SENTIMENT ANALYSIS AND OPINIONS SUMMARIZATION ON SOCIAL MEDIA

NGUYEN TIEN HUY

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

SENTIMENT ANALYSIS AND OPINIONS SUMMARIZATION ON SOCIAL MEDIA

NGUYEN TIEN HUY

Supervisor : Associate Professor NGUYEN Le Minh

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science
September, 2019

Doctoral Dissertation

SENTIMENT ANALYSIS AND OPINIONS SUMMARIZATION ON SOCIAL MEDIA

Nguyen Tien Huy

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Written under the direction of
Associate Professor Minh Le Nguyen

September, 2019

Abstract

The emergence of web 2.0, which allows users to generate content, is causing a rapid increase in the amount of data. Platforms (e.g. Twitter, Facebook, and YouTube), which enable millions of users to share information and comments, have a high demand for extracting knowledge from user-generated content. Useful information to be analyzed from those comments are opinions/sentiments, which express subjective opinions, evaluations, appraisals, attitudes, and emotions of particular users towards entities. If we can build a model to detect and summarize correctly and quickly opinions from comments of social media, we can extract/understand knowledge about the reputation of a person, organization or product. This task raises some challenges due to the unique characteristics of social media text such as: i) comments may not be in well-grammar text; ii) social media text covers a variety of domains (e.g., phone, education) that requires a robust approach against domains; iii) comments may not be related to topics or spams.

The aim of this study is to obtain an effective method for identifying and summarizing opinions on social media. To this end, the research question is as follows: how to employ deep learning architectures to deal with the challenges of this task. As the advantages of deep learning are to self-learn salient features from big data, we expect an efficient result from this approach for opinions summarization.

To answer the research question, we propose a framework with five subtasks as follows:

- Sentiment analysis - identifies the polarity (positive or negative or neutral) of a comment/review. We propose a freezing technique to learn sentiment-specific vectors from CNN and LSTM. This technique is efficient for integrating the advantages of various deep learning models. We also observe that semantically clustering documents into groups is more beneficial for ensemble methods.

- Subject toward sentiment analysis: determines the target subject which the comment gives its sentiment to or the comment contains spam. We propose a convolutional N-gram BiLSTM word embedding which represents a word with semantic and contextual information in short and long distance periods. Our model achieves strong performance and robustness across domains compared with previous approaches.

- Semantic textual similarity: measures the semantic similarity q_{ij} of two sentences i and j, which plays an important role in identifying the most informative sentences as well as redundant ones in summarization. We propose an M-MaxLSTM-CNN model for employing multiple sets of word embeddings for evaluating sentence similarity/relation. Our model does not use hand-crafted features (e.g., alignment features, Ngram overlaps, dependency features) as well as does not require pre-trained word embeddings to have the same dimension.

- Aspect similarity Recognition (ASR): identifies whether two sentences express one or some aspects in common. We propose this task to enhance the process of selecting

salient text for summarization where a summarized review needs to cover all aspects as well as avoid redundancy. To facilitate the application of supervised learning models for this task, we construct a dataset ASRCorpus containing two domains (i.e., LAPTOP and RESTAURANT). We propose an attention-cell LSTM model, which efficiently integrates attention signals into the LSTM gates.

- Opinions Summarization: employs those signals above for ranking sentences. A concise and informative summary of a product e is generated by selecting the most salient sentences from reviews. Applying ASR relaxes the constraint of predefined aspects in conventional aspect-based opinions summarization.

According to the results, our summarization approach obtains significant improvement compared to the previous works on social media text. Especially, the proposed Aspect Similarity Recognition subtask relaxes the limitation of predefining aspects and makes our opinions summarization applicable in domain adaptation. Further research could be undertaken to integrate transfer knowledge at sentence level as well as multitask learning for opinions summarization.

Keywords: Sentiment Analysis, Opinion Mining, Opinions Summarization, Deep Learning, Aspect Similarity Recognition, Semantic Textual Similarity

Acknowledgments

First of all, I wish to express my best sincerest gratitude to my principal advisor, Associate Professor Nguyen Le Minh of Japan Advanced Institute of Science and Technology (JAIST), for his constant encouragement, support and kind guidance during my Ph.D. course. He has gently inspired me in researching as well as patiently taught me to be strong and self-confident in my study. Without his consistent support, I could not finish the work in this dissertation

I would like to thank Professor Akira Shimazu, Professor Satoshi Tojo, Associate Professor Kiyoaki Shirai, Associate Professor Shinobu Hasegawa of JAIST, and Professor Ken Satoh of National Institute of Informatics for useful discussions and comments on this dissertation.

I would like to thank Associate Professor Le Hoai Bac from University Of Science, VNU-HCMC for his suggestion and recommendations to study at JAIST.

I am deeply indebted to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) for granting me a scholarship during the period of my research. Thanks also to JAIST Research Grant for Students for providing me with their travel grants which supported me to attend and present my work at international conferences.

I would like to thank JAIST staff for creating a wonderful environment for both research and life. I would love to devote my sincere thanks and appreciation to all members of Nguyens laboratory. Being a member of Nguyens lab and JAIST is a wonderful time of my research life.

Finally, I would like to express my sincere gratitude to my parents, brothers, and sisters for supporting me with great patience and love. I would never complete this work without their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Opinions Summarization is the collection of typical opinions mentioned in social media, blogs or forums on the web. This task helps customers to absorb better a large number of comments and reviews before making decisions as well as producers to keep track of what customers think about their products [Liu, 2012]. Table 1.1 shows a sample of summarization on Honda Accord performance.

Due to the fast growth of data over the Internet, automatically opinions summarization has received a lot of attention in recent years. Most studies focus on extractive summarization, where the most salient text units are identified and construct a summary. Ranking candidates for generic summarization usually bases on various handcrafted features such as sentence position and length [Radev et al., 2004], word frequency [Nenkova et al., 2006] or using neural networks for learning salient scores [Zhou et al., 2018].

Table 1.1: An sample of summarization on Honda Accord performance

Reviews	*Summary*
1) I owned this car for only a week, but I am pleasantly surprised by its performance and build quality.	The car is great, both with styling and performance. Overall performance is good but comfort level is poor.
2) I just put it on the highway this weekend and its performance was bad!	
3) Gas mileage is disappointing for a vehicle with this type of performance.	
4) Great performance and handling make this a real Winner!	
5) Overall performance is good but comfort level is poor.	71% positive
6) The car is great, both with styling and performance.	28% negative
7) It is delicious!	14% spam

In opinions summarization, however, this task is required to consider aspects and/or sentiments of text candidates for generating a concise and informative summary [Hu and Liu, 2006]. The popular framework of this problem involves three subtasks [Hu and Liu, 2004]: i) sentiment analysis which assigns sentiment polarity (positive and negative) towards subjects/topics mentioned in a piece of text; ii) semantic textual similarity which

measures the semantic similarity q_{ij} of two sentences i and j, which plays an important role in identifying the most informative sentences as well as redundant ones; iii) aspect discovery which extracts the properties of interested entities (e.g., battery life, design, customer service); and iv) summary generation which uses the three above signals for selecting the most salient opinions and discarding potentially redundant units to build a summary.

Recently, the sentiment analysis task is formulated as a classification problem and trained successfully via supervised learning methods. However, sentiment analysis on social media faces some challenges such as i) text may not be in well-grammar text; ii) content covers a variety of domains (e.g. phone, education); iii) some comments are unrelated to topics or spams.

For the aspect discovery task, there are two main techniques: supervised and unsupervised learning. The former models the aspect extraction as a sequence labeling task. Due to predefining a list of aspect and heavily relying on annotated data, this approach suffers from domain adaptation problems. The latter uses a large amount of unlabeled data for abstracting aspects via the statistical topic modeling LDA [Blei et al., 2003] or the aspect-based autoencoder model [He et al., 2017b]. However, these unsupervised techniques have limitations. First, we have to decide on a suitable number of aspects for each domain. Second, the existing methods require a sufficient amount of data while some domains may not have enough reviews, known as the cold-start problem [Moghaddam and Ester, 2013].

In the semantic textual similarity task, the main challenge is the diversity of linguistic expression. For example, two sentences with different lexicons could have a similar meaning. Moreover, the task requires to measure similarity at several levels (e.g., word level, phrase level, sentence level). These challenges give difficulties to conventional approaches using hand-crafted features.

In this thesis, we study deep learning approaches to address the mentioned challenges of social media text in each task. In addition, instead of using the aspect discovery subtask for discarding redundant information, we propose a novel subtask which does not require to redefine a list of aspect. In the next section, we will describe in details our research question as well as contribution.

1.2 Research Objective and Contribution

The objective of this research is to obtain an effective method for identifying and summarizing opinions on social media. To achieve this aim, the research question is as follow: how to employ deep learning architectures to deal with the challenges of this task. The emergence of deep learning models has provided an efficient way to learn continuous representation vectors for text (e.g., word2vec, fastText, Glove). These representations have a huge contribution to the success of deep learning in NLP area such as machine translation [Maruf and Haffari, 2018], summarization [Chen and Bansal, 2018], text classification [Wang et al., 2018]. The research question is answered through the five subtasks, which are shown in Figure 1.1, as follows:

Sentiment analysis identifies the polarity (positive or negative or neutral) of a comment/review. In this task, Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) are efficient methods. CNN employs filters to capture local dependencies

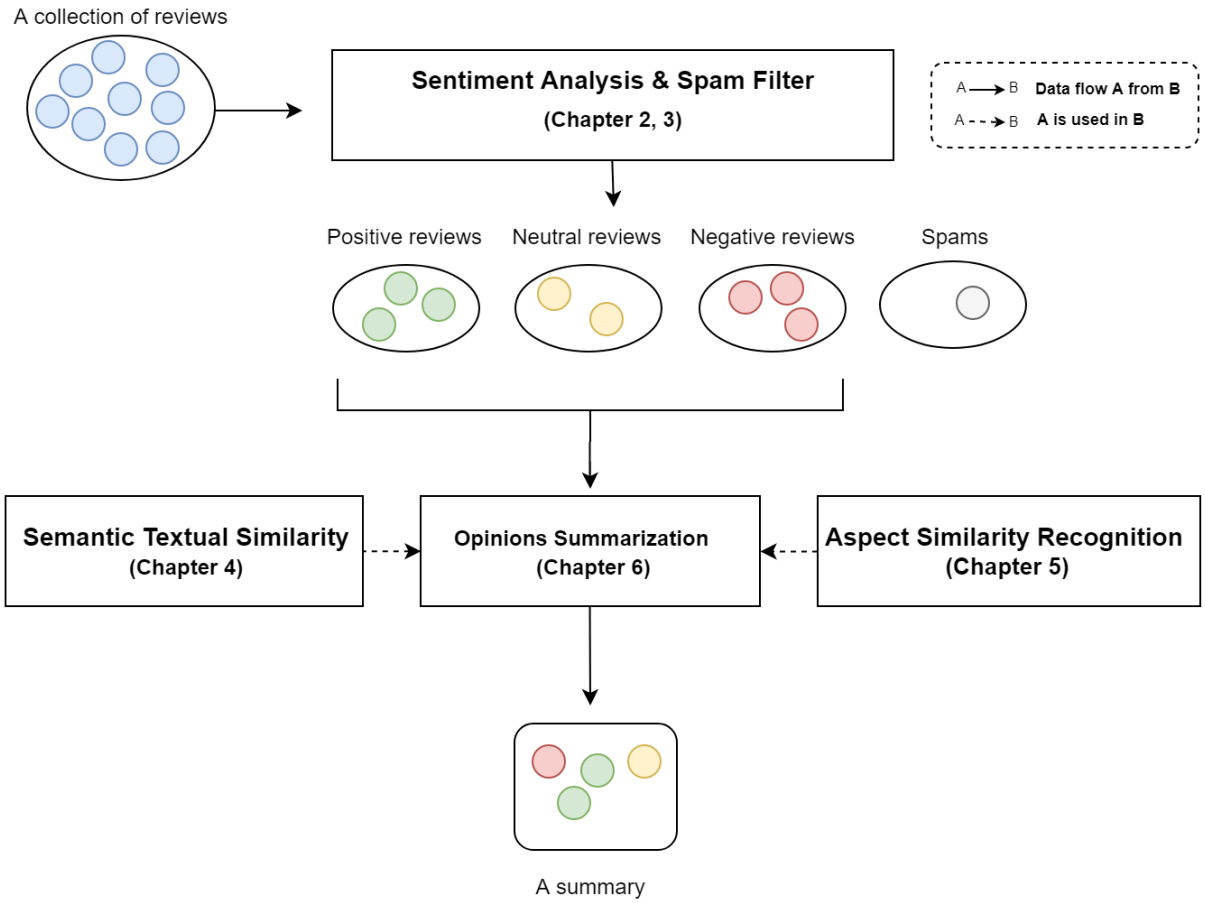


Figure 1.1: The overall framework for opinions summarization.

while LSTM designs a cell to memorize long-distance information. However, integrating these advantages into one model is challenging because of overfitting in training. To avoid this problem, we propose a freezing technique to learn sentiment-specific vectors from CNN and LSTM. This technique is efficient for integrating the advantages of various deep learning models. We also observe that semantically clustering documents into groups is more beneficial for ensemble methods. According to the experiments, our method achieves competitive results on the four well-known datasets: Pang & Lee movie reviews and Stanford Sentiment Treebank for sentence level, IMDB large movie reviews and SenTube for document level.

Subject toward sentiment analysis determines the target subject which the comment gives its sentiment to or the comment contains spam. In this subtask, we i) proposed a convolutional N-gram BiLSTM (CoNBiLSTM) word embedding which represents a word with semantic and contextual information in short and long distance periods; ii) applied CoNBiLSTM word embedding for predicting the type of a comment, its polarity sentiment (positive, neutral or negative) and whether the sentiment is directed toward the product or video; iii) evaluated the efficiency of our model on the SenTube dataset, which contains comments from two domains (i.e. automobile, tablet) and two languages (i.e. English, Italian). According to the experimental results, CoNBiLSTM generally outperforms the approach using SVM with shallow syntactic structures (STRUCT) - the current state-of-the-art sentiment analysis on the SenTube dataset. In addition, our model achieves more

robustness across domains than the STRUCT (e.g. 7.47% of the difference in performance between the two domains for our model vs. 18.8% for the STRUCT).

Semantic textual similarity (STS) measures the semantic similarity q_{ij} of two sentences i and j, which plays an important role in identifying the most informative sentences as well as redundant ones in summarization. Recently, using a pretrained word embedding to represent words achieves success in many natural language processing tasks. According to objective functions, different word embedding models capture different aspects of linguistic properties. However, the Semantic Textual Similarity task requires to take into account of these linguistic aspects. Therefore, this research aims to encode various characteristics from multiple sets of word embeddings into one embedding and then learn similarity/relation between sentences via this novel embedding. Representing each word by multiple word embeddings, the proposed MaxLSTM-CNN encoder generates a novel sentence embedding. We then learn the similarity/relation between our sentence embeddings via Multi-level comparison. Our method M-MaxLSTM-CNN consistently shows strong performances in several tasks (i.e., measure textual similarity, identify paraphrase, recognize textual entailment). Our model does not use hand-crafted features (e.g., alignment features, Ngram overlaps, dependency features) as well as does not require pre-trained word embeddings to have the same dimension.

Aspect similarity Recognition (ASR) predicts a probability r_{ij} that two sentences i and j shares at least one aspect. This task is useful in review summarization where a summarized review needs to cover all aspects as well as avoid redundancy. We propose an attention-cell LSTM model, which integrates attention signals into the LSTM gates. According to the experimental results, the attention-cell LSTM works efficiently for learning latent aspects between two sentences in both settings of in-domain and cross-domain. To facilitate the application of supervised learning models for this task, we construct a dataset ASRCorpus containing two domains (i.e., LAPTOP and RESTAURANT).

Opinions Summarization employs those signals above for ranking sentences. A concise and informative summary of a product is generated by selecting the most salient sentences from reviews. To extract aspects of an expression, most studies require a predefined list of aspects or at least the number of aspects. Instead of extracting aspects, we rate expressions by ASR, which relaxes the limitation of predefining aspects and makes our opinions summarization applicable in domain adaptation. The proposed extractive summarization method using ASR shows significant improvements over baselines on the Opinosis corpus.

1.3 Dissertation Outline

The remainders of this thesis are organized as follows:

Chapter 2 introduces the problem formulation of Sentiment Analysis. We do a literature review to analyze the gaps in current methods. The proposed freezing technique for learning features and clustering support for ensemble are explained in details and evaluated over four well-known datasets against strong baselines. We also discuss the results and analysis some typical error cases for making a conclusion.

Chapter 3 describes the problem formulation of Subject Toward Sentiment Analysis. We do a literature review to analyze the gaps in current methods. The proposed convolutional N-gram BiLSTM word embedding is explained in details and evaluated over the

SenTube dataset containing two domains: TABLET and RESTAURANT against strong baselines. We also discuss the results and analysis some typical error cases for making a conclusion.

Chapter 4 presents the problem formulation of Semantic Textual Similarity. We do a literature review to analyze the gaps in current methods. The proposed M-MaxLSTM-CNN model for employing multiple sets of word embeddings for evaluating sentence similarity/relation is explained in details and evaluated over the benchmark datasets of different tasks against strong baselines. We also discuss the results and analysis some typical error cases for making a conclusion.

Chapter 5 proposes the novel Aspect similarity Recognition task. In the literature review, we firstly survey some recent approaches for aspect category classification, then discuss some researches on measuring a relationship between two sentences. We describe the process of constructing an annotation dataset ASRCorpus containing two domains (i.e., LAPTOP and RESTAURANT) for this task. The proposed convolutional attention-cell LSTM model is explained in details and evaluated against over strong baselines. We also discuss the results and analysis some typical error cases for making a conclusion.

Chapter 6 explains the problem formulation of Opinions Summarization. We do a literature review to analyze the gaps in current methods. The proposed novel aspect-based summarization using Aspect Similarity Recognition is explained in details and evaluated over the Opinosis dataset against strong baselines. We also discuss the results and analysis some typical error cases for making a conclusion.

Chapter 7 concludes our research and discusses future directions based on our works.

Chapter 2

Sentiment Analysis

2.1 Introduction

The emergence of web 2.0, which allows users to generate content, is causing the rapid increase in the amount of data. From this data, we analyze various kinds of knowledges. One of them is sentimental information, which deliveries how evaluations, attitudes, emotions, opinions of particular users towards a person, a product, or an organization are. Analyzing sentiments/opinions from text is a fundamental study and has attracted many research in recent years [Pang and Lee, 2008, Liu, 2012].

Formulating sentiment analysis as a classification problem, Wang and Manning [2012] used a Support Vector Machine variant with Bag of bi-gram and Naive Bayes features (NBSVM). According to experiments on long and short reviews, NBSVM shows robust performances. However, the limitations of Bag of Word model is i)the sparse vectors; ii) not respect to the semantics of words as well as word order. Recently, Mikolov et al. [2013a] propose a word embedding technique for encoding word into a continuous representation. This model has a huge contribution to research of natural language processing. In Paragraph Vector [Le and Mikolov, 2014], the authors employed the technique of Word embedding representation using neural networks [Mnih and Hinton, 2009, Mikolov et al., 2013a] to represent a document or paragraph as a vector. This document modeling outperformed the Bag of Words model in sentiment analysis and information retrieval. Li et al. [2016] have enhanced the architecture of Paragraph Vector by allowing the model to predict not only words but also n-gram features (DVngram). Kim [2014] efficiently applies convolutional neural network (CNN) for semantic composition. In this technique, convolutional filters are utilized to capture local dependencies in term of context windows but these filters fail for long-distance dependencies. By using a memory cell for capturing information over a long period of time, LSTM can handle CNN's limitation. Our motivation is to build a combination approach taking the advantages of these methods.

We organize this paper as follows: Section 2.2 explains the research objective and contribution, Section 2.3 reviews the prior research on opinion mining, Section 2.4 introduces the proposed architecture of encoding and employing sentiment feature vectors, Section 2.5 describes the ensemble model with clustering support, Section 2.6 explains the dataset and experimental setup, Section 2.7 reports and discusses the results of the experiments, and Section 2.8 concludes our work.

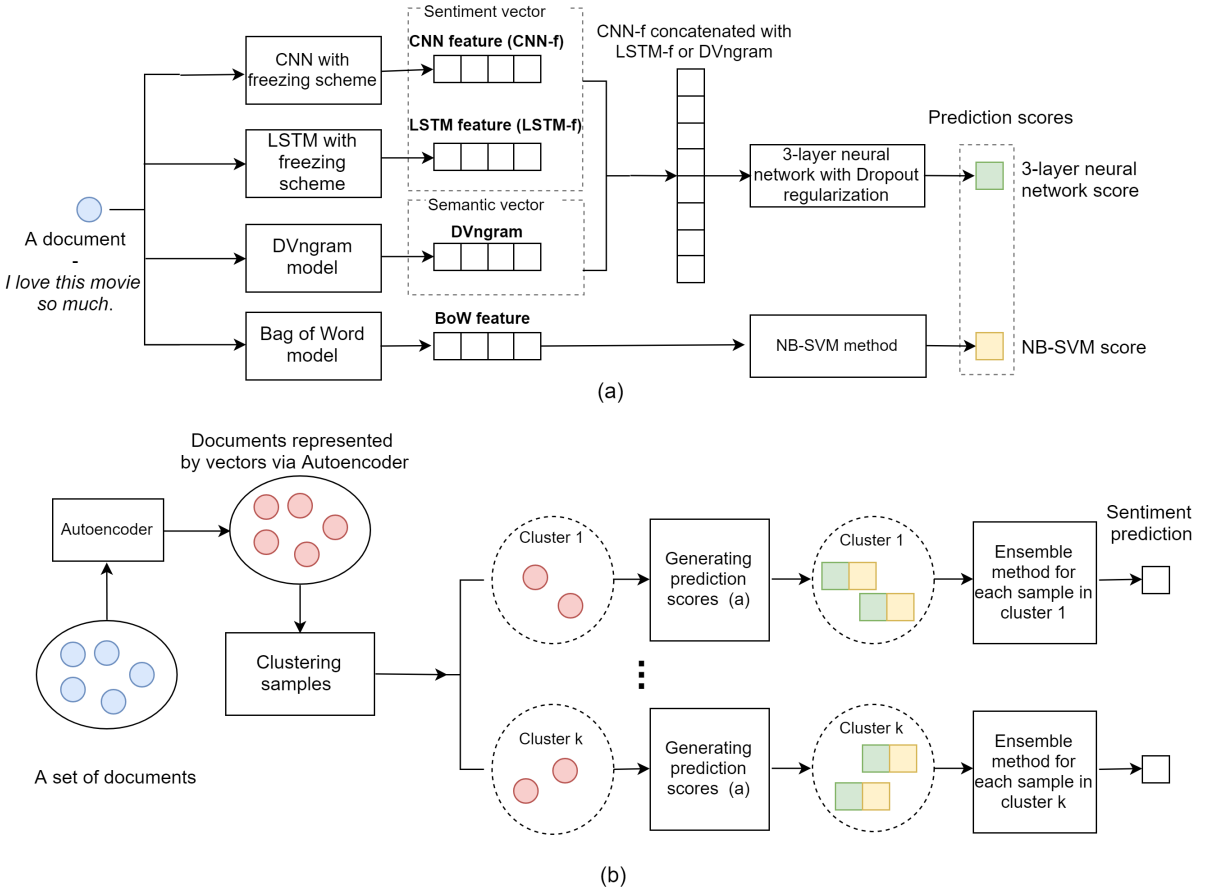


Figure 2.1: The proposed framework for sentiment analysis: (a) Extract two sentiment scores (3-layer neural network score and NV-SVM score), (b) Cluster sentences using autoencoder models and represent the sentences by these sentiment scores for sentiment prediction

2.2 Research Objective and Contribution

As mentioned in Section 1, each model has its own advantages, however, capturing all these characteristics of various models into one model is difficult. According to our experiments, which is described in Section 7, the approach of combining two networks into one model tends to be overfitting. Our objective is to build an ensemble model taking advantages of various models (e.g., generative models, discriminative models) and avoid to be overfitting. Particularly, CNN and LSTM are separately trained to encode sentiment information into feature vectors. To avoid overfitting, we propose a freezing technique during the training phase. For sentiment classification, these sentiment-specific vectors and the semantic-specific DVngram vector are passed into the 3-layer neural network. In sentiment analysis, two sentences with a slight difference could provide opposite sentiments. Generative models, however, have a tendency to encode similar sentences/documents into similar vectors. For that reason, we designed an autoencoder model to learn representation vectors for sentences/documents and used these vectors for clustering. The prediction score of NBSVM method is provided to enhance the sentiment prediction of each cluster. Figure 2.1 shows the architecture of our framework.

We compared our model with some competitive methods on the five well-known

datasets: IMDB large movie reviews [Maas et al., 2011a], Pang & Lee movie reviews [Pang and Lee, 2005], Stanford Sentiment Treebank [Socher et al., 2013], Stanford Twitter Sentiment [Go et al., 2009] and SenTube [Uryupina et al., 2014]. According to experiments, the proposed method achieves competitive performances on sentence level as well as document level. Our main contributions are as follows:

- We generate sentiment vectors via CNN and LSTM under the freezing scheme. These vectors provide a simple and efficient way to integrate the strong abilities of deep learning models.
- We propose a scenario to cluster data into groups of semantic similar sentences/documents. Then, each sentence/document in each group is represented by the prediction scores of the NBSVM method and the proposed 3-layer neural network. We propose an ensemble method to employ these scores.

2.3 Related work

Sentiment analysis studies how to extract people's opinion toward entities. Taboada et al. [2011] assigned sentiment labels to text by extracting sentiment words. This technique did not consider aspects of syntax, context. To deal with this limitation, Saif et al. [2016] introduced a lexicon-based approach for representing semantic sentiment information of words from their co-occurrence patterns, which can perform for both entity-level and tweet-level sentiment detection. Liu [2012] formulated the sentiment analysis as a classification task and applied machine learning techniques for this problem. In this approach, dominant research concentrated on designing effective features such as word ngram [Wang and Manning, 2012], emoticons [Zhao et al., 2012], sentiment words [Kiritchenko et al., 2014]. According to Fersini et al. [2016], several signals (i.e., adjectives, expressive lengthening) are investigated to evaluate their impact on sentiment analysis. The experimental results show that adjectives are more impacting and discriminative than others. Context is a factor for determining the polarity of a word (e.g., cheap design (negative) vs. cheap price (positive)). To take into account this fact, Vechtomova [2017] applies reference corpora with sentiment annotated documents for disambiguating sentiment polarity. This information retrieval method is efficient at word-level but sentence-level. However, the limitation of these above approaches is to require additional resources as well as an intensive effort for designing handcrafted features. In addition, designing features requires a comprehensive knowledge base and depends heavily on the typicality of its knowledge representation which is usually strictly defined and does not allow handling different concept nuances [Cambria, 2016]. By employing deep learning techniques, our proposed model could automatically learn efficient features for sentiment analysis.

Recently, the emergence of deep learning models has provided an efficient way to learn continuous representation vectors for sentiment classification. Mikolov et al. [2013a] introduced learning techniques for semantic word representation. By using a neural network in the context of a word prediction task, the authors generated word embedding vectors carrying semantic meanings. Embedding vectors of words which share similar meanings are close to each other. Because semantic information may provide opposite opinions in different contexts, some researches [Socher et al., 2011, Tang et al., 2014] worked on learning sentiment-specific word representation by employing sentiment text. Giatsoglou

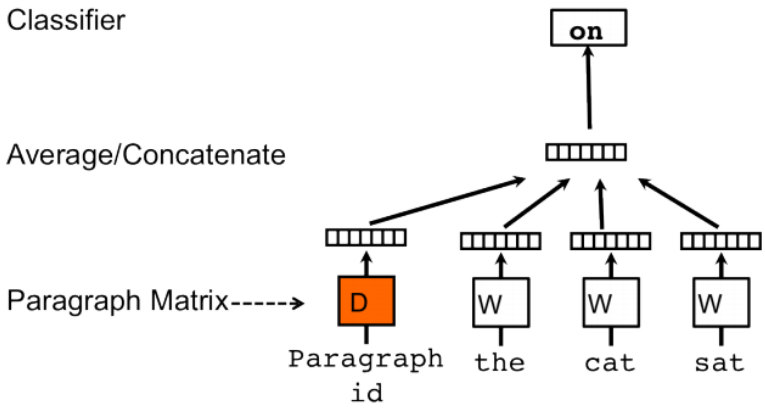


Figure 2.2: Paragraph Vector model which considers paragraph id as a word and uses the word embedding of this paragraph id as a paragraph representation [Le and Mikolov, 2014]

et al. [2017] enhance the word embedding by integrating emotional words. This hybrid approach of word embedding and bag-of-words representations shows competitive performances on English and Spanish tweets. Sentiment analysis requires models to take into account contextual information. Therefore, sentence and document modeling has attracted many studies. Yessenalina and Cardie [2011] modeled each word as a matrix and used iterated matrix multiplication to present a phrase. Le and Mikolov [2014] encoded paragraph into continuous representation by employing the word embedding technique with paragraph information. The authors extended the word embedding learning model by incorporating paragraph information. Given a paragraph, Le's method captures and encodes semantics into a representation vector or a semantic feature. Figure 2.2 illustrates the paragraph vector model, which is trained under a prediction task about the next word in a sentence. Deep recursive neural networks (DRNN) over tree structures were employed to learn sentence representation for sentiment classification such as DRNN with binary parse trees [Irsoy and Cardie, 2014]. Ma et al. [2018] proposed the Sentic LSTM cell effectively incorporating commonsense knowledge into the hierarchical attention encoder. According to experiments, the combination of the proposed attention architecture and Sentic LSTM can outperform the state-of-the-art methods in targeted aspect sentiment tasks. Besides, CNN is a strong technique in computer vision and also applied successfully in natural language processing. For instance, Kim [2014], Zhang and Wallace [2017] used convolutional filters to capture local dependencies in term of context windows and applied a pooling layer to extract global features. Tang et al. [2015] used CNN or LSTM to learn sentence representation and encoded these semantic vectors in document representation by Gated recurrent neural network. Zhang et al. [2016a] proposed Dependency Sensitive CNN to build hierarchically textual representations by processing pretrained word embeddings. As CNN captures local features while LSTM learns global features, there some researches try to integrate these models into one. Wang et al. [2016] used a regional CNN-LSTM to predict the valence arousal ratings of texts. Vo et al. [2017] employed CNN and LSTM as two channels for sentiment analysis. Gan et al. [2017] proposed a hierarchical CNN-LSTM architecture for modeling sentences. In this approach, CNN is used as an encoder to encode a sentence into a continuous representation, and LSTM is used as a

decoder. Nguyen and Le [2018] proposed the N-gram word embedding by employing CNN and LSTM. This approach captures context information of a word to disambiguate its sentiment polarity. Chaturvedi et al. [2016] also use two CNNs for extracting features in Spain and English, then employ Recurrent neural network with Lyapunov filter for subjectivity detection. The Lyapunov filter is a linear matrix inequality for deriving the stability criteria in multi-lingual subjectivity detection.

These prior researches show how well CNN and LSTM work for sentiment analysis. This inspires us to design a scheme for integrating the advantages of these networks into one model. We also observe that integrated models have more parameters than individual models. This fact makes the integrated model more prone to overfitting. To deal with it, we designed a freezing approach for efficiently learning sentiment document representations from two variant deep-learning models: CNN and LSTM. Afterward, these sentiment-specific vectors and the semantic DVngram vector were employed for sentiment classification. This strategy captures the advantages of variant models by using feature vectors, which each model generated. We also used NBSVM in clustering mode to boost the performance of classification. The clustering strategy enhances the problem of semantically similar sentences carrying opposite sentiments.

2.4 Sentiment representation learning

This section introduces the freezing scheme for generating sentiment vectors from two models: CNN and LSTM; and a classification model using these sentiment vectors.

Our motivation is to develop a document representation learning model to capture sentiment information. In our work, we proposed an approach to generate sentiment representation from CNN and LSTM models. Our idea is to train CNN and LSTM models under the sentiment classification task. A deep learning network is considered as a model with two parts: (i) **Building target feature** - from input samples, the first part encodes target information into vectors, (ii) **Classifying layer** - the second part tries to learn a layer (or a boundary) for classifying these vectors into target labels. Sentiment vectors generated by a model, however, are much fitting to the classifying layer of this model. It is not efficient to combine two sentiment vectors generated from two models because each sentiment vector is fitting to its particular classifying layer. To address this problem, we proposed a freezing scheme. According to this scheme, the parameters of the classifying layer are initialized from the uniform distribution and in the training phase, these parameters are kept unchanged. This technique makes sentiment vectors not too fit to a particular classifying layer.

2.4.1 LSTM for sentiment feature engineering - LSTM feature

LSTM is a variant of recurrent neural network [Goller and Kuchler, 1996]. To avoid suffering the exploding or vanishing gradient problem, the LSTM architecture contains a memory cell which preserves its state over a long period of time and non-linear gating units regulating information flow into and out of the cell.

Sentences are encoded into continuous representation vectors by recursively applying an LSTM unit to each input word x_t of sentences and the previous step h_{t-1}. At each time step t, the LSTM unit with l-memory dimension defines 6 vectors in $\mathbb{R}^l$: input gate

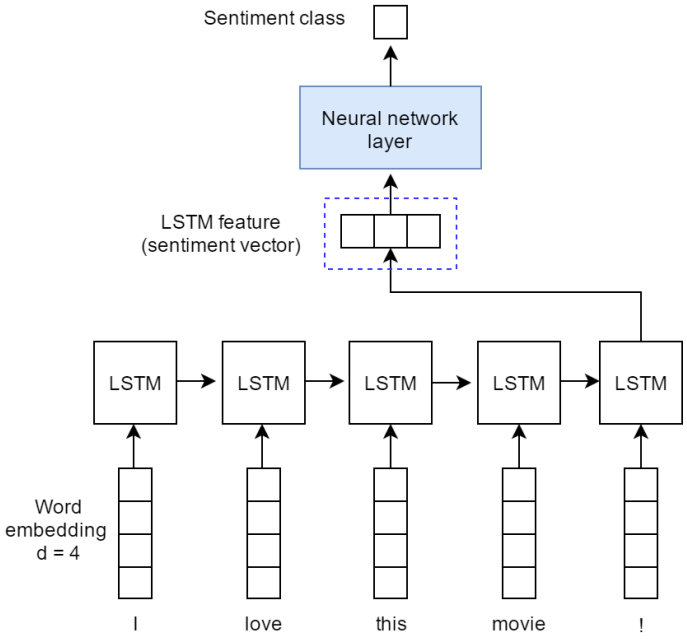


Figure 2.3: LSTM model for sentiment analysis. During training, the neural network layer's parameters (blue one) are frozen.

i_t, forget gate f_t, output gate o_t, tanh layer u_t, memory cell c_t and hidden state h_t as follows (from Tai et al. [2015b]):

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.1}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.2}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.3}$$
$$u_t = \tanh(W_u x_t + U_u h_{t-1} + b_u) \tag{2.4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{2.5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{2.6}$$

where $\sigma, \odot$ respectively denote a logistic sigmoid function and element-wise multiplication; W_i, U_i, b_i are respectively two weights matrices and a bias vector for input gate i. The denotation is similar to the others.

Intuitively, the forget gate decides which previous information in the memory cell should be forgotten, while the input gate controls what new information should be stored in the memory cell. Finally, the output gate decides the amount of information from the internal memory cell should be exposed. These gate units help an LSTM model remember significant information over multiple time steps. The hidden state h_l of the last step, which captures the whole information, is considered as a sentiment feature and classified by a neural network (NN) layer:

$$\hat{y} = \sigma(h_l W_{nn} + b_{nn}) \tag{2.7}$$

where $\hat{y}$ is the prediction output; W_{nn} and b_{nn} are the NN layer's parameters.

Figure 2.3 explains how to employ the LSTM architecture for memorizing sentiment information over sequential data. The model contains two parts: (i) **Building sentiment**

feature - the LSTM layer encodes sentiment information of input into a fixed-length vector; (ii) **Classifying layer** - this sentiment-specific representation vector will be classified by the last neural network layer (the blue layer in Figure 2.3). As applying the freezing scheme, this NN layer's parameters W_{nn} and b_{nn} are unchanged during the training process.

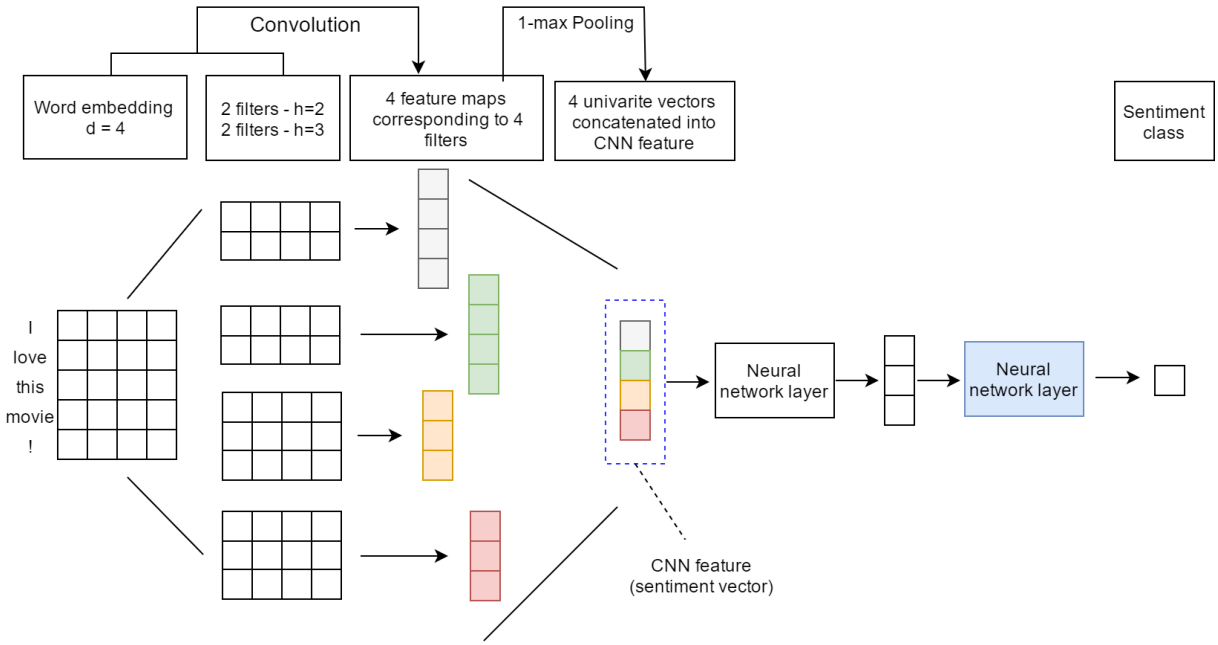


Figure 2.4: CNN for sentiment analysis. Given a sequence of d-dimension word embeddings ($d = 4$), the model applies 4 filters: 2 filters for region size $h = 2$ and 2 filters for region size $h = 3$ to generate 4 feature maps. During training, the last neural network layer's parameters (blue one) are frozen (untrained).

2.4.2 CNN for sentiment feature engineering - CNN feature

We present a sentence of length s as a matrix $d \times s$, where each row is a d-dimension word embedding vector of each word. Given a sentence matrix $\mathbf{S}$, CNN performs convolution on this input via linear filters. A filter is denoted as a weight matrix W of length d and region size h. W will have $d \times h$ parameters to be estimated. For an input matrix $S \in \mathbb{R}^{d \times s}$, a feature map vector $O = [o_o, o_1, ..., o_{s-h}] \in \mathbb{R}^{s-h+1}$ of the convolution operator with a filter W is obtained by applying repeatedly W to sub-matrices of S:

$$o_i = W \cdot S_{i:i+h-1} \tag{2.8}$$

where $i = 0, 1, 2, ..., s - h$, $(\cdot)$ is dot product and $S_{i:j}$ is the sub-matrix of S from row i to j.

Each feature map O is fed to a pooling layer to generate potential features. The common strategy is 1-max pooling [Boureau et al., 2010]. The idea of 1-max pooling is to capture the most important feature v corresponding to the particular feature map by selecting the highest value of that feature map:

$$v = \max_{0 \le i \le s-h} \{o_i\} \tag{2.9}$$

We have explained in detail the operation of one filter. Figure 2.4 shows an example of utilizing four filters with variant region sizes to obtain multiple 1-max pooling values. After pooling, these 1-max pooling values from feature maps are concatenated into a CNN feature k_{cnn} carrying sentiment information. Intuitively, the CNN feature k_{cnn} is a collection of maximum values from the feature maps. To make a connection to these values, an NN layer is employed to synthesize a high-level feature from the CNN feature. Afterward, this high-level feature is fed to an NN layer with sigmoid activation to generate the probability distribution over sentiment labels:

$$x_1 = \sigma(k_{cnn}W_1 + b_1) \tag{2.10}$$
$$\hat{y} = \sigma(x_1W_2 + b_2) \tag{2.11}$$

where $\hat{y}$ is the prediction output; W_1, W_2, b_1 and b_2 are the NN layers parameters.

In the training phase, similar to the strategy in our LSTM model, the last NN layer's parameters W_2 and b_2 are kept untrained to make the sentiment vectors not too fit to a particular classifying layer.

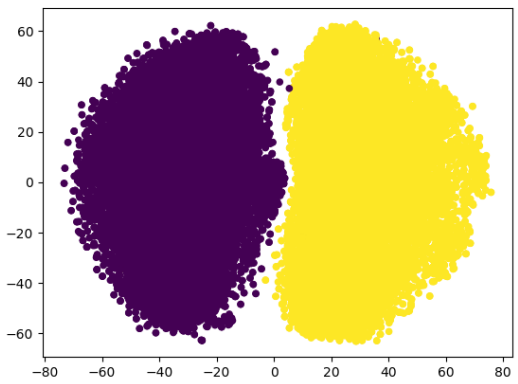
2.4.3 Classifying with sentiment vectors

We visualize the result of synthesizing feature vectors from CNN and LSTM in Figure 2.5. In the development set of CNN sentiment vectors, we observed some unambiguous cases. In other words, it is hard to determine sentiment polarities by only CNN sentiment vectors. Therefore, we add more information to CNN sentiment vectors by concatenating them with LSTM sentiment vectors or DVngram semantic vectors. From figure 2.5c and figure 2.5d, we observe that the classification boundary of CNN-LSTM sentiment features is clearer than the boundary of CNN sentiment features.

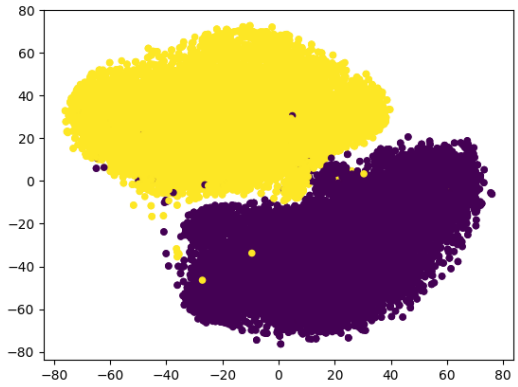
As CNN and LSTM sentiment vectors are, however, generated from models of sentiment classification, these vectors are easily separated in terms of sentimental categories by machine learning methods. In other words, a multi-layer NN sentiment classifier using both of these vectors as input reaches the state of perfect classification on the training set after a few epochs. In this case, the classifier's parameters are not efficiently optimized, and the classifier's performance has no improvement on the testing set, compared with using LSTM or CNN for classification (or we call the model overfitting).

To address this problem, we employ a **3-layer NN** with Dropout regularization Srivastava et al. [2014a] on the first and second layers (Figure 2.6). By randomly dropping out each hidden unit with a probability p on each presentation of each training case, Dropout prevents overfitting and provides a way to combine many variant NN architectures efficiently. By applying Dropout, our model can examine efficiently variant combination ways from feature vectors:

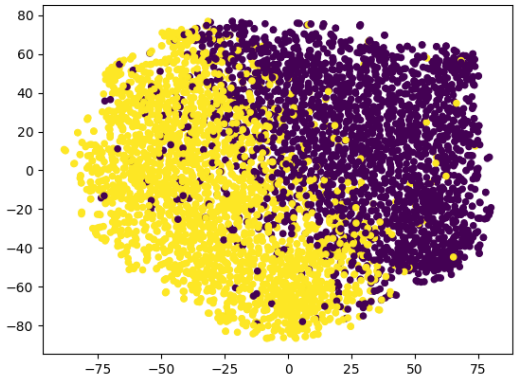
$$y_1 = \sigma(xW_{l_1} + b_{l_1}) \tag{2.12}$$
$$\widetilde{y_1} = dropout(y_1) \tag{2.13}$$
$$y_2 = \sigma(\widetilde{y_1}W_{l_2} + b_{l_2}) \tag{2.14}$$
$$\widetilde{y_2} = dropout(y_2) \tag{2.15}$$
$$\hat{y} = \sigma(\widetilde{y_2}W_{l_3} + b_{l_3}) \tag{2.16}$$



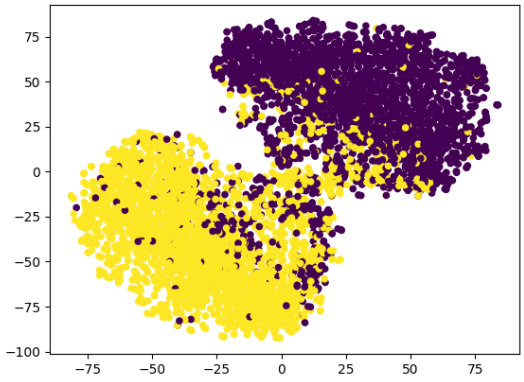
(a) CNN features in the train set



(b) CNN+LSTM features in the train set



(c) CNN features in the development set



(d) CNN+LSTM features in the development set

Figure 2.5: The t-SNE projection for IMDB dataset.

where x is a feature vector concatenated from the CNN sentiment vector and another vector (e.g. LSTM sentiment vector or DVngram semantic vector); $\widetilde{y_i}$ is the thinned output after applying Dropout to y_i; $\hat{y}$ is the prediction output.

2.5 Ensemble with clustering support

Because trained via context information, generative models focus on capturing semantic information rather than sentiment. Therefore, generative models have a tendency to transform semantically similar sentences into similar vectors, which are close in vector space. However, these vectors could carry opposite sentiments. In Figure 2.7, we visualize sentiment distribution of each group of similar documents in term of semantic. In each group, in spite of having similar semantic meanings, these documents carry opposite sentiment polarities. Therefore, it causes some difficulties in sentiment classification.

Our solution is to split data into clusters of semantically similar sentences/documents and then enhance the classification performance of each group by additional features. For clustering, we encode sentences/documents into fix-length vectors via an autoen-

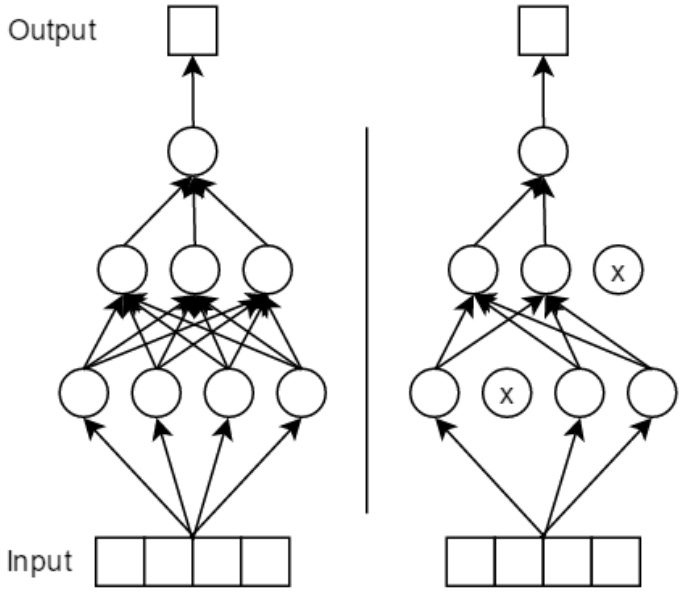


Figure 2.6: Illustration of Dropout technique. **Left**: a 3-layer NN model; **Right**: an example of a thinned network after applying Dropout to a 3-layer NN Srivastava et al. [2014b]

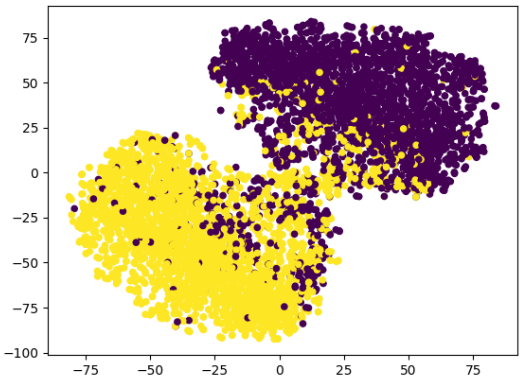
coder model. These vectors then are used for clustering sentences/documents. Each sentence/document in each cluster is represented by the prediction score of the method in Section 2 and the prediction score of NBSVM. The reason for choosing NBSVM is that NBSVM is an efficient method not based on neural network architectures, and using Bag of Word model to represent sentences/documents, which is different from the word embedding representation. We expect NBSVM's score to be a strong feature for boosting the performance of each group.

Given a sentence/document, we have two prediction scores: one from the proposed method in Section 2 and one from NBSVM. A voting approach is applied to these scores. This method considers each classifier f_i as a voter with a confident ratio r_i to the final probability score over classes distribution as follows:

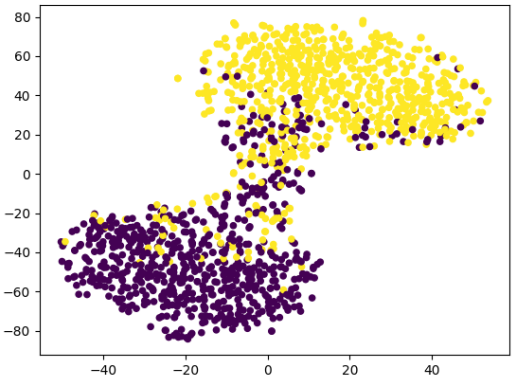
$$p(c_i|x) = \frac{1}{N}\sum_{k=1}^{N} p_k(c_i|x)r_k \tag{2.17}$$

where c_i is the ith sentiment class, N is the number of classifiers, $p_k(c_i|x)$ is the prediction score of the classifier k on the ith class for a sentence/document x.

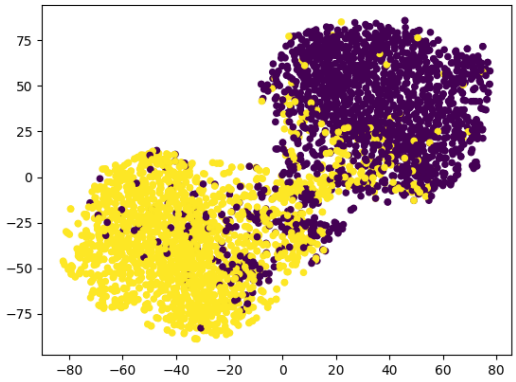
To optimize the classification performance, the ensemble model is trained to assign an optimal confident ratio to each classifier. We propose a neural network approach for learning these optimal values. We consider a feedforward process in a 2-layer NN as a scheme of voting and the NN's weights as confident ratios. The weights are optimized via Adamax algorithm [Kingma and Ba, 2015].



(a) Original set



(b) Cluster 1

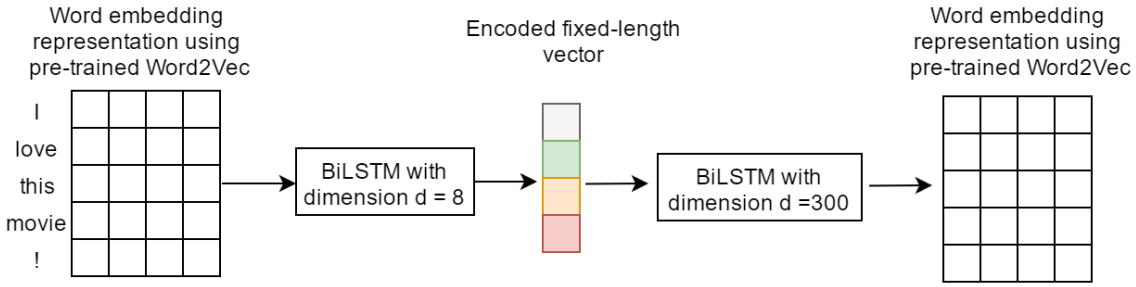


(c) Cluster 2

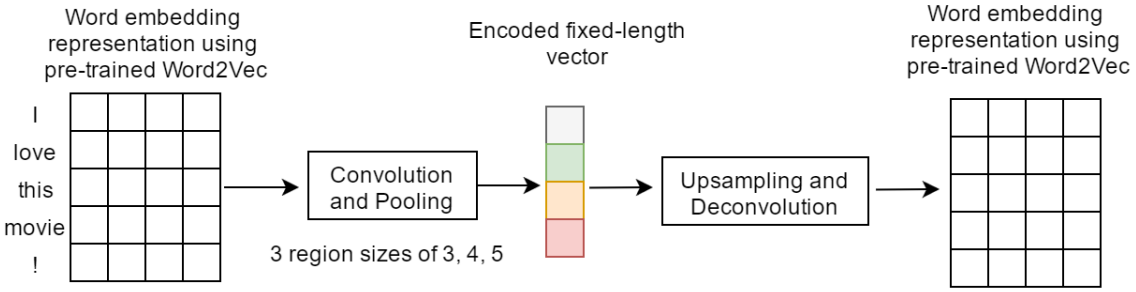
Figure 2.7: The t-SNE projection of CNN+LSTM sentiment features for the IMDB development set. BiLSTM autoencoder is used for clustering

Table 2.1: Statistic summary of datasets. *cv* is 10-fold cross validation. $|V|_{avai}$ is the proportion of vocabulary available in the *Word2Vec* embedding.

Dataset	*average length*	*train size*	*test size*	*vocabulary size*	$V_{\|avai\|}(\%)$
MR-L	300	25000	25000	169940	34
AUTO	98	3284	2472	19919	35
TAB	99	4609	2933	19693	34
MR-S	20	10662	*cv*	18765	88
SST	19	9613	1821	16185	82
Tweet	10	98000	359	129103	19



(a) BiLSTM model



(b) CNN model. In MR-L dataset, each region size has 300 filters. In MR-S and SST dataset, each region size has 100 filters

Figure 2.8: Autoencoder models semantically encode sentences into embedding vectors for clustering.

2.6 Datasets and Experiment setups

2.6.1 Datasets

For evaluation, we use the five well-known datasets: MR-L, MR-S, SST, Tweet, and SenTube. Table 2.1 shows the statistic summary of datasets.

- **MR-L** [Maas et al., 2011a] contains 50,000 reviews from IMDB, where each movie has no more than 30 reviews.
- **SenTube** [Uryupina et al., 2014] is a collection of 38,000 comments. The authors compiled a list of products in two domains: automobiles (AUTO) and tablets (TAB) (e.g. Apple iPad, Motorola xoom, Fiat 500), then collected and annotated comments from either commercials or review videos of those products.
- **MR-S** [Pang and Lee, 2005] labels 5,331 sentences as positive and 5,331 sentences as negative. These sentences are selected from Internet movie reviews.
- **Tweet** [Go et al., 2009] contains 1,6 million tweets, which are automatically labeled via positive/negative emoticons. Only the test set is human-annotated. Follow the

previous research [dos Santos and Gatti, 2014], we randomly select 80,000 tweets for training and 16,000 tweets for validation. Although tweets are short, their $|V|_{avai}$ is high. In other words, most of the words in tweets are unknown to the Word2Vec embedding[1]. This is a major challenge to word embedding based approaches.

- **SST** [Socher et al., 2013] is an extended set of MR-L. In addition to the review sentences, the authors also extract 215,154 phrases and label them via Amazon Mechanical Turk. In our experiments, these phrases are also used for training.

Table 2.2: Hyper-parameters configuration

Hyper-parameter	Value	Grid search's range
LSTM's dimension l	32	$[30, 32, ..., 256]$
CNN's region sizes	3,4,5	Kim (2014)
Number of each CNN's region size	100	$[50, 100, ..., 500]$
CNN's penultimate NN layer	100	$[50, 100, ..., 500]$
3-layer NN's first layer	input's dimension	
Dropout for 3-layer NN's first layer	0.9	$[0, 0.1, ..., 0.9]$
3-layer NN's second layer	64	$[10, 12, ..., 100]$
Dropout for 3-layer NN's second layer	0.5	$[0, 0.1, ..., 0.9]$
K-mean clustering k	2	$[2, 3, ..., 10]$
Ensemble's first NN layer	$3 \times$ input's dimension	$[1, 2, ..., 10]$

2.6.2 Experimental setups

To tune hyper-parameters of our models, we do a grid search on 30% of each dataset. Grid search is a greedy method searching an optimal value for each hyper-parameter in a defined range. In detail, we separately optimize the hyper-parameters of each model (i.e., LSTM and CNN) and then search the optimal value of the 3-layer neural network. Table 2.2 reports the optimal configuration for all the five datasets as well as ranges for grid search. However, the optimal number of each region size is 100 instead of 300 for MR-L.

For word vectors, we obtained the pre-trained word vectors *Word2Vec*. Its vectors have the dimension of 300. In our LSTM and CNN models, these pre-trained word vectors are optimized during the training process.

2.7 Results and Discussion

We compared our models against the other methods showed in table 2.3 on the binary sentiment classification task. In MR-S dataset, we could not reproduce the result 88.1% of CNN [Kim, 2014]. According to the empirical results, our method of combining feature vectors *3-layer NN* outperforms the individual methods: LSTM, CNN, and DVngram. That proves the efficiency of the feature combination strategy. In addition, our ensemble method with clustering support obtains competitive performances on the MR-L, MR-S,

[1]https://code.google.com/p/word2vec/

Table 2.3: Accuracy results on the binary sentiment classification task. ***3-layer NN*** $(F_1 + F_2)$ denotes using feature vector F_1 and F_2 as input; **CNN-f**, **LSTM-f** denote sentiment-specific feature vectors generated from the proposed CNN, LSTM respectively; ***Ensemble*** $(p_1 + p_2)$ denotes applying the proposed Ensemble for the prediction scores of p_1 and p_2.

Method		MR-S	SST	MR-L	AUTO	TAB	Tweet
LSTM		80.17	87.81	86.23	71.28	75.79	77.9
CNN (2014)		81.31	86.33	91.18	73.1	76.85	77.1
DVngram (2016)		73.51	74.2	92.14	70.43	74.05	72.15
NBSVM (2012)		79.26	80.39	91.87	72.17	73	77.43
DV-Ensemble (2016)		-	-	93.05	-	-	-
DAN (2015)		80.3	86.3	89.4	-	-	-
SA-LSTM (2015)		80.7	-	92.8	-	-	-
SkipThought (2015)		79.4	82.9	-	-	-	-
DSCNN-Pretrain (2016a)		82.2	**88.7**	90.7	-	-	-
Infersent (2017a)		81.1	84.6	-	-	-	-
CharSCNN (2014)		-	85.7	-	-	-	**86.4**
BERT (2019)		-	-	**94.9**	-	-	-
Proposed methods							
3-layer NN (CNN-f+LSTM-f) (1)		81.59	88.41	91.16	73.75	76.96	78.52
3-layer NN (CNN-f+DVngram) (2)		81.11	86.66	92.98	73.67	76.95	76.24
Without clustering	Ensemble (1)+NBSVM	82.18^a	88.36^a	92.50^a	**73.95**^{a}	**77**^{a}	78.62^a
	Ensemble (2) + NBSVM	81.1^a	87.31^a	93.25^a	73.82^a	76.95^a	77.75^a
CNN autoencoder	Ensemble (1)+NBSVM	82.2^b	88.46^b	92.55^b	73.93^b	76.94^b	79.31^b
	Ensemble (2)+NBSVM	81.74^b	86.87^b	93.29^b	73.8^b	76.9^b	79.02^b
BiLSTM autoencoder	Ensemble (1)+NBSVM	**82.22**^{b}	88.58^b	92.54^b	73.92^b	76.91^b	79.52^b
	Ensemble (2)+NBSVM	81.8^b	87.09^b	93.32^b	73.78^b	76.89^b	79.11^b

[a,b] denote results statistically significant at $p < 0.05$ via the pairwise t-test compared with the 3-layer NN method and the Without clustering method using the same features respectively.

SST and Tweet datasets. However, clustering support does not work in AUTO and TAB where the dataset size is small. We discuss more the size of clusters in Section 7.3. As we mentioned in Section 3, NBSVM uses a different way to present sentences/documents and a different approach for learning (a discriminative model), so it gives significant support in our ensemble method. On document level, LSTM method produces a much lower performance than DVngram method. As a result, the feature vectors generated from LSTM model does not support as well as DVngram's vectors when combining with CNN feature vectors. In case of low $V_{|avai|}$, Bert and CharSCNN achieve strong performances. It makes sense because these methods use external resources to pre-train word embeddings (BERT) or character embeddings (CharSCNN).

Table 2.4: Accuracy results of the NN model on various features. **CNNorg**, **LSTMorg** denote sentiment vectors generated from the proposed CNN, LSTM without freezing the last NN layer respectively

Feature	MR-S	SST	MR-L	AUTO	TAB	Tweet
CNNorg	80.61	86.05	91.22	73	76.85	77.5
CNN-f	**80.89***	**86.27***	**91.38***	**73.1***	**76.9***	**77.6***
LSTMorg	78.97	86.99	**85.5**	71.28	74.9	76.6
LSTM-f	**79.11***	**87.64***	85.14*	**72.05***	**75.79***	**78.15***
CNNorg + LSTMorg	80.95	87.31	90.34	72.49	76.81	77.16
CNN-f + LSTM-f	**81.59***	**88.41***	**91.16***	**73.75***	**76.96***	**78.52***
CNNorg + DVngram	80.6	85.34	92.66	73.38	76.34	76.04
CNN-f + DVngram	**81.11***	**86.66***	**92.98***	**73.67***	**76.95***	**76.24***
LSTMorg + DVngram	79.38	87.2	**90.41**	71.84	75.58	77.44
LSTM-f + DVngram	**79.59** *	**88.14***	88.04*	**71.97***	**75.96***	**77.83***

* denotes results statistically significant at $p < 0.05$ via the pairwise t-test compared with the method without using freezing.

2.7.1 Freezing vs Unfreezing

Compared against conventional approaches, our model freezes (untrain) the last NN layer's parameters to prevent the effect of overfitting. To evaluate the efficiency of this technique, we compared our vector's performance against the sentiment-specific vector from the unfreezing scheme (the conventional way). We passed these vectors to our 3-layer NN model to achieve the results (details in Table 2.4). One interesting observation is that the performance of a feature vector in freezing mode is better than one in unfreezing mode for most of the cases. In addition, we combined a sentiment-specific vector with the semantic-specific vector - DVngram for evaluating the performance. In general, our freezing scheme provides higher performance than the unfreezing scheme. The experimental results show that our freezing scheme works more efficiently on CNN than LSTM, especially in a case of combining a sentiment-specific vector and a semantic-specific vector.

2.7.2 Evaluation on combining features

In this section, we compared in performance our approach of combining features from variant models against **Merging scheme** which horizontally merges variant models (details in Figure 2.9).

From the result reported in Table 2.5, we found that our approach for feature vectors combination is applied more efficiently to CNN than LSTM. In the scheme of combining feature vectors, the CNN feature vector provides robust performance, while the LSTM feature vector provides inconsistent results: better when combining with the CNN feature vector, worse when combining with the DVngram vector (compared with Merging scheme). In most of the cases in Merging scheme, a composition model (i.e., CNN-LSTM) try to reproduce the result of its child models (e.g., CNN, LSTM) and does not provide a significant improvement.

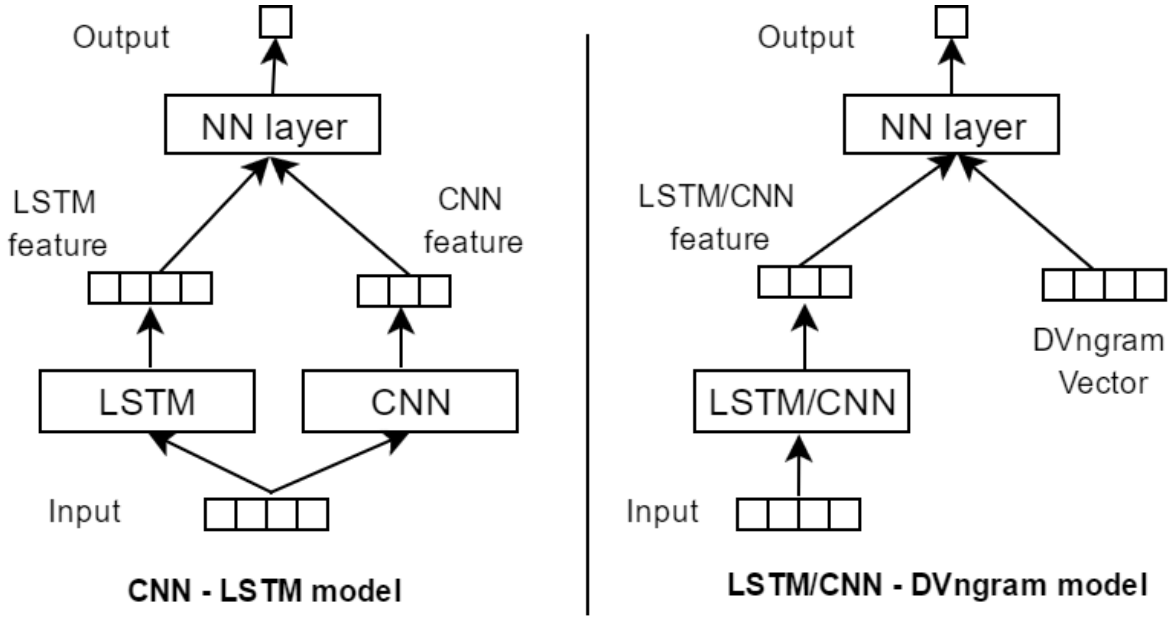


Figure 2.9: The architecture of merging models.

Table 2.5: Accuracy results of features combining scheme and Merging scheme

Method	MR-S	SST	MR-L	AUTO	TAB	Tweet
3-layer NN (CNN-f+LSTM-f)	**81.59***	**88.41***	**91.16***	**73.75***	**76.96***	**78.52***
CNN-LSTM	81.07	86.49	91.07	73.4	75.82	74.98
3-layer NN (CNN-f+DVngram)	**81.11***	**86.66***	**92.98***	**73.67***	**76.95***	**76.24***
CNN-DVngram	80.79	85.39	92.12	73.5	76.37	75.21
3-layer NN (LSTM-f+DVngram)	79.59*	**88.14***	88.04*	**71.97***	**75.96***	**77.83***
LSTM-DVngram	**80.61**	86.49	**92.08**	70.22	73.88	76.11

* denote results statistically significant at $p < 0.05$ via the pairwise t-test compared with the Merging scheme using the same features.

2.7.3 Evaluation on clustering support

In this section, we discuss how clustering methods contribute to the overall performance of classification. For experiments, we employ two clustering algorithms K-Means [Arthur and Vassilvitskii, 2007] and Birch [Zhang et al., 1996] with various settings of K (the number of clusters). We select these algorithms because K-Means is partitioning clustering while Birch is hierarchical clustering. In addition, we choose the largest dataset MR-L for this experiment.

According to the results in Table 2.6, we observe a difference in performance between K-Means and Birch. K-Means shows strong improvement in large clusters compared with smaller clusters while Birch shows reverse results. For too small clusters (e.g. Birch with $C_3 = 592$ samples), the proposed approach, however, does not work well. Generally, K-Means is more efficient in our ensemble approach compared with Birch.

We also evaluate our model with different numbers of clusters. Figure 2.10 shows the proposed model's performance with K in a range $[0, 10]$. K-Means and Birch have the same behavior when the value of K increases. In other words, their charts have similar patterns. When K is greater than 9, the contribution of clustering becomes negative. The

Table 2.6: Accuracy results on MR-L dataset with K-Means v.s. Birch. K denotes the number of clusters, C_i stands for cluster i and *Sample* is the number of samples in a cluster. The number in () denotes the increase/decrease in accuracy compared to the ensemble approach without using clustering.

		K-Means			Birch		
		Sample	*Accuracy*		*Sample*	*Accuracy*	
K=2	C_1	6703	93.38(0)	93.32(+0.07)	4217	93.48(+0.14)	93.26(+0.01)
	C_2	18297	93.3(+0.1)		20783	93.21(+0.01)	
K=3	C_1	3967	93.04(-0.13)	93.26(+0.01)	20780	93.23(+0.01)	93.24(-0.01)
	C_2	13505	93.32(+0.04)		3628	93.3(+0.2)	
	C_3	7528	93.26(+0.02)		592	93.26(-0.15)	

more clusters we have, the smaller each cluster is. This fact makes the training processing inefficient because of the small training set in each cluster.

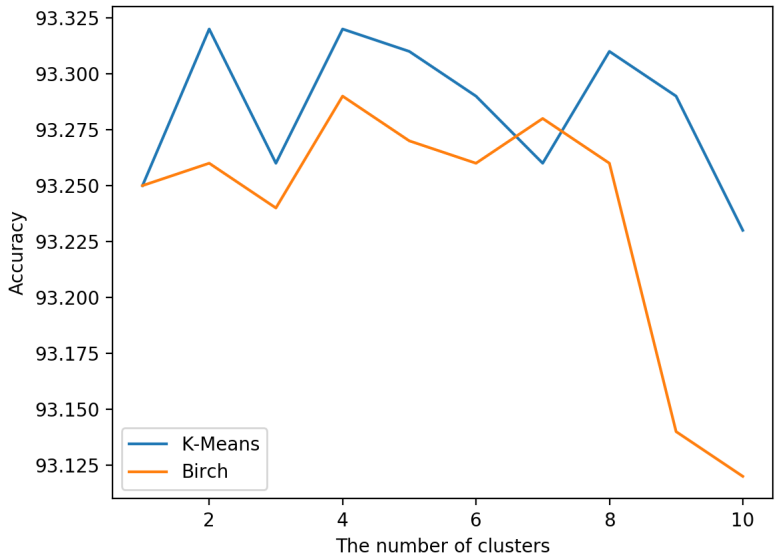


Figure 2.10: The ensemble model's performance on MR-L dataset.

2.7.4 Quality analysis

To obtain a better sense of the proposed model's advantages and disadvantages, we manually inspect some typical samples, which are shown in Table 2.7. These sentences are good examples of the proposed model's performance compared to other approaches.

For simple sentences which only carry words in the same sentiment polarity (example #1 and #2), the proposed model easily identifies the sentiment polarities. In long sentences (example #3, #4 and #5), the model using CNN and LSTM sentiment features tried to capture sentiment words in these sentences (e.g., romantic, fresh, anguished, bitter) but it failed to interpret the whole sentences, whereas the proposed model with the support of NBSVM can correctly classify their sentiment polarities. This result agrees

with the claim of Wang and Manning [2012] that NBSVM with bi-gram features and NB log-count ratios consistently performs well on long sentences/documents.

Table 2.7: Some typical samples. CNN+LSTM denotes the results of the 3-layer NN using CNN and LSTM sentiment features, True denotes the true label with 0, 1 for negative, positive sentiment labels respectively.

id	**Sentence**	CNN+LSTM	Proposed	True
1	*a refreshingly realistic, affectation free coming of age tale*	1	1	1
2	*a thoughtful, visually graceful work*	1	1	1
3	*apparently, romantic comedy with a fresh point of view just doesn't figure in the present hollywood program*	1	0	0
5	*the last scenes of the film are anguished, bitter and truthful mr koshashvili is a director to watch*	0	1	1
6	*clayburgh and tambor are charming performers neither of them deserves eric schaeffer*	1	0	0
8	*You've seen them a million times.*	1	1	0
9	*A whole lot foul, freaky and funny.*	0	0	1

In examples #8 and #9, we observe that the source of false hits is the lack of context information. Depending on context or domains, some words (e.g., foul, freaky, million times) carry a positive or negative sentiment. Without contextual information, it is difficult to decide the sentiment polarities of these cases. Therefore, we believe that the promising direction in future work will be to improve the model for capturing context information.

2.8 Conclusion

In this work, we apply the proposed freezing technique to CNN, LSTM for generating feature vectors. This approach is simple but efficient to combine the advantages of various models. To improve the weakness of generative models, we propose a strategy to cluster documents/sentences by their semantic similarity. A neural voting ensemble with additional NBSVM is used to boost the performance of each group. The approach obtains the strong performance in sentiment analysis.

In our work, we just researched on simple models. It is interesting to apply our freezing scheme approach to combination models (e.g., multi-channel CNN-LSTM, hierarchal LSTM) for generating feature vectors. In addition, our clustering is based on semantic similarity. Research on other kinds of similarity can lead to valuable results.

Chapter 3

Subject Toward Sentiment Analysis on Social Media

3.1 Introduction

Social network sites are platforms with many facets. These platforms are multi-domain, multilingual and multicultural since users from different countries can upload images/videos as well as comments about various topics in different languages. According to Aliaksei Severyn's study (2016), 60-80% of comments from YouTube, which is a well-know video-sharing website, do actually contain opinions. Therefore, a robust method of sentiment analysis in such an environment is a high interest for both the industry and the research community. For this reason, our research focuses on YouTube. This environment raises some challenges to opinion mining such as i) many comments may not be in well-grammar text; ii) YouTube covers a variety of domains (e.g. phone, education) that requires a robust approach to extract opinions from different topics; iii) words showing sentiment can refer to either the content itself of video or the advertised product; iv) some comments are unrelated to topics or spams; v) YouTube's content has a large variety of languages; thus it requires a method to be independent to grammar of natural languages.

To address these challenges, we proposed the Convolutional N-gram BiLSTM word embedding model for sentiment analysis by capturing semantic and contextual information. The advantage of this approach does not require any linguistic resources or handcraft features but achieves robust performance in a multi-lingual environment.

The remainder of this paper is organized as follows: Section 3.2 outlines the motivation and contribution of the work, Section 3.3 reviews the previous research on opinion mining, Section 3.4 introduces the architecture of our model, Section 3.5 describes the SenTube dataset and the tasks, Section 3.6 reports and discusses the results of the experiments, and Section 3.7 concludes our work.

3.2 Motivation and contribution of the work

Most prior research on sentiment analysis relied on Bag of Word (BOW) representation. For instance, Wang and Manning [2012] used a Support Vector Machine variant with Naive Bayes feature (NBSVM). Presenting a document or a sentence with Bag of bi-gram features, NBSVM consistently performs well across datasets of long and short reviews.

The winning system [Mohammad et al., 2013] of the SemEval 2013 shared task used a BOW representation together with a sentiment lexicon in Support Vector Machine. However, BOW representation loses the ordering of words, and it also ignores the semantics of words. The following example illustrates the problem of BOW representation:

iPad 2 is better. the superior apps just destroy the ***xoom.***

In the above comment about the product **xoom**, there are one negative word and two positive words, but the sentiment toward the product is negative. This situation is common in the YouTube environment where people can mention about a video and/or the product in that video and/or another product. Under BOW representation, we cannot determine which word a polarity word give an opinion toward. To address the weakness of BOW representation, Severyn et al. [2016] encoded a comment into a shallow syntactic tree with enriched tags (STRUCT). The advantage of the STRUCT is to capture sentiment words as well as essential concepts about the product and negation words. This approach requires a POS-tagger tool, a chunker tool and a set of sentiment lexicons for each language. Therefore, the applicability of this method in a multilingual environment is limited. In addition, which polarity a sentiment word depends on the context of that word. This information could not be captured by the tree structure.

Bengio et al. [2003] introduced an unsupervised framework that learns the continuous vector for each word. In this vector space, semantically similar words have similar vector representations (e.g., "*strong*" is close to "*powerful*"), whereas BOW representation gives the same distances between two words (e.g., $distance($"*strong*", "*powerful*"$) = distance($"*strong*", "*weak*"$)$). This word embedding representation has contributed to the success of deep learning methods in natural language processing, especially sentiment analysis.

A word can have different functions/meanings in different contexts. For example, the two words *"has"* from comment #1 and #2 in table 3.1 have two different functions (i.e. verb vs auxiliary verb); or in comment #3 and #4, the adjective *"cheap"* bears different sentiments (i.e. negative vs positive). In spite of different functions/meanings which a word could have, the word embedding model gives a unique vector for each word. Consequently, this representation loses the word's function as well as the word's contextual meaning.

By observing the neighbor words of a word, a human could identify the function, meaning, and sentiment of that word. Understanding correctly every word in a sentence helps to capture clearly the meaning of that sentence. This inspires us to design convolutional filters to encode words and their contextual information into a convolutional N-gram word embedding representation. However, the convolutional filters have two limitations: i) long distance contextual information is missing because of the relatively small size of filters. For example, the word *"Although"* restrains the negative sentiment of the word *"outdated"* in comment #5 in table 3.1. Because these two words have a long distance relationship, convolution filters miss this contextual information; ii) the position of a word in a sentence/document often describes how important that word is (e.g. placing an adjective in the first position of a review gives an emphasis on that adjective). However, the convolutional operator does not consider word's position, it applies the same filters to each word. To address the weakness of convolutional filters, Bidirectional Long Short Term Memory (BiLSTM) [Dyer et al., 2015] is applied to the convolutional N-gram word embedding representation for capturing long distance contextual information and taking into account of words' position. This representation is called the convolutional N-gram

Table 3.1: Some YouTube comments from the SenTube dataset

No.#	**Comment**
1	*as would I, jaguar always **has** a place in my heart. a place that BMW cannot fill*
2	*I agree however now Jaguar **has** been bought out the reliability should increase.*
3	*... Nobody wants it because it's made of **cheap** materials...*
4	*... i couldnt believe it when my friend told me about this site. and i can tell u , ive seen this car selling ridiculously **cheap** on this site.*
5	***Although** people say the iPads multitasking is **outdated**, i think it makes more sense - you pause one app and flick to another. I would very rarely want an app to run in the background on a tablet.*

BiLSTM (CoNBiLSTM) word embedding. Figure 3.1 shows an overview of our proposed framework for YouTube sentiment analysis.

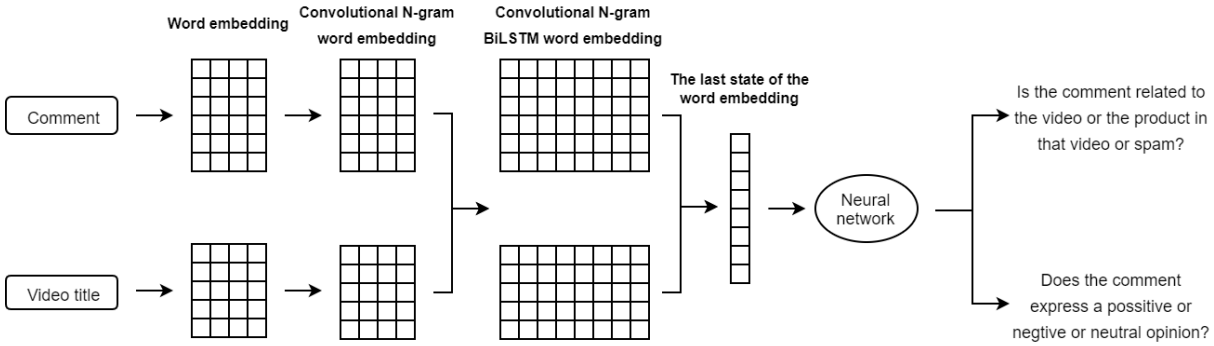


Figure 3.1: An overview of our sentiment analysis model

The contribution of our research is:

1. *CoNBiLSTM word embedding representation*: To enhance the conventional word embedding representation for capturing contextual information, we designed multiple convolutional filters with variant sizes. The convolution N-gram vectors generated by applying these filters on the word embedding representation are fed to a Bidirectional Long Short Term Memory (BiLSTM) for encoding long distance contextual dependencies and information of word's position.

2. *Applicable to any language*: because our approach relies on the word embedding representation learned in an unsupervised manner, our model does not require any linguistic preprocessor (e.g. POS-tagger, chunker). In some languages (e.g. Japanese, Chinese, Vietnamese) where words are not delimited by spaces, word embeddings could be trained on syllables or characters instead of words. Phung and De Vine

[2015] found that word segmentation does not give any advantage in learning Vietnamese word embedding representations for text summarization. In Vietnamese social media, Vo et al. [2017] observe that comments are usually informal and contain several grammar mistakes. These facts challenge word segmentation tools. As a result, syllable embedding models outperform word embedding models for Vietnamese sentiment analysis. Therefore, the applicability of our approach to multilingual environments is promising.

3. *Multilingual experiments*: to validate the robustness of this approach across languages, we carried out experiments on English and Italian comments in the SenTube dataset [Uryupina et al., 2014].

4. *Cross domain experiments*: our novel word representation is learned from context data, while each domain (e.g. tablets, automobiles) has its own word distribution. Therefore, it is important to evaluate the model's robustness in a cross-domain manner where the model is trained on a domain and tested on another domain. In our work, we performed experiments on two domains: automobiles and tablets in the SenTube dataset.

5. *CoNBiLSTM vs. BiLSTM*: since BiLSTM has an ability to capture long and short dependencies, there is an argument over whether or not convolutional filters have support to BiLSTM for capturing contextual information. To confirm the efficiency of using convolutional filters for encoding contextual information, we performed experiments to compare the performances of the two models (BiLSTM and CoNBiLSTM) for classification tasks.

In most of the experiments, our model outperforms BiLSTM model and the prior work [Severyn et al., 2016], which uses the shallow syntactic tree with Support Vector Machine (STRUCT). Especially in the cross-domain experiment, the proposed approach is more robust than the prior work.

3.3 Related work

Sentiment analysis is a study of determining people's opinions, emotions toward entities. We firstly review the work on English sentiment analysis and then focus on the work applied in multilingual settings.

3.3.1 Sentiment analysis in English

Feature based approach

Taboada et al. [2011] assigned sentiment labels to text by extracting sentiment-bearing words. To apply supervised machine learning techniques for this task, Liu [2012] formulated the sentiment analysis task as a classification problem. In this approach, dominant research concentrated on designing effective features such as word ngram [Wang and Manning, 2012], emoticon [Zhao et al., 2012], sentiment words [Kiritchenko et al., 2014]. Saif et al. [2016] introduced a lexicon-based approach for representing semantic sentiment information of words from their co-occurrence patterns, which can perform for both entity-level and tweet-level sentiment detection. For sentiment detection, Fersini et al. [2016]

investigates the impact of several expressive signals (i.e. adjectives, pragmatic particles, and expressive lengthening). These signals have been employed to enrich the feature space of baseline and ensemble classifiers. According to the experimental results, the author concluded that adjectives are more discriminative and impacting than pragmatic particles and expressive lengthening. The polarity of a single word is impacted by its context (e.g. cheap price (positive) vs. cheap material (negative)). To disambiguate contextual sentiment polarity at word-level, Vechtomova [2017] introduced an information retrieval approach which uses reference corpora with sentiment annotated documents. Although this approach was shown to be an effective alternative to machine learning approaches for disambiguating word-level contextual sentiment polarity, the method has not shown an improvement compared to other methods in sentence-level sentiment analysis. Instead of using additional reference corpora for disambiguating sentiment polarity, we design the CoNBiLSTM model to learn contextual sentiment polarity for each word. The experimental results show the efficiency of this approach at sentence-level.

Metaheuristic-based methods have also been applied to opinion mining. Gupta et al. [2015] proposed a Particle Swarm Optimization approach to select features and applied these features to the Conditional Random Field method for classifying sentiment. Compared to existing other systems, the method attained promising performance with the much reduced feature set. Pandey et al. [2017] employed K-means to resolve the problem of the random initialization in the cuckoo search. By optimizing the cluster-heads of sentiment dataset, the method outperformed the cuckoo search and the improved cuckoo search.

However, designing handcrafted features requires an intensive effort as well as linguistic preprocessing tools (e.g. POST-taggers, chunkers). This weakness limits the applicability on a multilingual environment like YouTube. In the next section, we review some deep learning techniques for sentiment analysis. One of the deep learnings main advantages is its capacity to learn new features from a limited set of features. Therefore, it is a promising approach for a multilingual environment.

Deep learning approach

Recently, the emergence of deep learning models has provided an efficient way to learn continuous representation vectors for sentiment classification. Bengio et al. [2003] and Mikolov et al. [2013a] introduced learning techniques for semantic word representation. By using a neural network in the context of a word prediction task, the authors generated word embedding vectors carrying semantic meanings. The embedding vectors of words which share similar meanings are close to each other. However, semantic information might provide opposite opinions in different contexts. Therefore, some research [Maas et al., 2011b, Socher et al., 2011, Tang et al., 2014] worked on learning sentiment-specific word representation by employing sentiment text. For sentence and document level, composition approach attracted many studies. Yessenalina and Cardie [2011] modeled each word as a matrix and used iterated matrix multiplication to present a phrase. Deep recursive neural networks (DRNN) over tree structures were employed to learn sentence representation for sentiment classification such as DRNN with binary parse trees [Irsoy and Cardie, 2014], Recursive tensor neural network with sentiment treebank [Socher et al., 2013]. Convolutional neural network (CNN) has recently been applied efficiently for semantic composition [Kalchbrenner et al., 2014b, Kim, 2014]. This technique uses

convolutional filters to capture local dependencies in term of context windows and applies a pooling layer to extract global features. Le and Mikolov [2014] applied paragraph information into the word embedding technique to learn semantic document representation. Tang et al. [2015] used CNN or LSTM to learn sentence representation and encoded these semantic vectors in document representation by a gated recurrent neural network. Zhang et al. [2016a] proposed Dependency Sensitive CNN to build hierarchically textual representations by processing pretrained word embeddings. Huy Tien and Minh Le [2017] propose a freezing scheme to learn sentiment features. This technique efficiently integrates the advantages of LSTM-CNN and avoids overfitting. Although contextual information might change the sentiment polarity of a word, this property is still not carefully considered in the prior work. To confirm the efficiency of the proposed CoNBiLSTM for encoding contextual sentiment polarity, we carried out experiments and quality analysis to compare the performances of CoNBiLSTM and BiLSTM.

3.3.2 Sentiment analysis in multi-lingual setting

Severyn et al. [2016] proposed a shallow syntactic tree with enriched tags. This structure captures not only words from the sentiment lexicons, but also important concepts about the product and negation words. For evaluation, the work has released a YouTube corpus (in Italian and English). According to the experimental results, the method improves performance for both the languages. Vilares et al. [2017] evaluated the performance of classifying multilingual polarity in various settings such as a multilingual model trained on a multilingual dataset, a dual monolingual model with/without language identification. The experimental results on English and Spanish tweets showed the efficiency and robustness of the multilingual approach.

To avoid using syntactic features, Giatsoglou et al. [2017] proposed a hybrid vectorization approach for integrating emotional words along with the word embedding approach. The experiments are carried out on English and Greek languages. By combining word embedding and Bag-of-Words representations, the hybrid method outperformed existing other approaches.

As relying on linguistic resources (e.g. emotional words, sentiment lexicons, POS-tagger), these above approaches could not be applied for low-resource languages. In contrast, our work enhanced the word embedding representation by capturing contextual information without using any additional linguistic resources. Employing convolutional filters and BiLSTM, the proposed contextual word embedding model achieves a better generalization across different domains, where the word distribution and vocabulary changes, compared to the prior work [Severyn et al., 2016].

3.4 Convolutional N-gram BiLSTM word embedding

In this section, we introduce i) the background of BiLSTM architecture, and then ii) the proposed model - CoNBiLSTM word embedding.

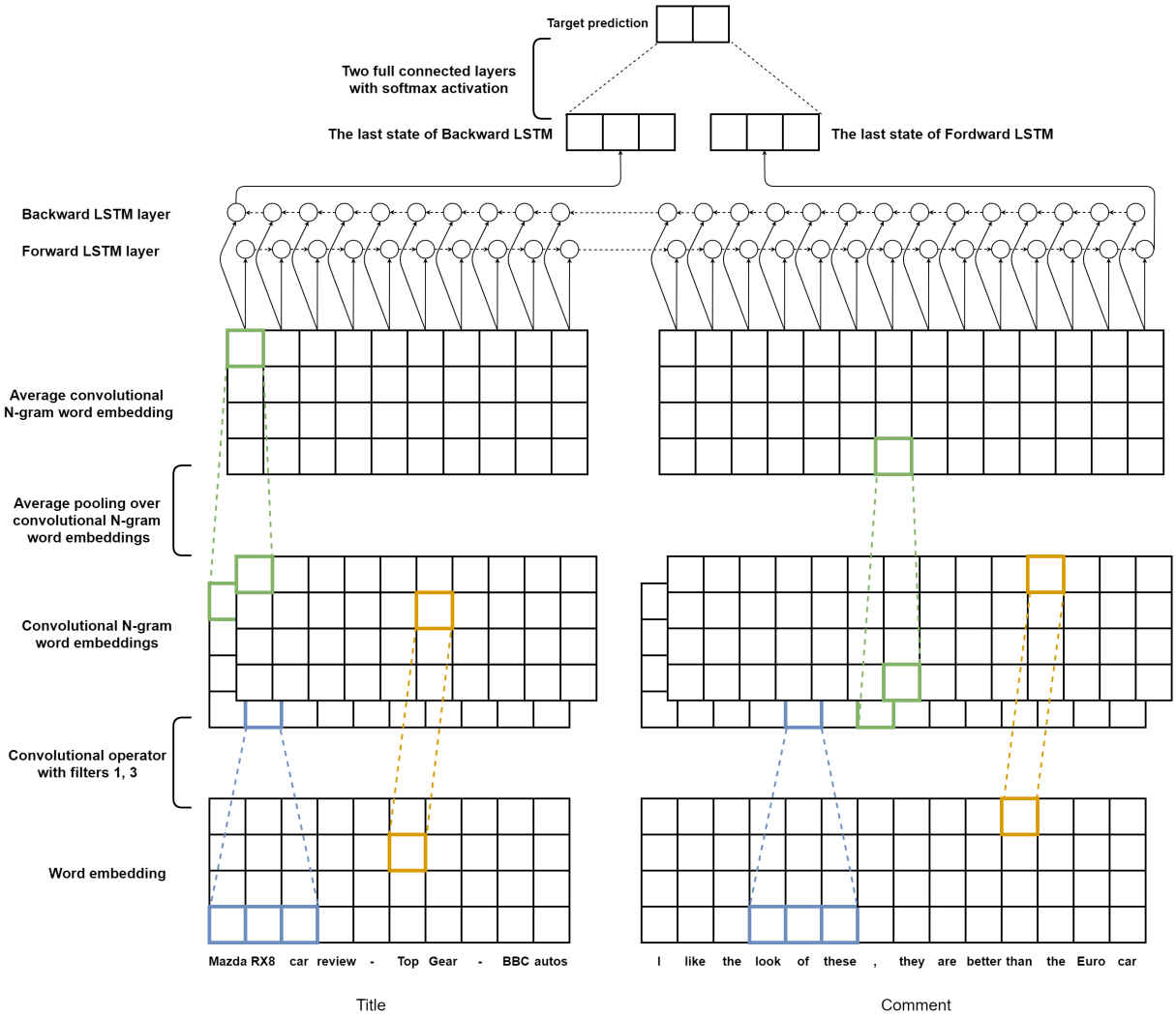


Figure 3.2: Illustration of our Convolutional N-gram BiLSTM word embedding model for classification.

3.4.1 Bidirectional Long Short Term Memory (BiLSTM)

In LSTM explained in Section 2.4.1, sentences of variable length are transformed to fix-length vectors as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.1}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.2}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.3}$$
$$u_t = \tanh(W_u x_t + U_u h_{t-1} + b_u) \tag{3.4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{3.5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{3.6}$$

However, the LSTMs hidden state h_t only takes the information of the left context and knows nothing about the right context. To address this weakness, Dyer et al. [2015] has proposed Bi-directional LSTM (BiLSTM). For capturing the left and right context, BiLSTM applies two separate LSTM units, one for forward direction and one for back-

ward direction. Two hidden states $h_t^{fordward}$ and $h_t^{backward}$ from these LSTM units are concatenated into a final hidden state h_t^{bilstm}:

$$h_t^{bilstm} = h_t^{forward} \oplus h_t^{backward} \tag{3.7}$$

where $\oplus$ is concatenation operator.

3.4.2 Convolutional N-gram BiLSTM word embedding

We present a document of length s as a matrix $d \times s$, where each column is a d-dimension word embedding vector of each word. Given a document matrix $\mathbf{S}$, we performs convolution on this input via linear filters H_k. Each filter H_k is denoted as a weight matrix W_{H_k} of length 1 and region size h_k and a bias value b_{H_k}. W_{H_k} will have $h_k + 1$ parameters to be estimated. Given an input matrix $S \in \mathbb{R}^{d \times s}$ and a filter H_k, the matrix S is converted into $S_k \in \mathbb{R}^{d \times h_k/2 + s + h_k/2}$ by padding zero, which makes the result of convolutional operator be the same dimension as the matrix S. Then a convolutional N-gram word embedding matrix $C_k \in \mathbb{R}^{d \times s}$ is obtained as follows:

$$C_k[i, j] = W_{H_k} \cdot S_k[i, j - h_k/2 : j + h_k/2] + b_{H_k} \tag{3.8}$$

where $\cdot$ is dot product operation and $S_k[i, l : t]$ is the sub-matrix of S_k from column l to t of row i.

To obtain a final convolutional N-gram word embedding matrix, an average pooling is applied over those convolutional N-gram word embedding matrices C_k as follows:

$$C[i, j] = \frac{1}{N} \sum_{k}^{N} C_k[i, j] \tag{3.9}$$

where N is the number of filters.

In the YouTube context, a comment can give sentiment to either a video or the product in that video or even other products. This challenge makes sentiment analysis in the YouTube environment more difficult. For facilitating the model's ability to determine which subject a comment refers to, we use the video title as an additional feature. The reason for this choice is that a video title usually describes the product in that video. Given a comment and the title of the video which the comment belongs to, we apply the above process to achieve two convolutional N-gram word embedding matrices $C^{comment}$ and C^{title} for the title and the comment respectively.

To make our convolutional N-gram word embedding take into account the word's position as well as capture long distance contextual information, we apply BiLSTM to this word embedding:

$$T = BiLSTM(C^{comment} \oplus C^{title}) \tag{3.10}$$

where $\oplus$ is concatenation operator, each column in $T \in \mathbb{R}^{2l \times s}$ is a $2l$-dimension CoNBiLSTM word embedding vector of each word, which is constructed by the equation (7), and l is the LSTM unit's dimension.

For classification tasks with CoNBiLSTM word embedding, the first and last columns of T, which are the word embedding vectors capturing a whole context in forward and backward direction respectively, are fed to a two full-connected-layers neural network:

$$x_0 = T[1,:] \oplus T[s,:] \tag{3.11}$$
$$x_1 = sigmoid(x_0 W_{l_1} + b_{l_1}) \tag{3.12}$$
$$\hat{y} = softmax(x_1 W_{l_2} + b_{l_2}) \tag{3.13}$$

where x_0 is a $4l$ dimension vector constructed by concatenating the first column $T[1,:]$ and the last column $T[s,:]$; $W_{l_1} \in \mathbb{R}^{4l \times d_1}$, $b_{l_1} \in \mathbb{R}^{d_1}$, $W_{l_2} \in \mathbb{R}^{d_1 \times d_2}$, and $b_{l_2} \in \mathbb{R}^{d_2}$ are the parameters of the neural network; and $\hat{y}$ is the prediction output of our model.

In summary, the input includes a comment and the title of the video which that comment belongs to. From this input, our model builds the CoNBiLSTM word embedding and then uses this word embedding to predict a target class. In Figure 3.2, we illustrate our CoNBiLSTM word embedding model, in which we use two filters with size 1, 3. In the training phase, the gradient descent optimization called ADADELTA is used to learn model parameters. Details of ADADELTA method can be found in [Zeiler, 2012]

3.5 SenTube dataset & Task description

In this section, we introduce (i) the description of three classification tasks, (ii) the SenTube dataset as well as the procedure of preparing train/validation/test sets.

3.5.1 Task description

In our experiment, we evaluated the proposed model on three tasks:

- **Sentiment task**: This task detects whether a comment expresses a positive, a negative, or a neutral sentiment. The sentiment can be general or about a specific topic (e.g. product or video).
- **Type task**: One challenge in the YouTube environment is that a comment expresses its sentiment toward not only the product in the video but also the video itself. Therefore, it is important to determine the target subject which the comment gives its sentiment to. Additionally, a comment could be irrelevant for both the product and the video (off-topic) or even contain spam. In this task, we classify a comment into video, product or uninformative (off-topic or spam) type.
- **Full task**: In this task, we want to jointly predict the sentiment and the type of each comment. The problem is cast into a multi-label classification with 7 labels: the Cartesian product between {product, video} type labels and {positive, neutral, negative} sentiment labels, and uninformative class.

3.5.2 SenTube dataset

SenTube [Uryupina et al., 2014] contains about 38,000 English comments and 10,000 Italian comments. The author compiled a list of products in two domains: automobiles (AUTO) and tablets (TABLET) (e.g. Apple iPad, Motorola xoom, Fiat 500), then collected and annotated comments from either commercials or review videos of those products. For several products, there is no corresponding Italian commercial or review

video. Table 3.2 highlights some differences between the two languages as well as the two domains. In YouTube, the number of Italian audiences is much less than the number of English audiences, so the number of Italian comments is quite small compared to English comments. However, the average length of Italian comments is quite longer than English comments.

Table 3.2: Corpus statistics for the SenTube dataset. *In-vocabulary size* denotes the number of terms existing in the pre-trained word embedding.

	English			**Italian**		
	AUTO	TABLET	ALL	AUTO	TABLET	ALL
Video in total	78	139	217	98	100	198
Comment in total	16787	22073	38860	4725	5539	10264
Avg.len. of comment	98	99	99	154	111	131
Avg.len. of title	41	36	38	39	47	43
Vocabulary size	39936	38790	66862	25362	20137	39955
In-vocabulary size	11536(29%)	10491(27%)	16013(24%)	10802(43%)	7729(38%)	14517(36%)

For each task, we prepare the dataset as follows:

- **Sentiment task**: Each comment is labeled as positive, negative or neutral sentiment. Comments with ambiguous sentiments (i.e. contain both positive and negative sentiments) and comments which are irrelevant for both the product and the video (off-topic) or contain spam are excluded.
- **Type task**: Each comment is labeled as video, product or uninformative (off-topic or spam) type.
- **Full task**: Each comment is labeled as product-positive, product-neutral, product-negative, video-positive, video-neutral, video-negative or uninformative. We excluded comments annotated as both video and product types as well as comments with ambiguous sentiments.

For both of the languages, we split the videos into 45% training set, 5% validation set and 50% test set, such that each video contains all its comments. Table 3.3, 3.4 show the data distribution of each class for English and Italian respectively. Since the number of comments in each video is different, the number of comments in the train and validation sets is not the same as the number of comments in the test set. Generally, the class distributions of the train set, validation set, and test set are similar. For example, in English **Sentiment** task for AUTO domain, the neutral class is the most frequent, and the negative class is the least frequent in the train set, validation set, and test set.

3.6 Experiment & discussion

In this section, we i) describe the model setting for English and Italian, and report the experimental results of ii) the in-domain experiment, iii) the cross-domain experiment, iv) the per class experiment, and v) give a quality analysis for our model.

Table 3.3: Summary of English YouTube comments data used in **Sentiment**, **Type** and **Full** tasks

Task	Class	AUTO			TABLET		
		Train	Validation	Test	Train	Validation	Test
Sentiment	Positive	1536(30%)	501(30%)	1601(30%)	2393(29%)	270(16%)	1666(27%)
	Negative	917(18%)	330(20%)	871(16%)	1441(17%)	505(29%)	1267(21%)
	Neutral	2591(52%)	843(50%)	2895(54%)	4476(54%)	963(55%)	3171(52%)
	Total	5044	1674	5367	8310	1738	6104
Type	Product	2755(43%)	952(44%)	2536(36%)	5938(57%)	1331(55%)	4411(59%)
	Video	2078(33%)	664(31%)	2525(36%)	1961(19%)	361(15%)	1435(19%)
	Uninfo	1565(25%)	538(25%)	1917(28%)	2608(25%)	723(30%)	1630(22%)
	Total	6398	2154	6978	10507	2415	7476
Full	Product-pos	1107(11%)	284(16%)	349(10%)	1096(12%)	472(13%)	712(11%)
	Product-neg	931(9%)	212(12%)	217(6%)	1152(12%)	452(12%)	869(14%)
	Product-neu	1921(20%)	376(20%)	447(13%)	2837(30%)	1073(29%)	2400(37%)
	Video-pos	924(9%)	175(10%)	444(13%)	698(7%)	247(7%)	412(6%)
	Video-neg	459(5%)	65(4%)	104(3%)	314(3%)	101(3%)	150(9%)
	Video-neu	1892(19%)	199(11%)	939(28%)	844(8%)	300(8%)	599(9%)
	Uninfo	2618(27%)	513(28%)	897(26%)	2583(27%)	1071(28%)	1314(20%)
	Total	9852	1824	3397	9524	3716	6456

3.6.1 Model configuration

English model

To tune the hyper-parameters of our models, we do a grid search on 30% of the dataset for **Full** task:

- *Word embedding layer*: we obtained the pre-trained word vectors *Word2Vec*[1]. It was trained on 100 billion words from English Google News, and its vectors have the dimension of 300. During the training process, these pre-trained word vectors are optimized.
- *Convolutional filters*: two filters ($h = 1, 3$) are employed to construct a convolutional N-gram word embedding.
- BiLSTM layer: the dimension for each direction the same as the word embedding layer ($l = 300$).
- First full-connected layer: the dimension is the same as BiLSTM layer ($d_1 = 600$)
- Dropout layer: For avoiding overfitting, we employ a dropout layer (p=0.5) between the first and second full-connected layers.
- Second full-connected layer: the dimension is the number of target labels. We used a softmax activation for this layer.

[1]https://code.google.com/p/word2vec/

Table 3.4: Summary of Italian comments data used in **Sentiment**, **Type** and **Full** tasks

Task	Class	AUTO			TABLET		
		Train	Validation	Test	Train	Validation	Test
Sentiment	Positive	665(35%)	115(32%)	265(24%)	521(22%)	50(22%)	364(23%)
	Negative	462(24%)	91(25%)	262(24%)	481(20%)	32(15%)	357(22%)
	Neutral	787(41%)	159(44%)	590(53%)	1379(58%)	139(63%)	881(55%)
	Total	1914	365	1117	2381	221	1602
Type	Product	1005(45%)	73(20%)	736(44%)	2045(63%)	287(56%)	836(62%)
	Video	647(29%)	181(50%)	525(31%)	498(15%)	120(24%)	249(18%)
	Uninfo	601(27%)	107(30%)	426(25%)	685(21%)	104(20%)	266(20%)
	Total	2253	361	1687	3228	511	1351
Full	Product-pos	253(11%)	50(7%)	176(14%)	218(10%)	172(16%)	154(10%)
	Product-neg	216(10%)	132(18%)	190(15%)	355(16%)	145(13%)	211(13%)
	Product-neu	283(13%)	148(20%)	272(21%)	719(33%)	451(41%)	551(35%)
	Video-pos	271(13%)	50(7%)	146(11%)	92(4%)	22(2%)	112(7%)
	Video-neg	127(6%)	51(7%))	36(3%)	41(2%)	11(1%)	62(4%)
	Video-neu	351(16%)	113(15%)	171(13%)	246(11%)	64(6%)	195(12%)
	Uninformative	661(31%)	206(15%)	294(13%)	527(24%)	243(22%)	285(18%)
	Total	2162	750	1285	2198	1108	1570

Italian model

To tune the hyper-parameters of our models, we do a grid search on 30% of the dataset for **Full** task:

- *Word embedding layer*: we obtained the pre-trained word vectors from Bojanowski et al. [2017b]. It was trained on Italian Wikipedia data, and its vectors have the dimension of 300. During the training process, these pre-trained word vectors are optimized.
- *Convolutional filters*: two filters ($h = 1, 17$) are employed to construct a convolutional N-gram word embedding.
- The other layers are the same as the English model.

3.6.2 In-domain experiment

To evaluate CoNBiLSTM word embedding model, we compared with BiLSTM and the previous work [Severyn et al., 2016] - STRUCT method on three tasks mentioned in Section 3.5.2. Table 3.5 shows the accuracy performance of three models on the SenTube dataset for the two languages and domains.

In most of the cases, the proposed method outperforms the others. In the previous work - STRUCT method, we observed that the performance on AUTO is much lower than on TABLET across all tasks. The author explained that i) TABLET contains more training data and (ii) the different types of audiences in AUTO and TABLET domains: well-informed users and geeks expressing better-motivated opinions about a product for the former vs. more general audiences for the latter. This makes the comments in AUTO more challenging to analyze. In our model, we achieved a smaller gap between AUTO and

Table 3.5: The result of the in-domain experiment

Language	Task	AUTO			TABLET		
		STRUCT	BiLSTM	CoNBiLSTM	STRUCT	BiLSTM	CoNBiLSTM
English	Sentiment	55.7	66.35	**66.89**^{a,b}	**70.5**	68.89	70.17^{b}
	Type	59.4	66.78	**68.36**^{a,b}	78.6	78.57	**79.49**^{a,b}
	Full	41.5	51.84	**53.81**^{a,b}	60.3	60.08	**61.28**^{a,b}
Italian	Sentiment	**61.6**	59	61.41^{b}	64.4	64.47	**65.6**^{a,b}
	Type	70.7	74.1	**74.75**^{a,b}	77.3	78.46	**79.64**^{a,b}
	Full	45.6	47.47	**51.05**^{a,b}	52.4	53.24	**55.03**^{a,b}

^{a,b} denote results statistically significant at $p < 0.05$ via the pairwise t-test compared with STRUCT and BiLSTM respectively.

TABLET domains (e.g. 7.47% of the difference in performance between the two domains for our model vs. 18.8% for the STRUCT in English **Full** task). It shows the robust of our model to domains with different types of audiences. More specifically, the performance on AUTO domain is much improved across all tasks except **Sentiment** task for Italian AUTO and English TABLET.

In the case of **Sentiment** task, although STRUCT outperformed our model in English TABLET and Italian AUTO domains, the difference is not statistically significant. STRUCT used a pre-defined list of sentiment words to determine sentiment words in a comment. Intuitively, this facilitates the process of sentiment analysis, especially when training data is small. In our approach, we aim to a multilingual task where labeled resources (e.g. sentiment dictionary, synonym dictionary) and linguistic preprocessing tools (e.g. POS-tagger, syntactic parser) are limited; hence we do not use any additional labeled data as sentiment dictionaries.

Table 3.6: The results of the cross-domain experiment

Language	Source	Target	Task	STRUCT	BiLSTM	CoNBiLSTM
English	AUTO	TABLET	Sentiment	66.6	68.23	**69.01**^{a,b}
			Type	64.1	66.51	**67.81**^{a,b}
			Full	38.3	46.2	**48.03**^{a,b}
	TABLET	AUTO	Sentiment	61.9	63.54	**64.94**^{a,b}
			Type	55.6	63.15	**64.57**^{a,b}
			Full	44.7	47.2	**47.6**^{a,b}
Italian	AUTO	TABLET	Sentiment	61.2	60.54	**62.92**^{a,b}
			Type	**63.8**	61.1	62.24^{a,b}
			Full	29.7	33.82	**37.34**^{a,b}
	TABLET	AUTO	Sentiment	54.3	56.27	**57.07**^{a,b}
			Type	56.4	59.8	**60.8**^{a,b}
			Full	31.7	38.24	**39.5**^{a,b}

^{a,b} denote results statistically significant at $p < 0.05$ via the pairwise t-test compared with STRUCT and BiLSTM respectively.

3.6.3 Cross-domain experiment

In this experiment, we trained a model on the data from one domain and tested on the data from the other domain. This experiment examines the adaptability of our models as

well as whether we need training data for a new domain. Table 3.6 reports the accuracy of the three tasks in the cross-domain setting.

According to the experimental results, our model shows a more robust and stable performance in the cross-domain setting. For example, in English **Full** task, the difference between the performances of our model trained on AUTO (48.03%) and on TABLET (47.6%) is 0.43%, while the difference of STRUCT is 6,4%. This proved that our model has more robustness and stronger generalization.

Generally, the proposed model provides an improvement in accuracy in both of the languages compared with BiLSTM and STRUCT, except the **Type** task with Italian AUTO as a source domain. In the Italian AUTO domain, the average length of comments ($l = 154$) is quite long compared with TABLET domain ($l = 111$) and the in-vocabulary size (in Table 3.2) of target domain TABLET (38%) is small compared with AUTO domain (43%). These differences give a challenge to our model.

Table 3.7: The precision, recall and F1 scores of CoNBiLSTM for each class in the English experiments

Task	Class	AUTO				TABLET			
		Precision	Recall	F1	Support	Precision	Recall	F1	Support
Sentiment	Positive	66.78	62.4	64.51	1601	78.95	60.56	68.55	1666
	Negative	41.46	17.57	24.68	871	55.29	33.39	41.63	1267
	Neutral	69.62	84.21	76.22	2895	70.2	89.91	78.84	3171
	Acc		66.89				70.17		
Type	Product	71.43	78.59	74.84	2536	84.55	91.20	87.75	4411
	Video	69.50	65.43	67.40	2525	79.15	63.48	70.46	1435
	Uninfo	62.12	58.69	60.35	1917	64.39	61.90	63.12	1630
	Acc		68.36				79.49		
Full	Product-pos	62.95	55.01	58.72	349	54.39	39.19	45.55	712
	Product-neg	35.71	34.56	35.13	217	45.67	37.63	41.26	869
	Product-neu	43.03	54.59	48.13	447	67.1	79.71	72.86	2400
	Video-pos	68.2	46.85	55.54	444	78.45	68.93	73.39	412
	Video-neg	21.05	7.69	11.27	104	27.68	20.67	23.66	150
	Video-neu	68.71	44.2	53.79	939	52.05	46.58	49.16	599
	Uninfo	50.15	76.48	60.57	897	61.71	64.16	62.91	1314
	Acc		53.81				61.28		

3.6.4 Performance on each class

In this section, we analyze the per-class performance of the three tasks. Table 3.7 reports Precision, Recall and F1 scores of each class. In **Sentiment** task, we observed that the negative class contributes the largest error in both of the domains. In fact, negative comments in the dataset take the smallest proportion and probably contains complicated grammars, so it is more difficult to learn an efficient classifier for the negative class compared with the positive class and the uninformative class. We discuss the difficulty of detecting negative sentiment in Section 3.6.5. In **Type** task, the uninformative class is considerably more difficult than the other classes. An uninformative comment could be a spam or an off-topic comment or even a comment about unrelated products, so it is

intuitively quite hard to classify it. In **Full** task, the Product-negative and Video-negative classes have the worst performance. This confirms with the result from **Sentiment** task.

3.6.5 Quality analysis

To obtain a better sense of the improvement and limitation of the proposed model, we manually inspect some typical cases, which are shown in table 3.8. These samples are a great way to show the proposed model's advantages and disadvantages.

Table 3.8: Some typical samples for quality analysis. The labels are 0:product-positive, 1:product-neutral, 2:product-negative, 3:video-positive, 4:video-neutral, 5:video-negative and 6:uninformative

No.	Title	Comment	BiLSTM	CoNBiLSTM	True
1	ferrari 430 review...	ferrari look so dull (spelling) and boring! lamborghini is so much more awsome! they look so mean! and just evil!	0	2	2
2	bugatti veyron vitesse video review	is it just me? or does the dash look pretty dull...	0	2	2
3	ferrari 430 review...	this is my dream car, and its getting cheaper and cheaper o yah!	2	0	0
4	2012 range rover evoque coupe hd video review	so it does have a rear wiper as in the new lexus rx eh?...	1	0	2
5	ferrari 430 review...	why did they change the music... the original was way more dramatic	4	4	5
6	2012 fiat 500 test drive	review but toyotas and hondas are still the most reliable cars on the planet. my 1997 honda civic still has the original engine even though it has 400,000 miles on it.	0	0	2

The first source of the improvement is to capture better long distance dependencies compared with BiLSTM. In sample #1, the comment gives two polarity sentiments: one negative sentiment toward the product (*Ferrari*) mentioned in the title and one positive sentiment toward the other (*Lamborghini*). Analyzing the main sentiment of this comment requires capturing a long distance relationship between the title and its comment. The second advantage of our model is to build a better word embedding, which captures contextual information and the part of speech. For example, word "*pretty*" in sample #2 is an adverb and almost contributes nothing to the meaning of this comment. However, BiLSTM considered this word as an adjective word bearing a positive sentiment. Consequently, BiLSTM made the wrong prediction for this sample. Another example is the word "*cheaper*". This word has two opposite sentiments depending on context. In sample #3, our model correctly assigned the positive sentiment for "*cheaper*" while BiLSTM did not.

As we mentioned in Section 3.6.4, our model has quite low performance in the negative class. To analyze the difficulty of negative comments, we also inspect some typical negative comments, which are reported in table 3.8. Comment #4 actually is a rhetorical question and carries a negation meaning. In sample #5 and #6, these comments do not directly

mention the products in the titles. However, these comments implicitly give negative sentiments toward those products by giving good reviews for other products. These facts make the negative class more challenging than the others in sentiment analysis.

3.7 Conclusion

In this work, we introduced a convolutional N-gram BiLSTM word embedding model. Our approach enhances the conventional word embedding by using i) multiple convolutional filters with variant sizes for capturing contextual information; ii) BiLSTM for encoding long distance contextual dependencies. Through the multilingual and cross-domain experiments on the SenTube dataset, our model shows the more robust and better performance compared with the previous work STRUCT - the state-of-the-art method on the SenTube dataset. While the previous work requires a pre-defined sentiment dictionary and some linguistic preprocessing tools, our model only requires a pre-trained word embedding which is trained in unsupervised learning scheme. Therefore, our model has larger applicability to multilingual environments as YouTube.

For future work, we plan to improve the ability to interpret implication, where main subjects are not mentioned explicitly as we analyzed in Section 3.6.5. In addition, we also plan to build a model for extracting helpful comments, which give polarity sentiments and explanation for those sentiments.

Chapter 4

Semantic Textual Similarity

4.1 Introduction

Measuring the semantic similarity/relation of two pieces of short text plays a fundamental role in a variety of language processing tasks (i.e., summarization, plagiarism detection, question answering, and machine translation). Semantic textual similarity (STS) task is challenging because of the diversity of linguistic expression. For example, two sentences with different lexicons could have a similar meaning. Moreover, the task requires to measure similarity at several levels (e.g., word level, phrase level, sentence level). These challenges give difficulties to conventional approaches using hand-crafted features.

Recently, the emergence of word embedding techniques, which encode the semantic properties of a word into a low dimension vector, leads to the successes of many learning models in natural language processing (NLP). For example, Kalchbrenner et al. [2014a] randomly initialize word vectors, then tunes them during the training phase of a sentence classification task. By contrast, Huy Tien and Minh Le [2017] initialize word vectors via the pre-train word2vec model trained on Google News [Mikolov et al., 2013b]. Wieting et al. [2015] train a word embedding model on the paraphrase dataset PPDB, then apply the word representation for word and bi-gram similarity tasks.

Several pre-trained word embeddings are available, which are trained on various corpora under different models. Levy and Goldberg [2014] observed that different word embedding models capture different aspects of linguistic properties: a Bag-of-Words contexts based model tends to reflect the domain aspect (e.g., scientist and research) while a paraphrase-relationship based model captures semantic similarities of words (e.g., boy and kid). From experiments, we also observed that the performance of a word embedding model is usually inconsistent over different datasets. This inspired us to develop a model taking advantages of various pre-trained word embeddings for measuring textual similarity/relation.

4.2 Research Objective and Contribution

Our research objective is to learn a novel word embedding capturing various linguistic properties from multiple sets of pretrained word embeddings and then measure similarity/relation between two sentences via this embedding. In this paper, we propose a convolutional neural network (CNN) to learn a multi-aspect word embedding from various

pre-trained word embeddings. We then apply the max-pooling scheme and Long Short Term Memory (LSTM) on this embedding to form a sentence representation. In STS tasks, Shao [2017] shows the efficiency of the max-pooling scheme in modeling sentences from word embedding representations refined via CNN. However, the max-pooling scheme lacks the property of word order (e.g., *sentence*("Bob likes Marry") = *sentence*("Marry likes Bob")). To address this weakness, we use LSTM as an additional scheme for modeling sentences with word order characteristics. For measuring the similarity/relation between two sentence representations, we propose Multi-level comparison which consists of word-word level, sentence-sentence level, and word-sentence level. Through these levels, our model comprehensively evaluates the similarity/relation between two sentences.

We evaluate our M-MaxLSTM-CNN model on three tasks: STS, textual entailment recognition, paraphrase identification. The advantages of M-MaxLSTM-CNN are: i) simple but efficient for combining various pre-trained word embeddings with different dimensions; ii) using Multi-level comparison shows better performances compared to using only sentence-sentence comparison; iii) does not require hand-crafted features (e.g., alignment features, Ngram overlaps, syntactic features, dependency features) compared to the state-of-the-art ECNU Tian et al. [2017] on STS Benchmark dataset.

Our main contributions are as follows:

- Propose the MaxLSTM-CNN encoder for efficiently encoding sentence embeddings from multiple word embeddings.
- Propose the Multi-level comparison (M-MaxLSTM-CNN) to learn the similarity/relation between two sentences. The model achieves strong performances over various tasks.

The remainder of this paper is organized as follows: Section 4.3 reviews the previous research, Section 4.4 introduces the architecture of our model, Section 4.5 describes the three tasks and datasets, Section 4.6 describes the experimental setting, Section 4.7 reports and discusses the results of the experiments, and Section 4.8 concludes our work.

4.3 Related work

Most prior research on modeling textual similarity relied on feature engineering. Wan et al. [2006] extract n-gram overlap features and dependency-based features, while Madnani et al. [2012] employ features based on machine translation metrics. Mihalcea et al. [2006] propose a method using corpus-based and knowledge-based measures of similarity. Das and Smith [2009] design a model which incorporates both syntax and lexical semantics using dependency grammars. Ji and Eisenstein [2013] combine the fine-grained n-gram overlap features with the latent representation from matrix factorization. Xu et al. [2014] develop a latent variable model which jointly learns paraphrase relations between word and sentence pairs. Using Dependency trees, Sultan et al. [2014] propose a robust monolingual aligner and successfully applied it for STS tasks. Ferreira et al. [2016] present a novel sentence representation at three layers: lexical, syntactical and semantic. Through the proposed statistical-semantic similarity measurement, the approach achieves strong performances in semantic textual similarity, redundancy elimination in multi-document summarization. According to AL-Smadi et al. [2017], these features (i.e., lexical, syntactic, and semantic) also achieve competitive performance in Arabic news tweets. Ferreira et al. [2018] also employ the three-layer representation with different machine learning

methods for identifying paraphrase. Although not achieving the state-of-the-art results in general terms, the model handles the problems of meaning and word order. Jiang et al. [2017] and Qu et al. [2018] propose some semantic computation approaches using features based on the structure of Wikipedia. These models are efficient in determining semantic similarity between concepts and have a better human correlation than previous methods such as Word2Vec and NASARI [Camacho-Collados et al., 2016].

The recent emergence of deep learning models has provided an efficient way to learn continuous vectors representing words/sentences. By using a neural network in the context of a word prediction task, Bengio et al. [2003] and [Mikolov et al., 2013a] generate word embedding vectors carrying semantic meanings. The embedding vectors of words which share similar meanings are close to each other. To capture the morphology of words, Bojanowski et al. [2017a] enrich the word embedding with character n-grams information. Closest to this approach, Wieting et al. [2016b] also propose to represent a word or sentence using a character n-gram count vector. However, the objective function for learning these embeddings is based on paraphrase pairs.

For modeling sentences, composition approach attracted many studies. Yessenalina and Cardie [2011] model each word as a matrix and used iterated matrix multiplication to present a phrase. Tai et al. [2015a] design a Dependency Tree-Structured LSTM for modeling sentences. This model outperforms the linear chain LSTM in STS tasks. Convolutional neural network (CNN) has recently been applied efficiently for semantic composition [Kalchbrenner et al., 2014a, Kim, 2014, Shao, 2017]. This technique uses convolutional filters to capture local dependencies in term of context windows and applies a pooling layer to extract global features. He et al. [2015] use CNN to extract features at multiple levels of granularity. The authors then compare their sentence representations via multiple similarity metrics at several granularities. Gan et al. [2017] propose a hierarchical CNN-LSTM architecture for modeling sentences. In this approach, CNN is used as an encoder to encode a sentence into a continuous representation, and LSTM is used as a decoder. Conneau et al. [2017b] train a sentence encoder on a textual entailment recognition database using a BiLSTM-Maxpooling network. This encoder achieves competitive results on a wide range of transfer tasks. To enhance the conventional word embedding representation for capturing contextual information, Nguyen and Le [2018] proposed an N-gram word embedding via convolutional filters. This approach achieves robust performance in multi-lingual sentiment analysis. As trained under one pre-trained embedding, these above approaches depend on the objective function of that embedding.

At SemEval-2017 STS task, hybrid approaches obtain strong performances. Wu et al. [2017] train a linear regression model with WordNet, alignment features and the word embedding *word2vec*[1]. Tian et al. [2017] develop an ensemble model with multiple boosting techniques (i.e., Random Forest, Gradient Boosting, and XGBoost). This model incorporates traditional features (i.e., n-gram overlaps, syntactic features, alignment features, bag-of-words) and sentence modeling methods (i.e., Averaging Word Vectors, Projecting Averaging Word Vectors, LSTM).

MVCNN model Yin and Schütze [2015] and MGNC-CNN model Zhang et al. [2016b] are close to our approach. In MVCNN, the authors use variable-size convolution filters on various pre-trained word embeddings for extracting features. However, MVCNN requires word embeddings to have the same size. In MGNC-CNN, the authors apply independently CNN on each pre-trained word embedding for extracting features and then concatenate

[1]https://code.google.com/p/word2vec/

these features for sentence classification. By contrast, our M-MaxLSTM-CNN model jointly applies CNN on all pre-trained word embeddings to learn a multi-aspect word embedding. From this word representation, we encode sentences via the max-pooling and LSTM. To learn the similarity/relation between two sentences, we employ Multi-level comparison.

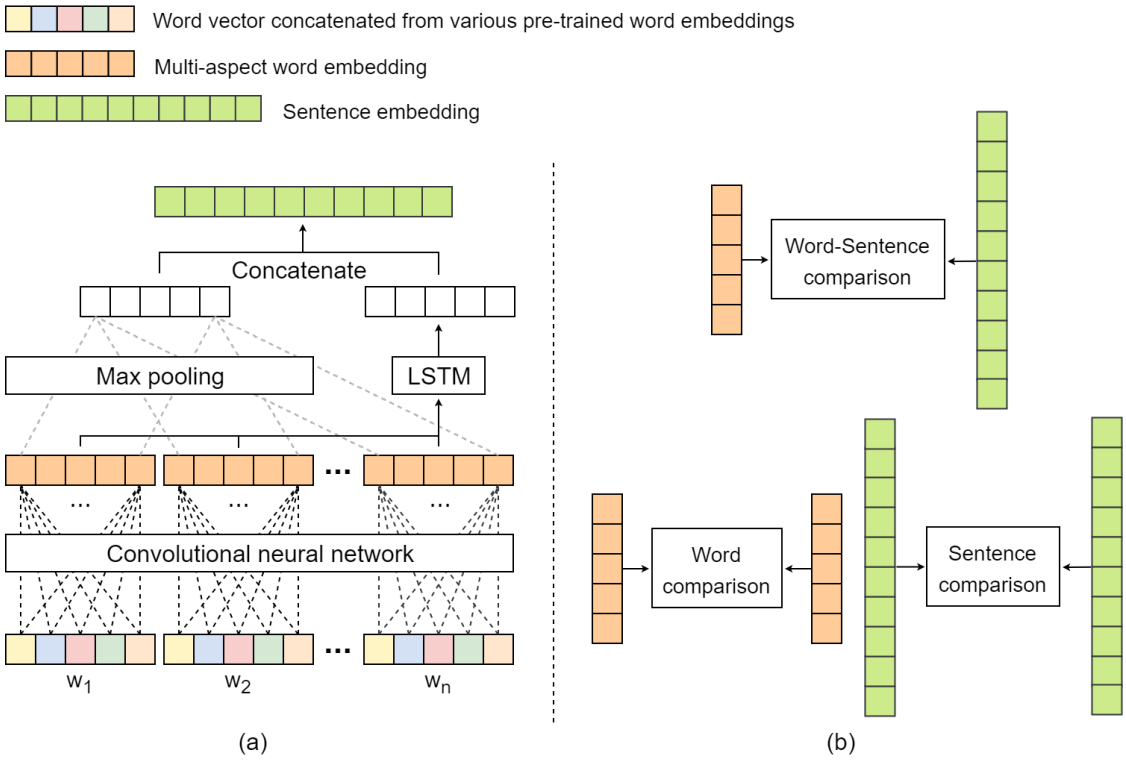


Figure 4.1: The proposed M-MaxLSTM-CNN model: (a) MaxLSTM-CNN encoder; (b) Multi-level comparison.

4.4 Model description

Our model (shown in Figure 4.1) consists of three main components: i) learning a multi-aspect word embedding (Section 4.3.1); ii) modeling sentences from this embedding (Section 4.3.2); iii) measuring the similarity/relation between two sentences via Multi-level comparison (section 4.3.3).

4.4.1 Multi-aspect word embedding

Given a word w, we transfer it into a word vector e_w^{concat} via K pre-trained word embeddings as follows:

$$e_w^{concat} = e_w^1 \oplus e_w^2 \oplus ... \oplus e_w^K \tag{4.1}$$

where $\oplus$ is concatenation operator, e_w^i is the word embedding vector of w in the ith pre-trained embedding.

To learn a multi-aspect word embedding e_w^{multi} from the representation e_w^{concat}, we design H convolutional filters. Each filter r_i is denoted as a weight vector with the same dimension as e_w^{concat} and a bias value b_{r_i}. The e_w^{multi} is obtained by applying these filters on the e_w^{concat} as follows:

$$e_w^{r_i} = \sigma(e_w^{concat} r_i^T + b_{r_1}) \tag{4.2}$$

$$e_w^{multi} = [e_w^{r_1}, e_w^{r_2}, ..., e_w^{r_H}] \tag{4.3}$$

where σ denotes a logistic sigmoid function.

The next section explains how to model a sentence from its multiple-aspect word embeddings.

4.4.2 Sentence modeling

Given an input sentence $s = [w_1, w_2, ..., w_n]$, we obtain a sequence of multiple-aspect word embeddings $s^{multi} = [e_{w1}^{multi}, e_{w2}^{multi}, ..., e_{w_n}^{multi}]$ using Eq. (1-3). For modeling the sentence from the representation s^{multi}, we use two schemes: max-pooling and LSTM.

Max-pooling scheme: To construct a max-pooling sentence embedding e_s^{max}, the most potential features are extracted from the representation s^{multi} as follows:

$$e_s^{max}[i] = max(e_{w_1}^{multi}[i], e_{w_2}^{multi}[i], ..., e_{w_n}^{multi}[i]) \tag{4.4}$$

where $e_{w_k}^{multi}[i]$ is the ith element of $e_{w_k}^{multi}$.

LSTM scheme: From Eq. (4.4), we find that the max-pooling scheme ignores the property of word order. Therefore, we construct a LSTM sentence embedding e_s^{lstm} to support the sentence embedding e_s^{max}. The representation s^{multi} is transformed to a fix-length vector by recursively applying a LSTM unit to each input $e_{w_t}^{multi}$ and the previous step h_{t-1}. At each time step t, the LSTM unit with l-memory dimension defines six vectors in $\mathbb{R}^l$: input gate i_t, forget gate f_t, output gate o_t, tanh layer u_t, memory cell c_t and hidden state h_t as follows (from Tai et al. [2015a]):

$$i_t = \sigma(W_i e_{w_t}^{multi} + U_i h_{t-1} + b_i) \tag{4.5}$$

$$f_t = \sigma(W_f e_{w_t}^{multi} + U_f h_{t-1} + b_f) \tag{4.6}$$

$$o_t = \sigma(W_o e_{w_t}^{multi} + U_o h_{t-1} + b_o) \tag{4.7}$$

$$u_t = \tanh(W_u e_{w_t}^{multi} + U_u h_{t-1} + b_u) \tag{4.8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{4.9}$$

$$h_t = o_t \odot \tanh(c_t) \tag{4.10}$$

$$e_s^{lstm} = h_n \tag{4.11}$$

where $\sigma, \odot$ respectively denote a logistic sigmoid function and element-wise multiplication; W_i, U_i, b_i are respectively two weights matrices and a bias vector for input gate i. The denotation is similar to forget gate f, output gate o, tanh layer u, memory cell c and hidden state h.

Finally, the sentence embedding e_s is obtained by concatenating the two sentence embeddings e_s^{max} and e_s^{lstm}:

$$e_s = e_s^{max} \oplus e_s^{lstm} \tag{4.12}$$

4.4.3 Multi-level comparison

In this section, we describe the process for evaluating the similarity/relation between two sentences. We compare two sentences via three levels: word-word, sentence-sentence, and word-sentence.

Word-word comparison

Given two input sentences s_1 and s_2, we encode them into two sequences of multi-aspect word embeddings s_1^{multi} and s_2^{multi} (Section 4.3.2). We then compute a word-word similarity vector sim^{word} as follows:

$$A_{ij} = \frac{s_1^{multi}[i] \cdot s_2^{multi}[j]}{\left\|s_1^{multi}[i]\right\| \left\|s_2^{multi}[j]\right\|} \tag{4.13}$$

$$sim^{word} = \sigma(W^{word} g(A) + b^{word}) \tag{4.14}$$

where $s_t^{multi}[i]$ is the ith multi-aspect word embedding of sentence s_t; $g()$ is a function to flatten a matrix into a vector; and W^{word} and b^{word} are a weight matrix and a bias parameter, respectively.

Sentence-sentence comparison

Given two input sentences s_1 and s_2, we encode them into two sentence embeddings e_{s_1} and e_{s_2} (Section 4.3.1 and 4.3.2). To compute the similarity/relation between the two embeddings, we introduce four comparison metrics:

Cosine similarity:

$$d_{cosine} = \frac{e_{s_1} \cdot e_{s_1}}{\|e_{s_1}\| \|e_{s_2}\|} \tag{4.15}$$

Multiplication vector & Absolute difference:

$$d_{mul} = e_{s_1} \odot e_{s_2} \tag{4.16}$$

$$d_{abs} = |e_{s_1} - e_{s_2}| \tag{4.17}$$

where $\odot$ is element-wise multiplication.

Neural difference:

$$x = e_{s_1} \oplus e_{s_2} \tag{4.18}$$

$$d_{neu} = W^{neu} x + b^{neu} \tag{4.19}$$

where W^{neu} and b^{neu} are respectively a weight matrix and a bias parameter.

As a result, we have a sentence-sentence similarity vector sim^{sent} as follows:

$$d^{sent} = d_{cosine} \oplus d_{mul} \oplus d_{abs} \oplus d_{neu} \tag{4.20}$$

$$sim^{sent} = \sigma(W^{sent} d^{sent} + b^{sent}) \tag{4.21}$$

where W^{sent} and b^{sent} are respectively a weight matrix and a bias parameter.

Word-sentence comparison

Given a sentence embedding e_{s_1} and a sequence of multi-aspect word embeddings s_2^{multi}, we compute a word-sentence similarity matrix $sim_{s_1}^{ws}$ as follows:

$$e_{s_1}^{ws}[i] = e_{s_1} \oplus s_2^{multi}[i] \tag{4.22}$$

$$sim_{s_1}^{ws}[i] = \sigma(W^{ws} e_{s_1}^{ws}[i] + b^{ws}) \tag{4.23}$$

where $s_2^{multi}[i]$ is the multi-aspect word embedding of the ith word in sentence s_2; W^{ws} and b^{ws} are respectively a weight matrix and a bias parameter.

As a result, we have a word-sentence similarity vector sim^{ws} for the two sentences as follows:

$$sim^{ws} = \sigma(W^{ws'}[g(sim_{s_1}^{ws}) \oplus g(sim_{s_2}^{ws})] + b^{ws'}) \tag{4.24}$$

where $g()$ is a function to flatten a matrix into a vector; $W^{ws'}$ and $b^{ws'}$ are respectively a weight matrix and a bias parameter.

Finally, we compute a target score/label of a sentence pair as follows:

$$sim = sim^{word} \oplus sim^{sent} \oplus sim^{ws} \tag{4.25}$$

$$h_s = \sigma(W^{l1} sim + b^{l1}) \tag{4.26}$$

$$\hat{y} = softmax(W^{l2} h_s + b^{l2}) \tag{4.27}$$

where W^{l1}, W^{l2}, b^{l1} and b^{l2} are model parameters; $\hat{y}$ is a predicted target score/label.

4.5 Tasks & Datasets

We evaluate our model on three tasks:

- **Textual entailment recognition**: given a pair of sentences, we predict a directional relation between the sentences (entailment/contradiction/neutral). We evaluate this task on **SICK** dataset. It was collected for the 2014 SemEval competition and includes examples of the lexical, syntactic and semantic phenomena and ignores other aspects of existing sentential datasets (i.e., idiomatic multiword expressions, named entities, telegraphic language).

- **Semantic textual similarity**: given a pair of sentences, we measure a semantic similarity score of this pair. We use two datasets for this task:

  - **STSB**: comprises a careful selection of the English data sets used in SemEval and *SEM STS shared tasks from 2012 to 2017. This dataset cover three genres: image captions, news headlines, and user forums. Each sentence pair is annotated with a relatedness score $\in [0, 5]$.
  - **SICK**: Each sentence pair is annotated with a relatedness score $\in [1, 5]$.

- **Paraphrase identification**: given a pair of sentences, we predict a binary label indicating whether the two sentences are paraphrases. Microsoft Research Paraphrase Corpus (MRPC) is used for this task. It includes pairs of sentences extracted from news source on the web.

Table 4.1 shows the statistic of the three datasets. Because of not dealing with name entities and multi-word idioms, the vocabulary size of SICK is quite small compared to the others.

Table 4.1: Statistic of datasets. $|V|$, l denote the vocabulary size, and the average length of sentences respectively.

Dataset	**Train**	**Validation**	**Test**	l	$\|V\|$
STSB	5,749	1,500	1,379	11	15,997
SICK	4,500	500	4,927	9	2,312
MRPC	3,576	500	1,725	21	18,003

4.6 Experimental setting

4.6.1 Pre-trained word embeddings

We study five pre-trained word embeddings[2] for our model:

- **word2vec** is trained on Google News dataset (100 billion tokens). The model contains 300-dimensional vectors for 3 million words and phrases.
- **fastText** is learned via skip-gram with subword information on Wikipedia text. The embedding representations in fastText are 300-dimensional vectors.
- **GloVe** is a 300-dimensional word embedding model learned on aggregated global word-word co-occurrence statistics from Common Crawl (840 billion tokens).
- **Baroni** uses a context-predict approach to learn a 400-dimensional semantic embedding model. It is trained on 2.8 billion tokens constructed from ukWaC, the English Wikipedia and the British National Corpus.
- **SL999** is trained under the skip-gram objective with negative sampling on word pairs from the paraphrase database **PPDB**. This 300-dimensional embedding model is tuned on SimLex-999 dataset [Hill et al., 2016].

4.6.2 Model configuration

In all of the tasks, we used the same model configuration as follows:

- Convolutional filters: we used 1600 filters. It is also the dimension of the word embedding concatenated from the five pre-trained word embeddings.
- LSTM dimension: we also selected 1600 for LSTM dimension.
- Neural similarity layers: the dimension of b^{word}, b^{sent}, b^{ws} and $b^{ws'}$ are respectively 50, 5, 5 and 100.
- Penultimate fully-connected layer: has the dimension of 250 and is followed by a drop-out layer ($p = 0.5$).

We conducted a grid search on 30% of STSB dataset to select these optimal hyper-parameters.

[2]These embeddings are available at *anonymous*

4.6.3 Training Setting

Textual entailment recognition & Paraphrase identification

In these tasks, we use the cross-entropy objective function and employ AdaDelta as the stochastic gradient descent (SGD) update rule with mini-batch size as 30. Details of Adadelta method can be found in Zeiler [2012]. During the training phase, the pre-trained word embeddings are fixed.

Semantic Textual Similarity

To compute a similarity score of a sentence pair in the range $[1, K]$, where K is an integer, we replace Eq. (27) with the equations in Tai et al. [2015a] as follows:

$$\hat{p}_\theta = softmax(W^{l2}h_s + b^{l2}) \tag{4.28}$$

$$\hat{y} = r^T \hat{p}_\theta \tag{4.29}$$

where W^{l1}, W^{l2}, b^{l1} and b^{l2} are parameters; $r^T = [1, 2, ..., K]$; $\hat{y}$ is a predicted similarity score.

A sparse target distribution p which satisfies $y = r^T p$ is computed as:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & otherwise \end{cases} \tag{4.30}$$

for $i \in [1, K]$, and y is the similarity score.

To train the model, we minimize the regularized KL-divergence between p and $\hat{p}_\theta$:

$$J(\theta) = \frac{1}{m} \sum_{k=1}^{m} KL(p^{(k)} || \hat{p}_\theta^{(k)}) \tag{4.31}$$

where m is the number of training pairs and θ denotes the model parameters. The gradient descent optimization Adadelta is used to learn the model parameters. We also use mini-batch size as 30 and keep the pre-trained word embeddings fixed during the training phase. We evaluate our models through Pearson correlation r.

4.7 Experiments and Discussion

This section describes two experiments: i) compare our model against recent systems; ii) evaluate the efficiency of using multiple pre-trained word embeddings.

4.7.1 Overall evaluation

Besides existing methods, we also compare our model with several sentence modeling approaches using multiple pre-trained word embeddings:

- **Word Average**:

$$e_s = \frac{1}{n} \sum_{i=1}^{n} e_{w_i}^{concat} \tag{4.32}$$

Table 4.2: Test set results with Pearson's r score×100 for STS tasks, and accuracy for other tasks. Boldface values show the highest scores in each dataset. SICK-R and SICK-E denote the STS task and the entailment task in SICK dataset respectively.

Method	STSB	SICK-R	SICK-E	MRPC
Ensemble models/Feature engineering				
DT_TEAM [Maharjan et al., 2017]	79.2	-	-	-
ECNU [Tian et al., 2017]	81	-	-	-
BIT [Wu et al., 2017]	80.9	-	-	-
TF-KLD [Ji and Eisenstein, 2013]	-	-	-	**80.41/85.96**
Neural representation models (NNM) with one embedding				
Multi-Perspective CNN [He et al., 2015]	-	86.86	-	78.6/84.73
InferSent [Conneau et al., 2017b]	75.8	88.4	**86.1**	76.2/83.1
GRAN [Wieting and Gimpel, 2017]	76.4	86	-	-
Paragram-Phrase [Wieting et al., 2016a]	73.2	86.84	85.3	-
HCTI [Shao, 2017]	78.4	-	-	-
NNM with the five embeddings using sentence-sentence comparison (S)				
S-Word Average	71.06	81.18	80.88	71.48/81.1
S-Project Average	75.12	86.53	85.12	75.48/82.47
S-LSTM	77.14	85.15	85.6	70.43/79.71
S-Max-CNN	81.87	88.3	84.33	76.35/83.75
S-MaxLSTM-CNN	82.2	88.47	84.9	77.91/84.31
NNM with the five embeddings using Multi-level comparison (M)				
M-Max-CNN	82.11	88.45	84.7	76.75/83.64
M-MaxLSTM-CNN	**82.45**	**88.76**	84.95	78.1/84.5

where e_s is the sentence embedding of a n-words sentence, and $e_{w_i}^{concat}$ is from Eq. (1)

- **Project Average**:

$$e_s = \sigma(W(\frac{1}{n}\sum_{i=1}^{n} e_{w_i}^{concat}) + b) \tag{4.33}$$

where W is a 1600 × 1600 weight matrix, and b is a 1600 bias vector.

- **LSTM**: apply Eq. (5-11) on $e_{w_i}^{concat}$ to construct the 1600-dimension e_s sentence embedding.

- **Max-CNN**: apply Eq. (2-4) on $e_{w_i}^{concat}$ to construct the 1600-dimension e_s sentence embedding.

We report the results of these methods in Table 4.2. Overall, our M-MaxLSTM-CNN shows competitive performances in these tasks. Especially in the STS task, M-MaxLSTM-CNN outperforms the state-of-the-art methods on the two datasets. Because STSB includes complicated samples compared to SICK, the performances of methods on STSB are quite lower. In STSB, the prior top performance methods use ensemble

approaches mixing hand-crafted features (word alignment, syntactic features, N-gram overlaps) and neural sentence representations, while our approach is only based on a neural sentence modeling architecture. In addition, we observed that InferSent shows the strong performance on SICK-R but quite low on STSB while our model consistently obtains the strong performances on both of the datasets. InferSent uses transfer knowledge on textual entailment data, consequently it obtains the strong performance on this entailment task.

According to Wieting et al. [2016a], using Word Average as the compositional architecture outperforms the other architectures (e.g., Project Average, LSTM) for STS tasks. In a multiple word embeddings setting, however, Word Average does not show its efficiency. Each word embedding model has its own architecture as well as objective function. These factors make the vector spaces of word embeddings are different. Therefore, we intuitively need a step to learn or refine a representation from a set of pre-trained word embeddings rather than only averaging them. Because Project Average model, LSTM model, and Max-CNN model have their parameters for learning sentence embeddings, they significantly outperform Word Average model.

Table 4.3: Evaluation of exploiting multiple pre-trained word embeddings. $|V|_{avai}$ is the proportion of vocabulary available in a word embedding. In case of using all word embeddings, $|V|_{avai}$ denotes the proportion of vocabulary where each word is available in at least one word embedding.

Word embedding	**STSB**		**SICK-R & SICK-E**			**MRPC**	
	Pearson	$\|V\|_{avai}$(%)	Pearson	Acc	$\|V\|_{avai}$(%)	Acc/F1	$\|V\|_{avai}$(%)
word2Vec	78.9	75.64	87.27	84.09	98.53	75.42/82.13	67.81
fastText	79.95	84.27	87.59	83.42	99.18	74.31/81.75	79.04
Glove	80.1	91.71	88.21	84.71	99.78	74.9/82.782	89.85
SL999	80.31	94.76	87.26	84.55	99.83	76.46/83.13	94.19
Baroni	79.81	90.54	86.9	83.99	98.83	74.84/82.4	87.92
Glove+SL999	81.14	95.07	88.28	84.45	99.83	76.17/83.01	94.29
Glove+SL999+fastText	81.73	95.45	88.38	84.91	99.83	76.46/83.22	94.83
Glove+SL999+fastText+Baroni	82.16	95.65	88.74	84.94	99.83	76.63/82.99	95.06
All	**82.45**	95.65	**88.76**	**84.95**	99.83	**78.1/84.5**	95.97

We observed that MaxLSTM-CNN outperforms Max-CNN in both of the settings (i.e., sentence-sentence comparison, Multi-level comparison). As mentioned in Section 4.1, Max-CNN ignores the property of word order. Therefore, our model achieves improvement compared to Max-CNN by additionally employing LSTM for capturing this property.

We only applied Multi-level comparison on Max-CNN and MaxLSTM-CNN because these encoders generate multi-aspect word embeddings. The experimental results prove the efficiency of using Multi-level comparison. In the textual entailment dataset SICK-E, the task mainly focuses on interpreting the meaning of a whole sentence pair rather than comparing word by word. Therefore, the performance of Multi-level comparison is quite similar to sentence-sentence comparison in the SICK-E task. This is also the reason why LSTM, which captures global relationships in sentences, has strong performance in this task. In Section 4.6.3, we provide a deeper analysis explaining why LSTM's performance is better than our model's in SICK-E.

4.7.2 Evaluation of exploiting multiple pre-trained word embeddings

In this section, we evaluate the efficiency of using multiple pre-trained word embeddings. We compare our multiple pre-trained word embeddings model against models using only one pre-trained word embedding. The same objective function and Multi-level comparison are applied to these models. In case of using one pre-trained word embedding, the dimension of LSTM and the number of convolutional filters are set to the length of the corresponding word embedding. Table 4.3 shows the experimental results of this comparison. Because the approach using five word embeddings outperforms the approaches using two, three, or four word embeddings, we only report the performance of using five word embeddings. We also report $|V|_{avai}$ which is the proportion of vocabulary available in a pre-trained word embedding. SICK dataset ignores idiomatic multi-word expressions, and named entities, consequently the $|V|_{avai}$ of SICK is quite high.

We observed that no word embedding has strong results on all the tasks. Although trained on the paraphrase database and having the highest $|V|_{avai}$, the SL999 embedding could not outperform the Glove embedding in SICK-R. HCTI [Shao, 2017], which is the current state-of-the-art in the group of neural representation models on STSB, also used the Glove embedding. However, the performance of HTCI in STSB (78.4) is lower than our model using the Glove embedding. In SICK-R, InferSent [Conneau et al., 2017b] achieves a strong performance (88.4) using the Glove embedding with transfer knowledge, while our model with only the Glove embedding achieves a performance close to the performance of InferSent. These results confirm the efficiency of Multi-level comparison.

In STSB and MRPC, as employing the five pre-trained embeddings, the $|V|_{avai}$ is increased. This factor limits the number of random values when initializing word embedding representations because a word out of a pre-trained word embedding is assigned a random word embedding representation. In other words, a word out of a pre-trained word embedding is assigned a random semantic meaning. Therefore, the increase of the $|V|_{avai}$ improves the performance of measuring textual similarity. In STSB and MRPC, our multiple pre-trained word embedding achieves a significant improvement in performance compared against using one word embedding. In SICK-R and SICK-E, although the $|V|_{avai}$ is not increased when employing five pre-trained embeddings, the performance of our model is improved. This fact shows that our model learned an efficient word embedding via these pre-trained word embeddings.

4.7.3 Quality analysis

We manually inspect some samples to analysis the advantages and disadvantages of our model (listed in Table 4.4). To answer sample #1, our multi-word embeddings model well evaluates the sharing meaning between words *(man∼person* and *horse∼animal)* compared to single-word embedding models. This capability is a fundamental requirement for STS tasks. However, totally basing on this measurement is not enough for textual entailment tasks. As in sample #2, the rule from sample #1 could not be applied (i.e., *people* and *spectator* are not interchange in the context of sample #2). The degree of word similarity has to consider contextual information. A Context-based similarity evaluator is a promising approach for textual entailment tasks. That is the reason why LSTM focusing on comparing the whole context meanings rather than each word has a strong

Table 4.4: Some typical samples for quality analysis.

No	**Sentence pair**	**Proposed**	**True**
1	a man is riding a horse the person is riding the animal	Entailment	Entailment
2	two spectators are kickboxing and some people are watching two people are kickboxing and spectators are watching	Entailment	Neutral
3	microwave would be your best bet. your best bet is research.	2.8	0
4	it 's not a good idea. it 's not just a good idea, it 's an excellent idea.	3.9	1
5	“ this is america, my friends, and it should not happen here, ” he said to loud applause. “ this is america, my friends, and it should not happen here. ”	Paraphrase	Not paraphrase
6	the victims were last seen at church last sunday ; their bodies were discovered tuesday . the family was last seen july 6 and their bodies were found tuesday .	Not paraphrase	Paraphrase

performance in this task.

In sample #3 and #4, we observed these two pairs share some phrases (e.g., *your best bet*, *a good idea*). Although having the same phrases, these pairs are manually assigned low similarity scores by human, which contradicts our model. In these samples, each word or phrase contributes to its sentence at different degrees. For example, *“microwave”* and *“research”* are more important than *“your best bet”*. The word "just" usually does not contribute so much to its sentence meaning. However, in sample #4, it changes the whole meaning of the sentence. Therefore, the role or contribution of each word in a sentence should be considered for evaluating sentence similarity or textual entailment.

Compared to STS tasks, Paraphrase identification has a little different rule which requires a paraphrase sentence pair to share the same meaning and use different words, phrases or forms. In sample #5, the sentence pair shares the form and almost words, so they are not called paraphrases. As a result, similarity measurement models without the constrain of using different forms, words fail to handle the cases as sample #5. In addition, some cases require extra knowledge as sample #6. Without prior-knowledge, it is hard to link *“victims”* to *“family”*. These challenges make the paraphrase task difficult.

4.8 Conclusion

In this work, we study an approach employing multiple pre-trained word embeddings and Multi-level comparison for measuring semantic textual relation. The proposed M-MaxLSTM-CNN architecture consistently obtains strong performances on several tasks. Compared to the state-of-the art methods in STS tasks, our model does not require handcrafted features (e.g., word alignment, syntactic features) as well as transfer learning knowledge. In addition, it allows using several pre-trained word embeddings with different dimensions.

Future work could apply our multiple word embeddings approach for transfer learning tasks. This strategy allows making use of pre-trained word embeddings as well as available resources.

Chapter 5

Aspect Similarity Recognition

5.1 Introduction

In natural language processing, there are many tasks measuring a relationship between two sentences: semantic similarity, paraphrase identification and textual entailment recognition. In this work, we propose a new task - Aspect Similarity Recognition (ASR) - for measuring sentence similarity in term of aspects. Two sentences are identified as aspect similar if they mention at least one aspect in common.

In Table 5.1, we introduce some examples for this task. The first sentence of sample #1 mentions the aspect of performance while the second sentence mentions the aspects of performance and design. Therefore, this sentence pair shares the aspect of performance. In sample #2, there is no aspect in common between the two sentences where the first is about food and the second is about location. Through sample #3, we could find out how the ASR task is different from the semantic similarity task. The two sentences mention food but the semantic meanings of these two sentences are different. Compared to the aspect category classification task (ACC), which identifies aspects expressed in each sentence, the aspect similarity recognition task learns patterns and terms between two sentences for identifying aspect similarity. In other words, the ACC task learns patterns and terms directly belonging to some particular aspects, while the ASR task learns the aspect similarity of patterns and terms between two sentences without directly using the information of aspects. Therefore, the ASR approach is promising for cross-domain application, where models are trained in a domain and applied to other domains. The ASR task has a high demand in review summarization, which requires a summarized review to be short, accurate, fluent and to cover all aspects of its original reviews.

The ASR task is challenging because of the diversity of linguistic expression. For example, different lexicons and syntaxes could be used to express the same aspect or topic. In addition, one sentence could carry one or many aspects. This fact makes measuring aspect similarity more difficult. According to our knowledge, there has been no research for this task. One reason is that a training corpus is not available so far.

The main contributions of this work are as follows:

- Propose a new task - Aspect Similarity Recognition task and build an Aspect Similarity Recognition corpus (ASRcorpus) with two domains (e.g., restaurants and laptops) which is constructed from the SemEval 2016 Aspect Category Detection

Table 5.1: Some samples of Aspect Similarity Recognition

No.	Sentence	Aspect Similarity
1	The Dell runs so fast. I like its performance and design.	Yes
2	The food is great. The restaurant's location is good.	No
3	I love its food. I bought pizza from the restaurant yesterday.	Yes

dataset[1].

- Design and analysis some conventional deep learning models for this task under in-domain and cross-domain setting.
- Propose an attention-cell LSTM model (ACLSTM) for ASR which enhances the LSTM model via employing attention signals into the input gate and the memory cell. ACLSTM shows improvements compared to the conventional attention models for both settings of in-domain and cross-domain.

The remainder of this paper is organized as follows: Section 2 reviews the related research, Section 3 introduces our model, Section 4 describes the procedure of constructing the dataset, Section 5 describes the experimental setting and discusses the results of the experiments, and Section 6 concludes our work and future work.

5.2 Related work

In this section, we firstly review some recent approaches for aspect category classification, then discuss some researches on measuring a relationship between two sentences.

5.2.1 Aspect Category Classification

Zhou et al. [2015] propose a semi-supervised embedding learning along with a hybrid feature extraction approach for aspect recognition. This model captures semantic relations between word-word, word-aspect and sentiment word-aspect in a unified framework. Toh and Su [2016] introduce an ensemble approach with handcrafted features (i.e., word, head word, name list, word embedding, word cluster) and convolutional features. This proposed model achieves the best performance in SemEval-2016 for aspect category classification. Xue et al. [2017] design a multi-task model based on neural networks. By using BiLSTM and CNN to encode sentences, the model classifies aspect categories and extracts aspect terms simultaneously. He et al. [2017a] employ an attention model (ABAE) to de-emphasize irrelevant words and improve the coherence of aspects. By learning word

[1]http://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools

co-occurrence patterns, ABAE discovers more meaningful and coherent aspects. However, these approaches explicitly use aspect information for training. Therefore, the application on cross-domain setting is limited.

5.2.2 Sentence Relationship Measurement

For modeling sentences, Tai et al. [2015a] extend LSTM with a Dependency Tree-Structured architecture. This model works more efficiently in semantic textual similarity tasks compared to the linear LSTM. There are some researches applying successfully Convolutional Neural Network (CNN) for semantic composition [Kalchbrenner et al., 2014b, Huy Tien and Minh Le, 2017, Shao, 2017]. This technique uses convolutional filters to capture local dependencies in context windows and applies pooling layers to extract global features. He et al. [2015] learn multi-level granularity features via CNN. The author then compares sentence representations via multiple similarity metrics at several granularities. Gan et al. [2017] introduce a hierarchical CNN-LSTM model. In this approach, a continuous representation is generated via CNN, and LSTM is used as a decoder. Conneau et al. [2017b] employ textual entailment transfer learning to encode sentences. This BiLSTM-Maxpooling network achieves competitive results on a wide range of transfer tasks. Tien et al. [2018] propose an M-MaxLSTM-CNN to encode sentences via multi-word embeddings. The author also proposes multiple comparisons to measure sentence relations.

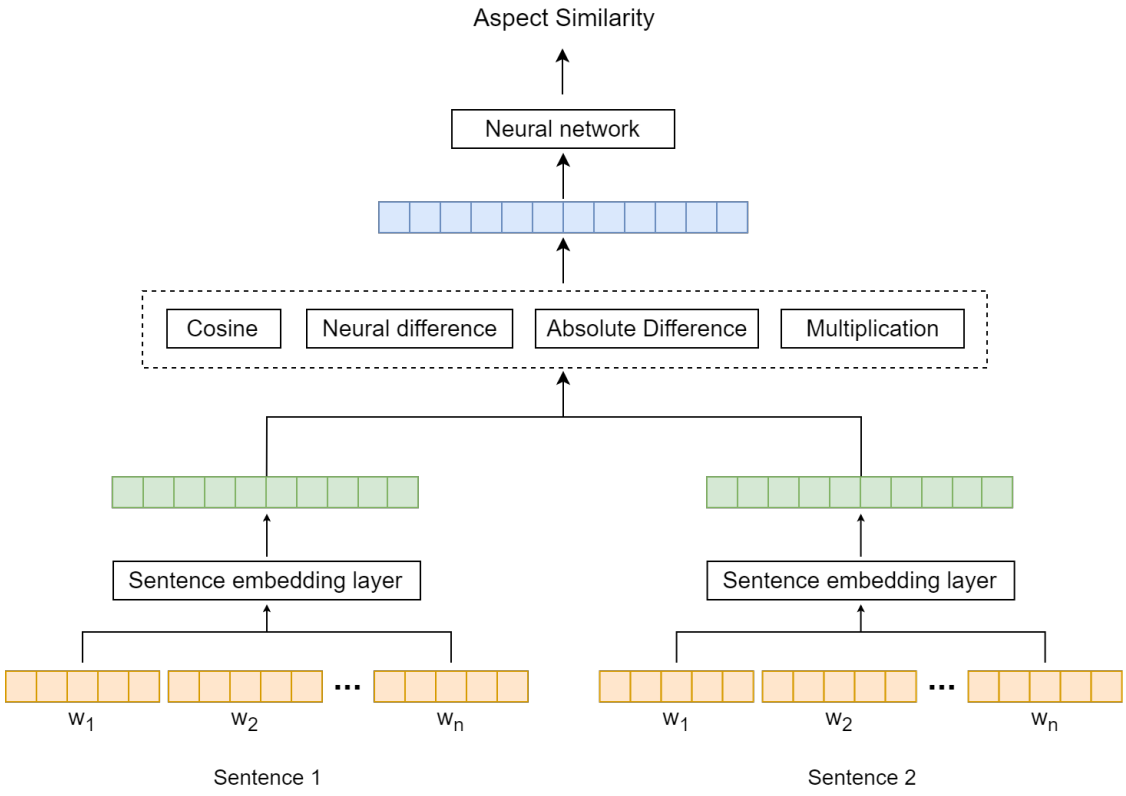


Figure 5.1: The proposed framework for the aspect similarity task

In the SemEval-2017 semantic textual similarity task, ensemble approaches achieve strong results. Wu et al. [2017] train a linear regression model with WordNet, align-

ment features and the word embedding *word2vec*[2]. Tian et al. [2017] develop a hybrid model with multiple boosting techniques (i.e., Random Forest, Gradient Boosting, and XGBoost). This model incorporates conventional features (i.e., n-gram overlaps, syntactic features, alignment features, bag-of-words) and sentence embedding techniques (i.e., Averaging Word Vectors, Projecting Averaging Word Vectors, LSTM).

In this work, we propose a task to measure aspect similarity between two sentences. To the best of our knowledge, this is the first work doing research on this task.

5.3 Methodology

The proposed model of measuring aspect similarity contains three parts: i) Sentence modeling; ii) Sentence comparison; and iii) Aspect similarity transferring. Figure 5.1 shows an overview of our model.

5.3.1 Sentence modeling

For this part, we describe the four sentence embeddings models: Word Average, CNN, LSTM, and BiLSTM.

Word Average[Wieting et al., 2016a]: By representing a word w_i by a pre-trained word embedding e_{w_i}, we construct a sentence S of n words as a sequence of n word embeddings $S = [e_{w_1}, e_{w_2}, ..., e_{w_n}]$. The sentence embedding $e_s$is obtained via the following equation:

$$e_s = \frac{1}{n}\sum_{i=1}^{n} e_{w_i} \tag{5.1}$$

Convolutional neural network (CNN)[Kim, 2014]: A sentence S could be considered as a matrix $d \times s$, where each row is a d-dimension word embedding vector of each word. CNN performs convolution operator on this sentence matrix **S** via linear filters. A filter is represented as a weight matrix W of length d and region size h. W will have $d \times h$ parameters to be learned. For an input matrix $S \in \mathbb{R}^{d \times s}$, a feature map vector $O = [o_o, o_1, ..., o_{s-h}] \in \mathbb{R}^{s-h+1}$ of the convolution operator with a filter W is obtained by applying repeatedly W to sub-matrices of S:

$$o_i = W \cdot S_{i:i+h-1} \tag{5.2}$$

where $i = 0, 1, 2, ..., s-h$, $(\cdot)$ is dot product operation and $S_{i:j}$ is the sub-matrix of S from row i to j.

A pooling layer is applied to each feature map O to capture potential features. The common strategy is 1-max pooling [Boureau et al., 2010]. The idea of 1-max pooling is to extract the most important feature v corresponding to the particular feature map by selecting the highest value of that feature map:

$$v = \max_{0 \le i \le s-h} \{o_i\} \tag{5.3}$$

We have described in detail the process of one filter. In our model, we apply multiple filters with variant region sizes to obtain multiple 1-max pooling values. After pooling,

[2]https://code.google.com/p/word2vec/

these 1-max pooling values from feature maps are concatenated into a sentence embedding e_s.

Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]: By employing a memory cell, LSTM has the ability to capture efficiently long distance dependencies of sequential data without suffering the exploding or vanishing gradient problem of Recurrent neural network [Goller and Kuchler, 1996].

A sentence s is transformed to a fix-length vector e_s by recursively applying a LSTM unit to each word embedding e_{w_t} and the previous step h_{t-1}. At each time step t, the LSTM unit with l-memory dimension defines six vectors in $\mathbb{R}^l$: input gate i_t, forget gate f_t, output gate o_t, tanh layer u_t, memory cell c_t and hidden state h_t as follows (from Tai et al. [2015a]):

$$i_t = \sigma(W_i e_{w_t} + U_i h_{t-1} + b_i) \tag{5.4}$$
$$f_t = \sigma(W_f e_{w_t} + U_f h_{t-1} + b_f) \tag{5.5}$$
$$o_t = \sigma(W_o e_{w_t} + U_o h_{t-1} + b_o) \tag{5.6}$$
$$u_t = \tanh(W_u e_{w_t} + U_u h_{t-1} + b_u) \tag{5.7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{5.8}$$
$$h_t = o_t \odot \tanh(c_t) \tag{5.9}$$
$$e_s = h_n \tag{5.10}$$

where $\sigma, \odot$ respectively denote a logistic sigmoid function and element-wise multiplication; W_i, U_i, b_i are respectively two weights matrices and a bias vector for input gate i. The denotation is similar to forget gate f, output gate o, tanh layer u, memory cell c and hidden state h.

Bi-directional LSTM (BiLSTM): According to the LSTM's equations, the hidden state h_t only employs information of the left context, and does not take the right context into account. To handle this weakness, Dyer et al. [2015] have designed Bi-directional LSTM (BiLSTM) which captures both the left and right context. BiLSTM contains two separate LSTM units, one for forward direction and one for backward direction. Two hidden states $h_t^{fordward}$ and $h_t^{backward}$ from these LSTM units are concatenated into a final hidden state h_t^{bilstm}:

$$h_t^{bilstm} = h_t^{forward} \oplus h_t^{backward} \tag{5.11}$$
$$e_s = h_n^{bilstm} \tag{5.12}$$

where $\oplus$ is concatenation operator.

Attention cell LSTM To enhance recurrent neural networks, we aim to emphasize salient words as encoding sentences over LSTM. A straightforward approach is to learn attention signals by self-attention and then apply these signals into inputs before feeding them into LSTM. In other words, these attention signals are applied to all gates of an LSTM cell. However, we assume emphasized input makes the cell forget more information on the previous state (the forget gate's function) while this state stores the most salient information by the support of attention signals. This conflict causes the inefficiency of integrating attention signals with LSTM. Therefore, we propose a novel LSTM cell which prevents the state from forgetting too much salient information as employing attention signals for encoding sentences. For the ASR task, the proposed attention-cell LSTM

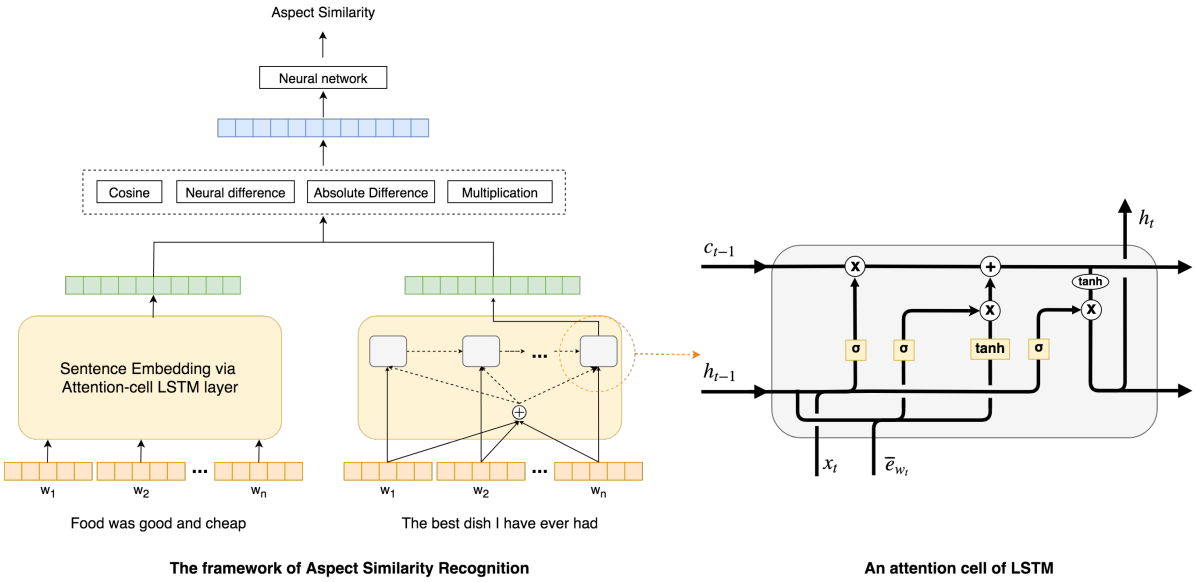


Figure 5.2: The proposed attention cell LSTM

outperforms the conventional LSTM with/out using attention in both of settings: in-domain and cross-domain.

By representing a word w_i by a pre-trained word embedding e_{w_i}, we construct a sentence S of n words as a sequence of n word embeddings $S = [e_{w_1}, e_{w_2}, ..., e_{w_n}]$. Contextual information is incorporated in the word embeddings over the bidirectional GRU [Bahdanau et al., 2014] and then the self-attention signal a_i of w_i is learned as follows (from Yang et al. [2016]):

$$\overleftarrow{h_i} = \overleftarrow{GRU}(e_{w_i}) \tag{5.13}$$

$$\overrightarrow{h_i} = \overrightarrow{GRU}(e_{w_i}) \tag{5.14}$$

$$h_i = \overleftarrow{h_i} \oplus \overrightarrow{h_i} \tag{5.15}$$

$$u_i = tanh(W_a h_i + b_a) \tag{5.16}$$

$$a_i = \frac{exp(u_i^T u_a)}{\sum_i exp(u_i^T u_a)} \tag{5.17}$$

$$\bar{e}_{w_i} = e_{w_1} a_i \tag{5.18}$$

where $\oplus$ is concatenation operator, w_a, b_a, u_a are respectively a weight matrix, a bias, and a context vector. These parameters are randomly initialized and optimized during training.

A sentence s is transformed to a fix-length vector e_s by recursively applying a LSTM cell to each word embedding e_{w_t} and the previous step h_{t-1}. At each time step t, the LSTM unit with l-memory dimension defines six vectors in $\mathbb{R}^l$: input gate i_t, forget gate f_t, output gate o_t, tanh layer u_t, memory cell c_t and hidden state h_t [Tai et al., 2015a]. We modify the conventional LSTM cell to employ attention signals without the conflict

of remembering and forgetting as follows:

$$i_t = \sigma(W_i \bar{e}_{w_t} + U_i h_{t-1} + b_i) \tag{5.19}$$
$$f_t = \sigma(W_f e_{w_t} + U_f h_{t-1} + b_f) \tag{5.20}$$
$$o_t = \sigma(W_o e_{w_t} + U_o h_{t-1} + b_o) \tag{5.21}$$
$$u_t = \tanh(W_u \bar{e}_{w_t} + U_u h_{t-1} + b_u) \tag{5.22}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{5.23}$$
$$h_t = o_t \odot \tanh(c_t) \tag{5.24}$$
$$e_s = h_n \tag{5.25}$$

where $\sigma, \odot$ respectively denote a logistic sigmoid function and element-wise multiplication; W_i, U_i, b_i are respectively two weights matrices and a bias vector for input gate i. The denotation is similar to forget gate f, output gate o, tanh layer u, memory cell c and hidden state h. In the attention-cell LSTM, we introduce the attention signal a_t to only the input gate i_t and the tanh layer u_t, which are in charge of deciding what new information is going to be stored in the cell state. This approach allows the LSTM cell to employ attention for remembering salient information and avoid the unexpected effect of attention on the forget gate. We visualize how the attention-cell LSTM manipulates attention signals in Figure 5.2.

5.3.2 Sentence comparison

After encoding two input sentences s_1 and s_2 into two sentence embeddings e_{s_1} and e_{s_2} (Section 5.3.1), we compute the aspect relation between the two embeddings via the following comparison metrics:

Cosine similarity:

$$d_{cosine} = \frac{e_{s_1} \cdot e_{s_1}}{\|e_{s_1}\| \, \|e_{s_2}\|} \tag{5.26}$$

Multiplication vector & Absolute difference:

$$d_{mul} = e_{s_1} \odot e_{s_2} \tag{5.27}$$
$$d_{abs} = |e_{s_1} - e_{s_2}| \tag{5.28}$$

where $\odot$ is element-wise multiplication.

Neural difference:

$$x = e_{s_1} \oplus e_{s_2} \tag{5.29}$$
$$d_{neu} = W^{neu} x + b^{neu} \tag{5.30}$$

where W^{neu} and b^{neu} are respectively a weight matrix and a bias parameter.

As a result, we have a sentence-sentence similarity vector d^{sent} as follows:

$$d^{sent} = d_{cosine} \oplus d_{mul} \oplus d_{abs} \oplus d_{neu} \tag{5.31}$$

5.3.3 Aspect similarity transferring

The sentence-sentence similarity vector is transferred into an aspect similarity label y through a two layers neural network as follows:

$$sim^{sent} = \sigma(W^{sent} d^{sent} + b^{sent}) \tag{5.32}$$

$$\bar{sim}^{sent} = dropout(sim^{sent}) \tag{5.33}$$

$$\hat{y} = \sigma(W^{y} \bar{sim}^{sent} + b^{y}) \tag{5.34}$$

where $W^{sent}, W^{y}, b^{sent}$, and b^{y} are weight matrices and bias parameters, respectively.

We apply the Dropout layer [Srivastava et al., 2014a] for our model. This technique prevents networks from overfitting by randomly dropping out each hidden unit with a probability p on each presentation of each training case. To train this model, we employ the cross entropy loss function and AdaDelta as the stochastic gradient descent (SGD) update rule. Details of Adadelta method can be found in [Zeiler, 2012]. During the training phase, the pre-trained word embeddings are fixed.

Table 5.2: Statistic of ASRCorpus

	RESTAURANT			LAPTOP		
	Train	Dev	Test	Train	Dev	Test
Sentences	1,239	469	587	1,657	382	573
Sentence pairs	458,676	68,778	98,066	447,506	26,270	44,560
Similarity	229,338	34,389	49,033	223,753	13,135	22,280
Not similarity	229,338	34,389	49,033	223,753	13,135	22,280
Vocabulary		3,769			3,649	

5.4 Aspect Similarity Recognition Dataset

To construct the dataset ASRcorpus for ASR, we employ the Aspect-Based Sentiment dataset from SemEval-2016. In the SemEval dataset, each sentence is labeled with one or some category-opinion pairs. The reason for selecting this dataset is that aspect annotation is much specific. For example, the sentence in Figure 5.3 mentions the four aspects (i.e., FOOD#QUALITY, RESTAURANT#PRICES, FOOD#STYLE_OPTIONS, and FOOD#PRICES) and the four polarities are assigned to these aspects.

In Table 5.2, we report the statistic of ASRCorpus in details. The procedure of constructing ASRCorpus is followed as:

1. For each domain, we construct pairs of sentences from the Semeval-2016 dataset.

2. For each pair: if the two sentences share one or some categories, this pair is annotated as aspect similarity ($label = 1$). Otherwise, it is not aspect similarity ($label = 0$).

```
<sentence id="en_BlueRibbonSushi_478219459:1">
    <text>- The sushi here is perfectly good, but for $5 a piece,
    either the slices of fish should be larger, or there should be
    no pretense that this is a moderately priced restaurant (even
    for NYC).</text>
    <Opinions>
        <Opinion target="sushi" category="FOOD#QUALITY" polarity=
        "positive" from="6" to="11"/>
        <Opinion target="restaurant" category="RESTAURANT#PRICES"
        polarity="negative" from="164" to="174"/>
        <Opinion target="sushi" category="FOOD#STYLE_OPTIONS"
        polarity="negative" from="6" to="11"/>
        <Opinion target="sushi" category="FOOD#PRICES" polarity=
        "negative" from="6" to="11"/>
    </Opinions>
</sentence>
```

Figure 5.3: A sentence in the Aspect-Based Sentiment dataset from SemEval-2016

3. Because the number of not aspect similarity pairs is much greater than aspect similarity pairs, some of not aspect similarity pairs are randomly removed to make these numbers equal.

Table 5.3: Hyper-parameters setting on both of domains: RESTAURANT and LAPTOP

Hyper-parameter	Value
Sentence embedding CNN\LSTM\BiLSTM	300 \ 300 \ 600
BiGRU attention dimension	300
tanh attention dimension	300
Neural difference dimension	5
Penultimate layer sim^{sent} dimension	250
Dropout p	0.5
Batch size	60

5.5 Experiment

5.5.1 Experimental Setting

To tune hyper-parameters of the four models, we do a grid search on 30% of LAPTOP domain. As a result, we obtain these hyper-parameters listed in Table 5.3. We employ Glove word embedding[3] to encode words. For CNN, we used 3 region sizes of $3, 4, 5$; the number of each region size is 100.

[3]https://nlp.stanford.edu/projects/glove/

Table 5.4: The in-domain and cross-domain experimental results on the two domains: RESTAURANT and LAPTOP. "→Y" denotes that models are tested on Y but trained on the other. Accuracy metric is used for evaluation. The results are statistically significant at $p < 0.05$ via the pairwise t-test.

Method	RES	LAP	→RES	→LAP
Word Average	70.75	65.12	54.5	54.59
CNN	77.57	67.23	54.08	54.49
LSTM	79.4	70.21	59.1	57.59
BiLSTM	79.2	71.14	59.2	57.95
Attention	78.79	68	57.92	54.55
LSTM-attention	50	50	50	50
ACLSTM-AspectReg	73	65	-	-
Attention-Cell LSTM	**80**	**72.73**	**59.77**	**58.1**
Attention-Cell BiLSTM	79.42	71.65	59.3	58

5.5.2 Results & discussion

We compare our model to some strong baselines as well as the conventional recurrent networks using attention. In addition, we also compare with a two-steps approach (ACLSTM-AspectReg): extract aspects from a sentences via an Attention-Cell LSTM model, then identify the aspect similarity between two sentences via their extracted aspects. The F1 scores of ACLSTM-AspectReg in the aspects extraction task for RES and LAP are respectively 70 and 47. Table 5.4 shows the results of our experiments. According to the results, the recurrent neural networks (i.e., LSTM and BiLSTM) obtain strong performances. The RESTAURANT domain is quite simpler than the LAPTOP domain where the number of categories is large. Consequently, the performances of LSTM and BiLSTM are similar in the RESTAURANT domain while BiLSTM shows a large improvement compared to LSTM in the LAPTOP domain. By employing efficiently attention signals, the attention-cell LSTM outperforms the conventional recurrent models using attention. As we analysis in Section 4, applying attention to all gates of an LSTM cell causes the conflict of remembering and forgetting. This drawback makes the training of the LSTM-attention model inefficient. Consequently, the trained LSTM-attention model predicts the same label for all inputs. According to the results of ACLSTM-AspectReg, our approach considering aspects as latent variables works more efficiently in this task.

We also evaluate how the models perform in cross-domain setting where the models are trained on one domain dataset and tested on the other. We reports these results in Table 5.4. The cross-domain results are consistent with the in-domain results where the recurrent neural networks outperform the others. We observe that CNN, which captures the best features from sentences by filters, is better than Average Word in in-domain setting but fails in cross-domain setting. These results also prove that these approaches are potential for cross-domain application. We observe that a set of salient words in each domain is different. Therefore, the support of attention signals in domain adaptation is not significant compared to the recurrent models without attention.

To obtain deeper analysis, we inspect the attention-cell LSTM's performance on each

Table 5.5: The attention-cell LSTM's performance on each class.

Domain	*Class*	*Precision*	*Recall*	*F*1
RES	Not Similarity	0.76	0.88	0.81
	similarity	0.86	0.82	0.78
LAP	Not Similarity	0.68	0.87	0.76
	similarity	0.82	0.59	0.68
→RES	Not Similarity	0.58	0.68	0.63
	similarity	0.62	0.52	0.56
→LAP	Not Similarity	0.56	0.72	0.63
	similarity	0.61	0.44	0.51

class (e.g., "similarity" and "not similarity") by precision, recall and F1 scores reported in Table 5.5. In both of the domains and settings, the model performs better on "not similarity" class than "similarity" class in term of F1 score. According to the results in cross-domain setting, we could conclude that the models learn rules, patterns for identifying aspect similarity rather than remembering topic words and keywords in a particular domain.

Table 5.6: Some error samples of the Attention-Cell LSTM

No.	Sentence	Aspect	Similarity
1	I went there with some friends one night to play bingo and watch the sox game and it was a blast!	RESTAURANT#GENERAL	No
	Everything I've had here is good, taco salads, burritos, enchiladas i love this place.	RESTAURANT#GENERAL	
2	This is right up there with places in Tokyo as far as the freshness is concerned.	FOOD#QUALITY	No
	There is only one place on the east coast that has it all, plus a lot more.	RESTAURANT#GENERAL	
3	Food was good and cheap.	FOOD#QUALITY	Yes
	The best Chuwam Mushi I have ever had.	FOOD#QUALITY	
4	Cannot even get to bios.	LAPTOP#OP_PERFORMANCE	Yes
	Just a click and a BLACK SCREEN.	LAPTOP#OP_PERFORMANCE	

5.5.3 Error analysis

To evaluate the challenges of this task, we manually inspect some cases of the wrong prediction by the Attention-Cell LSTM model, which are shown in table 5.6. These pairs are typical examples of the Attention-Cell LSTM model's weakness.

In Sample #1, the second sentence gives positive opinions toward lots of foods (i.e., taco salads, burritos, enchiladas) but its overall content is highlighted by the phrase "love this place" (i.e., FOOD#QUALITY→RESTAURANT#GENERAL). This fact gives challenges to our model. Similarly, the word "freshness" modifies the aspect mentioned in the latter sentence of Sample #2 although these sentences carry the word "place". To handle this problem, models need to pay more attention to polarity words (e.g., freshness, overpriced) and make a connection between aspect terms and polarity words. Therefore, an attention mechanism is promising to deal with this difficulty.

Another source of the wrong prediction comes from proper nouns. For example, the word "Chuwam Mushi" refers to a Japanese food. Without this prior knowledge, it is hard to answer Sample #3. In addition, terminology words (e.g., bios and back screen) also make this task more challenge. The word "BLACK SCREEN" usually expresses a design aspect but in the context of LAPTOP domain, it refers to an error in performance. To tackle this problem, we could apply a preprocessing to replace proper nouns or terminology words with more abstractive words (e.g., "Chuwam Mushi" → "food" and "BLACK SCREEN"→"error").

5.6 Conclusion

In this work, we propose a new task - Aspect Similarity Recognition for identifying whether two sentences share some aspects. We also construct a dataset ASRcorpus from the SemEval-2016 Aspect Sentiment Analysis dataset with two domains: RESTAURANT and LAPTOP. To evaluate how the performances of supervised learning models, we compare our proposed attention cell LSTM model to some strong baselines as well as the conventional recurrent networks using attention for this task. Through the experimental results, the proposed model which integrates efficiently attention signals into LSTM outperforms the baselines on both settings of in-domain and cross-domain. In addition, the cross-domain results show that these models are also applicable to cross-domain learning. According to the error analysis, we identified some typical difficulties of this task. An aspect-polarity attention with pre-processing is expected to handle these challenges.

Chapter 6

Extractive Opinions Summarization

6.1 Introduction

In extractive opinions summarization, most existing approaches use the aspects information for discarding potentially redundant units. For minimizing repeated information on the same aspect, we only need to identify whether two text units have at least one aspect in common, which is called Aspect Similarity Recognition - ASR [Nguyen et al., 2018], rather than explicitly extracting aspects of each text unit. Follow this observation, we propose an aspect-based summarization using ASR instead of aspect discovery. The advantage of ASR is to learn patterns and relations between two text units and not need to identify the aspects of each unit, therefore it is potential to cross-domain application. Our contributions in this work are as follows:

- We introduce a novel aspect-based summarization using Aspect Similarity Recognition. This approach relaxes the constraint of predefined aspects.
- According to the experiments, our method outperforms strong baselines on Opinosis corpus. We also evaluate our method in regard to domain adaptation.

The remainder of this paper is organized as follows: Section 6.2 reviews the related research, Section 6.3 describes the problem formulation, Section 6.4 introduces the proposed summarization using ASR, Section 6.5 discusses the experiments for ASR and summarization, and Section 6.6 concludes our work and future work.

6.2 Related work

In the scope of this paper, we focus on discussing neural-based systems for generic and opinions summarization. For a comprehensive literature of non-neural techniques, we refer the reader to Liu and Zhang [2012].

For extractive generic summarization, Cao et al. [2015] rank sentences in a parsing tree via a recursive neural network. However, the model requires handcrafted features as input. Cheng and Lapata [2016] propose an end-to-end model for extracting words and sentences. In this system, a document is encoded via convolutional and recurrent layers, then an attention architecture is employed to extract sentences and words. Follow this work, Zhou et al. [2018] enhance the previous system by jointly learning to score

and select sentences. By integrating sentence scoring and selecting into one phase, as the model selects a sentence, the sentence is scored according to the partial output summary and current extraction state.

To our knowledge, the first neural-based model of extractive opinions summarization is proposed by Kågebäck et al. [2014], which uses an unfolding recursive auto-encoder to learn phrase embeddings and measures similarity by Cosine and Euclidean distance. The limitation of this system is to purely rely on semantic similarity without taking into account the aspect information. Yang et al. [2017] use the unsupervised neural attention-based aspect autoencoder (ABAE) [He et al., 2017b] for presenting each aspect in an aspect embedding space. Then, the representative sentence for each aspect is selected via its distance with the centroid of that aspect. For summarization, however, ABAE is not efficient compared to K-mean in the aspects which occur more frequently in the dataset. Angelidis and Lapata [2018] introduce seed words of each domain to the autoencoder ABAE. This weakly-supervised model which is trained under multi-task objective outperforms the unsupervised model for aspect extraction. Different from the previous work in aspect-based opinions summarization, we apply aspect similarity recognition (ASR) instead of aspect extraction. ASR facilitates the problem of domain adaptation in summarization.

6.3 Problem Formulation

Every product e contains a set of reviews $R_e = \{r_i^e, ..., r_n^e\}$ expressing users' opinions on that product. A review r_i^e is viewed as a sequence of sentences $(s_1, ..., s_m)$. For each product e, our goal is to select the most salient sentences in reviews R_e for producing a summary. The proposed approach is divided into subtasks as follows:

1. **Sentiment prediction** determines the overall polarity $p_s \in [-1, +1]$ a sentence carries, where $-1, +1$ respectively indicate maximally negative and positive. According to Angelidis and Lapata [2018], highly positive or negative opinions are more likely to contain informative text than neutral ones. In our system, we use the ensemble sentiment classifier proposed in Chapter 2, which achieves strong performances at sentence level.

2. **Semantic textual similarity** measures the semantic similarity q_{ij} of two sentences i and j, which plays an important role in identifying the most informative sentences as well as redundant ones. We use the multi-word embeddings and multi-level comparison model proposed in Chapter 4 for this task.

3. **Aspect similarity recognition (ASR)** predicts a probability r_{ij} that two sentences i and j shares at least one aspect. This subtask facilitates the elimination of redundant text in summarization, especially for domain adaptation. The attention cell LSTM proposed in Chapter 5 is applied for this task.

4. **Summarization generation** employs the three signals above for ranking sentences. A concise and informative summary of a product e is generated by selecting the most salient sentences from reviews R_e.

Section 6.4 explains how to combine the polarity, semantic and aspect similarity to produce a summary.

6.4 Opinion Summarization

Given a product e, we aim to rank a set of sentences $D = \{s_i\}$ from the reviews talking about the product e. The procedure of scoring and selecting sentences for constructing an opinion summary K of the product e is as follows:

1. In the first step $t = 0$, we score each sentence $s_i \in D$ and select the most salient sentence $\hat{s}^0$ for the summary K:

$$asp_{s_i}^{t=0} = \frac{1}{|D|} \sum_{j \in D} r_{ij} \tag{6.1}$$

$$sim_{s_i}^{t=0} = \frac{1}{|D|} \sum_{j \in D} q_{ij} \tag{6.2}$$

$$sal_{s_i}^{t=0} = (1 + \alpha |p_{s_i}|) * asp_{s_i}^{t=0} * sim_{s_i}^{t=0} \tag{6.3}$$

$$\hat{s}^0 = \arg\max_{s_i \in D} \{sal_{s_i}^{t=0}\} \tag{6.4}$$

$$K^{t=1} = K \cup \{\hat{s}^0\} \tag{6.5}$$

$$D^{t=1} = D \setminus \{\hat{s}^0\} \tag{6.6}$$

   At the step $t = 0$, the salient sal_{s_i} is computed by the semantic similarity sim_{s_i}, the aspect coverage sim_{s_i} and the polarity p_{s_i}. Different from the previous works, we also take into account the aspect coverage in which a sentence carrying more aspects has a higher salient score. In addition, the polarity of a sentence contributes to its ranking by a coefficient $\alpha \in [0, 1]$.

2. In the next step t, the salient sentence $\hat{s}^t$ is selected as follows:

$$asp_{s_i}^{t} = \frac{1}{|D^t|} \sum_{j \in D^t} r_{ij} \tag{6.7}$$

$$sim_{s_i}^{t} = \frac{1}{|D^t|} \sum_{j \in D^t} q_{ij} \tag{6.8}$$

$$\bar{sal}_{s_i}^{t=0} = (1 + \alpha |p_{s_i}|) * asp_{s_i}^{t} * sim_{s_i}^{t} \tag{6.9}$$

   To avoid the redundant information, we penalize each sentence s_i by the aspect similarity $acov_{s_i}^t$ and semantic similarity $scov_{s_i}^t$ of that sentence with the selected sentences, in which β is a coefficient:

$$acov_{s_i}^{t} = \frac{1}{|K^t|} \sum_{j \in K^t} r_{ij} \tag{6.10}$$

$$scov_{s_i}^{t} = \frac{1}{|K^t|} \sum_{j \in K^t} q_{ij} \tag{6.11}$$

$$sal_{s_i}^{t} = \bar{sal}_{s_i}^{t} - \beta * acov_{s_i}^{t} * scov_{s_i}^{t} \tag{6.12}$$

$$\hat{s}^t = \arg\max_{s_i \in D^t} \{sal_{s_i}^{t}\} \tag{6.13}$$

$$K^{t+1} = K^t \cup \{\hat{s}^t\} \tag{6.14}$$

$$D^{t+1} = D^t \setminus \{\hat{s}^t\} \tag{6.15}$$

3. We repeat step 2 until the number of selected sentences is reached or the most salient score at the current step t is lower than a threshold. To avoid missing topic words in a summary, in step 1 and 2, we only select sentences containing words belonging to the list of frequent words on that topic. According to our observation, the topic words are the most frequent.

6.5 Experiments & Results

The Opinosis dataset [Ganesan et al., 2010] includes user reviews of 51 different topics (e.g., hotel, car, product). Each topic includes between 50 and 575 sentences made by various authors and around 4 reference summaries created by human. The corpus is suited for opinion summarization as well as evaluating the ability of domain adaptation.

We use ROUGE to assess the agreement of generated summaries and gold summaries. Our experiments include ROUGE-1, ROUGE-2 and, ROUGE-SU4, which base on one-gram, bi-gram, and skip-bigram co-occurrences respectively.

The model for each subtask in our summarization system is implemented as follows:

- Sentiment prediction: the ensemble classifier proposed in Chapter 2 is trained on Stanford Sentiment Treebank [Socher et al., 2013] with the accuracy of 88.6%.
- Semantic textual similarity: the multi-level comparison model proposed in Chapter 4 is trained on STSbenchmark[1] with the accuracy of 82.45%.
- Aspect similarity recognition: the attention-cel LSTM proposed in Chapter 5 is trained on the ASRcorpus of both domains with the accuracy of 76.2%.
- Summary generation: we set $\alpha = 1.67$ and $\beta = 0.1$. The number of the most frequent words is three. These parameters are optimized over a set of 5 topics randomly selected from the Opinosis dataset. According to the analysis of Ganesan et al. [2010], the size of a summary is two sentences.

For comparison, we use MEAD [Radev et al., 2000] and CW-Add$_{Euc}$ [Kågebäck et al., 2014] as baselines. MEAD is an extractive method based on cluster centroids which selects the salient sentences by a collection of the most important words. CW-Add$_{Euc}$ measures the Euclidean similarity between two sentences by their continuous vector space. In addition, we also report the contribution of using aspect and sentiment information in summarization. The results denoted OPT$_R$ and OPT$_F$ in Table 6.1 describe the upper bound score of recall and F-score respectively. As the reference summaries of Opinosis are generated in abstractive approach by humans, our generated summaries cannot fully match with the reference summaries. For example, the maximum recall which an extractive method could achieve in ROUGE-1 is 57.86%.

While MEAD selects long sentences (around 75 words) containing a lot of salient words to achieve a high score in recall but low in precision, our approach obtains a balance between these scores with quite shorter sentences (around 17 words).

To analyze why sentiment signals cause negative impacts on the summarization generation, we inspect the most polarity sentences in the corpus. Some typical sentences are

[1] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

Table 6.1: Performace comparison between the proposed methods and baselines.

Method	**ROUGE-1**			**ROUGE-2**			**ROUGE-SU4**		
	Recall	Precision	F	Recall	Precision	F	Recall	Precision	F
OPT_R	57.86	21.96	30.28	22.96	12.31	15.33	29.5	13.53	17.7
OPT_F	45.93	48.84	46.57	20.42	19.94	19.49	23.17	26.5	23.7
MEAD	**49.32**	9.16	15.15	**10.58**	1.84	3.08	**23.16**	1.02	1.89
CW-Add_{Euc}	29.12	22.75	24.88	5.12	3.6	4.1	10.54	7.59	8.35
The proposed summarizer									
Semantic	28.24	28.63	27.62	7.34	7.19	7	10.69	10.94	10.4
Semantic+Aspect	29.2	**29.19**	**28.24**	7.45	**7.29**	**7.12**	11.25	**11.26**	**10.78**
Aspect+Polarity	27.77	27.86	26.92	7.24	7.09	6.93	10.42	10.55	10.04
Semantic+Aspect+Polarity	28.56	28.31	27.5	7.06	6.84	6.71	10.92	10.83	10.4

listed in Table 6.2. We observe that most of these sentences express individual experiences and too subjective to be selected for summarization. According to the Opinosis dataset, overstrong words (i.e., rude, extremely) and subjective words (i.e., my wife, I, we) are seldom present in a summary. These factors lead to an unexpected result of using polarity information in summarization although sentences carrying the most polarity are still informative.

Table 6.2: Some sentences carrying the most polarity in the Opinosis dataset.

Positive sentences
I purchased a 2007 Camry because of the looks of the redesigned model and because of the legendary Toyota quality and reliability.
The Concierge staff, exceptional and extremely helpful, right from suggestions on transportation excursion options to recommending an amazing restaurant.
When I checked in, I asked to be shown several rooms and the staff was happy to do so.
Negative sentences
My wife does say the vehicle is not as comfortable for long trips as other cars we've owned.
We had to go up a floor and into a service area to find ice.
The rude and poor service started from the concierge who was curt when I asked a question.

We expect that aspect signals support to generate an informative summary, which is a summary carrying salient information on various aspects. However, the ROUGE metric measures the number of matches between two pieces of text, so it is difficult to compare which one is more informative. Therefore, we execute an **informative test** to understand whether aspect signals help to generate a more informative summary. Given reference summaries and two summaries generated by the system with/out using aspect signals respectively, three persons are asked to select one of the three answers: which

Table 6.3: Informative test for using Semantic with Aspect against without Aspect.

Domain	*Class*	*Semantic + Aspect*
Tablet	More informative	33%
	Less informative	13%
	Equally informative	54%
Others	More informative	17%
	Less informative	8%
	Equally informative	72%

system's summary is more informative, or both of them are equally informative. The inter-rater agreement Cohen's Kappa score for each pair of assessors is more than 0.74. The overall answer is concluded by the majority vote scheme. In case of receiving three different answers, that pair of summaries is assigned as equally informative. The result reported in Table 6.3 includes domain specification (15 samples in *Tablet* and 36 samples in *Others*), which facilitates the evaluation of domain adaptation. As the ASR system is trained on the restaurant and laptop dataset, we consider tablet's topics in the Opinosis corpus as in-domain and others as out-of-domain. According to the informative test, the system with aspect dominates in both of the domains (*Tablet* and *Others*). This result proves the contribution of aspect signals and the domain adaptation of the ASR system.

To obtain a better view of the advantages and disadvantages in our system, we show some generated summaries against reference summaries in Table 6.4. In extractive methods, the most salient sentences are selected from different reviewers, so it is possible to have repeated information in a summary. For instance in the case #1, the first sentence mentions *quiet* and *comfortable ride* while the second one contains *ride* and *seating*. Although these sentences still have different opinions (i.e., *quiet* vs *seating*), the repeat of *comfortable ride* downgrades the generated summary's quality. For improvement, we suggest a post-processing for a more concise summary by filtering redundant information. As the proposed aspect-based system ranks a sentence by not only semantic cover but also aspect cover, it selects the more salient opinions for summarization. For instance, although both of the systems extract different features (e.g., interior vs seating, breakfast vs tube and bus), the opinions (i.e., seating, tube and bus) chosen by the system with aspect support are more suited to the reference summaries.

In each topic, although the reference summaries and generated summary share most of the meaning, they deliver information in different ways as well as words. This fact makes the quality evaluation of generated summaries difficult. In addition to the ROUGE metric, we conducted the informative test for quality evaluation. However, for a large corpus or multiple systems comparison, this test requires a huge amount of human effort. Therefore, it is a high demand to have a reliable metric for summaries evaluation without human involvement.

Table 6.4: Human and system summaries for some products/services. For each topic, we list three summaries by human.

Summary on the Comfort of Toyota Camry 2007	
Human	[1] The Camry offers interior comfort, while providing a quiet ride. Comfortable seating and easy to drive. [2] Overall very comfortable ride front and back. Nice and roomy. [3] Its very comfortable and a quiet ride with low levels of noise.
Semantic	The ride is quiet and comfortable. Very comfortable, quiet interior.
Semantic+Aspect	The ride is quiet and comfortable. Very comfortable ride and seating.
Summary on the location of Holiday Inn London	
Human	[1] Location is excellent, very close to the Glouchester Rd. Tube stop. [2] Excellent location. Near the tube station. [3] The location is excellent. The hotel is very convenient to shopping, sightseeing, and restaurants. It is located just minutes from the tube stations.
Semantic	Great location but don't bring the car! Great location great breakfast!
Semantic+Aspect	Great location but don't bring the car! Great location for the tube and bus!

6.6 Conclusion

In this work, we introduced a novel aspect-based opinions summarization framework using aspect similarity recognition. This subtask relaxes the constraint of predefined aspects in conventional aspect categorization tasks. For summarization, we evaluated our system on the Opinosis corpus. In addition to ROUGE metric, an informative test with human involvement was implemented to show the domain adaptation ability of our system and how informative our generated summaries are. In the corpus, we observe that sentences carrying the most polarity are not suited to summarization. Therefore, employing sentiment for summarization needs deeper analysis. Due to the ASR task's advantage, we believe that it has a high demand in some fundamental tasks of natural language processing such as information retrieval, and sentence comparison.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Our thesis is motivated by the fact that opinions summarization on social media will benefit for many applications, and deep learning are a promising approach for solving that task.

The main contributions of this dissertation are summarized as follows:

- **Sentiment analysis** (Chapter 2): We combine the advantages of various models (e.g., LSTM and CNN) via the proposed freezing technique. This approach efficiently prevents joint networks from over-fitting. In addition, we also observe that clustering semantically documents/sentences supports generative models' drawbacks. A neural voting ensemble with additional NBSVM is used to boost the performance of each cluster. The approach obtains the strong performance in sentiment analysis.

- **Subject Toward Sentiment Analysis on Social Media** (Chapter 3): To encode contextual information into the conventional word embedding, a convolutional N-gram BiLSTM word embedding model is learned via i) multiple convolutional filters with variant sizes for capturing contextual information; ii) BiLSTM for encoding long distance contextual dependencies. Our model achieves significant improvements and robustness in the multilingual and cross domain experiments on the SenTube dataset compared with the previous work STRUCT - the state-of-the-art method on the SenTube dataset. While the previous work requires a predefined sentiment dictionary and some linguistic preprocessing tools, our model only requires a pre-trained word embedding which is trained in unsupervised learning scheme. Therefore, our model has larger applicability to multilingual environments as YouTube.

- **Semantic textual similarity** (Chap 4): The proposed M-MaxLSTM-CNN architecture, which employs multiple pre-trained word embeddings and Multi-level comparison for measuring semantic textual relation, consistently obtains competitive performances on several tasks. The advantage is that the model allows using several pre-trained word embeddings with different dimensions. Compared to the state-of-the art methods in STS tasks, our model does not require handcrafted features (e.g., word alignment, syntactic features) as well as transfer learning knowledge.

- **Aspect Similarity Recognition** (Chap 5): We proposed an attention cell LSTM model for a new task - Aspect Similarity Recognition for identifying whether two sentences share some aspects. A dataset ASRcorpus is also constructed from the SemEval-2016 Aspect Sentiment Analysis dataset with two domains: RESTAURANT and LAPTOP. Through the experimental results, the proposed model which integrates efficiently attention signals into LSTM outperforms the baselines on both settings of in-domain and cross-domain. In addition, the cross-domain results show that these models are also applicable to cross-domain learning.

- **Extractive Opinions Summarization** (Chap 6): an opinions summary for product's reviews is generated vi a novel aspect-based opinions summarization framework using aspect similarity recognition. By applying this subtask, we relax the constraint of predefined aspects in conventional aspect categorization tasks. For experiments, we evaluated our system on the Opinosis corpus. In addition to ROUGE metric, an informative test with human involvement was implemented to show the domain adaptation ability of our system and how informative our generated summaries are.

7.2 Future Work

The next study will focus on the following things:

- **Sentiment analysis** (Chapter 2): In our work, we just researched on simple models. It is interesting to apply our freezing scheme approach to combination models (e.g., multi-channel CNN-LSTM, hierarchal LSTM) for generating feature vectors. In addition, our clustering is based on semantical similarity. Research on other kinds of similarity could lead to valuable results.

- **Subject Toward Sentiment Analysis on Social Media** (Chapter 3) For future work, we plan to improve the model's inference for cases where main subjects are not mentioned explicitly. In addition, it is useful if a model can extract helpful comments, which give polarity sentiments and explanation for those sentiments.

- **Semantic textual similarity** (Chap 4): Recently, transfer learning at sentence level achieves many successes. The combination of our multiple word embeddings approach and transfer learning can lead to interesting results.

- **Aspect Similarity Recognition** (Chap 5): To improve the performance, the model has to pay more attention to aspect and polarity words as analyzed in Section 5.5.3. In addition, a preprocessing step to replace proper nouns or terminology words with more abstractive words should be considered.

- **Extractive Opinions Summarization** (Chap 6): Through the error analysis on the Opinosis corpus, we observe that sentences carrying the most polarity are not suited to summarization. Therefore, employing sentiment for summarization needs deeper analysis. Due to the robust results of applying the aspect similarity recognition in summarization, we believe that it is promising to apply this subtask for other tasks such as information retrieval, and sentence comparison.

Bibliography

M. AL-Smadi, Z. Jaradat, M. AL-Ayyoub, and Y. Jararweh. Paraphrase identification and semantic text similarity analysis in arabic news tweets using lexical, syntactic, and semantic features. *Information Processing & Management*, 53(3):640 – 652, 2017. ISSN 0306-4573.

S. Angelidis and M. Lapata. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *EMNLP*, 2018.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003. ISSN 1532-4435.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5:135–146, 2017a.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017b. ISSN 2307-387X.

Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36 – 64, 2016. ISSN 0004-3702.

E. Cambria. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31 (2):102–107, Mar 2016. ISSN 1541-1672.

Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou. Ranking with recursive neural networks and its application to multi-document summarization. In *In AAAI 2015*, pages 2153–2159. AAAI Press, 2015. ISBN 0-262-51129-0.

I. Chaturvedi, E. Cambria, and D. Vilares. Lyapunov filtering of objectivity for spanish sentiment model. In *International Joint Conference on Neural Networks(IJCNN)*, pages 4474–4481. IEEE, 2016.

Y.-C. Chen and M. Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/P18-1063.

J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of ACL (Volume 1: Long Papers)*, pages 484–494. ACL, 2016.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017a. Association for Computational Linguistics.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017b.

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

D. Das and N. A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics, 2009.

C. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland, Aug. 2014. Dublin City University and Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C14-1008.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 334–343. Association for Computational Linguistics, 2015.

R. Ferreira, R. D. Lins, S. J. Simske, F. Freitas, and M. Riss. Assessing sentence similarity through lexical, syntactic and semantic analysis. *Computer Speech & Language*, 39:1 – 28, 2016. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2016.01.003.

R. Ferreira, G. D. Cavalcanti, F. Freitas, R. D. Lins, S. J. Simske, and M. Riss. Combining sentence similarities measures to identify paraphrases. *Computer Speech & Language*, 47:59 – 73, 2018. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2017.07.002.

E. Fersini, E. Messina, and F. A. Pozzi. Expressive signals in social media languages to improve polarity detection. *Information Processing & Management*, 52(1):20–35, 2016.

Z. Gan, Y. Pu, R. Henao, C. Li, X. He, and L. Carin. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2380–2390, 2017.

K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd COLING*, pages 340–348. ACL, 2010.

M. Giatsoglou, M. G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, and K. C. Chatzisavvas. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214–224, 2017.

A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

C. Goller and A. Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, Jun 1996.

D. K. Gupta, K. S. Reddy, A. Ekbal, et al. Pso-asent: Feature selection using particle swarm optimization for aspect based sentiment analysis. In *International Conference on Applications of Natural Language to Information Systems*, pages 220–233. Springer, 2015.

H. He, K. Gimpel, and J. J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, 2015.

R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397. Association for Computational Linguistics, 2017a. doi: 10.18653/v1/P17-1036.

R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*, Vancouver, Canada, July 2017b. ACL.

F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1.

M. Hu and B. Liu. Opinion extraction and summarization on the web. In *Proceedings of AAAI*, volume 2, 01 2006.

N. Huy Tien and N. Minh Le. An ensemble method with sentiment features and clustering support. In *Proceedings of the Eighth IJCNLP (Volume 1: Long Papers)*, pages 644–653. Asian Federation of Natural Language Processing, 2017.

O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2096–2104. Curran Associates, Inc., 2014.

M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics, 2015.

K. L. Jacob Devlin, Ming-Wei Chang and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, USA, June 2019. Association for Computational Linguistics.

Y. Ji and J. Eisenstein. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896. Association for Computational Linguistics, 2013.

Y. Jiang, W. Bai, X. Zhang, and J. Hu. Wikipedia-based information content and semantic similarity computation. *Information Processing & Management*, 53(1):248 – 265, 2017. ISSN 0306-4573.

M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39. Association for Computational Linguistics, 2014.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modeling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics, 2014a. doi: 10.3115/v1/P14-1062.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665. Association for Computational Linguistics, 2014b.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

S. Kiritchenko, X. Zhu, and S. M. Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1188–1196. PMLR, 22–24 Jun 2014.

O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308, 2014.

B. Li, T. Liu, X. Du, D. Zhang, and Z. Zhao. Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. In *Proceedings of the Workshop of 4th International Conference on Learning Representations (ICLR)*, 2016.

B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

B. Liu and L. Zhang. *A Survey of Opinion Mining and Sentiment Analysis*, pages 415–463. Springer US, Boston, MA, 2012. ISBN 978-1-4614-3223-4.

Y. Ma, H. Peng, and E. Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011a. Association for Computational Linguistics.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011b.

N. Madnani, J. Tetreault, and M. Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics, 2012.

N. Maharjan, R. Banjade, D. Gautam, L. J. Tamang, and V. Rus. Dt_team at semeval-2017 task 1: Semantic similarity using alignments, sentence-level embeddings and gaussian mixture model output. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 120–124, 2017.

S. Maruf and G. Haffari. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284. Association for Computational Linguistics, 2018.

R. Mihalcea, C. Corley, C. Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

S. Moghaddam and M. Ester. The flda model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22Nd International Conference on WWW*, pages 909–918, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2035-1. doi: 10.1145/2488388.2488467. URL http://doi.acm.org/10.1145/2488388.2488467.

S. Mohammad, S. Kiritchenko, and X. Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327. Association for Computational Linguistics, 2013.

A. Nenkova, L. Vanderwende, and K. McKeown. A compositional context sensitive multi-document summarizer: Exploring the factors that influence summarization. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, SIGIR '06, pages 573–580, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7.

H. T. Nguyen and N. M. Le. Multilingual opinion mining on youtube a convolutional n-gram bilstm word embedding. *Information Processing and Management*, 54(3):451 – 462, 2018. ISSN 0306-4573.

H. T. Nguyen, Q.-H. Vo, and M.-L. Nguyen. A deep learning study of aspect similarity recognition. In *Proceedings of the 10th International Conference KSE*, 2018.

A. C. Pandey, D. S. Rajpoot, and M. Saraswat. Twitter sentiment analysis using hybrid cuckoo search method. *Information Processing & Management*, 53(4):764–779, 2017.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124. Association for Computational Linguistics, 2005.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

V. Phung and L. De Vine. A study on the use of word embeddings and pagerank for vietnamese text summarization. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 7:1–7:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-4040-3.

R. Qu, Y. Fang, W. Bai, and Y. Jiang. Computing semantic similarity based on novel models of semantic representation using wikipedia. *Information Processing & Management*, 54(6):1002 – 1021, 2018. ISSN 0306-4573.

D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. Mead - a platform for multidocument multilingual text summarization. In *Proceedings of the Fourth LREC'04*. ELRA, 2004.

D. R. Radev, H. Jing, and M. Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*, 2000.

H. Saif, Y. He, M. Fernandez, and H. Alani. Contextual semantics for sentiment analysis of twitter. *Information Processing & Management*, 52(1):5–19, 2016.

A. Severyn, A. Moschitti, O. Uryupina, B. Plank, and K. Filippova. Multi-lingual opinion mining on youtube. *Information Processing & Management*, 52(1):46–60, 2016.

Y. Shao. Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 130–133, 2017.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, volume 1631, page 1642. Citeseer, 2013.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014a.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014b.

M. A. Sultan, S. Bethard, and T. Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association of Computational Linguistics*, 2:219–230, 2014.

M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1556–1566. ACL, 2015a.

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566. Association for Computational Linguistics, 2015b.

D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565. Association for Computational Linguistics, 2014.

D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432. Association for Computational Linguistics, 2015.

J. Tian, Z. Zhou, M. Lan, and Y. Wu. Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 191–197, 2017.

H. N. Tien, M. N. Le, Y. Tomohiro, and I. Tatsuya. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *CoRR*, abs/1805.07882, 2018. URL http://arxiv.org/abs/1805.07882.

Z. Toh and J. Su. Nlangp at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 282–288. Association for Computational Linguistics, 2016. doi: 10.18653/v1/S16-1045.

O. Uryupina, B. Plank, A. Severyn, A. Rotondi, and A. Moschitti. Sentube: A corpus for sentiment analysis on youtube social media. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014.

O. Vechtomova. Disambiguating context-dependent polarity of words: An information retrieval approach. *Information Processing & Management*, 53(5):1062–1079, 2017.

D. Vilares, M. A. Alonso, and C. Gómez-Rodríguez. Supervised sentiment analysis in multilingual environments. *Information Processing & Management*, 53(3):595–607, 2017.

Q. H. Vo, H. T. Nguyen, B. Le, and M. L. Nguyen. Multi-channel lstm-cnn model for vietnamese sentiment analysis. In *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, pages 24–29, Oct 2017.

S. Wan, M. Dras, R. Dale, and C. Paris. Using dependency-based features to take theparafarceout of paraphrase. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 131–138, 2006.

G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/P18-1216.

J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 225–230. Association for Computational Linguistics, 2016.

S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 90–94. Association for Computational Linguistics, 2012.

J. Wieting and K. Gimpel. Revisiting recurrent networks for paraphrastic sentence embeddings. In *ACL (1)*, pages 2078–2088. Association for Computational Linguistics, 2017.

J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*, 2015.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards universal paraphrastic sentence embeddings. *The International Conference on Learning Representations*, 2016a.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, 2016b.

H. Wu, H. Huang, P. Jian, Y. Guo, and C. Su. Bit at semeval-2017 task 1: Using semantic information space to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 77–84, 2017.

W. Xu, A. Ritter, C. Callison-Burch, W. B. Dolan, and Y. Ji. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association of Computational Linguistics*, 2:435–448, 2014.

W. Xue, W. Zhou, T. Li, and Q. Wang. Mtna: A neural multi-task model for aspect category classification and aspect term extraction on restaurant reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume*

2: Short Papers), pages 151–156. Asian Federation of Natural Language Processing, 2017.

Y. Yang, C. Chen, M. Qiu, and F. Bao. Aspect extraction from product reviews using category hierarchy information. In *Proceedings of the 15th Conference of EACL: Volume 2, Short Papers*, pages 675–680. ACL, 2017.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the NAACL: Human Language Technologies*, pages 1480–1489. ACL, 2016. doi: 10.18653/v1/N16-1174.

A. Yessenalina and C. Cardie. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics, 2011.

W. Yin and H. Schütze. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 204–214. Association for Computational Linguistics, 2015.

M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

R. Zhang, H. Lee, and D. R. Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1512–1521. Association for Computational Linguistics, 2016a.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263. Asian Federation of Natural Language Processing, 2017.

Y. Zhang, S. Roller, and B. C. Wallace. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527. Association for Computational Linguistics, 2016b. doi: 10.18653/v1/N16-1178.

J. Zhao, L. Dong, J. Wu, and K. Xu. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1528–1531. ACM, 2012.

Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of ACL*, pages 654–663. ACL, 2018.

X. Zhou, X. Wan, and J. Xiao. Representation learning for aspect category detection in online reviews. In *AAAI Conference on Artificial Intelligence*, 2015.

Publications and Awards

Journals

[1] Nguyen Tien Huy, and Nguyen Le Minh. "Multilingual opinion mining on YouTube A convolutional N-gram BiLSTM word embedding." Information Processing & Management, Volume 54 Issue 3, pp. 451-462, May 2018.

[2] Nguyen Tien Huy, and Nguyen Le Minh. "An Ensemble Method with Sentiment Features and Clustering Support." Neurocomputing, May 2019.

[3] Nguyen Tien Huy, and Nguyen Le Minh. "Sentence Modeling via Multiple Word Embeddings and Multi-level Comparison for Semantic Textual Similarity." Information Processing & Management, July 2019.

Conference papers

[4] Vo Hoang Quan, Nguyen Tien Huy, Le Hoai Bac, and Nguyen Le Minh. "Multi-channel LSTM-CNN model for Vietnamese sentiment analysis." Knowledge and System Engineering (KSE) 9th International Conference on IEEE, pp 24-29, Oct 2017.

[5] Nguyen Tien Huy, and Nguyen Le Minh. "An Ensemble Method with Sentiment Features and Clustering Support." Proceeding of the Eighth International Joint Conference on Natural Language Processing, Vol.1: Long Papers, pp 644653, Nov 2017.

[6] Minh-Tien Nguyen, Dac Lai Viet, Nguyen Tien Huy and Nguyen Le Minh. "TSix: A Human-involved-creation Dataset for Tweet Summarization." The Eleventh International Conference on Language Resources and Evaluation, pp 3204-3208, May 2018.

[7] Nguyen Tien Huy, Vo Hoang Quan, and Nguyen Le Minh. "A Deep Learning Study of Aspect Similarity Recognition." Knowledge and System Engineering (KSE) 10th International Conference on IEEE, pp 181-186, Oct 2018.

[8] Nguyen Tien Huy, Le Thanh Tung, and Nguyen Le Minh. "Opinions Summarization: Aspect Similarity Recognition Relaxes The Constraint of Predefined Aspects." Recent Advances in Natural Language Processing (RANLP) 2019.