set is an order of magnitude larger than listed in Table 5.2
⁶http://cogcomp.cs.illinois.edu/Data/QA/QC/
5.4. Experimental Results and Discussions

Two Naive Bayes-based models are also included so as to compare with methods which do not use word embedding. The comparison systems are described as follows:

- **NB-SVM** and **MNB** (Naive Bayes SVM and Multinomial Naive Bayes): They are developed by taking Naive Bayes log-count ratios of uni- and bi-gram features as input to the SVM classifier and Naive Bayes classifier respectively [165]. The word embedding technique is not used.

- **cBoW** (Continuous Bag-of-Words): This model uses average or max pooling to compose a set of word vectors into a sentence representation.

- **RAE, MV-RNN** and **RNTN** (Recursive Auto-encoder [166], Matrix-vector Recursive Neural Network [167] and Recursive Neural Tensor Network [144]): These three models belong to recursive neural networks and recursively compose word vectors into sentence vector along a parse tree. Every word in the parse tree is represented by a vector, a vector and a matrix and a tensor-based feature function in RAE, MV-RNN, and RNTN respectively.

- **RNN** and **BRNN** (Recurrent Neural Network [128] and Bidirectional Recurrent Neural Network [130]): The RNN composes words in a sequence from the beginning to the end into a final sentence vector while the BRNN does the composition from both the beginning to the end and the end to the beginning.

- **CNN, one-hot CNN** and **DCNN** (Standard Convolutional Neural Network [46], One-hot vector Convolutional Neural Network [129] and Dynamic Convolutional Neural Network [96]): The CNN and the DCNN use pre-trained word vectors while the one-hot CNN employs high dimensional ‘one-hot’ vector representation of words as input. The CNN and the one-hot CNN employ max pooling and the DCNN uses k-max pooling at the pooling stage.

- **P.V.** (Paragraph Vector [102]): It learns representations for sentences or paragraphs in the same way as learning word vectors using CBOW and skip-grams.

The characteristics of the comparison systems are summarized in Table 5.3.

\[\text{RecursiveNN, RNN, and CNN stand for recursive neural networks, recurrent neural networks, and convolutional neural networks respectively.}\]
Table 5.3: Characteristics of the comparison systems

<table>
<thead>
<tr>
<th>Comparison systems</th>
<th>Word embedding</th>
<th>Category of neural systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB-SVM and MNB</td>
<td>×</td>
<td>–</td>
</tr>
<tr>
<td>cBoW</td>
<td>✓</td>
<td>average/max</td>
</tr>
<tr>
<td>RAE</td>
<td>✓</td>
<td>RecursiveNN</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>✓</td>
<td>RecursiveNN</td>
</tr>
<tr>
<td>RNTN</td>
<td>✓</td>
<td>RecursiveNN</td>
</tr>
<tr>
<td>RNN</td>
<td>✓</td>
<td>RNN</td>
</tr>
<tr>
<td>BRNN</td>
<td>✓</td>
<td>RNN</td>
</tr>
<tr>
<td>CNN</td>
<td>✓</td>
<td>CNN</td>
</tr>
<tr>
<td>one-hot CNN</td>
<td>×</td>
<td>CNN</td>
</tr>
<tr>
<td>DCNN</td>
<td>✓</td>
<td>CNN</td>
</tr>
<tr>
<td>P.V.</td>
<td>✓</td>
<td>CBOW/skip-gram</td>
</tr>
</tbody>
</table>

5.4.3 Parameter Settings

The authors of [158] provide a guide regarding CNN architecture and hyperparameters for practitioners who deploy CNNs for sentence classification tasks. We follow their suggestions to select parameters for our model. The paper shows that using word2vec or GloVec pre-trained word vectors may result in different performances, but with slight differences. We choose word2vec for all the datasets in this chapter. The dimension of each word vector is 300. For all tasks, the word vectors are fine-tuned during the training phase [46].

It is claimed in [158] that the filter window size and the number of feature maps may have large effects on the performance while regularization may have relatively small effects. We employ coarse grid search and cross-validation to determine these parameter values. As suggested by authors of [158], it is appropriate to set the region sizes for parallel CNNs near the single best size. We first determine the best single filter window size (odd number) and then use two adjacent region sizes. Three parallel CNNs are used in order to compare fairly with models in [129] and [46]. Settings for the four parameters for the datasets are listed in Table 5.4. However, sensitivity analysis over the four parameters shows that their choices do not affect the final performance too much as long as they are in an appropriate range.

The output dimension of the BLSTM for each dataset is set the same as the number of feature maps in order to compare local representations with intermediate sentence representation. The training batch size for SST-1/SST-2 is set as 100 while
5.4. Experimental Results and Discussions

Table 5.4: Parameter settings of different datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Region size</th>
<th>Feature maps</th>
<th>dropout rate</th>
<th>l2 norm constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>(4,5,6)</td>
<td>200</td>
<td>0.5</td>
<td>4</td>
</tr>
<tr>
<td>SUBJ</td>
<td>(6,7,8)</td>
<td>200</td>
<td>0.4</td>
<td>3</td>
</tr>
<tr>
<td>CR</td>
<td>(6,7,8)</td>
<td>200</td>
<td>0.3</td>
<td>4</td>
</tr>
<tr>
<td>MPQA</td>
<td>(4,5,6)</td>
<td>100</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>SST-1</td>
<td>(2,3,4)</td>
<td>100</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>SST-2</td>
<td>(2,3,4)</td>
<td>100</td>
<td>0.5</td>
<td>3</td>
</tr>
<tr>
<td>TREC</td>
<td>(2,3,4)</td>
<td>100</td>
<td>0.4</td>
<td>5</td>
</tr>
</tbody>
</table>

that for the other datasets as 50 because the number of training dataset size of SST-1/SST-2 is very large. Every experiment is conducted with 30 epochs. The learning rate, decay factor, and fuzzy factor of Adadelta are set as 1.0, 0.95, and $1 \times 10^{-6}$ respectively. Our model is developed based on keras\(^8\). All simulation studies are conducted using a GeForce 510 GPU on a Windows PC with 2.0 GHz CPU and 4 GB RAM.

5.4.4 Comparison of Classification Accuracy

The classification accuracy of APCNN compared with other approaches is given in Table 5.5. The results of NB-SVM, MNB, RAE, MV-RNN, RNTN, CNN and DCNN are extracted from their original papers. The results of one-hot CNN are extracted from [158]. Public implementation of the P.V. is used and the logistic regression is used on top of the pre-trained paragraph vectors for prediction. For cBoW, RNN, and BRNN, they were implemented by ourselves and the best results are reported.

We can see from the table that the two Bayesian models achieve very good performance on datasets with long sentences but perform not well on datasets with short sentences. This results from the sparsity of n-gram encoding for short sentences. The cBoW is not performing very well which should result from losing word order information in the sentence. It is surprising that the three recursive neural network structures, namely RAE, MV-RNN and RNTN do not achieve very satisfactory performances. Their performances largely rely on the construction of the parse trees. The models may be so complicated that the problem of over-fitting

\(^8\)http://keras.io
Table 5.5: Classification accuracy results of APCNN against other approaches on benchmark datasets

<table>
<thead>
<tr>
<th>System</th>
<th>MR</th>
<th>SUBJ</th>
<th>CR</th>
<th>MPQA</th>
<th>SST-1</th>
<th>SST-2</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB-SVM</td>
<td>79.4</td>
<td>93.2</td>
<td>81.8</td>
<td>86.3</td>
<td></td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>MNB</td>
<td>79.0</td>
<td>93.6</td>
<td>80.0</td>
<td>86.3</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BoW</td>
<td>77.2</td>
<td>91.3</td>
<td>79.9</td>
<td>86.4</td>
<td>42.8</td>
<td>81.5</td>
<td>87.3</td>
</tr>
<tr>
<td>RAE</td>
<td>77.7</td>
<td>–</td>
<td>–</td>
<td>86.4</td>
<td>43.2</td>
<td>82.4</td>
<td>–</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>79.0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>44.4</td>
<td>82.9</td>
<td>–</td>
</tr>
<tr>
<td>RNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>45.7</td>
<td>85.4</td>
<td>–</td>
</tr>
<tr>
<td>RNN</td>
<td>77.2</td>
<td>92.7</td>
<td>82.3</td>
<td>90.1</td>
<td>47.2</td>
<td>85.8</td>
<td>90.2</td>
</tr>
<tr>
<td>BRNN</td>
<td>81.6</td>
<td>93.2</td>
<td>82.6</td>
<td>90.3</td>
<td>48.1</td>
<td>86.5</td>
<td>91.0</td>
</tr>
<tr>
<td>CNN</td>
<td>81.5</td>
<td>93.4</td>
<td>84.3</td>
<td>89.5</td>
<td>48.0</td>
<td>87.2</td>
<td>93.6</td>
</tr>
<tr>
<td>one-hot CNN</td>
<td>77.8</td>
<td>91.1</td>
<td>78.2</td>
<td>83.9</td>
<td>42.0</td>
<td>79.8</td>
<td>88.3</td>
</tr>
<tr>
<td>DCNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>48.5</td>
<td>86.8</td>
<td>93.0</td>
</tr>
<tr>
<td>P.V.</td>
<td>74.8</td>
<td>90.5</td>
<td>78.1</td>
<td>74.2</td>
<td>48.7</td>
<td>87.8</td>
<td>91.8</td>
</tr>
<tr>
<td>APCNN</td>
<td><strong>82.5</strong></td>
<td><strong>94.3</strong></td>
<td><strong>85.8</strong></td>
<td><strong>90.7</strong></td>
<td><strong>50.1</strong></td>
<td><strong>89.9</strong></td>
<td><strong>93.9</strong></td>
</tr>
</tbody>
</table>

affects classification accuracy. Compared with RNN, the bidirectional structure helps BRNN extract more information and enhance the learning performance.

When we compare the two one-layer CNN structures with word vectors and one-hot vectors as input respectively, we find that the one-hot approach performs much worse than the word-embedding approach. This shows that one-hot CNN may not be suitable for sentence classification although it achieves good performance on document classification tasks in [129]. The reason may be that the sentences are too short to provide enough information for high-dimensional encoding, resulting in extreme sparsity. The classification accuracy of DCNN with multiple layers and k-max pooling does not improve too much compared with the one-layer CNN model, proving that simple models can already be very effective in achieving competitive performance. Another surprising result is that P.V. yields very bad performance on several datasets while it demonstrates state-of-the-art performance on almost all the tasks in [102]. This indicates that P.V. may only work for certain datasets.

We can conclude from the table that the APCNN consistently outperforms the other systems in all the tasks. We believe that it is the attention pooling strategy that helps the new model outperform the max pooling-based CNN approach because it can extract the most significant information contained in the sentence. Furthermore, the combination of the BLSTM model and convolutional structure enables the model to extract comprehensive information, namely historical, future
and local context of any position in a sequence. The effectiveness of the new model has been verified by experiments.

In order to verify that the performance improvement of the APCNN over other approaches is statistically significant, we perform some statistical tests. We also use paired comparison t-tests which have been introduced in Chapter 3.2.4. For each approach, we conducted the experiment ten times for statistical comparison (the accuracy is supposed to be the same for ten times if the results are extracted from the original papers). The associated p-values for each pair of approaches on all the datasets are all smaller than 0.001. Therefore, we can conclude that our proposed method obtains superior performance compared with the other methods at nearly 100% confidence.

5.4.5 Sensitivity Analysis

In this section, the sensitivity analysis over predefined parameters has been performed to confirm that they are not problem-specific. The four key predefined parameters are filter window size, the number of feature maps, dropout rate and $l_2$ norm constraint. They are not included in the parameters which are updated during the training phase. The authors of [158] claimed the filter window size and the number of feature maps may have large effects on the performance while regularization may have relatively small effects. However, Kim [46] demonstrated that dropout is a very good regularizer to improve learning performance. The two statements contradict each other in some way. Therefore, it is necessary to do a sensitivity analysis on the key parameters.

We conducted experiments on the seven datasets to evaluate the parameter effects. The basic configurations for the four parameters are set as (3, 100, 0.5, 3) for (filter window size, the number of the feature maps, dropout rate, $l_2$ norm constraint) respectively. When analyzing the effect of one parameter, we hold all the other parameters constant at the basic configuration values. Because the accuracy ranges of different datasets are quite different, we show the percent change compared with the base point rather than the actual accuracy for each dataset.

The effects of the four parameters are depicted in Figure 5.3. We can see from Figure 5.3a that different datasets may have distinct optimal filter window sizes. For
Figure 5.3: Sensitivity analysis of (A) filter window size, (B) number of feature maps, (C) dropout rate, and (D) $l_2$ norm constraint. The plots depict the percent changes of accuracy compared with basic configurations for the four parameters respectively on seven datasets. The basic configurations are set as (3, 100, 0.5, 3). In (C), the zero value of dropout rate means that no dropout is used. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

those datasets with very long sentences (e.g. CR and SUBJ), it is appropriate to choose a relatively large window size. Figure 5.3b demonstrates that the number of feature maps has a strong impact on the learning performance of the model. When the number of feature maps is small, the accuracy can be very small. However, as the size increases, the performance does not improve too much and can even deteriorate because of the problem of over-fitting. Furthermore, the model complexity increases quickly with the increasing number of feature maps. Therefore, we set the number of feature maps as 100 or 200 in our experiments.

The authors of [158] claimed that the dropout regularization does not help too much in performance improvement. However, our experiments indicate that the dropout is a very effective regularizer. As indicated by Figure 5.3c, the performance of some
datasets can improve a lot (more than 1%) with the help of dropout. We believe that the power of dropout can be even stronger if we use more complex models. However, it is true that the performance varies very little when the dropout rate ranges from 0.2 to 0.6. Too large values of dropout rate can lead to under-fitting thus resulting in very bad performance. We set the dropout rates for different datasets in the range of \([0.3, 0.5]\). At last, Figure 5.3d shows that the \(l_2\) norm constraint indeed has limited effect on the learning performance as long as the gradient does not become too large.

### 5.4.6 Comparison of Pooling Strategies

In this section, we verify the effectiveness of the attention pooling strategy by comparing with existing pooling strategies. For local max pooling, we set the pooling size as 10. For k-max pooling, \(k\) is set as 3. Only pooling strategies are different while all the other parameters are set the same. For our model, the intermediate sentence representation generated by the BLSTM model is not included in the input of the softmax layer in the testing phase for fairness in comparison. Comparison of accuracy is depicted in Figure 5.4. We leave SST-1 out for a better comparison because the accuracy range for SST-1 is much smaller than those of other datasets. The comparison result of SST-1 is similar to that of SST-2.

We can see from the figure that the attention pooling strategy outperforms all the other pooling strategies across all the datasets. The average pooling strategy performs the worst, showing that setting equal weights to all the features is unreasonable. It is surprising that 1-max pooling strategy achieves better performance than k-max and local max pooling in most cases although the latter two preserve more information. The 1-max pooling even achieves competitive performance in the dataset TREC compared with attention pooling. This demonstrates the significance of the maximal feature. After all, we can conclude that the new pooling scheme is effective because it achieves the best performance. The primary reason is that the new pooling scheme can extract the most significant information contained in the sentences.
Figure 5.4: Accuracy comparison by the impact of pooling strategies. Five pooling strategies are compared on six datasets. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.4.7 Visualization of Sentence Representation Space

To further show that the proposed model is able to learn appropriate sentence representations, we visualize the semantic space of sentence vectors of SST-2 dataset produced by our model by mapping the high-dimension sentence vector to a two-dimension plane. This mapping procedure is done by using PCA and t-SNE [168]. We first map the sentence representation vectors to the 50-dimensional space with PCA and then map the obtained 50-dimensional vectors to the 2-dimension plane with t-SNE. Visualization results are shown in Figure 5.5 and Figure 5.6.

Figure 5.5 shows the distribution of sentence vectors where red triangles denote positive sentiment and green dots denote negativeness. It is clear that sentence vectors belonging to the same classes are tightly clustered together in the semantic space, which enables better classification performance. This is very interesting because no explicit attempt has been made to separate sentences from different classes. We also depict in Figure 5.6b and Figure 5.6a examples of positive and negative areas extracted from Figure 5.5. The dots and triangles are replaced by the sentences they represent. It can be seen that the sentences with similar semantics lie close to each other. This proves that the new model is very effective in modelling sentence representation.

The two dimension reduction procedures are done using the scikit-learn toolkit [169].
5.5 Summary

In this chapter, a new neural sentence model termed Attention Pooling-based Convolutional Neural Network has been successfully developed and an innovative attention pooling strategy has been proposed. The bidirectional long-short term memory model is exploited to generate an intermediate sentence representation which is used to obtain attention weights for pooling. The attention weights help extract the most significant information contained in the sentence. Furthermore, combining the bidirectional long-short term memory with the convolutional structure enables the model to extract comprehensive information, namely historical, future and local context of any position in a sequence.

Our new model can be trained end-to-end with limited hyper-parameters. It can extract comprehensive and the most significant information contained in a sentence. The sentence representation learning ability of the proposed model is very powerful in that it can implicitly separate the sentences from different classes in the semantic space. Experimental results demonstrate that the new model outperforms state-of-the-art approaches on seven benchmark datasets for text classification. We believe the new method can be applied to other natural language processing tasks and
Figure 5.6: Enlarged views of the visualization of the sentence representation space of SST-2 dataset. (a) Enlarged view from green area of Figure 5.5. (b) Enlarged view from red area of Figure 5.5. The original dots and triangles are replaced by the represented sentences. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
even computer vision applications. The proposed method uses bidirectional long-
short term memory to generate intermediate sentence presentation. Other models,
such as the gated recurrent units and auto encoders, may also be subject of future
investigation.
Chapter 6

Demography Prediction

6.1 Introduction

Nowadays people spend a great amount of time online doing all kinds of things, like reading news, playing games, shopping, etc. This consumer behavior results in advertisers putting great efforts and investment in online advertising. However, advertising audiences or web users, in this case, seldom click on online ads, which significantly decreases the effectiveness of the promotion. Targeted advertising is a practice delivering personalized ads relevant to users as opposed to pushing same ads to all. According to studies of TM advertising, targeted advertising gains 115% more business traffic a year and the targeted consumers have higher brand awareness than random viewers [170]. Targeted advertising is based on information of individual users like geographical location, behavioral variables, and demographic attributes. In this chapter, we are interested in demographic attributes (e.g., gender, age, income, etc.) of website audience as they play a key role in providing personalized services.

Traditional methods for determining demographic attributes are mainly based on panel data. The demographic tendency of websites is estimated statistically by known demographic information of panel members who have visited those sites. This approach is reliable for websites visited by a large number of panel members but may lead to biases for those sites not covered by a sufficient number of panel members. Besides, panel data may be difficult and expensive to obtain. Machine
learning approaches construct the relationship between website audience’s demographics and their online features (e.g., web content, web category, search keywords, etc.) by building a model on a subset of users with known demographics. Then the model can be used to predict other web users’ demographic attributes. Various online data is used to build prediction models, among which social network data is a popular choice. Facebook and Twitter profiles have been proven to be very effective in determining the users’ personality and demographics [171, 172]. However, these models require having access to extensive information about people’s social network, which may step into the sensitive area of privacy.

In this chapter, we also take advantage of machine learning approach to predict users’ demographic attributes. The proposed method constructs a model to build the relationship between users’ browsing history and their demographic attributes. The general view of the method is depicted in Figure 6.1. The various information contained in the browsing history has been employed to do the prediction job in literature. The authors of [173] used web-category information, users’ browsing time, browser type and other information to predict personal attributes. In [174], the authors employed more features by including visit frequency, visit duration, web ID, etc. Search keywords entered by users were employed to do the prediction job in [175, 176]. However, such information is usually not available for most users, resulting in a weak generalization of the trained models. The most robust method may be content-based methods [170, 177]. The web content is always available as long as the website is still there. The proposed model in this chapter is also purely based on web content.

Under the proposed framework, firstly we have built a web crawler to retrieve the content of the websites visited by users, and represent the content as website feature vectors. Exhaustive experiments have been done to determine the crawling rule. Secondly, we aggregate the vectors of websites that users have visited and arrive at feature vectors representing the users. Feature representations of users
are the most important step of the entire framework because existing literature demonstrates that even a simple classifier can achieve good performance as long as it takes in good features as input. At last, a classification model is exploited to predict the audiences’ demographic attributes. The support vector machine [178] is used as the top classifier in our method because it outperforms other baseline classifiers. Our new model proposes an innovative method to obtain feature vectors representing users. The new feature representation method takes advantage of word embedding technique and is easy-to-implement and effective.

Website feature representation in this chapter is the same as document representation in natural language processing community. Traditional research works in document representation without using word embedding usually employ the bag-of-words model which uses occurrence frequency of words as features. They face the problem of high dimensionality and sparsity and losing word order of sentences [28, 30]. To address these issues, most recent works exploit deep learning methods together with word embedding technique and leverage neural networks to construct non-linear interactions between words so as to capture the word order information. The usually used deep learning models include convolutional neural network [8, 27, 46, 96], recurrent neural networks [128, 130], recursive neural networks [44, 144], paragraph vector [102], etc. The problems with deep learning models are that they are very sophisticated, have a lot of parameters to tune, and are time-consuming to train.

Considering the specific characteristics of the task in this chapter, we propose a simple but effective model to represent website features. As the crawled content of websites is constituted by fragmented information, the word order information may not be very significant. The simplest document representation method based on word embedding is taking the sum or average of all the vectors of words contained in the document if the word order is not important. Our approach is developed based on this simple idea but incorporates a term weighting function based on term frequency-inverse document frequency (tf-idf) formula. With the tf-idf weighting scheme, the document representation vector not only leverages word co-occurrence probabilities but also takes into consideration the information about the general distribution of terms appearing in different documents. The idea is inspired by [179], in which the authors adopted variants of original tf-idf to represent documents for sentiment analysis and achieved significant improvement in classification.
accuracy. Instead of using the tf-idf vectors to represent documents, our method takes them as weights of word vectors. The proposed method has both the power of word embedding capturing semantic and syntactic information as well as the statistical nature of tf-idf. This intuitive representation approach is unsupervised, requiring no human annotations. It is computationally efficient compared with the neural network language models. Experimental results demonstrate that the new method outperforms the baseline document representation methods and achieves even better performance than sophisticated deep learning methods. The main contribution of this chapter is summarized as follows:

- A simple but effective unsupervised document representation method is proposed for website audience demography prediction. The method takes advantage of pre-trained word embedding and variants of tf-idf weighting scheme.
- Exhaustive experiments have been done to determine an internal protocol to crawl web content.
- Two benchmark datasets are created for demographic prediction task.

This chapter is based on the paper: *Targeted Advertising Based on Browsing History*.

### 6.2 Related Works

Many researchers have explored to predict users’ demographics based on browsing behavior. Baglioni *et al.* [173] predicted gender of users by analyzing users’ navigational behavior. They also analyzed the hierarchical ontology design of URLs visited and employed other click-through data (e.g., timestamps, browser type, etc.) as features. The decision tree was used to predict the attribute but only achieved slightly better accuracy than the random guess. Similarly, the authors of [174] used more click-through data (e.g. click frequency, browsing duration, etc.) to build features. They used the random forest to predict demographic attributes including gender, age, occupation, and education. The performance can hardly be named satisfactory as well. Jones *et al.* [175] exploited users’ query logs to build feature vectors and used support vector machine (SVM) to predict gender and age.
They achieved very good prediction performance on both attributes. In [176], the authors used both search keywords and accessed web-pages as input features, and mapped them to a reduced vector space using latent semantic analysis (LSA). Then the reduced vectors are fed into a neural network-based model to predict attributes of unknown users. A common characteristic of all the aforementioned methods is that they all used user-specific data. However, such information is usually not available for most users. Therefore, these models may not be generalizable. Goel et al. [180] used web domains themselves as features to predict five demographic attributes. Each user is represented as a sparse vector whose elements are binary features (e.g., 1 for visited and 0 for not visited). The SVM classifier was trained on a huge panel data and achieved good prediction performance. However, due to that a large number of web domains are not included, the model’s prediction capability as it scales by is doubtful.

The literature so far demonstrates that content-based methods may be the most robust and effective method among all the approaches. In [170], Hu et al. used both content-based features and web-category-based features to complement each other, but only achieved a slight performance improvement compared with using content-based features alone. They first predicted the demographic tendency of web-pages and then used a Bayesian network to derive representation vectors for users. In [177], only web content is used and the prediction task is completed by regression-based approaches. The web content in the above two papers is represented by bag-of-words model and standard tf-idf term weighting scheme respectively. The new method proposed in this chapter is also only based on website content.

### 6.3 Methodology

In our framework, we first represent the websites browsed by a user as vectors and then aggregate the website vectors to derive a vector representing the user. At last, a supervised classification model is used to predict the user’s demographic attributes. The flowchart of our framework is depicted in Figure 6.2. The details of the proposed methodology are discussed in the remainder of this section. The website representation approach is introduced in Section 6.3.2. Three aggregation methods are discussed in Section 6.3.3. The classifier is introduced in Section 6.3.4.
6.3. Methodology

Figure 6.2: The flow chart that explains the general methodology

6.3.1 Problem Formulation

Before introducing the proposed methodology, we define the problem in this section. A user’s browsing history is a set of websites he or she has visited. Suppose there are totally M users $U = \{u_1, u_2, ..., u_i, ..., u_M\}$ and N websites $S = \{s_1, ..., s_j, ..., s_N\}$. Then the browsing data of all the users can be represented by an adjacency matrix $R$, where the element $r_{ij}$ denotes the weight from user $u_i$ to website $s_j$. The weight is deemed as the visit frequency of user $u_i$ on website $s_j$. Given the demographic attributes $Y = \{y_1, y_2, ..., y_M\}$ of some users are known, the problem is to train a general model ($\mathcal{M} : U \sim Y$) on the known users and use the trained model to predict demographic attributes of unknown users.

6.3.2 Website Representation

The first step of our methodology is to represent websites browsed by users as site vectors. Only content-based features are used in our model. In this chapter, we propose a new method by taking a powerful technique named word embedding. Variants of the tf-idf weighting scheme are employed as weights of word vectors to help improve the representation power.

Word Embedding

There are $M$ websites which need to be represented as vectors in our problem setting. Each website $s_j$ is constituted by words. Recent research results have
demonstrated that word vectors are more effective in representing words than traditional bag-of-words method. We also use word embeddings in this chapter as well.

Suppose website $s_j$ is constituted by $K$ words. As every word is represented as a $d$-dimensional vector, the website can be represented as a dense matrix as $V_i = \{v_1, v_2, \ldots, v_K\} \in \mathbb{R}^{d \times K}$. The simplest way to reduce the site matrix to a site vector is to take the sum or the average of all the word vectors. Some recent works fulfill the document (website\(^1\)) representation task by employing neural networks to model non-linear interactions between words. Our proposed method follows the simple summation fashion but exploits variants of tf-idf weighting scheme to capture the distribution information of words amongst documents. It remains to be very simple and intuitive while our experimental results demonstrate that it even outperforms the sophisticated neural network approaches.

**Tf-idf Weighting Scheme**

Tf-idf, short for “term frequency-inverse document frequency”, is used to reflect the contribution of a word to a document in a corpus. It is widely used in information retrieval, text mining, and user modeling. Typically, tf-idf consists of two components, namely the term frequency (tf) and the inverse document frequency (idf). The two terms are calculated as:

\[
\begin{align*}
    tf(t, d) &= f_{t,d} \\
    idf(t, D) &= \log\left(\frac{N}{n_t}\right)
\end{align*}
\]

where the term $f_{t,d}$ is the raw frequency of term $t$ in a document $d$, $N$ is the total number of documents in the corpus $D$, and $n_t$ is the number of documents with term $t$ in it. And then the weight of term $t$ in the document $d$ is just the product of the two components:

\[
w_{t,d} = tf(t, d) \times idf(t, D)
\]

The weight increases proportionally to the term frequency in the given document and is offset by the frequency that the term appears in the entire corpus. This helps

\(^1\)A website is regarded as a document in this chapter.
Methodology

Table 6.1: Variants of tf weight

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaulted (d)</td>
<td>$f_{t,d}$</td>
</tr>
<tr>
<td>binary (b)</td>
<td>$\begin{cases} 1, &amp; f_{t,d} &gt; 0 \ 0, &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>logarithm (l)</td>
<td>$1 + \log(f_{t,d})$</td>
</tr>
<tr>
<td>augmented (a)</td>
<td>$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max{f_{t',d}: t' \in d}}$</td>
</tr>
</tbody>
</table>

Table 6.2: Variants of idf weight

<table>
<thead>
<tr>
<th>Weighting scheme</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary (u)</td>
<td>1</td>
</tr>
<tr>
<td>defaulted (d)</td>
<td>$\log(\frac{N}{n_t})$</td>
</tr>
<tr>
<td>smoothed (s)</td>
<td>$1 + \log(\frac{N}{1+n_t})$</td>
</tr>
</tbody>
</table>

to filter the common terms (e.g., the, I, you, etc.) which are actually non-relevant to the specific document.

Many variations of the tf-idf weighting scheme have been proposed for various applications. Some widely used variants of tf and idf weights [181] are listed in Table 6.1 and Table 6.2. The augmented term frequency is able to prevent a bias towards longer documents. The smoothed inverse document frequency is proposed in case that $n_t = 0$.

A two-letter notation scheme is used to denote the variants of tf-idf weighting scheme in the following sections according to the notations in Table 6.1 and Table 6.2. The first letter denotes the term frequency factor and the second the inverse document frequency. The classic tf-idf weighting scheme is denoted as $dd$.

The experiment results demonstrated that the $ad$ scheme results in the best information representing power. The weight of term $t$ in a document $d$ under the $ad$ scheme is given as follows:

$$w_{t,d}^{ad} = \left(0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d}: t' \in d\}}\right) \cdot \log\left(\frac{N}{n_t}\right) \quad (6.4)$$

Site Vector

Following the assumption that the website $s_j$ contains $K$ words, the website can be represented by a dense matrix $V_j = \{v_1, v_2, \ldots, v_K\} \in \mathbb{R}^{d \times K}$ by using pre-trained
word vectors. The term $d$ is the dimension of pre-trained word vectors. By taking advantage of the aforementioned tf-idf weighting scheme, we can obtain the weights of the $K$ terms in the website $s_j$. Let's denote the weights of the $K$ terms as a vector $W_{s_j} = \{w_{1,s_j}, w_{2,s_j}, \ldots, w_{K,s_j}\}$. Then we can get the site vector using the following equation:

$$s_j = W_{s_j}V_j^T \in \mathbb{R}^d$$  \hspace{1cm} (6.5)

Therefore, the website is represented by a low-dimensional vector.

### 6.3.3 Aggregation

After representing each website as a vector, a user's vector can be obtained by aggregating the vectors of all the websites browsed by him or her. This can be easily done by taking a weighted average of all the website vectors by using the weight matrix $R \in \mathbb{R}^{M \times N}$ which gives the visit frequency of each user to all the websites $S = \{s_1, s_2, \ldots, s_N\} \in \mathbb{R}^{d \times N}$.

$$u_i^{WA} = g(R_i)S^T \in \mathbb{R}^d$$  \hspace{1cm} (6.6)

where $g(\cdot)$ is a function to rescale values in a vector so that they sum up to be 1. The weighted average method faces a problem that some websites have much higher visit frequency thus weights than others. To address the problem, we narrow down the weight differences by mapping visit frequency to log space as follows:

$$u_i^{LA} = g(\log(R_i))S \in \mathbb{R}^d$$  \hspace{1cm} (6.7)

The implementation can effectively eliminate the influences of large weight difference. For example, supposing the visit frequency to sites $\{s_1, s_2\}$ are $\{10, 1000\}$, the weight ratio would be $1 : 100$ using Equation (6.6) and $1 : 3$ using Equation (6.7).

Another even simpler aggregation method is to take the simple average of the browsed website vectors by ignoring the user weights on websites as follows:

$$u_i^{SA} = \tilde{R}_iS \in \mathbb{R}^d$$  \hspace{1cm} (6.8)
Where $\tilde{R}$ is derived from $R$ with the following formula:

$$\tilde{R}_{ij} = \begin{cases} 
1, & R_{ij} > 0 \\
0, & R_{ij} = 0
\end{cases}$$  \quad (6.9)

### 6.3.4 Classification

We use support vector machine (SVM) as our classifier. However, we have still done several experiments comparing the learning performance of SVM with several other baseline machine learning algorithms, i.e., logistic regression, neural network and random forest. The results demonstrate that SVM achieves similar performance as neural network and outperforms logistic regression and random forest.

### 6.4 Datasets

#### 6.4.1 Demographic Attributes

There are many important demographic attributes associated with a person, e.g., age, gender, race, household income and so on. In this chapter, we focus on gender and age. However, the predictive model that we developed can be easily applied to any other types of audience classification. The gender attribute is categorized into male and female, while the age attribute is broken down into four groups, i.e., Teenage (<18 years), Young (18-34 years), Mid-age (35-49 years) and Elder (50+ years). The breakdown method closely corresponds to groups that are of interest to advertising agencies.

#### 6.4.2 Data Preparation

In our experiments, we use panelist data for training our learning model, and the trained model is applied to cookie browsing data for prediction. The panel data used contains users’ browsing log data in May 2015. Each record of the data consists of user id, web-page clicked, visit frequency of that web-page, and gender/age information. The data contains 175640 distinct users and 2476338 unique websites. The websites with very low traffic (visit frequency of all users
Table 6.3: Data distribution over gender and age of Demo20

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Male</td>
<td>Female</td>
</tr>
<tr>
<td>Teenage</td>
<td>3.5%</td>
<td>3.1%</td>
</tr>
<tr>
<td>Young</td>
<td>21.9%</td>
<td>16.7%</td>
</tr>
<tr>
<td>Mid-age</td>
<td>16.7%</td>
<td>17.2%</td>
</tr>
<tr>
<td>Elder</td>
<td>10.3%</td>
<td>10.5%</td>
</tr>
<tr>
<td>Total</td>
<td>52.5%</td>
<td>47.5%</td>
</tr>
</tbody>
</table>

Table 6.4: Data distribution over gender and age of Demo100

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Male</td>
<td>Female</td>
</tr>
<tr>
<td>Teenage</td>
<td>1.6%</td>
<td>1.3%</td>
</tr>
<tr>
<td>Young</td>
<td>18.3%</td>
<td>13.2%</td>
</tr>
<tr>
<td>Mid-age</td>
<td>18.7%</td>
<td>18.2%</td>
</tr>
<tr>
<td>Elder</td>
<td>16.4%</td>
<td>12.4%</td>
</tr>
<tr>
<td>Total</td>
<td>54.9%</td>
<td>45.1%</td>
</tr>
</tbody>
</table>

less than 100) are removed as the content of such websites falls out of the scope of most users. If a website is visited by a user by less than five times, it is taken away from the list of visited websites of that user, since it is not reasonable to refer a user’s attributes using websites that he or she randomly visits. Then the websites that cannot be crawled are filtered out. We also filter the websites whose crawled content has less than 10 words because such few content can hardly convey useful information. The number of websites clicked by a user is an important variable for user’s demography prediction, because it is not reasonable to predict a user’s demographic information based on very few websites. In this chapter, we choose two threshold values for the minimum number of websites visited, namely 20 and 100 to investigate the impact. Two datasets, namely Demo20 and Demo100, are generated by filtering out the users who visit less than the corresponding number of websites. After all the processing, Demo20 has 70868 distinct users and 3499537 entries in sum, while Demo100 has 4742 distinct users and a total of 667019 entries. The gender and age distribution of the two datasets are shown in Table 6.3 and Table 6.4.
6.4.3 Web Crawling

Web crawling of websites is necessary for our task, in order to extract key information from a web page. As websites are now designed to appear in different formats and styles to appeal to a diverse audience, and different web platforms (e.g., mobile and computer), it is necessary for us to identify an internal protocol to extract web content, which is ubiquitous to most websites, in a generalized manner. In our implementation, the HTML tag-based method is used to crawl the websites.

Specifically, the title tag <title>, headline tags <h1-h6>, paragraph tags <p>, link tags <a>, and image tags <img> are chosen and tested on the gender attribute of a website demography tendency dataset. This dataset contains the top 2000 visited websites in the USA, with the visitors’ demographic tendency for each website. To build our protocol, we tested a combination of the tag information collected in our experiments. Concretely, we want to understand if a certain (combination of) tag information will be more reliable and useful to succinctly describe a website. Intuitively, we believe that the <title> tag conveys very important information about a website, therefore, it is always included in our protocol. As these five HTML tags do not encompass the entire website’s content, we have also crawled the visible text in each website. Although the visible text contains more information on each website, it also includes more noise for our learning model. This is demonstrated in our experimental results.

The website demography tendency dataset uses continuous scores to indicate visitors’ demographic tendency (e.g., 0.45 for male and 0.55 for female). As such, we employ a regression model named $\epsilon$-support vector regression ($\epsilon$-SVR) [182], which is an extension of SVM, to predict the tendency scores for a given website. The prediction root mean square error (RMSE) results of all the tag combinations are demonstrated in Figure 6.3. We observe that the combination $hpai$ achieves the smallest RMSE, while the inclusion of all visible text does not improve the performance due to the presence of additional noise. Therefore, the combination of $hpai$ is employed to crawl website content in our model.
Chapter 6. Demography Prediction

Figure 6.3: RMSE results by tag combinations on website demography tendency prediction. The terms h, p, a, i, v are used to represent headlines, paragraphs, link words, image descriptions, and visible texts respectively for brevity. Tag combination xy means content associated with the corresponding tags x and y are used. For instance, hp stands for using both headlines and paragraph content.

6.5 Experiment Analysis

In this section, we evaluate the performance of the proposed model on the two user demography datasets, namely Demo20 and Demo100. The impact of different tf-idf weighting schemes on website representation is evaluated in Section 6.5.2. The three aggregation methods are compared in Section 6.5.3. The prediction performance using different classifiers are shown in Section 6.5.4. Section 6.5.5 demonstrates the power of new proposed website representation method by comparing with existing baseline and state-of-the-art methods. One more experiment is done in Section 6.5.6 to demonstrate that our aggregation-classification framework is better than regression-aggregation framework used in [170].

6.5.1 Experiment Setup

The greatest advantage of the proposed model is that the user feature representation procedure has no parameters to tune. We directly use the word embedding matrix word2vec which has been pre-trained on Google news. The dimension of each word vector is 300. Gensim toolkit [183] is exploited to obtain the tf-idf weights. The only parameters to tune are the parameters in the SVM classifier. The SVM classifier is implemented using the LinearSVC package in scikit-learn.
6.5. Experiment Analysis

Fig. 6.4: Accuracy comparison results on gender prediction using various tf-idf weighting scheme on Demo20 and Demo100 datasets. The first letter in the scheme notation denotes the name of the variant of term frequency weight. The second letter accounts for the variant of inverse document frequency weight.

toolkit [169]. Ridge norm penalty is used and the penalty parameter $C$ is determined using grid search and cross-validation. The two datasets are split into training and testing datasets by 3:2 ratio. The linear SVC is trained on the training dataset and all the prediction results in the following sections are obtained on the testing dataset.

6.5.2 Performance of Different Tf-idf Weighting Schemes

Four variants of tf weight and three variants of idf weight are shown in Table 6.1 and Table 6.2. Therefore, there are totally 12 different tf-idf weighting schemes. In this section, we compare the demography prediction results on both Demo20 and Demo100 using all the 12 weighting schemes. As the comparison results on gender attribute and age attribute are similar, only gender prediction results are shown for conciseness. The log average aggregation implementation is used.

Figure 6.4 demonstrates the accuracy comparison results on gender prediction using various tf-idf weighting scheme on Demo20 and Demo100 datasets. The results on both Demo20 and Demo100 show that binary tf weight scheme performs better than the default raw tf weight, e.g., $bd$ (83.45%) vs. $dd$ (81.81%). Furthermore, other variants of tf weight also outperform the raw tf weight. The logarithm and augmented weighting schemes perform similarly, both slightly better than binary
features, e.g., \(ld\) (83.83\%) vs. \(ad\) (83.95\%). We can conclude that sub-linearly scaled tf weight improves representation capability compared with raw term frequency.

When comparing the idf weighting schemes, we can find that the default idf weight helps improve representation power compared with unary idf weight, e.g., \(ad\) (83.95\%) vs. \(au\) (82.81\%). The \(du\) weighting scheme actually takes the weighted average of vectors of all words in the website, without considering the word distribution information over the corpus. With the incorporation of idf weighting scheme, the website representation vector not only leverages word co-occurrence probabilities but also takes into consideration the information about the general distribution of terms appearing in different documents. In [179], the authors claimed that the idf smoothing greatly improved the classification accuracy on sentiment analysis task. However, our results show that smoothing idf weight does not provide any advantage over the classic idf weight. This is because all the terms appear in the corpus at least once \(n_t \geq 1\). In our task, the \(ad\) weighting scheme achieves the best prediction performance on both \textit{Demo20} and \textit{Demo100}.
6.5.3 Performance of Different Aggregation Methods

After obtaining the representation vectors for websites, an aggregation step is needed to move from website vectors to user vectors. In Section 6.3.3, three aggregation methods are introduced. They are compared in this section. The best tf-idf weighting scheme in our task, namely $ad$, is used in all the experiments in this section. The comparison results on two datasets for both gender and age attributes are demonstrated in Figure 6.5. We can see that the prediction accuracy using weighted average method is much poorer than that using simple average aggregation although the former taking into consideration the user’s visit frequency on each website. This results from the fact that frequency of few websites may be tens of or hundreds of times larger than that of other sites. Thus few most frequently visited websites dominate the final user representation and contributions of other websites can almost be neglected. The best aggregation method among the three is the log average method. It still leverages the visit frequency difference, but rescale the weights to a much narrower range so that all the useful websites’ information is included in the final user representation. According to the experimental results in Section 6.5.2 and Section 6.5.3, the $ad$ tf-idf weighting scheme and log average aggregation method are employed to obtain the final user vector representation.

6.5.4 Performance of Different Classification Models

Several popular baseline classifiers are used as the classification model in our framework and their prediction performance on gender attribute is shown in Table 6.5. All the classifiers have been implemented using scikit-learn toolkit [169]. For the neural network, one hidden layer perceptron classifier has been used. Random forest and logistic regression are implemented using base classifiers in the toolkit. Default settings are used for most parameters. The most important hyper-parameters, like regularization strength of logistic regression, the number of maximal features and estimators of random forest, and the number of hidden nodes of the neural network, are determined using grid search and cross-validation. The results in Table 6.5 demonstrate that SVM achieves similar performance as neural network and outperforms logistic regression and random forest. Random forest performs rather poor in this task, which should be expected from the noisy characteristic
Table 6.5: Gender prediction accuracy of SVM against other baseline classifiers (%)

<table>
<thead>
<tr>
<th>Models</th>
<th>Demo20</th>
<th>Demo100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>73.39</td>
<td>82.53</td>
</tr>
<tr>
<td>Random Forest</td>
<td>67.63</td>
<td>77.60</td>
</tr>
<tr>
<td>Neural Network</td>
<td>74.28</td>
<td>83.99</td>
</tr>
<tr>
<td>SVM</td>
<td><strong>74.42</strong></td>
<td>83.95</td>
</tr>
</tbody>
</table>

of web data. The fact that a linear SVM achieves rather satisfactory performance demonstrates that the feature representations of users are effective and informative.

### 6.5.5 Comparison of Website Representation Methods

One of the main contributions of the chapter is our proposed website representation approach. Our method is purely based on content and does not require any other user-specific input data. To demonstrate the representation power of the proposed method. We compare it with some other website representation methods. The comparison methods are described as follows:

- **Category-based method**: represents websites using category-based features. Websites are spanned over 460 two-level categories from Open Directory Project (ODP\(^2\)). ODP is the largest, most comprehensive human-edited directory of the Web. Each website is represented by a 460-dimensional vector with each value indicating the possibility of belonging to each category. The possibility values are obtained by comparing the web content with keyword library built for each category.

- **Latent Semantic Analysis (LSA)**: is a widely used topic modeling method which is able to find the relationships between terms and documents. Gensim toolkit [183] is used to implement LSA and the number of topics is set as 300.

- **Recurrent Neural Network (RNN)**: is a powerful deep learning model which is able to construct non-linear interactions between words. It takes in word vectors in a website one by one and outputs the vector representation of the website. The model is trained using demographic tendency scores of

\(^2\)http://dmoz.org
Table 6.6: Comparison of the percentage (%) accuracy of different websites representation methods

<table>
<thead>
<tr>
<th>Representation methods</th>
<th>Demo20 Gender</th>
<th>Demo20 Age</th>
<th>Demo100 Gender</th>
<th>Demo100 Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category-based</td>
<td>71.71</td>
<td>53.65</td>
<td>79.34</td>
<td>53.03</td>
</tr>
<tr>
<td>LSA</td>
<td>71.93</td>
<td>53.49</td>
<td>79.81</td>
<td>53.55</td>
</tr>
<tr>
<td>RNN</td>
<td>68.35</td>
<td>49.65</td>
<td>75.49</td>
<td>50.71</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>70.56</td>
<td>52.21</td>
<td>77.96</td>
<td>52.78</td>
</tr>
<tr>
<td>TF-IDF,word2vec</td>
<td><strong>74.42</strong></td>
<td><strong>53.95</strong></td>
<td><strong>83.95</strong></td>
<td><strong>54.93</strong></td>
</tr>
</tbody>
</table>

websites which are propagated from attributes of users who have browsed the websites. It is implemented using keras\(^3\).

- **TF-IDF**: uses tf-idf weights directly as feature vectors of websites without using word vectors. We also use gensim toolkit to obtain tf-idf representations and the default tf-idf is used.

The comparison results are demonstrated in Table 6.6. The last row is the representation method proposed in this chapter. The \(ad\) variant of tf-idf weighting scheme is used. We can see that the new representation method achieves a much better performance than other approaches. The performance of category-based method relies on the completeness and precision of keyword library of each category, which is not easy to build. This may be the reason that category-based method only achieves average performance. LSI has not achieved very satisfactory performance either, demonstrating that topic extraction from messy website content is not an easy task. Comparing TF-IDF and our word embedding method, we can conclude that the incorporation of word vectors help significantly improve the representation power. TF-IDF representations are very sparse and with high dimension, resulting in difficulty in training a good classifier. What is of interest is that the sophisticated deep learning model RNN shows the worst performance. One reason is that website content usually contains a lot of words. RNN faces great challenges to memorize long sequences because of the problems of ‘vanishing gradient’ and ‘explosive gradient’. Another reason is that the content in websites is usually fragmented, therefore, this greatly limits the use of deep learning within our framework. Furthermore, sophisticated RNN model requires a great amount of time to train while our model is computationally efficient. The comparison results demonstrate the effectiveness and power of the new website representation method.

\(^3\)http://keras.io
Figure 6.6: Accuracy comparison results using aggregation-classification and regression-aggregation framework. The upper two are age and gender prediction on Demo100 dataset, while the bottom two are prediction results on Demo10.

6.5.6 Study on Overall Framework

In [170], Hu et al. first trained a supervised regression model to predict a web-page’s gender and age tendency. Then, a user’s gender and age are predicted based on the age and gender tendency of the web-pages browsed by the user within the Bayesian framework. The Bayesian framework assumed that the web-pages visited by the users are independent. Our model does user attribute prediction after aggregating information from websites browsed by the user. In this section, we compare the two frameworks. The framework in Hu’s paper is denoted as regression-aggregation, while our framework as aggregation-classification. The comparison results are depicted in Figure 6.6. We can see that the aggregation-classification framework achieves much higher accuracy than regression-aggregation framework when predicting both gender and age attribute on Demo20 and Demo100. The results indicate the independent assumption is problematic. The fact is that websites visited by the same user are usually similar or correlated. The experimental results demonstrate that our framework is more reasonable.

6.6 Summary

Audience classification is of significant importance for targeted advertising. In this chapter, a new method is proposed to estimate on-line users’ demographic
attributes based on their browsing histories, as an example of audience classifications. We first retrieve the content of the websites visited by users, and employ an innovative website representation method to represent the content as feature vectors. Word embedding with variants of tf-idf weighting scheme turns out to be a simple but effective website representation method. The method is unsupervised and requires no human annotations. Various variants of tf-idf weighting schemes have been tested and it was observed that sub-linearly scaled term frequency weight together with the default inverse document frequency weight can significantly improve the final classification accuracy. After obtaining website vectors, a log average aggregation method is shown to be effective in composing vectors of websites browsed by the user into a vector representing that user. The experimental results demonstrate that our new user feature representation method is more powerful than existing baseline methods, leading to a significant improvement of prediction accuracy. Another contribution of the chapter is examining various web crawling rules and identifying the best crawling protocol for audience classification. The audience classification task is a practical application backed by natural language understanding techniques. The final objective of doing research in natural language understanding area is to solve such practical tasks in real life.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

Natural language understanding aims at endowing comprehension power to machines and has received great attention from both industry and academia in recent years. Most existing works in this area follow rigid language assumptions and demand great efforts in feature engineering. The objective of the thesis is to develop machine learning models that can learn effective semantic representations from raw texts automatically without the limit of rigid language assumptions and laborious feature engineering. Specifically, I have proposed several new algorithms to induce effective semantic feature representations for natural language understanding applications, such as document summarization, sentiment analysis, sentence modeling, and so on.

In Chapter 3, two new models have been proposed for document summarization task. The sentences in documents are represented as dense vectors and given scores so that they can be ranked. Sentences with high scores are chosen to form a final summary according to a certain selection procedure. The first model, termed window-based sentence representation (WSR), employs context window of vectors to preserve word order information in sentences. The sentence vectors are given scores by the extreme learning machine (ELM) model. Experimental results demonstrate that the simple structure achieves good summarization performance with very fast learning speed. The window-based model only preserves partial
order information of sentences and owns weak representation capability. The second model, termed *multi-view convolutional neural network* (MV-CNN), improves the ability to extract knowledge from data by leveraging on the latent representation power of the convolutional neural network. The key principles of multi-view learning, namely complementary and consensus principles are exploited to mine knowledge from different perspectives. I have also proposed an innovative sentence position embedding technique to further improve summarization performance. The MV-CNN model demonstrates much greater representing power than the WSR method and achieves better performance than state-of-the-art approaches.

Chapter 4 introduces a novel recurrent structure and three recurrent models, termed *comprehensive attention recurrent neural networks* (CA-RNN), *comprehensive attention long short-term memory* (CA-LSTM), and *comprehensive attention gated recurrent unit* (CA-GRU). The new models take advantage of the recurrent models’ power to extract sequential information and the convolutional neural network’s ability of extracting local information, to access the historical, future, and local context information of any position in a sequence. The new models have been tested on three sentiment analysis datasets and proved to be superior to the standard recurrent models. By utilizing the idea of the *comprehensive attention recurrent models*, I have proposed another model, termed *attention pooling-based convolutional neural networks*, by combining the bidirectional long short-term memory and convolutional structures so that it can extract comprehensive knowledge contained in sentences as well in Chapter 5. Furthermore, I proposed a new pooling scheme called *attention pooling* in order to overcome the shortcoming of the max pooling, which loses order information of sequences and intensity information of the same feature. The new convolutional model can well separate sentences from different classes in the latent feature space, demonstrating that the learned semantic representations are effective and useful. I have tested the new model on some text classification benchmark datasets and proved that it outperforms the state-of-the-art approaches.

At last, Chapter 6 turns to a practical application which is addressed using natural language understanding techniques. The task aims at predicting on-line users’ demographic attributes based on their browsing histories. Our model is purely based on web content. The content of a website is represented as a feature vector by employing word embedding and term frequency inverse document frequency (tf-idf)
Table 7.1: Comparison of all models in this thesis: Composition, Classifier/Regressor, and Task addressed. The “RNN” in the third algorithm can be replaced by “LSTM” and “GRU”.

<table>
<thead>
<tr>
<th>Models</th>
<th>Composition</th>
<th>Classifier/Regressor</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSR Ch. 3.1</td>
<td>window summation</td>
<td>regression: ELM</td>
<td>document summarization</td>
</tr>
<tr>
<td>MV-CNN Ch 3.2</td>
<td>CNN</td>
<td>regression: sigmoid</td>
<td>document summarization</td>
</tr>
<tr>
<td>CA-RNN Ch. 4</td>
<td>CNN + RNN</td>
<td>classification: softmax</td>
<td>sentiment analysis</td>
</tr>
<tr>
<td>AP-CNN Ch. 5</td>
<td>CNN + LSTM</td>
<td>classification: softmax</td>
<td>sentence modeling</td>
</tr>
<tr>
<td>TF-IDF,word2vec Ch. 6</td>
<td>tf-idf</td>
<td>classification: SVM</td>
<td>demography prediction</td>
</tr>
</tbody>
</table>

weighting scheme after it has been retrieved from the website. The new site representation method owns both the power of word embedding capturing semantic and syntactic information and the statistical nature of tf-idf. Experimental results demonstrate that the simple representation method outperforms baseline document representation methods and even achieves better prediction performance than sophisticated deep learning models which may result from the specific characteristic of the addressed task.

Table 7.1 summarizes the composition framework, final classifier or regressor, and the task addressed for the five models proposed in this thesis. All the models have taken advantage of the power of word embedding. The WSR employed simple summation method to compose word vectors to sentence vectors after taking the context window into consideration. The last model takes a certain tf-idf weighting scheme to do a weighted summation on word vectors so as to arrive at the sentence vectors. The other three models exploit deep neural networks to capture the non-linear interaction between words when composing word vectors to sentence vectors. The third column in Table 7.1 shows whether the model deals with regression or classification problem, and the corresponding regressor or classifier. The first two models are regression models and the last three classification. The first and the last model use standalone prediction models (ELM for regression and SVM for classification respectively), while the other three utilize simple regressor or classifiers that are part to the deep neural network models. The three deep neural network models use stochastic gradient descent methods to optimize parameters.
and the input word vectors can be trained as well. The last column demonstrates
the natural language understanding task addressed by each model, although the
models can be easily adapted to other tasks.

At last, the contributions that we have made to the natural language understanding
community in this thesis are summarized as follows:

- A general natural language understanding framework is proposed in this the-
  sis, which is word embedding plus machine learning models. The word embed-
  ding together with neural network-based models (i.e., the first four models)
  prove to be very effective in learning semantic representations of texts. Es-
  pecially, deep neural network models built over word vectors can be trained
  end-to-end and demonstrate great learning power, which to my belief will be
  the foundation framework for future natural language understanding appli-
  cations. However, my last model shows that deep learning models may not
  be suitable in certain cases. Our mind should always be open and analyze
  the problems we are addressing case by case.

- Two new approaches are proposed for the document summarization task. The
  first model is very simple but can preserve partial word order information.
  The usage of the extreme learning machine provides a new idea for this task if
  the training speed is the key consideration. The second model applies multi-
  view learning into convolutional neural networks for document summarization
  for the first time. The multi-view learning suits document summarization
  task because it is very similar to the behavior of human summarizers. We
  demonstrate that multi-view learning is a feasible direction to improve CNN
  learning capability.

- A new recurrent structure that can access to historical, future and local
  context information contained in sentences is proposed. This structure can
  be easily plugged into various recurrent models.

- A new pooling scheme leveraged on attention mechanism has been proposed
  for convolutional neural networks. The new pooling scheme is able to preserve
  word order information and feature intensity information lost by the popular
  max pooling strategy.
• We have proposed a new thought of solution for predicting demography attributes in the targeted advertising campaign. Two benchmark datasets have been made for the demography prediction task. We prove that natural language understanding is essential and can play a significant role in many practical applications in our daily life.

• Last but not the least, the new models do not demand the texts in any rigid form. No expert linguistic knowledge is needed. They learn underlying semantics and structures of texts purely from training data with no or few manual interventions. I believe it is the future trend of natural language processing applications that machines extract semantic knowledge automatically from raw text or speech without consulting human being.

7.2 Future Work

The models proposed in this thesis have demonstrated the great advantages of word embeddings. Words with similar semantics are close to each other in the vector space, therefore similar words can lead to similar predictions. The conventional bag-of-words model can hardly achieve this property. The success of the same pre-trained word vectors (word2vec) across distinct tasks suggest that word vectors are good universal feature extractors. Word embeddings have almost become an indispensable tool in natural language processing, especially neural network-based natural language processing studies. However, people still have the right to doubt whether a single vector can fully capture the semantic and syntactic information in multiple words, let alone long sentences and documents. Actually, the models indeed come across with difficulty when trying to represent long sequences. Recurrent models, even long short-term memory models, have the problem of forgetting words seen long ago because the back-propagated error signal would nevertheless suffer from gradient vanishing in modeling long texts with hundreds of words. It needs a lot more explorations in building representations for long texts. The attention mechanism is a useful tool addressing the problem because all hidden states in a sequence contribute to higher level understanding instead of compressing the entire input sequence into a fixed representation [12]. The other promising technique we may consult is memory networks [184], which incorporate an external memory component that can be read and written to into the successful learning
strategies like CNNs and RNNs. The memory networks efficiently improve the memory capability of models. There are also many other technologies worth of studying. Learning semantic representations for long texts is a challenging but useful exploration direction.

The greatest difficulty in understanding natural language may result from language’s ambiguous nature. The models have not shown to be able to understand vague meanings in ambiguous sentences. For example, the machines can hardly judge the sentiment contained in the sentence “The food in the restaurant is delicious but I am not happy with their service.” Our models cannot resolve the referential ambiguity problems mentioned in Section 2.1 as well. It is even harder, actually almost impossible, for machines to understand an author’s real meaning if the writer uses sarcasm. The machine must lose itself when reading the famous sarcastic quote by George Carlin: “Weather forecast for tonight: dark.”. The good news for machines is that our human being also makes mistakes when attempting to understand ambiguous language. To understand ambiguous language is an even more challenging task but we cannot just run away from it. Possible solutions include understanding the paragraph or document level meaning first and employ the holistic meaning to understand the partial context using recurrent or recursive techniques.

All the models in this thesis are discriminative models which model the dependence of unobserved variables on observed variables. Discriminative models are usually supervised models and can yield superior performance for tasks that do not require the joint distribution of input and output variables [185]. However, discriminative models cannot generate samples from the joint distribution and can hardly be extended to unsupervised learning. A generative model, as opposed to discriminative models, is a full probabilistic model of all variables that can generate values of any variables given some hidden parameters. Therefore, generative models are more flexible. The document summarization models in this thesis can only address the extractive summarization task because they are discriminative models. In comparison, a generative model may generate an abstractive summary that is as readable as human-generated summaries. As foreshadowed in Section 2.1, natural language processing involves another significant branch named natural language generation which is very important for tasks like machine translation and question answering. The applications take in a sequence and output a sequence,
thus generative models are more suitable. Question answering, to my belief, is the final task that can solve everything related to natural language and even images and other knowledge sources. All the natural language understanding tasks in this thesis can be addressed in a question answering framework. For instance, the judgment of sentiment in a movie review can be transformed into the question “What’s the sentiment contained in the review?”.

There are a lot of generative models in the literature, such as Naive Bayes, Hidden Markov model, Latent Dirichlet allocation, and so on. Generative adversarial networks [186] is one popular generative model proposed recently. The model has seen great success in computer vision and ignited great research passion in artificial intelligence community. The model has not achieved any breakthrough results in the area of natural language processing area because of the discrete nature of language, but I think it is a very promising technology in understanding language. Another interesting and promising technique that will help build generative natural language understanding systems is reinforcement learning (RL) [187, 188]. The RL technique enables machines to learn best actions under certain states based on reward or punishment. So far, the RL is mainly applied to robot control and computer games. There are not too many applications in NLP tasks yet. However, we see a lot of NLP tasks which can take advantage of RL. For example, a question answering system should select an answer sentence based on forward-looking, long-term reward.

One problem with deep learning models is that they must be trained on a large volume of task-specified data and are very prone to overfitting. Although there are massive data available nowadays, it is still of high cost to obtain labeled data. Most models are trained on data from one domain and can only be used in that certain domain. It would save great cost if we could develop a universal model trained on limit-size data but generalizable to multiple tasks. Although the structures of our models can be adapted to other tasks, the models have to be trained again on new data domain when used to address a different task. Transfer learning, domain-independent learning, and multi-task learning [189] are approaches to address the issue by learning multiple tasks simultaneously to significantly improve performance relative to learning each task independently. Our long-term goal is to develop a general model that can jointly address various natural language processing tasks. Therefore, multi-task natural language understanding needs intensive future research explorations.
7.2. Future Work

The last problem is the computation complexity of deep learning models. Three models in this thesis are developed based on deep learning models and I believe deep neural networks are most promising technology in NLP research area. With the help of more powerful computing equipment like GPUs and more advanced parallelized computing technology, the computation problem seems to be no longer a limitation. However, the cost is still high for those who cannot afford the expensive computing hardware. As people’s life become more mobile, the need for high-performance algorithms on mobile phone, tablets and other light equipment skyrockets. Therefore, it is necessary to design computationally economic algorithms while maintaining good learning power. Algorithms that can be easily parallelized and decentralized will become more and more popular in the future. And this is also an inevitable issue when commercializing the developed approaches.

I hope that the models in this thesis can be extended to efficiently understand various language formats (no matter short or long, concrete or fuzzy), deal with various language understanding tasks, and be smart enough to interact with human being in one coherent framework. A universal question answering system may be the possible solution.
Appendix A

Backpropagation Procedure

We have the approximation function:

\[ y = f(wx + b) \]  \hspace{1cm} (A.1)

where \( w = [w_1, \cdots, w_d] \) denotes the weights connecting inputs to the output neuron. The annotation \( z = wx + b \) is introduced to make derivative calculation procedure neat. Therefore, we have \( y = f(z) \). As mentioned in Section 2.1, the approximation function can hardly return real output but only an approximated value. The difference between the real and approximated outputs given an input is used to evaluate the model. We use the simple difference function defined in Equation (2.2). Then, we have the loss function:

\[ L = \frac{1}{2N} \sum_n (y_n - f(wx_n + b))^2 \]  \hspace{1cm} (A.2)

We can use the GD algorithm to determine the parameters \( \theta = [w, b] \). Observe that:

\[ \nabla L = \frac{\partial L}{\partial w} = \left( \frac{\partial L}{\partial w_1}, \cdots, \frac{\partial L}{\partial w_d} \right) \]
We only show the derivation of a single weight $w_j$ for ease of exposition (N is ignored as the derivation is not affected):

$$\frac{\partial L}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_n (y_n - f(wx_n + b))^2$$

$$= \sum_n (y_n - f(z_n)) \frac{\partial}{\partial w_j} (y_n - f(z_n))$$

$$= \sum_n (y_n - f(z_n)) f'(z_n) \frac{\partial z_n}{\partial w_j}$$

$$= \sum_n (y_n - f(z_n)) f'(z_n) \frac{\partial}{\partial w_j} \sum_{k=1}^d (x_{nk} w_k + b)$$

$$= \sum_n (y_n - f(z_n)) f'(z_n) x_{nj}$$

$$= \sum_n \delta_n x_{nj} \quad (A.3)$$

The term $\delta_n = (y_n - f(z_n)) f'(z_n)$ is the error signal of $n$th sample. The full gradient of $\mathbf{w}$ is:

$$\frac{\partial L}{\partial \mathbf{w}} = \sum_n \delta_n x_n \quad (A.4)$$

The gradient of bias is calculated similarly:

$$\frac{\partial L}{\partial b} = \sum_n \delta_n \quad (A.5)$$
Author’s Publications

Journal Articles


Conference Proceedings


Book Chapters


Bibliography


