Sentiment Induction for Attention and Lexicon Regularized Neural Sentiment Analysis

Improving Aspect-based Neural Sentiment Analysis with Lexicon Enhancement, Attention Regularization and Sentiment Induction

Lingxian Bao

TESI DOCTORAL UPF / year 2021

THESIS SUPERVISOR
Patrik Lambert, Toni Badia
Department of Translation and Language Sciences

Dedication

This work is dedicated to my grandfathers: Maoheng Bao (鲍茂恒) and Weijie Zhang(张维杰); my grandmothers: Xiuzhen He (何秀珍) and Dazhen Qin (秦大珍); my parents: Hu Bao (鲍虎) and Xinhong Zhang (张鑫红); and my beloved wife: Jingyi Han (韩静怡).

Acknowledgements

What a ride! Six years of working and researching in parallel is finally coming to a closure. The decision of enrolling a PhD program has definitely changed my life. Coming from a linguistic background and trying to make something computational work might seem impossible at the beginning. Without the kind help and support that I have received from so many, it would not have been possible. Here I would like to express my most heartfelt gratitude to the following people.

First of all, a huge thank you to Patrik Lambert and Toni Badia who accepted me as a PhD candidate in the first place, and being such great advisors over the years. Your guidance, patience and encouragement really helped me going through this journey.

A special thank you to Juan Camilo Dorado for giving me a hand at the lowest point of my life, and connecting me with the wonderful people of ToolsGroup. Thank you Yossi Shamir and Eugenio Cornacchia for offering me a job at ToolsGroup so I can work and study at the same time.

Another special thank you goes to Pietro Peterlongo, a great colleague, mentor, and friend. I have learned a great deal from you not just on the technical side, your passion, dedication and knowledge made me a better version of myself.

Last but not least, I would also like to thank my parents for raising me and supporting me unconditionally. And my loving wife Jingyi Han, who inspired me and encouraged me to step onto the path of NLP. Thank you for always believing in me and supporting me, I am truly gratefully for having you in my life.

Abstract

Deep neural networks as an end-to-end approach lacks flexibility and robustness from an application point of view, as one cannot easily adjust the network to fix an obvious problem, especially when new training data is not available: e.g. when the model consistently predicts *positive* when seeing the word "*disappointed*". Meanwhile, it is less stressed that the attention mechanism is likely to "over-focus" on particular parts of a sentence, while ignoring positions which provide key information for judging the polarity. In this thesis, we describe a simple yet effective approach to leverage lexicon information so that the model becomes more flexible and robust. We also explore the effect of regularizing attention vectors to allow the network to have a broader "focus" on the input sequence. Moreover, we try to further improve the proposed lexicon enhanced neural sentiment analysis system by applying sentiment domain adaptation.

Resumen

Las redes neuronales profundas como enfoque integral carecen de flexibilidad y robustez desde el punto de vista de la aplicación, ya que no se puede ajustar fácilmente la red para solucionar un problema evidente, especialmente cuando no se dispone de nuevos datos de entrenamiento: por ejemplo, cuando el modelo predice sistemáticamente positivo al ver la palabra ”decepcionado”. Por otro lado, se hace menos hincapié en que es probable que el mecanismo de atención “se concentre demasiado” en partes concretas de una oración, mientras ignora posiciones que proporcionan información clave para juzgar la polaridad. En esta tesis, describimos un enfoque sencillo pero eficaz para aprovechar la información del léxico de modo que el modelo sea más flexible y robusto. También exploramos el efecto de regularizar los vectores de atención para permitir que la red tenga un ”enfoque” más amplio en la secuencia de entrada. Además, tratamos de mejorar aún más el sistema que proponemos de análisis profundo de sentimiento con el soporte de léxico aplicando sobre el mismo la adaptación del análisis de sentimiento al dominio.

摘要

深度神经网络作为一种端到端的方法，从应用的角度来看缺乏灵活性和鲁棒性。例如，当模型看到词语“失望”却始终预测正值时，在没有新的训练数据的情况下，很难轻易地通过调整模型来解决问题。另外，常用的注意力机制可能会“过度关注”句子的某些特定部分，从而忽略能提供判断极性关键信息的位置；此情况在业界鲜有提及。在本文中，我们描述一种简单却行之有效的方法将词典信息与深度神经网络相结合，从而改进模型的灵活性及鲁棒性。我们亦探索通过正则化注意力向量来抑制注意力机制“过度关注”的问题。此外，我们尝试通过应用情感域自适应来进一步改进所提出的词典增强型神经情感分析系统。

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Natural Language Processing (NLP) is the field of study that aims to have computers understand human languages, where machines are capable of processing and analyzing natural language data. After decades of countless efforts that could date back to the 1950s (A M Turing, 1950), many NLP applications have made significant breakthroughs in recent years and have quietly become essential parts of our daily lives. Virtual assistants such as *Siri* leverage speech recognition and many other NLP techniques to create an anthropomorphic interaction experience; machine translation helps people to communicate in different languages with unprecedented translation quality; language models comfortably complete your unfinished sentences and even suggest better expressions.

These breakthroughs are fundamentally based on two factors: the increasing size of digitized data and the growth of computing power. As human society enters the Third Industrial Revolution (also known as the Digital Revolution), data such as text, image, audio and video has been stored in binary and thus paved the way for computers to process. Meanwhile, the growth of computer performance has made it possible to process this huge amount of data at scale and unleash the true power of modern algorithms such as deep neural networks (DNN, also known as deep learning).

Within the realm of NLP, one task named sentiment analysis has become more and more important due to its value to both business and society. It consists of automatically extracting opinions (polarities such as *positive*, *neutral* and *negative*) expressed in natural languages; for instance, one should extract *positive* from the sentence: "*I'm in love with this place!*". However, as opinion expressed by words is highly context dependent (e.g. "*This camera has a **long** battery life.*" vs "*This camera takes **long** to focus.*"); and opposite polarities can be expressed in the same sentence (e.g. "*The food is **great** but the service is **awful**.*"); there is thus the need to perform sentiment analysis at a more fine-grained level: aspect level.

Sentiment analysis at aspect level, also known as aspect-based sentiment analysis (ABSA), consists of extracting the opinion associated with a predefined aspect in a sentence. For instance, consider the earlier example: "*The food is **great** but the service is **awful**.*", ABSA will extract *positive* for the aspect *food* and *negative* for the aspect *service*. In the case of a restaurant review domain, aspects are usually defined as *food*, *price*, *service*, *ambience* and *miscellaneous*.

Similar to other NLP tasks, ABSA leverages deep learning to achieve the state of the art performance. However, as an end-to-end approach, DNN are considered to lack robustness and flexibility as one cannot easily adjust the network to fix an obvious problem: when the network always predicts *positive* when seeing the word "*disappointed*", or when the network is not able to recognize the word "*dungeon*" as an indication of *negative* polarity. It could be even trickier in a low-resource scenario where more labeled training data is simply not available.

In this thesis, focusing on ABSA, we start from searching for an efficient way to bridge DNN and existing language resources (sentiment lexicons) for a more robust and adaptive model architecture. Along the way, we find that the commonly used attention mechanism is likely to over-fit and force the network to "focus" too much on a particular part of a sentence, while ignoring key positions for judging the polarity. Moreover, we also explore the possibility of further improving the lexicon enhanced neu-

ral system through domain specific sentiment induction. Before jumping into the details, let us start by reviewing the sentiment analysis task.

1.1 Sentiment Analysis

Sentiment analysis (also known as *sentiment classification* or *opinion mining*) has flourished as one of the most active fields in NLP since the beginning of the 21st century due to its important value to both business and society. It is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes (Liu, 2012).

Opinions have always been an important part in all human activities, they are key influencers of our behaviors as our beliefs and perceptions of reality are conditioned at a certain level upon how others see and evaluate the world. As a result, opinions affect us deeply considering the fact that people tend to seek opinions from others when it comes to decision making. On the other hand, as the human society develops, for the first time in history, information such as blogs, reviews, forums and social media give us a huge amount of opinionated data in digital form ready to be explored.

Early researches of sentiment analysis date back to 1999, when researchers began to realise the value of this field (Wiebe et al., 1999; Wiebe, 2000; Turney, 2002; Pang et al., 2002). However, the terms of sentiment analysis and opinion mining did not come out until later in 2003 when Nasukawa and Yi (2003) and Dave et al. (2003) first adopted them in their works. As of today, the sentiment analysis task itself is generally categorized into three different levels based on the granularity: document level, sentence level and aspect level.

At document level, early works by Pang et al. (2002) and Turney (2002) tackle the problem by classifying whether a given document is expressing a *positive* or *negative* opinion. For instance, a blog post or a full descrip-

tion of a product review with multiple sentences is considered a document; and the task is to extract an overall opinion of that document. One might already think of a problem for the document level classification, which is the fact that multiple opinions can be expressed in the same document; thus the difficulty of extracting an overall opinion at document level increases as the content length grows. Hence, it is necessary to push the granularity to a lower level.

As shown in works by Wiebe et al. (1999) and Wilson et al. (2004), sentence level sentiment classification aims to classify sentences as *positive*, *negative* or *neutral* (non subjective expressions that do not contain any sentiment). However, opinion expressed by words is highly context dependent (e.g. "*A **cheaper** price should not equal a **cheap** product.*"); and opposite polarities can be expressed in the same sentence (e.g. "*Although the service is not that great, I still love this restaurant.*"). It is clear that both document level and sentence level sentiment classification are not sufficient to cover the variety and complexity of the way human or natural language expresses people's opinions; thus it is needed to push the granularity to an even lower level: aspect level.

1.2 Aspect-based Sentiment Analysis (ABSA)

Sentiment analysis on aspect level, also known as aspect-based sentiment analysis (ABSA), was first proposed by Hu and Liu (2004a) as feature-based opinion mining. Formally, it aims to extract and summarize opinion expressed on entities and aspects of entities; for instance, in the sentence "*The Apple Care has never failed me.*", the system should extract *positive* sentiment expressed on the aspect *service* of the entity *Apple*. Thus, the task can be broken down into three sub-tasks: entity extraction, aspect detection and aspect-based sentiment classification.

Normally, the first and second sub-tasks: entity extraction and aspect detection can be treated as an information extraction task, which is very

similar to another well studied NLP problem: named entity recognition (NER). It consists of identifying and grouping textual elements into predefined categories (e.g. *person*, *organization*, *location* and *time*); in other words, extract structured information from unstructured data. In the case of ABSA, the system should identify entities (e.g. *McDonald's*, *Apple*, etc.) and aspects (e.g. *food*, *service*, *lens*, *battery*, etc.) from unstructured corpora. As for the third task: aspect-based sentiment classification, the previously extracted aspect and the relevant text data will be given to a classifier that decides whether *positive*, *neutral* or *negative* opinion is expressed towards that aspect.

Generally speaking, the three sub-tasks of ABSA are often tackled by different models, which require completely different strategies; thus it is common to separate these tasks and study them individually. In this thesis, our main focus will be on the third sub-task: aspect-based sentiment classification. As an individual task, we will have predefined aspects and opinion text as joint inputs to predict the polarity of the given aspect-text pairs.

In recent years, DNN have been adopted extensively for tackling almost all NLP tasks as they yield significant improvement across a variety of tasks compared to previous state of the art methods. For instance the encoder-decoder structure (Cho et al., 2014) and the pure attention model (Vaswani et al., 2017) for neural machine translation; recurrent neural networks (RNN) and convolutional neural networks (CNN) for sentiment analysis (Tai et al., 2015; Kim, 2014). A number of approaches that leverage both attention mechanisms and RNN or CNN architectures have also been proposed for sentiment analysis on different levels (Wang et al., 2016c; Shin et al., 2017).

However, as an end-to-end approach, deep learning based systems lack flexibility as one cannot easily adjust the network to fix an obvious problem: e.g. in the ABSA case, when the network always predicts *positive* when seeing the word *disappointed*, or when the network is not able to recognize the word *dungeon* as an indication of *negative* polarity. It could

be even trickier to fix this issue in a low-resource scenario where more labeled training data is simply not available (Bao et al., 2019).

1.3 Attention Mechanism

The idea of allowing the model to "look" back at the input sequence and "focus" on different parts accordingly has been one of the most influential ideas in the NLP world in recent times. Through the years, it has inspired many great innovations such as Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019).

Briefly speaking, the attention mechanism consists of learning a weight vector in the model and apply it to obtain a weighted representation of the input. The attention mechanism was first successfully applied in the sequence-to-sequence model for machine translation in order to overcome poor performance when translating long sentences (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015); later similar idea has been widely adopted across different fields including computer vision (Guan et al., 2018; Ramachandran et al., 2019).

As an active field of NLP, the idea of attention arrives to ABSA rather quickly as Wang et al. (2016c) introduced AT-LSTM with the state of the art performance on the SemEval14 dataset. Since in ABSA the model is expected to extract opinion from the same input sentence according to different given aspects, it makes perfect sense to allow the model to look at the input sequence differently given different aspects and being able to "highlight" relevant parts when predicting. However, it is less stressed that the commonly used attention mechanism is likely to over-fit and force the network to "focus" too much on a particular part of a sentence, while in some cases ignoring positions which provide key information for judging the polarity. In recent studies, Niculae and Blondel (2017); Zhang et al. (2019a) proposed approaches to incentivize the sparsity in the attention vector; however, it would only encourage the over-fitting effect in such

scenarios, especially when attention is applied early in the model.

1.4 Sentiment Lexicon

An obvious way that could help the model to distinguish *positive* and *negative* words is leveraging existing language resources: sentiment lexicons. A sentiment lexicon, sometimes known as opinion lexicon, is a thesaurus with word-polarity pairs where each opinion word has an associated polarity notation (e.g. numerical values within the range of $[-1, +1]$; or labels such as *positive* or *negative*).

Over the years, there have been numerous approaches proposed by researchers for compiling and inducing sentiment lexicons (Turney, 2002; Turney and Littman, 2003; Hu and Liu, 2004a; Mohammad et al., 2009; Hamilton et al., 2016; Mudinas et al., 2018). Thanks to their contributions, with a great number of sentiment lexicons freely available now, a lot of works have been done focusing on leveraging existing sentiment lexicons to enhance the performance of neural sentiment classifiers; however, most works are performed at document level and sentence level; and they usually require some sort of transformation on the lexicon inputs instead of directly taking polarities as features (Teng et al., 2016; Zou et al., 2018; Shin et al., 2017; Lei et al., 2018; Wu et al., 2018).

Nevertheless, one of the biggest limitations of a sentiment lexicon is that the polarity of an opinion word is domain dependent and context dependent. For example, under a general context, "*fallout*" and "*excel*" carry *negative* and *positive* sentiments respectively; but in the electronics or the laptop review domain, "*fallout*" or "*excel*" are both *neutral* proper nouns referring to a video game and a software. Additionally, in some cases, the same word may carry opposite sentiments in the same domain under different contexts, and this is common in ABSA: for instance, "*cheap*" is *positive* when describing the aspect *price* but it is definitely *negative* when describing the aspect *quality*. Thus it is necessary to not only enable models to

leverage sentiment lexicons, but also adapt lexicons according to different domains and aspects.

1.5 Research Questions

1.5.1 Lexicon Enhancement

As previously described in Section 1.2, sometimes when the size of the training data is limited, the deep learning based ABSA system lacks robustness and flexibility; thus our research starts by asking the question: **how to improve a neural ABSA system by using sentiment lexicons?** The advantage of using sentiment lexicons here is: first, as freely available language resources, it requires no extra efforts for feature engineering; second, by having a secondary input, the model should learn to leverage the information provided by the lexicon; compared to pure end-to-end approaches, a lexicon is easier to be maintained: for instance, the polarities of opinion words can be added, removed or updated accordingly, so that the model becomes overall more robust. On the other hand, compared to existing approaches of merging lexicon with DNN models, **is there a simpler yet effective way of doing it?**

1.5.2 Attention Over-fit

As mentioned in Section 1.3, we believe it is possible that the commonly used attention mechanism could over-fit by being too sparse, and this extreme sparsity in the attention vector could hurt the model by "over focusing" in particular parts of the sentence and thus "blinding" the model on key positions for polarity judgement. Hence we ask the question: **can attention over-fit?** And when it does, **how do we overcome it?**

1.5.3 Sentiment Induction

Moreover, as discussed in Section 1.4, although sentiment lexicons can directly provide polarity information of opinion words to the model, it is true that the polarity of an opinion word is both domain and context dependent. Thus we wonder: **is it possible to further improve the model through sentiment induction** (domain adaptation and fine-grained aspect adaptation)**?**

1.6 Research Overview

1.6.1 ATLX

In this thesis, to address the research questions in Section 1.5.1, we aim to search for a simple yet effective approach to merge lexicon information with an attention LSTM model (AT-LSTM by Wang et al. (2016c)) for ABSA in order to leverage both the power of deep neural networks and existing linguistic resources, so that the framework becomes more flexible and robust without requiring additional labeled data.

We start by replicating the AT-LSTM model as our baseline system on the SemEval 2014 Task 4, restaurant domain dataset. Later, we design and experiment different approaches to effectively merge sentiment lexicons with the baseline model. One of the approaches, which we name ATLX yields notable improvement, while requiring less complexity in terms of model architecture and feature engineering. It consists of applying the attention vector α on the lexical features given directly by the sentiment lexicon as numerical values, and the fusion of all weighted representations is connected to the final output layer for prediction. We later validate the same approach on the SemEval 2015 Task 12, laptop domain dataset. A similar performance improvement is observed compared to the baseline.

In Chapter 5, the baseline AT-LSTM model is explained in Section 5.1, followed by the details of the ATLX model in Section 5.2. The related

experiments are discussed in Section 6.1 of Chapter 6.

1.6.2 Attention Regularization

Regarding the over-fitting problem of the attention vector mentioned in Section 1.5.2, we do observe evidence that the attention vector could be over-fitting by being too sparse. Meanwhile, we also observe in ATLX that by applying the attention vector on the lexical features, it is able to reduce the over-fitting effect of the attention mechanism. Namely, the standard deviation of the attention weights distribution for the test set in ATLX (0.0219) is significantly lower than in the baseline (0.0354).

Following this idea, we explore the effect of regularizing attention vectors by introducing an attention regularization term in the loss function to allow the network to have a broader "focus" on different parts of the sentence. We design and experiment two regularizers: a standard deviation regularizer and a negative entropy regularizer. Experimental results suggest that both regularizers are able to improve the baseline, where the negative entropy regularizer yields the largest improvement.

Details of the regularization method are described in Section 5.3 of Chapter 5. Experiment details are discussed in Section 6.2 of Chapter 6.

1.6.3 Domain & Aspect Adaptation

As mentioned in Section 1.4 and Section 1.5.3, domain and context dependent problems can be a major obstacle for sentiment analysis. Thus we are interested to see whether it is possible to further improve the ATLX system with a more fine-grained lexicon.

To do that, we adopt one of the state of the art sentiment domain adaptation methods, the one by Mudinas et al. (2018), which consists of a word vector based semi supervised approach; and apply it to convert the general lexicon constructed for ATLX to a domain specific one of electronics reviews. We then compare the performance gain of the general lexicon,

the domain specific lexicon, and a gold lexicon for laptop reviews labeled by ourselves by applying them in the ATLX system on the SemEval 2015 Task 12, laptop domain dataset.

As a result, we find that in general, domain-specific lexicons do improve the model performance compared to a generic one; however, the performance ceiling suggested by the gold lexicon is rather low. Moreover, as most domain adaptation works are done by recreating an existing domain specific lexicon and *neutral* words are often ignored, we find that the role of *neutral* words are rather important when applying the adapted lexicon in the model.

In addition, we also intend to create a fine-grained aspect specific sentiment lexicon with a similar approach. However, no performance improvement can be achieved. We believe the main reason for that is that not enough seed examples can be constructed easily to reflect the aspect dependent cases (e.g. "***cheap*** *price*" vs "***cheap*** *quality*"). On the other hand, the number of cases which is related to the aspect dependent issue is rather low, so that it may be difficult to reflect on the evaluation metrics.

Details of domain adaptation approach are described in Section 5.4 of Chapter 5. Experiment results are discussed in Section 6.3 of Chapter 6.

1.6.4 Contribution

In this thesis, we propose a novel method named ATLX for aspect level sentiment analysis that leverages an attention LSTM neural network and sentiment lexicons to improve the robustness and flexibility of the model. The proposed method is experimented on the SemEval 2014 Task 4, restaurant domain dataset. A paper that describes this method and its related experiments has been published in Bao et al. (2019).

On the other hand, we expose the fact that the attention mechanism could over-fit and hurt the model when applied early in the network; and two regularizers are proposed to validate and overcome this effect. Details of the approaches have also been described in the paper Bao et al. (2019).

An additional batch of experiments on a different dataset (SemEval 2015 Task 15, laptop domain) that validate the ATLX model and the attention regularization methods are carried out as well. A paper that describes the details of the experiments and extended analysis has been submitted to the *Language Resource and Evaluation Journal*.

Moreover, we apply a vector-based domain adaptation method for sentiment induction and evaluate the performance gain of a domain specific lexicon in the ATLX system. Compared to the commonly used evaluation method for sentiment domain adaptation that recreates an existing domain-specific lexicon, we find that a good evaluation score on the lexicon evaluation does not necessarily translate to a performance improvement when applied in the ATLX model.

Chapter 2

LITERATURE REVIEW

Sentiment Analysis as a valuable NLP field has been extensively studied in the past decades. In this chapter, we will review the formal definition of sentiment analysis and take a deeper look at sentiment analysis at different levels with their advantages and disadvantages. We will also review the common methods proposed by researchers along the years. Meanwhile, we will also cover common methods for sentiment lexicon generation and sentiment domain adaptation, as they are important aspects of this thesis. On the other hand, existing methods that integrate sentiment lexicons with deep neural networks will be mentioned. Finally, we will discuss the works on attention regularization that try to shape the attention vector to be more sparse.

As *Machine Learning (ML)* methods are extensively adopted in the NLP field, having some knowledge on ML could be helpful for navigating in this chapter. The commonly used methods of both classical machine learning and deep learning are briefly introduced in the Chapter 4, with a focus of the methods that are directly related to our research. For readers that are not familiar with ML, it is recommended to read Chapter 4 first.

2.1 Sentiment Analysis (SA)

Opinions have always been an important part in all human activities and key influencers of our behaviors as our beliefs and perceptions of reality and the choices we make are conditioned at a certain level upon how others see and evaluate the world. As a result, opinions affect us deeply considering the fact that people tend to seek opinions from others when it comes to decision making. In an era of social media and connectivity, the size of data has been growing exponentially as web users are becoming increasingly enthusiastic about interacting, sharing, and working together through online collaborative media (Cambria, 2016). In order to effectively process these data and extract valuable insights from them, there thus has been the need of an automated method to extract opinions at scale: sentiment analysis.

Sentiment analysis has many different names: e.g. *opinion mining*, *opinion extraction*, *sentiment classification*, *sentiment mining*, *subjectivity analysis*, *affect analysis*, *emotion analysis*, *review mining*, etc. It is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes (Liu, 2012). As the meaning of opinion itself is still very broad, sentiment analysis and opinion mining mainly focuses on opinions that express or imply *positive*, *neutral* or *negative* sentiments.

Early researches of sentiment analysis date back to the beginning of the 21st century, when researchers began to realise the value of this field (Wiebe et al., 1999; Wiebe, 2000; Turney, 2002; Pang et al., 2002; Morinaga et al., 2002; Das et al., 2004). However, the terms of sentiment analysis and opinion mining did not come out until later in 2003 when Nasukawa and Yi (2003) and Dave et al. (2003) first adopted them in their works. Before the year 2000, unlike fields such as linguistics or other NLP tasks, there have been few researches conducted in the field of sentiment analysis. The main reason for that is that the industries that value

sentiment analysis also started to flourish in the 21st century, for instance, e-commerce giants such as *Amazon*, *eBay*, *Alibaba*, etc. On the other hand, as the center of online contents has been shifting from websites, blogs and forms to micro-blogs and social media, sentiment analysis also contributes dearly to researches such as finance (Pagolu et al., 2017), social science (Zhou et al., 2013) and political science (Caetano et al., 2018).

2.1.1 Applications

According to Liu (2012), opinions are central to almost all human activities because they are key influencers of our behaviors, as whenever we need to make a decision, we want to know others' opinions. In the real world, businesses and organizations always want to find consumer or public opinions about their products and services. Individual consumers also want to know the opinions of existing users of a product before purchasing it, and others' opinions about political candidates before making a voting decision in a political election. In the past, when an individual needed opinions, he/she asked friends and family. When an organization or a business needed public or consumer opinions, it conducted surveys, opinion polls, and focus groups. Acquiring public and consumer opinions has long been a huge business itself for marketing, public relations, and political campaign companies.

With the explosive growth of social media (e.g., reviews, forum discussions, blogs, micro-blogs, twitter, comments and posts in social networks) on the internet, individuals and organizations are increasingly using the content in these media for decision making. Nowadays, if one wants to buy a consumer product, it is no longer limited to asking one's friends and family for opinions because there are already many user reviews and discussions about the product existing on the internet. For an organization, it may no longer be necessary to conduct surveys, opinion polls, and focus groups in order to gather public opinions because there is an abundance of such information publicly available. However, finding and monitor-

ing opinion sites on the internet and distilling the information contained in them remains a formidable task because of the proliferation of diverse sites. Each site typically contains a huge volume of opinion text that is not always easily deciphered in long blogs and forum postings. The average human reader will have difficulty identifying relevant sites and extracting and summarizing the opinions in them. On the other hand, many organizations also have internal data, e.g. customer feedback collected from emails and call centers or results from surveys conducted by the organizations. Thus automated sentiment analysis systems are needed.

Due to the high demand from businesses and organizations for an automated sentiment analysis system, in recent years, besides the commonly seen review summarization function on e-commerce websites such as *Amazon* or applications such as *Google Maps*; there have been a great number of sentiment analysis tool vendors ranging from start-ups to tech giants. For example, in business intelligence, there is usually the need of brand monitoring or social listening, which consists of a sentiment analysis system deployed on real-time data flow; marketers use tools such as *Awario*, *Brandwatch*, *HubSpot*, *Hootsuite*, *Lexalytics*, etc. to monitor the public opinion of their brand or product in real time. More recently, cloud computing platforms such as *Microsoft Azure*, *Google Cloud*, *Amazon Web Service*, etc. have been providing sentiment analysis as a SaaS (software as a service) service within their AI and NLP functionalities, which allows users to consume their trained models through web API in order to build more customized solutions.

From an applied research's perspective, along the years there have been many application oriented research papers published; for instance, Liu et al. (2007) proposed a sentiment-aware model for predicting sales performance using blogs; McGlohon et al. (2010) uses reviews to rank products and merchants in order to measure their quality; Hong and Skiena (2010) studied the relationship between the NFL betting line and public opinions expressed in blogs and micro-blogs (twitter). On political science, in O'Connor et al. (2010), public opinion measured from polls are

connected with sentiment measured from text; in Tumasjan et al. (2010), twitter sentiment is studied with the hypothesis of mirroring offline political sentiment; in Chen et al. (2010), political standpoints were studied using an opinion scoring model; in Yano and Smith (2010), a method was proposed to model the relationship between the text of a political blog post and its comment volume. When it comes to social matters, in some works (Asur and Huberman, 2010; Sadikov et al., 2009; Joshi et al., 2010), twitter data, movie reviews and blogs were used to predict the success and revenues of movies; Miller et al. (2011) investigated sentiment flow in social networks; Sakunkoo and Sakunkoo (2009) studied social influences in online book reviews; Groh and Hauffa (2011), used sentiment analysis to characterize social relations; Mohammad and Yang (2011) studied sentiments in emails to find gender differences on emotional axes; and later, emotions in novels and fairy tales were studied as well (Mohammad, 2012). Regarding finance, in Bollen et al. (2011) twitter moods were adopted to predict the stock market; in Bar-Haim et al. (2011); Feldman et al. (2011), expert investors in micro-blogs were identified and sentiment analysis of stocks was performed; in Zhang and Skiena (2010), blog and news sentiment was used to study trading strategies.

More recently, Ceron et al. (2014) used sentiment analysis to study the online popularity of Italian political leaders throughout 2011 and the voting intention of French users in both the 2012 presidential ballot and the subsequent legislative election. Fan et al. (2017) proposed a novel method that combines time series forecasting and sentiment analysis that uses historical sales data and online reviews product sales forecasting. Ren et al. (2019) studied the investor psychology reflected by online textual data as an indicator of stock market. Ilyas et al. (2020) evaluated the public sentiment and opinion on Brexit during September and October 2019 using 16 million tweets by quantifying daily public sentiment towards Brexit and using it to evaluate Brexit's impact on the British currency exchange rate and stock markets in Britain. Pokharel (2020) studied the sentiment on Twitter during the COVID-19 outbreak. Muthukumar and Zhong (2021)

proposed a novel deep learning model called ST-GAN (*Stochastic Time-series Generative Adversarial Network*), that analyzes both financial news texts and financial numerical data to predict stock trends.

2.1.2 Problem Definition

As briefly mentioned in Section 2.1, sentiment analysis is the field of study that analyses people's opinion; however, opinion has a very broad meaning, it can mean people's sentiments, evaluations, appraisals, attitudes and emotions. In NLP as the problem definition, sentiment analysis is the study of opinions that express or imply *positive*, *neutral* or *negative* sentiments expressed in natural languages. In order to fully capture the complexity and inner connections between sub-problems that exist within the problem itself, it is necessary to abstract a structure from the complex unstructured natural language text.

Opinion Definition

Taken from Liu (2012), we use the following example of a *Canon* camera review to illustrate the key components of an opinion. Although it is a review from almost 10 years ago, the fact that nowadays cameras still uses battery, picture quality still matters, and people still buy cameras online and leave reviews on websites; the example still feels appropriate for defining what is an opinion:

Posted by *John Smith* on *September 10th, 2011*: "(1) *I bought a Canon G12 camera six months ago.* (2) *I simply love it.* (3) *The picture quality is amazing.* (4) *The battery life is also long.* (5) *However, my wife thinks it is too heavy for her.*"

In this piece of review document, we can observe a few things:

1. **Sentiment** (s): there are multiple opinions expressed, both *positive*, *neutral* and *negative*. E.g. in sentence (2), there is a general *positive* opinion on the *Canon G12* camera as a whole; sentence (3) expresses a *positive* opinion on the picture quality; sentence (4) expresses a *positive* opinion on the battery life. On the other hand, sentence (5) expresses a *negative* opinion on the weight of the camera; and sentence (1) is considered *neutral* as it simply states a fact and no opinion is expressed there.

2. **Holder** (h): the opinions in this review document come from two different persons, which are also know as *opinion sources* or *opinion holders* (Kim and Hovy, 2004; Wiebe et al., 2005). In sentences (2), (3) and (4), the opinions come from the author of the review (*John Smith*); while in the sentence (5), it comes from the author's wife.

3. **Time** (t): there is a timestamp associated with the review document: *September 10th, 2011*. The time factor is important as the expressed opinion may change over time; and in practice, one often likes to study how opinion shifts through time.

4. **Entity** (e): there is a target for every sentiment expressed, however, the targets could be different even in the same review document. For instance, in sentence (2) the target is *Canon G12*, while in sentence (3) and (4), the targets are *picture quality* and *battery life* respectively. It is important to understand that in fact the target in sentence (3) should be *picture quality of Canon G12*, and the target in sentence (4) should be *battery life of Canon G12*; otherwise, without knowing the sentences are evaluating attributes of the *Canon G12* camera, the opinions alone in sentence (3) and (4) are of little use.

   Thus, let us use the term ***entity*** to denote the target object that has been evaluated. In practice, the target can often be decomposed and described in a structured manner with multiple levels, which greatly

facilitate both mining of opinions and later use of the mined opinion results. For example, *picture quality of Canon G12* can be decomposed into an entity and an attribute of the entity and represented as a pair: *(Canon-G12, picture-quality)*.

5. **Aspect** (a): typically, under the context of a product or service review, the entity and its attributes form a hierarchical relation and opinions are usually expressed either on a specific attribute of the entity or on the entity as a whole. For instance, in our example, sentence (2) expresses a *positive* opinion on the entity *Canon G12* as a whole; sentences (3), (4) and (5) each express opinion on one attribute of the entity, namely *picture quality*, *battery life* and *weight* respectively. To capture this tree structured relation, let us use the term ***aspect*** to denote each of the leafs of an ***entity*** node. For instance, *Canon G12* is the node entity with leaf aspects such as: *picture quality*, *battery life* and *weight*.

Thus, given the 5 observed points, we can now define an **opinion** as a ***quintuple***:

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$$

where e_i is the name of an entity, a_{ij} is an aspect of e_i, s_{ijkl} is the sentiment expressed by opinion holder h_k at time t_l on the aspect a_{ij} of entity e_i. The sentiment s_{ijkl} is *positive*, *neutral* or *negative*, or is expressed with different strength/intensity levels, e.g. 1 to 5 stars as used by most review websites. When an opinion is on the entity itself as a whole, the special aspect *GENERAL* is used to denote it. Here, e_i and a_{ij} together represent the opinion target.

In this definition, all five components are essential, as missing any of them will be problematic for sentiment analysis; meanwhile, as indicated by the subscripts, all five components must have a strict and corresponding relation with one another. For instance, if the time component is missing, it will be impossible to analyze opinions on an entity with respect to time,

which is often very important in practice as an opinion on something two years ago might not be the same as the one of today. In addition, knowing an opinion without knowing its opinion holder is hard to have any value; for instance, consider the sentence "*The mayor is loved by the people in the city, but he has been criticized by the state government.*", it is clear that knowing the opinion holders ("*people in the city*" and "*state government*") in this sentence is crucial for applications. Thus, the definition provides a framework to transform unstructured text to structured data; and in practice, the quintuple above is often adopted as a database schema, based on which the extracted opinions can be put into a database table, from which a rich set of qualitative, quantitative, and trend analyses of opinions can be performed.

However, this definition is not perfect as it covers most but not all possible faces of the semantic meaning of an opinion, which can be arbitrarily complex in natural languages. For instance, it does not cover the situation in "*The viewfinder and the lens are too close.*", which expresses an opinion on the distance of two parts; it also does not cover the context of the opinion, e.g. "*This car is too small for a tall person.*", which does not say the car is too small for everyone. In some other cases, the quintuple framework may not be able to catch nested relations between entities and aspects; for instance, in a printer review that says "*The ink of this printer is too expensive.*", *ink* can be treated as an aspect of the printer entity, but it can also be treated as a entity itself as the opinion expressed in the example is targeting *ink* as an object that is related to a printer. Despite the limitations, in practice, the quintuple definition does cover the essential information of an opinion which is sufficient for most practical applications.

Sentiment Analysis Tasks

With this definition of opinion, we can now define the objective of sentiment analysis as: **discover all opinion quintuples** $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ **in a given opinion document** d (Liu, 2010, 2011).

Naturally, the first step is to extract entities. In fact, the extraction is very similar to another well studied NLP problem: named entity recognition (NER) (Mooney and Bunescu, 2005; Hobbs, Jerry; Riloff, 2010; Ma et al., 2016; Lample et al., 2016; Li et al., 2020a), which consists of identifying and grouping textual elements into predefined categories such as *Person*, *Organization*, *Location*, *Time*, etc.; in this case, entities such as *Canon G12* camera. As it is common in natural language text that the same entity is often referred to with different expressions (e.g. *Motorola* can be written as: *Mot*, *Moto* or *Motorola*), the extracted entity expressions need to be categorized as referring to the same entity. In practice, each entity should have a unique name in a particular application. The process of grouping entity expressions into entities is also known as *entity categorization*.

Next, when identifying aspects, the same problem occurs. For instance, *picture*, *image*, and *photo* are different expressions referring to the same aspect. Thus the same approach for entity extraction and entity categorization is applied to aspects. Similarly, each aspect should have a unique name in a particular application; the process of grouping aspect expressions is called *aspect categorization*.

Later, with the extracted and categorized entities and aspects, the following task is to classify the polarity expressed on the entity-aspect pairs as *positive*, *neutral* or *negative*.

Formally, let e_i be an entity, which can be expressed by any one of a finite set of its entity expressions $\{ee_{i1}, ee_{i2}, ..., ee_{im}\}$; a_{ij} be an aspect of e_i, where a_{ij} can be expressed with any one of its finite set of aspect expressions $\{ae_{ij1}, ae_{ij2}, ..., ae_{ijn}\}$. Let d be an opinion document that contains a set of entities $\{e_1, e_2, ..., e_r\}$ and a subset of their aspects from a set of opinion holders $\{h_1, h_2, ..., h_p\}$ at some particular time point t_l.

Thus, given a set of opinion documents $D = \{d_1, d_2, ..., d_u\}$, sentiment analysis can be broken down into into 6 tasks:

1. **Entity Extraction and Categorization**: Extract all entity expres-

sions in D and categorize synonymous entity expressions into entity clusters, where each entity expression cluster corresponds to a unique entity e_i.

2. **Aspect Extraction and Categorization**: Extract all aspect expressions of the entities and categorize them into aspect clusters; each aspect expression cluster of entity e_i represents a unique aspect a_{ij}.

3. **Opinion holder extraction and categorization**: Extract opinion holders for opinions from D and categorize them (similar to previous tasks). Each opinion must have an opinion holder h_k.

4. **Time Extraction and Standardization**: Extract timestamps when opinions are given and standardize different time formats (similar to previous tasks). Each opinion must have a corresponding timestamp t_l.

5. **Aspect Sentiment Classification**: Determine whether an opinion expressed on an aspect a_{ij} is *positive*, *neutral* or *negative*, or assign a numeric polarity value to the opinion. The polarity of the opinion is denoted as s_{ijkl}.

6. **Opinion Quintuple Generation**: Generate all opinion quintuples expressed in each document $d \in D$ as $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$.

To better understand the tasks of sentiment analysis, consider the following example (Liu, 2012):

Posted by *bigJohn* on *September 15th, 2011*: "(1) *I bought a Samsung camera and my friends brought a Canon camera yesterday.* (2) *In the past week, we both used the cameras a lot.* (3) *The photos from my Samy are not that great, and the battery life is short too.* (4) *My friend was very happy with his camera and loves its picture quality.* (5) *I want a camera that can*

take good photos. (6) *I am going to return it tomorrow.*"

According to the previous definitions, **Task** 1 should extract the entity expressions: "*Samsung*", "*Samy*" and "*Canon*", and group "*Samsung*" and "*Samy*" to represent the same entity. **Task** 2 should extract the aspect expressions: "*picture*", "*photo*" and "*battery life*", and group "*picture*" and "*photo*" as the same aspect as they share the same meaning in the camera domain. **Task** 3 should identify the opinion holder in sentence (3) as the author of the post *bigJohn*, and the opinion holder in sentence (4) as *bigJohn*'s friend. **Task** 4 should find the timestamp when the post was published, namely "*2011-09-15*". **Task** 5 should determine that: sentence (3) expresses a *negative* opinion on the *picture quality* of the *Samsung* camera, and also a *negative* opinion on its *battery life*; sentence (4) gives a *positive* opinion on the *Canon* camera as a whole, and also on its *picture quality*. In practice, the rest of the sentences should be identified as *neutral* as they do not contain opinions on entities or aspects; however, for the purpose of avoiding over-complication, *neutral* cases are not considered here. Finally, **Task** 6 should generate the following opinion quintuples:

(*Samsung*, *picture_quality*, *negative*, *bigJohn*, *2011-09-15*)

(*Samsung*, *battery_life*, *negative*, *bigJohn*, *2011-09-15*)

(*Canon*, *GENERAL*, *positive*, *bigJohn's_friend*, *2011-09-15*)

(*Canon*, *picture_quality*, *positive*, *bigJohn's_friend*, *2011-09-15*)

Sentiment analysis based on these exact tasks is also called *aspect-based sentiment analysis* or *aspect level sentiment analysis* (Hu and Liu, 2004a; Liu et al., 2005). In fact there are generally three levels of sentiment analysis with increasing granularities: *document level*, *sentence level* and *aspect level*. In the following sections (Section 2.1.3, Section 2.1.4, Section 2.1.5), details of the different levels of sentiment analysis and related researches will be discussed.

2.1.3 Document Level Sentiment Analysis

According to the definition by Liu (2012), the task of document level sentiment analysis is to classify whether generally an opinion document expresses a *positive*, *neutral* or *negative* sentiment (Pang et al., 2002; Turney, 2002; Teng et al., 2016; Zou et al., 2018). For instance, an online review about a certain product can be considered as a document, thus the system needs to determine whether the review expresses an overall *positive*, *neutral* or *negative* opinion about that product. The task is also commonly known as *document-level sentiment classification* because it considers the whole document as a basic information unit.

Formally, based on the framework defined in Section 2.1.2, given an opinion document d evaluating an entity, the task of document level sentiment analysis is to determine the overall sentiment s of the opinion holder about the entity; i.e. determine the sentiment s expressed on the aspect *GENERAL* in the quintuple:

(_, *GENERAL*, *s*, _, _)

where s can be a categorical label and the task is treated as a classification problem (e.g. *positive*, *slight-positive*, *negative*, etc.); or a numerical value in a given range that indicates the intensity and polarity of the sentiment (e.g. 1 to 5); in the case of numerical value, the task is treated as a regression problem. The entity e, opinion holder h and timestamp t are assumed to be known or irrelevant; the reason for that is the fact that document level sentiment analysis is usually based on an assumption: it assumes that the opinion document d (e.g. a product review) expresses opinions on a single entity e from a single opinion holder h at a single time t.

This assumption holds true for most product/service reviews as a post is usually targeting a specific product or service and is written by a single author; thus extracting e, h and t is rather easy. However, it does not hold in the case of a forum or a blog post, as in them the author usually expresses opinions on multiple entities. Therefore, in practice, when an opinion doc-

ument evaluates more than one entity, or multiple opinion holders express different opinions in a single document, it does not make sense to apply document-level sentiment analysis to assign one sentiment orientation to the entire document.

Sentiment Classification with Supervised Learning

Sentiment classification is essentially a text classification problem. Similar but different to other text classification problems such as topic classification (e.g. *news*, *politics*, *science*, *sports*, etc.), where topic related words are important features. Sentiment classification is usually a binary (*positive*, *negative*) or ternary (*positive*, *neutral*, *negative*) classification problem, where opinion words (e.g. *good*, *bad*, *great*, *amazing*, etc.) are key features.

As a text classification problem, any existing supervised learning method can be applied, for instance, *Naïve Bayes* (Wiebe et al., 1999; Weichselbraun et al., 2013), *Support Vector Machines (SVM)* (Joachims, 1998; Scholkopf and Smola, 2018), *Maximum Entropy* (Lu et al., 2011), *K-nearest Neighbor (KNN)* (Wu et al., 2009) and *Neural Networks* (Tang et al., 2014). On the other hand, besides the general supervised learning algorithms, many works have been done exploring different linguistic features to push the boundary of sentiment analysis. For instance, *word terms and frequencies (TF-IDF)* (Martineau et al., 2008), *part of speech (POS)* (Wu et al., 2018), *sentiment lexicons* (Zhang et al., 2011) and *syntactic dependency* (Di Caro and Grella, 2013). In addition, instead of applying standard machine learning methods, Dave et al. (2003) proposed a score function based on words in positive and negative reviews; Teng et al. (2016) proposed a weighted-sum model which consists of representing the final prediction as weighted sum of model prediction and polarities provided by lexicon. Zou et al. (2018) described a framework to assign higher weights to opinion words found in lexicon by transforming lexicon polarity to sentiment degree.

Prior to the take off of deep learning, there have already been a great number of researches done in the literature which mostly combine machine learning algorithms with hand-crafted linguistic features. For instance, Pang and Lee (2004) proposed a graph based minimum cut algorithm for sentiment classification. Mullen and Collier (2004) used syntactic relations together with diverse information sources in a SVM classifier. Kennedy and Inkpen (2006) studied the role of sentiment shifters (negations, intensifiers, and diminishers) on movie reviews. Cui et al. (2006) compared passive-aggressive algorithm based classifiers (a family of margin based online learning algorithms for binary classification, e.g. an online version of SVM) with language model based classifiers using n-grams features. Ng et al. (2006) studied the role of linguistic knowledge sources in review identification and classification. Abbasi et al. (2008) proposed an entropy weighted genetic algorithm (EWGA) that incorporates information-gain heuristics for feature selection in different languages. Martineau et al. (2008) proposed a new weighted representation scheme (Delta-TFIDF) to improve sentiment analysis. Li et al. (2009) proposed a non-negative matrix factorization method. Dasgupta and Ng (2009) experimented with a semi-supervised approach for sentiment classification combining active learning and ensemble learning. Kim et al. (2009) proposed a new term weighting scheme leveraging collection statistics, topic characteristics and opinion properties. Qiu et al. (2009) proposed a self-supervised, lexicon-based and corpus-based model for sentiment classification. Nakagawa et al. (2010) proposed a dependency tree based method using *conditional random fields (CRF)* (Lafferty et al., 2001) with hidden variables. Yessenalina et al. (2010) proposed a joint two-level approach that distinguishes useful sentences from unuseful ones to help sentiment classification. Liu (2010) compared different linguistic features for both blog and review sentiment classification. Wang et al. (2011) proposed a graph based approach to classify twitter hastag sentiment. Maas et al. (2011) presented a probabilistic model that uses a mix of unsupervised and supervised techniques to learn word vectors capturing sentiment content. Bespalov et al. (2011)

proposed an efficient embedding method for projecting n-gram phrases to low-dimensional latent semantic space through a deep neural network, and applied it in sentiment classification. Glorot et al. (2011) proposed a system based on stacked *denoising autoencoder* with sparse rectifier units, which learns to extract a meaningful representation for each review in an unsupervised fashion.

In the last decades, computation power and digital data has been increasing exponentially, which enables deep neural networks such as *Long Short-term Memory (LSTM)* (Hochreiter and Schmidhuber, 1997) and *Convolutional Neural Network (CNN)* (LeCun et al., 1998) to be back under the spotlight as they yield significant improvements across a variety of tasks compared to previous state of the art methods. For instance, CNN based *ImageNet* (Krizhevsky et al., 2012) in computer vision and *Recurrent Neural Network (RNN)* based encoder-decoder structure for machine translation (Cho et al., 2014) are both considered milestones that had a great impact on both academia and industry in the recent years.

In addition, compared to traditional text representation methods such as *bag of words (BOW)* or *TF-IDF*, where each word is represented as a single dimension in a discontinuous high-dimensional space (i.e. each index of a vector represents a word where the length of the vector is equal to the vocabulary size); inspired by training language models using deep neural networks, Mikolov et al. (2013a,b) proposed a new method for learning *distributed word representations* (also known as *word embeddings* or *word vectors*) which is able to represent a word with multidimensional features. Intuitively, the meaning of a word can be different under different contexts, so that it is reasonable that distributed word representation is better than traditional representation methods as it encodes richer contextual information and forms a continuous vector space for word representations.

With distributed word representations, sentiment classification can be conducted using a variety of neural network models following the traditional supervised learning setting. In some cases, neural networks may only be used to extract text features/text representations, and these features

are fed into some other non-neural classifiers (e.g. SVM) to obtain a final global optimum classifier. The properties of neural networks and SVM complement each other in such a way that their advantages are combined (Zhang et al., 2018a).

More specifically, early works focused on improving the word representations. For example, Moraes et al. (2013) compared SVM with *artificial neural networks (ANN)* for document level sentiment analysis and demonstrated that ANN achieved competitive results compared to SVM in most cases. To overcome the drawbacks of BOW representation (losing the ordering and semantics of words), Le and Mikolov (2014) introduced *paragraph vector*, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, and experimented with sentiment analysis. Johnson and Zhang (2015) designed a new variation of CNN and tested it with sentiment classification, namely, a BOW specific convolution layer to directly learn embeddings of small text regions on the high-dimensional text representation. Meanwhile, by concatenating the one-hot vector of multiple words, the sequential information of the text is able to be captured as well. Tang et al. (2015b) incorporated user and product level information through vector space models into a neural network approach for document level sentiment classification; thus the network is capable of capturing important global clues such as individual preferences of users or overall qualities of products to boost performance.

Later, more complicated neural networks such as CNN, LSTM and attention became more popular. For instance, Tang et al. (2015a) introduced a hybrid neural network framework to learn vector-based document representation in a unified bottom-up fashion by combining sentence representations obtained from CNN or LSTM in a RNN network to adaptively encode semantics of sentences and their relations as document representation. The learned document representation is then tested on several sentiment classification corpus. Xu et al. (2016) proposed a *Cached Long Short-term Memory Neural Network (CLSTM)* to capture the overall semantic information in long texts; where a cache mechanism was introduced to divide

memory into several groups with different forgetting rates and thus enables the network to keep sentiment information better within a recurrent unit. The intuition is to enable the memory groups with low forgetting rates to capture global semantic features and the ones with high forgetting rates to learn local semantic features. Yang et al. (2016) proposed a hierarchical attention network that mirrors the hierarchical structure of documents to construct document representation by applying two levels of attention mechanisms at the word and sentence level. Zhou et al. (2016) proposed an attention-based bilingual representation learning model which learns the distributed semantics of the documents in both the source and the target languages; later a hierarchical attention mechanism for the bilingual LSTM network is used for document representation and classification. In this setting, it effectively adapts the sentiment information from a resource-rich language (English) to a resource-poor language (Chinese) and helps improve the sentiment classification performance. Chen et al. (2016) proposed a hierarchical neural network to incorporate global user and product information into sentiment classification. The model is capable of leveraging user and product information via attention over different semantic levels due to its ability of capturing crucial semantic components. Similarly, Dou (2017) proposed a deep memory network for document-level sentiment classification which could capture the user and product information at the same time; where LSTM is first applied to learn the document representation, and later, a deep memory network consisting of multiple computational layers (hops) is used to predict the review rating for each document.

Later on, some other models and techniques (e.g. autoencoder and adversarial network) that come from other machine learning practices are brought to sentiment analysis. For example, Zhai and Zhang (2016) proposed a semi-supervised autoencoder that handles scalability problems when large vocabulary size leads to high dimensionality, and deals with task-irrelevant words at the same time. The model considers sentiment information in its learning stage in order to obtain better document vectors for

sentiment classification. Li et al. (2017b) proposed an adversarial memory network for cross-domain sentiment classification in a transfer learning setting, where the data from the source and the target domain are modelled together. It jointly trains two networks for sentiment classification and domain classification. Yin et al. (2017) tackled the task of document level sentiment classification as a machine comprehension problem where pseudo question-answer pairs are constructed by a small number of aspect-related keywords and aspect ratings; a hierarchical interactive attention-based model was adopted to represent both word level and sentence level information. Rao et al. (2018) propose a new neural network model (SR-LSTM) to capture the semantic relations between sentences in document-level sentiment classification. The model is composed by two hidden layers: one layer learns sentence vectors to represent semantics of sentences with LSTM, and in the other layer, the relations of sentences are encoded in document representation.

More recently, Li et al. (2018) proposed *Hierarchical User Aspect Rating Network (HUARN)* that adopts a hierarchical architecture to encode word, sentence, and document level information, so that user preference and overall ratings are considered jointly and merged through attention mechanism for document representation. The document representation is then combined with user and overall rating information for prediction. Rhanoui et al. (2019) extended the traditional document level sentiment analysis setting from short documents to long texts (e.g. newspaper articles) and proposed a hybrid model combining CNN, Bi-LSTM (Bidirectional LSTM) and document embeddings to deal with long texts. Pröllochs et al. (2019) adopted a novel strategy for learning fully interpretable negation rules via weak supervision (i.e. reinforcement learning) to find a policy that reconstructs negation rules from sentiment predictions at document level. Barnes (2019) reported a hierarchical multi-task network that uses shared lower-layers in a deep Bi-LSTM model to predict negation, while the higher layers are dedicated to predict sentiment at document-level. The multi-task component shows promise as a way to incorporate information

on negation into deep neural sentiment classifiers. Jain et al. (2020) proposed a hybrid framework, *Senti-NSetPSO*, to analyse large-sized text data, which consists of a binary and a ternary classifier based on hybridization of *particle swarm optimization (PSO)*. Liu et al. (2020b,a) proposed a novel hierarchical neural network model based on *Dynamic Word Embeddings (HieNN-DWE)* for document level sentiment classification. The model consists of two encoders designed to capture both lower (sentence) and higher (document) level features given ELMo embeddings (Embeddings from Language Models); namely, a *Bidirectional Gated Recurrent Unit (BiGRU)* with attention mechanism for sentence encoding, and a BiGRU with CNN combination for global level representation. The multi level representations are then concatenated to make the final prediction. Wen et al. (2020) proposed a novel speculative sentiment classification model (SSC), in which the idea that users with similar rating behaviours are more likely to write documents of similar sentiments toward a product is explored. In SSC, three different components that encode user-product interaction information, document representation and aggregated speculative information are applied and integrated into a unified model.

Sentiment Classification with Unsupervised Learning

Since a labeled corpus for training a supervised sentiment analysis model is not always available, it is definitely worth the effort to develop an unsupervised approach; however, the amount of published works in the literature is far less than it is with supervised learning. The reason for that could be: first, more labeled corpus on a variety of domains had become available as more resources were invested into the field. Second, sentiment expressed by words is highly context and domain dependent so that it is impossible to achieve great performance through fixed language resources such as sentiment lexicons, which is what most of the unsupervised approaches rely on.

The approaches that rely on sediment lexicons are also known as lexicon-

based methods, which consists of using a dictionary of opinion words and phrases with their associated sentiment orientation or strength; and it incorporates intensification and negation to compute a sentiment score for each document (Taboada et al., 2011). As an important resource for many sentiment analysis tasks, details about sentiment lexicons will be discussed in Section 2.2. Similar works at different levels such as Ding et al. (2008), where *Opinion Observer* was proposed. It consists of a holistic lexicon-based approach by exploiting external evidences and linguistic conventions of natural language expressions. Based on sets of linguistic patterns and an aggregation function, the system is capable of handling opinion words that are context dependent; dealing with special words and phrases that have an impact on the polarity; and effectively aggregating multiple conflicting opinion words in a sentence. In Paltoglou and Thelwall (2012), focusing on the more common informal textual communication on the internet, such as online discussions, tweets and social network comments, they purposed an intuitive unsupervised lexicon-based approach that estimates the level of emotional intensity contained in text in order to make a prediction. Later, Hu et al. (2013) studied the problem of unsupervised sentiment analysis with emotional signals, where they investigated the possibility to help sentiment analysis by providing a unified way to introduce emotion signals to an unsupervised learning framework. Dayalani et al. (2014) proposed a simple yet unsupervised method for twitter sentiment analysis. The system is built by clustering tweets using emoticons and inferring a polarity thesaurus at the same time. Jiménez-Zafra et al. (2016) described an unsupervised approach for aspect-based sentiment analysis by adopting a lexicon-based method combining different linguistic resources and ensemble several distinct classifiers to improve the classification. The scores are then aggregated to give a document level prediction. Cheng et al. (2017) studied a novel problem of unsupervised sentiment analysis with signed social networks, in which they incorporated explicit sentiment signals in textual terms and implicit sentiment signals from signed social networks into a coherent model *SignedSenti* for unsupervised sentiment analysis.

Fernández-Gavilanes et al. (2018) proposed a novel approach to predict sentiments expressed by emojis in online textual messages (e.g. tweets) by automatically constructing a novel emoji sentiment lexicon using an unsupervised sentiment analysis system based on the definitions given by emoji creators in *Emojipedia*; additionally lexicon variants were also created automatically by considering the sentiment distribution of the texts around emojis. Vashishtha and Susan (2019) proposed a fusion system with fuzzy rules involving multiple lexicons and datasets to compute the sentiment of social media posts, where a set of NLP and word sense disambiguation techniques are combined.

Besides most approaches that involve a sentiment lexicon in some different ways, there are a few other interesting unsupervised methods as well. For instance, Turney (2002) proposed an algorithm based on some fixed syntactic patterns such as *part-of-speech (POS)* tags, that are likely to be used to express opinions. The algorithm consists of three steps: 1) two consecutive words are extracted if their POS tags conform to any of the predefined patterns; 2) the sentiment orientation (SO) of the extracted phrases is estimated using the *pointwise mutual information (PMI)* measure based on its association with the positive reference word "*excellent*" and the negative reference word "*poor*"; 3) given a review, the algorithm computes the average SO of all phrases in the review and classifies the review as positive if the average SO is positive and negative otherwise. Zhang et al. (2013) proposed a hierarchical generative model based on Naïve Bayes and LDA for unsupervised sentiment analysis at sentence level and document level simultaneously; the model named NB-LDA assumes that each sentence instead of words has a latent sentiment label, and then the sentiment label generates a series of features for the sentence independently in a Naïve Bayes manner. Zhang et al. (2018b) investigated the feasibility of quantum probability theory for twitter sentiment analysis using the density matrix defined on the quantum probabilistic space to perform unsupervised sentiment analysis. The idea is to artificially create two sentiment dictionaries, generate density matrices of documents and dictionaries using an extended

Quantum Language Model (QLM), then employ the quantum relative entropy to judge the similarity between density matrices of documents and dictionaries. Yadav and Chakraborty (2020) introduced methods that use different kinds of multilingual and cross-lingual embeddings to efficiently transfer knowledge from monolingual text to code-mixed text for sentiment analysis of code-mixed texts. Zeng et al. (2020) proposed a new approach to unsupervised sentiment analysis: instead of using gold labels provided by domain experts, target-opinion word pairs are used as a supervision signal (e.g. in "*the room is big*", *(room, big)* is a target-opinion word pair, which can be extracted by dependency parsing and simple rules). By introducing a latent variable, i.e. the sentiment polarity to the objective function, a sentiment classifier is trained by optimizing the evidence lower bound during prediction of an opinion word given a target word. Inspired by Radford et al. (2017), where a single unit of the byte-level recurrent language model, trained in a completely unsupervised manner, is capable of encoding sentiment in the representation. Saji and Whig (2020) explored the possibility of learning sentiment representation through simple LSTM networks trained on a limited amount of training samples, in which language models were trained separately on positive and negative review datasets (e.g. IMDB movie reviews) and evidence for differentiating positive word vectors and negative word vectors were found.

Sentiment Classification as Regression

Since sentiment can also be expressed as degrees or scores (e.g. a value between the range of $[-1, +1]$ or 1-5 stars), instead of treating sentiment analysis as a classification problem, it can be formulated as a regression problem as well.

Pang and Lee (2005) addressed the problem by determining the evaluation with respect to a multi-point scale (1 to 5 stars), where SVM regression, SVM multi-class classification and a meta labeling method were compared; the results suggested that the regression method is similar to the

classification method in terms of performance. Goldberg and Zhu (2006) improved this approach by presenting a graph-based semi-supervised learning algorithm to address the sentiment analysis task as rating inference, which used both labeled and unlabeled reviews. The unlabeled reviews were also the test reviews whose ratings need to be predicted. The idea is that, in the graph, each node is a document and the link between two nodes is the similarity value between the two documents; a large similarity weight implies that the two documents tend to have the same sentiment rating. In the paper, experiments with several different similarity schemes were tested. The algorithm also assumes that initially a separate learner has already predicted the numerical ratings of the unlabeled documents. The graph based method only improves them by revising the ratings through solving an optimization problem to force ratings to be smooth throughout the graph with regard to both the ratings and the link weights.

Instead of predicting the rating of each review, Snyder and Barzilay (2007) studied the problem of predicting the rating for each aspect by formulating the task as a multiple aspect ranking problem, where the goal is to produce a set of numerical scores, one for each aspect. Thus two models were proposed: aspect model (which works on individual aspects) and agreement model (which models the rating agreement among aspects). Both models were combined in learning, and lexical features such as unigram and bi-grams from each review were used for training. Thus the algorithm jointly learns ranking models for individual aspects by modeling the dependencies between assigned ranks, and guides the prediction of individual rankers by analyzing meta-relations between opinions, such as agreement and contrast. Liu and Seneff (2009) proposed a parse-and-paraphrase paradigm to assess the degrees of sentiment for product reviews, which consists of extracting adverb-adjective-noun phrases (e.g., "*very nice car*") based on clause structure obtained by parsing sentences into a hierarchical representation. A robust general solution was also proposed for modeling the contribution of adverbials and negation to the score for degree of sentiment, in which sentiment scores were assigned according to a heuris-

tic method which computes the contribution of adjectives, adverbials and negations to the sentiment degree based on the ratings of reviews where these words occurred.

Later, Qu et al. (2010) addressed the limitations of uni-gram and n-gram text representations in opinion mining: uni-grams cannot capture important expressions such as "*could have been better*"; and n-grams of words, on the other hand, capture such phrases, but typically occur too sparsely in the training set and fail to yield robust predictors. Thus a new bag-of-opinions representation was proposed to overcome the limitations of these two models, which is different from the traditional bag-of-words representation. In the opinion representation, each of the opinions is a triplet: a sentiment word, a modifier, and a negator (e.g. in "*not very good*", "*good*" is the sentiment word, "*very*" is the modifier and "*not*" is the negator). For sentiment classification of two classes (positive and negative), the opinion modifier is not crucial but for rating prediction, it is very important and so is the impact of negation. A constrained ridge regression method was developed to learn the sentiment score or strength of each opinion from domain-independent corpora of rated reviews. The key idea of learning was to exploit an available opinion lexicon and the review ratings to transfer the regression model to a newly given domain-dependent application. The algorithm derives a set of statistics over the opinion scores and then uses them as additional features together with the standard uni-grams for rating prediction.

More recently, as social media users are increasingly using additional images and videos to express their opinions and share their experiences, You et al. (2016) extended the data source of sentiment analysis from textual to visual and proposed a cross-modality consistent regression (CCR) model, which is able to utilize both the state-of-the-art visual and textual sentiment analysis techniques. The model consists of a fine-tuned convolutional neural network (CNN) for image sentiment analysis and a paragraph vector model for textual sentiment analysis, and the multi-modality regression model is trained on top of them. Later, Zhang et al. (2018c) improved

the CCR model by fusing semantic embeddings, sentiment embeddings and lexicon embeddings into a CNN framework through three different attention models including attention vector, LSTM attention and attentive pooling. Jiang et al. (2018) Proposed an ensemble regression algorithm that effectively integrates four different features extracted from text data, namely linguistic features, sentiment lexicon features, domain-specific features and word embedding features. The solution was tested in the *SemEval 2017 Task 5* competition and ranked 1st and 5th in *subtask 1* and *subtask 2* respectively. Saad and Yang (2019) studied twitter sentiment analysis based on ordinal regression using machine learning techniques, which consists of first pre-processing tweets using a feature extraction method that creates an efficient feature representation; then, the extracted features are scored and balanced under several classes. Finally, *multinomial logistic regression (SoftMax)*, *Support Vector Regression (SVR)*, *Decision Trees (DTs)*, and *Random Forest (RF)* algorithms are used for sentiment analysis classification in the proposed framework.

Summary

Sentiment classification at the document level provides an overall opinion on an entity, topic or event. It has been studied by a large number of researchers. However, this level of classification has some shortcomings for applications (Liu, 2012):

- In many applications, the user needs to know additional details, e.g., what aspects of entities are liked and disliked by consumers, which is very important for gaining business insights. In typical opinion documents, such details are provided, but document sentiment classification does not extract them for the user.

- Document sentiment classification is not easily applicable to non-reviews such as forum discussions, blogs, microblogs, and news articles, because many such postings can evaluate multiple entities and

compare them. In reality, it is sometimes hard to determine whether a post actually evaluates the entities that the user is interested in, and whether the post expresses any opinion at all, let alone to determine the sentiment about them. Without in-depth and more complicated natural language processing, document-level sentiment classification cannot perform such fine-grained tasks. In fact, online reviews do not need sentiment classification because almost all reviews already have user-assigned star ratings; in practice, it is the forum discussions, blogs and microblogs that need sentiment classification to determine people's opinions about different entities and topics.

2.1.4 Sentence Level Sentiment Analysis

As document level sentiment analysis may not offer all of what is needed for most applications, it is natural to move to a lower level: sentence level, i.e., to classify sentiment expressed in each sentence. One could also regard sentence level sentiment classification as a special case of the document level task where each document consists of a single sentence. In this way, there is no fundamental difference between document and sentence level classifications because sentences can be seen as just short documents. Thus, as defined in Section 2.1.3, sentence level sentiment analysis can be described as: given a sentence x evaluating an entity, determine the overall sentiment s of x. Same as document level, the task can be seen as to determine the sentiment s expressed on the aspect *GENERAL* in the quintuple:

(_, *GENERAL*, *s*, _, _)

where s is often a class of *positive*, *neutral* or *negative*. The entity e, opinion holder h and timestamp t are assumed to be known and singular (s expresses opinions on a single entity e from a single opinion holder h at a single time t). Besides, sentence level sentiment classification also makes the assumption that a sentence usually contains a single opinion (although not true in all cases) (Liu, 2012).

So far, there have been two approaches for solving sentence sentiment classification: applying directly ternary classification on the input sentence; or treating it as a two step classification problem. In the second case, the first step is to classify whether a sentence expresses an opinion or not. Then, the second step is to classify opinion sentences into *positive* and *negative* as a binary classification. The first step is usually called *subjectivity classification*, which determines whether a sentence expresses a piece of subjective information or factual (objective) information (Hatzivassiloglou and Wiebe, 2000; Riloff and Wiebe, 2003; Yu and Hatzivassiloglou, 2003; Wilson et al., 2004; Wiebe et al., 2004; Riloff et al., 2006; Wilson et al., 2006). In practice the two classes *opinionated* and *not opinionated* are commonly used instead of *subjective* and *objective*; as under the context of sentiment analysis an objective sentence could also express opinion. For instance, consider the sentence "*It stopped working yesterday.*"; it is an objective sentence but it implies a negative sentiment about the product due to an undesirable fact. Thus, it is more appropriate for the first step to classify each sentence as *opinionated* or not *opinionated*, regardless whether it is subjective or objective. However, due to the common practice, the term *subjectivity classification* is still usually used in the domain of sentiment analysis.

In reality, however, it is more common to adopt the other approach that directly performs ternary classification on a given sentence for sentence level sentiment analysis. For example, Hatzivassiloglou and Wiebe (2000); Yu and Hatzivassiloglou (2003) adopted a Bayesian classifier for discriminating subjectivity at document level, and later at sentence level, a modified log-likelihood ratio algorithm was proposed to determine the sentiment orientation for each adjective, adverb, noun and verb. The sentence level opinion is then aggregated from a lower level, using the average log-likelihood scores. Hu and Liu (2004a) proposed a lexicon-based algorithm for aspect level sentiment classification, which can determine the sentiment orientation of a sentence as well. It was based on a sentiment lexicon generated using a bootstrapping strategy with some given positive

and negative sentiment seed words and their synonyms and antonyms relations extracted from *WordNet*. The sentiment orientation of a sentence was determined by summing up the orientation scores of all sentiment words in the sentence. A positive word was given the sentiment score of +1 and a negative word was given the sentiment score of -1. Negation words and contrary words (e.g., "*but*" and "*however*") were also considered. Kim and Hovy (2004) adopted a similar approach, which consists of scoring sentiment words and determining the sentiment orientation of a sentence by multiplying the scores of the sentiment words in the sentence. Again, a *positive* word was given the sentiment score of $+1$ and a *negative* word was given the sentiment score of -1. Two other aggregation methods for sentiment scores were also experimented but they yielded inferior results. Gamon et al. (2005) presented a system named *Pulse* for mining topics and sentiment orientation jointly from free text customer feedback. The system consists of a semi-supervised learning algorithm, based on *Expectation Maximization (EM)* using the Naïve Bayes classifier (Nigam et al., 2000), that learns from a small set of labeled sentences and a large set of unlabeled sentences.

Different from the generative models mentioned previously, discriminative models have been widely adopted later. For instance, McDonald et al. (2007) proposed a hierarchical sequence learning model that takes into account information at varying levels of granularity. The model is similar to CRF (Lafferty et al., 2001), which jointly learns and infers sentiment at both the sentence-level and the document-level. Each sentence and document in the training set was labeled with a sentiment individually. Experiment results showed that learning at both levels simultaneously improved classification accuracy for both levels. Similarly, Täckström and McDonald (2011) reported a partially semi-supervised model which consists of a fusion of a fully supervised structured conditional model and its partially supervised counterpart. Both models leverage abundant natural supervision in the form of review ratings, as well as a small amount of manually crafted sentence labels, to perform sentence-level sentiment classification.

Hassan et al. (2010) present a method to identify the attitude of participants in an online discussion toward one another. As the focus was on the discussion recipient, the algorithm only used sentence segments with second person pronouns. The first step was to find sentences with attitudes using supervised learning, where the input features were generated by Markov models. Followed by the second step that determines the orientation of the attitudes, for which a lexicon-based method similar to that in Ding et al. (2008) was used; except that the shortest path in the dependence tree was utilized to determine the orientation when there were conflicting sentiment words in a sentence, while Ding et al. (2008) used words distance. Davidov et al. (2010) proposed a supervised sentiment classification framework based on data from Twitter where each tweet is considered a single sentence. By utilizing 50 Twitter tags and 15 smileys as sentiment labels, this framework avoids the need for labor intensive manual annotation, allowing identification and classification of diverse sentiment types of short texts.

Same as document level sentiment analysis, the focus on sentence level has also been shifting to deep learning, as it does not require hand-crafted features, yet is able to push the state of the art performance. Compared to traditional methods, architectures such as CNN and RNN use distributed word representations (word embeddings) as input which already encode some semantic and syntactic information. Moreover, these models can help learn intrinsic relationships between words in a sentence as well.

For instance, according to Zhang et al. (2018a), Socher et al. (2011) first proposed a *Semi-supervised Recursive Autoencoder Network* for sentence level prediction of sentiment label distributions; then followed by Socher et al. (2012, 2013), a model named *Recursive Neural Tensor Network (RNTN)* that captures tree structured syntactic relations was presented. The model naturally gains advantage on understanding compositional meaning of longer phrases. Later, the tree structured model was further improved by Tai et al. (2015) combining with *Long Short-term Memory (LSTM)* (Hochreiter and Schmidhuber, 1997), a basic building block that deals with vanishing or exploding gradient when training deep neural networks.

Meanwhile, based on the tree structured network, Qian et al. (2015) proposed two more advanced models: *Tag-guided Recursive Neural Network (TG-RNN)*, which chooses a composition function according to the part-of-speech tags of a phrase; and *Tag-embedded Recursive Neural Network (TE-RNN/RNTN)*, which learns tag embeddings and then combines tag and word embeddings together. Kalchbrenner et al. (2014) proposed *Dynamic Convolutional Neural Network (DCNN)* for the semantic modelling of sentences. By using the dynamic K-Max pooling operator as a non-linear subsampling function, the network handles input sentences of varying length and induces a feature graph over the sentence that is capable of explicitly capturing short and long-range relations. Kim (2014) proposed to use CNN directly for sentence-level sentiment classification and experimented with several variants, namely CNN-rand (where word embeddings are randomly initialized), CNN-static (where word embeddings are pre-trained and fixed), CNN-non-static (where word embeddings are pre-trained and fine-tuned) and CNN-multichannel (where multiple sets of word embeddings are used). Dos Santos and Gatti (2014) proposed a new deep neural network model: *Character to Sentence CNN (CharSCNN)*, which uses two convolutional layers to extract relevant features from character to sentence-level to perform sentiment analysis of short texts (tweets).

On the other hand, Wang et al. (2015) adopted a LSTM recurrent network for twitter sentiment analysis by simulating the interactions of words during the compositional process. With the help of gates and constant error carousels in the memory block structure, the model could handle interactions between words through a flexible compositional function. Multiplicative operations between word embeddings through gate structures are used to provide more flexibility and to produce better compositional results compared to the additive ones in simple recurrent neural networks. Besides, the unidirectional LSTM can be extended to a bidirectional LSTM (Graves and Schmidhuber, 2005) by allowing bidirectional connections in the hidden layer. Wang et al. (2016a) proposed a regional CNN-LSTM model, which consists of two parts: regional CNN and LSTM, to pre-

dict the valence arousal (VA) ratings of text. Unlike a conventional CNN which considers a whole text as input, the regional CNN uses an individual sentence as a region, dividing an input text into several regions such that useful information in each region can be extracted and weighted according to their contribution to the VA prediction. The regional information is then sequentially integrated by a LSTM. Wang et al. (2016b) described a jointed CNN and RNN architecture that takes advantage of the coarse-grained local features generated by CNN and the long-distance dependencies learned via RNN for sentiment analysis of short texts. Guggilla et al. (2016) conducted experiments using CNN and LSTM on two claim data sets compiled from online user comments for claim classification (classifying sentences to be *factual* or *feeling*), where word embeddings and linguistic embeddings were used. Huang et al. (2017) proposed a framework to learn tag-specific composition functions and tag embeddings in recursive neural networks, in order to utilize POS tags to control the gates of tree-structured LSTM networks and enhance sentence/phrase representations. Akhtar et al. (2017) proposed a novel method for combining deep learning and classical feature based models using a *Multi-Layer Perceptron (MLP)* network for financial sentiment analysis, in which several deep learning models (e.g. CNN, LSTM, GRU) are trained on top of financial word embeddings and lexicon features pre-trained on autoencoders. Finally, an ensemble is constructed combining all deep learning models and a classical supervised SVM model.

Moreover, hybrid approaches have been widely adopted as well. For instance, Guan et al. (2016) proposed a novel CNN framework for review sentiment classification for sentences which employs previously available ratings as weak supervision signals. The framework consists of two steps: first, learn a high level representation (embedding space) which captures the general sentiment distribution of sentences through rating information; second, add a classification layer on top of the embedding layer and use labeled sentences for supervised fine-tuning. Teng et al. (2016) proposed a context-sensitive lexicon-based method based on a simple weighted-sum

model, using a bidirectional LSTM network to learn the sentiments strength, intensification and negation of lexicon sentiments in composing the sentiment value of sentences. Yu and Jiang (2016) studied the problem of learning generalized sentence embeddings for cross-domain sentence sentiment classification and designed a neural network model containing two separated CNNs that jointly learn two hidden feature representations from both the labeled and unlabeled data. Zhao et al. (2017) propose a recurrent random walk network learning framework for the problem by exploiting both users' posted tweets and their social relations in microblogs. Mishra et al. (2017) proposed a framework based on CNN that learns cognitive features from both gaze data (eye-movement) and text data from readers, and uses them to classify the input text. The experiment results showed that cognitive enriched features improve the model performance. Qian et al. (2017) proposed a linguistically regularized LSTM for sentence-level sentiment classification, in which the linguistic role of sentiment lexicons, negation words, and intensity words were modeled, so as to capture the sentiment effect in sentences more accurately.

More recently, Lutz et al. (2018) proposed a method that uses distributed word representations and multi-instance learning to transfer information from the document-level to the sentence-level in the financial domain, so that investors and professionals can apply the utmost attention and detailed, domain-specific knowledge by assessing the information on a fine-grained basis. Hayashi and Fujita (2019) presented a method to calculate sentence vectors from word vectors, and a sentence level sentiment classification model is trained using sentence vectors and sentence polarities. In addition, a weight mechanism was proposed to help better compose sentence vectors by assigning higher value to word vectors. Zhang et al. (2019b) proposed a three-way enhanced convolutional neural network model named 3W-CNN, which consists of the ensemble of the deep learning based method and the traditional feature-based method. The model puts together a CNN model and a SVM with Naïve Bayes selected features (NB-SVM). Lin et al. (2020) proposed the use of a word-level sen-

timent bidirectional LSTM together with the self-attention mechanism for sentence-level sentiment classification, along with a finance report dataset for sentence-level financial risk detection. Wang et al. (2020) studied the problem of learning with noisy labels for sentence-level sentiment classification, where a novel DNN model called NETAB (as shorthand for convolutional neural networks with AB-networks) was proposed to handle noisy labels during training. NETAB consists of two convolutional neural networks, one with a noise transition layer for dealing with the input noisy labels and the other for predicting "clean" labels. The two networks were trained using their respective loss functions in a mutual reinforcement manner.

Summary

Compared to document level sentiment classification, the task at sentence level has a higher granularity and moves closer to sentiments on the targets. However, in real-life applications, several shortcomings can still be seen:

- In reality, it is usually not enough to provide users only the sentence level sentiment information, additional details such as which entities or aspects of entities are liked or disliked are more important for gaining business insights.

- It is often seen that one sentence may carry different (or even opposite) sentiments on different targets. For example, "*Apple is doing very well in this lousy economy.*", or "*I like the food here but the service is awful!*". In the last sentence, only aspect level sentiment analysis can precisely extract valuable sentiment information from the text; namely, *positive* sentiment on the target (aspect) *food*, and *negative* sentiment on the target (aspect) *service*.

2.1.5 Aspect-based Sentiment Analysis

As described in Section 2.1.3 and Section 2.1.4, document level and sentence level sentiment analysis are often insufficient for applications because they do not extract the relation between the sentiment and the target which the sentiment is expressed on. Moreover, even assuming that each document and sentence evaluates a single entity, it is not guaranteed that all aspects in the same document/sentence receive the same sentiment. For instance, in the sentence "*This car is beautiful but difficult to drive.*", multiple opinions are expressed towards different aspects of a car (*positive* on the aspect *APPEARANCE*, and *negative* on the aspect *FUNCTION*). In the sentence "*The screen of this phone is amazing, but the battery sucks.*", a *positive* opinion is expressed on the aspect *SCREEN* and a *negative* opinion is expressed on the aspect *BATTERY_LIFE*.

On the other hand, as opinion words are domain dependent, it could create big problems for sentiment analysis. For example, in the electronics or laptops review domain, *fallout* or *excel* are *neutral* proper nouns referring to a video game and a software; however, in a generic context *fallout* and *excel* usually carry *negative* and *positive* sentiment respectively. Sometimes, the same opinion word may even carry different sentiment when describing different targets/aspects. For instance, "*cheap price*" carries a *positive* sentiment while "*cheap plastic*" is definitely *negative*. For more examples, consider: "*black screen*" vs "*black macbook*"; "*loud speaker*" vs "*loud click*"; "*low price*" vs "*low grade*".

Thus, to extract a detailed information and overcome the domain- and context-dependent problem of opinion words, it is absolutely necessary to move to a more fine-grained level: aspect level, and perform aspect-based sentiment analysis (ABSA). As described in Section 2.1.2, the objective of ABSA is to discover every quintuple $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ in a given document d; and six tasks are performed to achieve this goal. For research purposes, as different tasks usually require different strategies, ABSA has been simplified to consider three of the six tasks. Namely, entity extrac-

tion, aspect detection and aspect sentiment classification. Due to the fact that most of the corpora used for ABSA are online reviews, it is common to assume that the entities are already known when studying ABSA. Adding the fact that entity detection and aspect extraction are essentially the same task with different outputs, we summarize ABSA to the following two core tasks:

1. **Entity/Aspect extraction**: This task extracts aspects (or entities) that have been evaluated. For example, in the sentence, "*The voice quality of this phone is amazing*", the aspect is *VOICE_QUALITY* of the entity represented by "*this phone*". In the case of "*I love this phone*.", the sentence evaluates the phone as a whole, i.e., the *GENERAL* aspect of the entity represented by "*this phone*". As mentioned previously, it is common to assume the entity is known when working with online reviews, and thus consider the aspects only.

2. **Aspect sentiment classification**: This task determines whether the opinions on different aspects are *positive*, *neutral*, or *negative*. In the first example above, the opinion on the *VOICE_QUALITY* aspect is *positive*. In the second example, the opinion on the aspect *GENERAL* is also positive.

Entity/Aspect Extraction

In most cases, entity detection and aspect extraction are treated as an information extraction task, using similar methods to *named entity recognition (NER)*. NER is a NLP task with the objective of identifying and grouping textual elements into predefined categories (e.g. *person*, *organization*, *location* and *time*); in other words, extracting structured information from unstructured data. In the case of ABSA, the system should identify entities such as "*SONY camera*", "*iPhone*", "*HP laptop*"; and aspects such as *FOOD*, *SERVICE*, *LENS*, *BATTERY*, etc. from unstructured corpora.

Commonly, approaches such as rule-based methods and sequence learning models are applied for the task. For instance, in Riloff and Jones (1999); Cimiano and Völker (2005), a bootstrapping method for NER was proposed by searching patterns such as "*offices in X*", "*facilities in X*" or "*city/organization such as X*" where "*X*" represents seed entities. Similarly in ABSA, frequent nouns and nouns phrases are firstly searched to be candidate entities or aspects, later by following the dependency pattern, more targets can be found given the same sentiment words. For example, given "*picture*" as a known aspect and "*amazing*" as a known sentiment word. In the sentence "*The pictures are absolutely amazing.*", the sentiment word and the aspect word form a dependency pattern. By following this dependency pattern, more aspects such as "*software*" can be found in sentences like "*The software is amazing.*" (Hu and Liu, 2004b). In Kobayashi et al. (2007), a more sophisticated approach was proposed that first identifies opinion words using a dictionary and then looks for its candidate aspects using syntactic patterns. On the other hand, Popescu and Etzioni (2005) proposed an approach based on *pointwise mutual information (PMI)*, which consists of a measurement of the co-occurrence of two terms in a large corpus, so that the higher the score, the more similar two terms are. In this case, they first defined a set of phrases based on patterns associated with a given term; for instance, aspects of the term "*scanner*" often come with phrases like "*X of scanner*", "*scanner has X*" or "*scanner comes with X*" etc., where "*X*" stands for candidate aspects. The PMI score is then measured by searching in a large corpus, and words occurring together more frequently with the predefined terms tend to have higher scores and thus are more likely to be the correct aspect.

Regarding sequence-based learning models, as in information extraction tasks, algorithms such as *Hidden Markov Model (HMM)* (Rabiner, 1989) and *Conditional Random Fields (CRF)* (Lafferty et al., 2001) are commonly adopted. Just like in NER, finding entities and aspects in ABSA can be treated as a sequence labeling problem, and thus be dealt with both HMM (Jin et al., 2009; Jin and Ho, 2009) and CRF (Jakob and Gurevych,

2010; Li et al., 2010). As HMM measures the joint probability of a sequence based on the *Markov Assumption*: it assumes that the probability of the current state is only based on its immediate predecessor and is independent of all its ancestors. However, CRF models a conditional probability over a hidden sequence given its observation sequence. It releases the strong independence assumption made by HMM as it takes the whole input sequence features into account to predict on each state. Thus, in this case, when comparing generative models (e.g. Naïve Bayes, HMM, etc.) to discriminative models (e.g. Logistic Regression, CRF, etc.), discriminative models generally perform better and are favored in most cases (Zhang and Liu, 2015).

In some other cases, unsupervised topic modeling algorithms such as *Probabilistic Latent Semantic Analysis (pLSA)* and *Latent Dirichlet Allocation (LDA)* were also adopted for aspect detection by a number of researchers (Lu et al., 2009; Moghaddam and Ester, 2011; Titov and McDonald, 2008; Titov and Mcdonald, 2008). More recently, deep neural networks such as CNN, LSTM, Bi-LSTM and Seq2Seq have also been widely adopted (Poria et al., 2016; Jebbara and Cimiano, 2016; Giannakopoulos et al., 2017; Schmitt et al., 2020; Gandhi and Attar, 2020; Ma et al., 2020).

Aspect Sentiment Classification

Regarding the second task of ABSA: sentiment classification, there are in general two main approaches in the literature for determining the sentiment orientation expressed on each aspect in a sentence: the lexicon-based approach and the supervised learning approach (Liu, 2012).

The lexicon-based approaches are typically unsupervised (Ding et al., 2008; Hu and Liu, 2004a), which have been shown to perform quite well in a large number of domains. Usually, the sentiment lexicons are dictionaries that contain sentiment words, phrases, and idioms as keys; and their associated polarities as values. To perform sentiment classification, they are commonly used jointly with methods such as composite expressions,

heuristic rules and sentence parse trees to determine the sentiment orientation on each aspect in a sentence. Meanwhile, special conditions such as *negations*, *sentiment shifters*, *but-clauses* and many other constructs which may affect sentiments are considered as well.

For instance, in Ding et al. (2008), a four steps approach was taken. 1) All sentiment words and phrases in the sentence are marked and assigned with a score of $+1$ for *positive* or -1 for *negative*. 2) Sentiment shifters (also called *valence shifters* in Polanyi and Zaenen (2005)), which consist of words and phrases that can reverse sentiment orientations (e.g. *not*, *never*, *none*, *cannot*, etc.), are used to shift the sentiment orientation of certain polarities marked in step 1. 3) The system handles words or phrases that indicate *contrary*, like *but-clauses* by changing the sentiment orientation. 4) Finally, an opinion aggregation function is applied to the resulting sentiment scores to determine the final orientation of the sentiment on each aspect in the sentence. Formally, let the sentence be s, which contains a set of aspects $a_1, a_2, ..., a_m$ and a set of sentiment words or phrases $sw_1, sw_2, ..., sw_n$ with their sentiment scores obtained from steps 1-3. The sentiment orientation for each aspect a_i in s is determined by:

$$score(a_i, s) = \sum_{ow_j \in s} \frac{sw_j.so}{dist(sw_j, a_i)}$$

where sw_j is a sentiment word/phrase in s, $dist(sw_j, a_i)$ is the distance between aspect a_i and sentiment word sw_j in s. $sw_j.so$ is the sentiment score of sw_i. The multiplicative inverse is used to give lower weights to sentiment words that are far away from aspect a_i. If the final score is *positive*, then the opinion on aspect a_i in s is *positive*; if the final score is *negative*, then the sentiment on the aspect is *negative*; otherwise, it is neutral. In fact, there are many opinion aggregation methods. For example, Hu and Liu (2004a) simply summed the sentiment scores of all sentiment words in a sentence; Kim and Hovy (2004) used multiplication of sentiment scores; some other similar methods were also employed by other researchers (Wan, 2008; Zhu et al., 2009).

Regarding the supervised learning approach, in general, methods mentioned previously in Section 2.1.3, 2.1.4 on document and sentence level are applicable as well. Meanwhile, some methods designed specifically for ABSA were proposed. For instance, Wei and Gulla (2010) proposed a hierarchical classification model using a sentiment ontology tree that leverages the knowledge of hierarchical relationships of products attributes to better capture sentiment aspect relations. Similar to Boiy and Marie-Francine (2008), Jiang et al. (2011) applied a dependency parser to generate a set of aspect dependent features and context features for twitter sentiment classification. Dong et al. (2014) proposed *Adaptive Recursive Neural Network (AdaRNN)* for target-dependent twitter sentiment classification, which adaptively propagates the sentiments of words to targets depending on the context and syntactic relationships between them. The model uses the representation of the root node as features, and feeds them into the softmax layer to predict the distribution over classes. In Vo and Zhang (2015), instead of relying on parse trees, which are subject to noise for informal text such as tweets, they proposed a method to automatically extract a rich set of features from unsupervised learning methods to enhance the performance of the classifier. In particular, a tweet is split into a left context and a right context according to a given target, then distributed word representations and neural pooling functions are used to extract features. Zhang et al. (2016) proposed a novel model based on Dong et al. (2014) and Vo and Zhang (2015) to address the limitation of pooling functions, which do not explicitly model tweet-level semantics. First, a bi-directional gated neural network is used to connect the words in a tweet so that pooling functions can be applied over the hidden layer instead of words for better representing the target and its contexts. Second, a three-way gated neural network structure is used to model the interaction between the target mention and its surrounding contexts.

On the other hand, the LSTM network and the attention mechanism became widely popular for ABSA. For example, Tang et al. (2016a) proposed *Target Dependent LSTM (TD-LSTM)* and *Target Connection LSTM*

(TC-LSTM) to extend LSTM by taking the target into consideration. The given target is regarded as a feature and concatenated with its context features for aspect sentiment classification, thus the influences of different context words on determining the sentiment polarity of a sentence towards a target is considered. Ruder et al. (2016) modeled the interdependencies of sentences in a review with a hierarchical bidirectional LSTM for ABSA, where the model is capable of leveraging both intra- and inter-sentence relations. The model consists of feeding first word embeddings to a sentence-level bidirectional LSTM. Hidden states of the forward and backward LSTM are concatenated together with the target embedding and fed to a bidirectional review-level LSTM. At every time step, the output of the forward and backward LSTM is concatenated and fed into a final layer, which outputs a probability distribution over sentiments. Wang et al. (2016c) proposed an attention-based LSTM with aspect embeddings, which was proven to be an effective way to enforce the neural model to attend to the related part of a sentence given different aspects. The model consists of learning an attention vector with the concatenation of LSTM hidden states and aspect embeddings, and then use it to obtain a final representation of the input sentence given an aspect. Tang et al. (2016b) introduced a deep memory network for aspect level sentiment classification that explicitly captures the importance of each context word when inferring the sentiment polarity of an aspect. Such importance degree and text representation are calculated with multiple computational layers, each of which is a neural attention model over an external memory. Lei et al. (2016) proposed to use a neural network approach to extract pieces of input text as rationales (reasons) for sentiment analysis, where two modular components are combined, namely a generator and an encoder. The generator specifies a distribution over text fragments as candidate rationales, which are passed through the encoder for prediction.

Later, Yang et al. (2017) proposed two attention-based bidirectional LSTMs to improve the classification performance. Cheng et al. (2017) proposed a *HiErarchical ATtention (HEAT)* network for ABSA, which con-

tains a hierarchical attention module, consisting of aspect attention and sentiment attention. The aspect attention extracts the aspect-related information to guide the sentiment attention to better allocate aspect-specific sentiment words of the text. Chen et al. (2017) propose a novel framework based on neural networks for ABSA that adopts a multiple-attention mechanism to capture sentiment features separated by a long distance, so that it is more robust against irrelevant information. The results of multiple attentions are non-linearly combined with a GRU recurrent neural network and a weighted-memory mechanism. Liu and Yue (2017) extended the attention modelling by differentiating the attention obtained from the left context and the right context of a given target/aspect. They further controlled their attention contribution by adding multiple gates. Li et al. (2017a) merged the target identification task and the sentiment classification task together to perform ABSA by proposing an end-to-end deep memory network (*AttNet*), in which the two sub-tasks are interleaved by a deep memory network. In this way, signals produced in target detection provide clues for polarity classification, and reversely, the predicted polarity provides feedback to the identification of targets. Ma et al. (2017) proposed *Interactive Attention Networks (IAN)* to interactively learn attention in the contexts and targets, and generate the representations for targets and contexts separately. With this design, the IAN model can well represent a target and its collocating context, which is helpful for ABSA. Tay et al. (2017) proposed a novel extension of end-to-end memory networks named *Dyadic Memory Networks (DyMemNN)* that models dyadic interactions between aspect and context, by using either neural tensor compositions or holographic compositions for memory selection operation.

More recently, Liu et al. (2018) proposed a content attention based ABSA model, which consists of two attention enhancing mechanisms: a sentence-level content attention mechanism and a context attention mechanism. He et al. (2018) proposed two novel approaches for improving the effectiveness of the attention mechanism for ABSA, which consists of a method for obtaining target representation that better captures the seman-

tic meaning of the opinion target, and an attention model that incorporates syntactic information into the attention mechanism. Zhu and Qian (2018) proposed a deep memory network with auxiliary memory to learn aspect features and term features simultaneously, where a main memory module is used to capture the important context words for sentiment classification, and an auxiliary memory module is used to implicitly convert aspects and terms to each other, and feed both of them to the main memory module. Ma et al. (2018) proposed a solution for ABSA that leverages commonsense knowledge by combining a LSTM network with a hierarchical two level attention mechanism (target-level and sentence-level); an extension of LSTM named *Sentic LSTM* was proposed to carefully encode the commonsense knowledge into a recurrent order. Xing et al. (2019) proposed a variant of LSTM: *aspect-aware LSTM (AA-LSTM)*. Compared to most attention LSTM methods that learn the attention vector after the LSTM encoding, AA-LSTM incorporates aspect information into LSTM cells in the context modeling stage before the attention mechanism. Therefore, the model can dynamically produce aspect-aware contextual representations. Xu et al. (2019) extended ABSA to *Review Reading Comprehension (RRC)* that aims to turn customer reviews into a large source of knowledge that can be exploited to answer user questions, where BERT was used for post-training. Sun et al. (2019) constructed auxiliary sentences from aspect terms and converted ABSA to a sentence-pair classification task, similar to question answering and natural language inference. Similar to Xu et al. (2019), the pre-trained language model BERT was used for fine-tuning. García-Díaz et al. (2020) proposed an ontology-driven aspect-based sentiment analysis for infodemiology where ontologies are used to extract features on the infectious disease domain with concepts such as risks, symptoms, transmission methods and drugs; and the relationship between these concepts is measured in order to determine the degree to which one concept influences other concepts. Finally, all information is applied in a deep learning model for ABSA. Huang et al. (2020) propose a weakly-supervised approach for ABSA, which uses only a few keywords describ-

ing each aspect/sentiment without using any labeled examples. The model first learn sentiment-aspect joint topic embeddings in the word embedding space by imposing regularizations to encourage topic distinctiveness, and then use neural models to generalize the word-level discriminative information by pre-training the classifiers with embedding-based predictions and self-training them on unlabeled data. Karimi et al. (2020) applied adversarial training to produce artificial examples that act as a regularization method for the BERT model on the tasks of both aspect extraction and aspect sentiment classification.

Summary

Aspect-level sentiment analysis is usually performed at the level of detail required for practical applications. Although a great deal of work has been done in the research community and many systems have also been built, the problem is still far from being solved. Sentiment analysis in general seems to be a long tail problem (Liu, 2012), while sentiment words can handle most of the cases (more in some domains than in others), the rest are highly diverse, scarce and infrequent, which make it hard for algorithms to learn patterns because there are simply not enough training data for all the cases. In fact, there seem to be an unlimited number of ways that people can use to express positive or negative opinions. Every domain appears to have something special. As advanced as the latest developments on deep learning are, it is yet unable to achieve human level performance. Since deep learning based methods rely heavily on large scale training data, it still remains challenging to train a robust model that behaves well in many domains.

Thus in this thesis, we would like to highlight that, as an end-to-end approach, deep learning based systems lack flexibility and robustness because one cannot easily adjust the network to fix an obvious problem. For example, in the state of the art system by Wang et al. (2016c) when this research work started, the attention LSTM network always predicts *posi-*

tive when seeing the word "*disappointed*", and the network is not able to recognize the word "*dungeon*" as an indication of *negative* polarity. One way to fix the problem is gathering more training data, but it is not always feasible. Thus, another way to solve this problem would be leveraging existing linguistic resources such as sentiment lexicons; however, unlike traditional learning approaches, deep learning models take word vectors as input that are not combined in a straightforward way with other features. Therefore one of the main objectives of this thesis is to find an effective way of merging lexicon information into the attention-LSTM based ABSA model.

2.2 Sentiment Lexicon

In many sentiment classification tasks, a *sentiment lexicon*, sometimes called *sentiment dictionary* is often used to help differentiate the polarity of a given opinion word. It consists of a dictionary containing words as keys and their corresponding polarities as values (e.g. $\{$"*good*": $+1$, "*bad*": $-1\}$). One way to build a lexicon is relying on human annotation; although it generally guarantees higher quality, it is rather time consuming and relatively expensive in terms of resources. During the years, many automatic and semi-automatic approaches have been proposed to construct or expand a sentiment lexicon, they are also referred as *dictionary-based* and *corpus-based* approaches. Usually, such approaches start from a predefined set of seed words. By using thesauri or corpora, combining with rules that leverage word relations such as synonym, antonym and hypernym (dictionary-based), and linguistic patterns such as "and" rule, "but" rule and negation rule (corpus-based), the seed words are expanded to a larger lexicon. For instance, given a seed word "*nice*" and its associated polarity *positive*, it is possible to find more *positive* words like "*beautiful*" and *negative* words like "*ugly*" by looking for its synonyms and antonyms in a thesaurus. Meanwhile, given "*beautiful*" as a known *positive* opinion word, by applying linguistic rules we can expand the lexicon through patterns such as "*This car is beautiful and spacious.*" ("and" rule), or "*This car is beautiful but difficult to drive.*" ("but" rule), to learn that "*spacious*" and "*difficult*" are new *positive* and *negative* words respectively.

2.2.1 Sentiment Lexicon Generation

There have been a variety of researches conducted on constructing a sentiment lexicon using existing thesauri; for instance, Hu and Liu (2004b) first proposed a bootstrapping method leveraging the dictionary-based approach and Mohammad et al. (2009) extended the method with additional morphological patterns. Kamps et al. (2004) followed a more sophisti-

cated path by computing the distance between terms and seed words in a thesaurus to assign sentiment to a new term. Turney (2002); Turney and Littman (2003) applied *Pointwise Mutual Information (PMI)* to assign polarity of a given word by comparing the co-occurrence of the word with a set of positive/negative opinion words. Esuli and Sebastiani (2005); Kim and Hovy (2006) expanded a set of positive/negative seed words through synonyms and antonyms in a dictionary and later built a classifier with supervised learning algorithms to distinguish sentiment orientation.

Although dictionary-based approaches are relatively simple and fast to deploy (they require only a set of seed words), the collected words are generally domain and context independent. In reality, opinion words are highly domain- and context- dependent. For example, "*huge*" is positive when talking about a hotel room, as in "*huge room*" , but *negative* when it refers to price, as in "*huge price*"; "*cheap*" is *positive* when someone says "*The ink for the printer is cheap.*", but *negative* when saying "*cheap appearance*". In a dictionary-based approach, it is impossible to extract domain-dependent polarities like these. Therefore, a corpus-based approach was firstly proposed by Hatzivassiloglou and McKeown (1997) to bootstrap new sentiment words from a corpus with a set of linguistic rules; so that domain-specific polarities can be extracted. The idea is also referred as *sentiment consistency*, which basically means people tend to express the same sentiment in a compound sentence connected by conjunctions such as "*and*", "*or*", "*either-or*" and "*neither-nor*"; and opposite sentiment in sentences connected by "*but*", "*however*" and "*although*". Kanayama and Nasukawa (2006) extended the consistency rule from intra-sentence to inter-sentence, in order to find domain dependent opinion words and their polarities. Wu and Wen (2010) adopted syntactic patterns, PMI and web search hit together to address the context-dependent problem for a set of Chinese adjectives. Lu et al. (2011) proposed a method to automatically construct a sentiment lexicon as an optimization problem, combining general sentiment lexicon, overall sentiment score from online reviews, thesaurus, and linguistic rules into an optimization framework in order to automatically

build a domain and aspect dependent lexicon.

More recent approaches are mainly hybrid methods which combine linguistic rules, machine learning, optimization techniques and other language resources such as knowledge bases and ontologies. For instance, Lu et al. (2011) proposed a unified optimization framework which combines 4 heuristics as constraints to calculate the sentiment orientation of a set of aspect-opinion pairs parsed from a review corpus; namely, general sentiment score, overall sentiment score, synonyms adding the "*and*" consistency rule and antonym adding the "*but*" consistency rule. Weichselbraun et al. (2013) took a different approach to tackle the context-dependent problem by learning the sentiment orientation of an ambiguous term through its context terms, and adding later concept information from an additional knowledge base into the contextualized lexicon for boosting the system's performance. They first extract ambiguous terms by looking into the sentiment distribution of opinion words in a labeled corpus with statistical methods. Indicators such as standard deviation, average value and a set of empirically determined thresholds are used to determine if a term is ambiguous. The ambiguous term and its context terms (all other terms appearing with the sentiment term in the sentence or paragraph not contained in a stop-words list) are extracted together and stored in a contextualized sentiment lexicon. The polarity of the ambiguous term was later decided by a Naïve Bayes classifier given its context terms. To enhance the lexicon with a clearer distinction between contexts, the additional knowledge base *WordNet* was also introduced into the contextualized lexicon by calculating the similarity between context terms and entries. Agarwal et al. (2015) combined common sense through language resources into the process of context-aware ABSA with the help of a contextualized sentiment lexicon. They first constructed a domain-specific ontology using *ConceptNet* to ground aspects and entities and later give importance value (weight) to each clause at the final sentiment aggregation step. Then, a contextualized lexicon was built similar to Weichselbraun et al. (2013): a general lexicon and a set of labeled review documents were firstly used to find

ambiguous sentiment terms through statistical parameters; all the nouns, adjectives, adverbs, and verbs that occurred in two sentences before and after the occurrence of the ambiguous term in all the review documents are considered as context terms. The sentiment orientation of the ambiguous term was decided by the sum of all its co-occurring context term polarities, which are calculated based on the prior probability of appearing in a positive or negative review document.

From a different perspective, Mikolov et al. (2013b) discovered that vectorized word representations in a continuous space can capture semantic regularities. More specifically, after training a RNN based language model, in addition to the model itself, word representations can also be learned as parameters in the input/embedding layer (word embeddings). A simpler architecture was later proposed by Mikolov et al. (2013a) for more efficient learning of word embeddings using *skip-gram* and *CBOW* techniques, which consist of predicting a word through a feedforward neural network given its context words or the other way around. As distributed word representations, compared to traditional single dimension word representations (e.g. one-hot encoding, TF-IDF, etc.), word embeddings are significantly better in capturing semantic relations. For example, the male/female analogous relation is automatically encoded, and with the learned vector representations, a semantic relation was able to be modeled through vector operations: $v_{king} - v_{man} + v_{woman}$ results in a vector very close to v_{queen}. Similarly, the singular/plural relation can be learned as well, e.g. in the learned vector space, $v_{apple} - v_{apples} \approx v_{family} - v_{families} \approx v_{car} - v_{cars}$.

Following the idea of distributed word representations, Rothe and Schütze (2016) proposed a novel method named *DENSIFIER*, which aims to transform existing embedding space into an ultra-dense subspace by learning an orthogonal transformation matrix. The transformed ultra-dense representation is able to induce extra information such as sentiment without losing its original sense. The idea was to learn a transformation matrix by minimizing the distance between words that carry the same sentiment orienta-

tion and maximizing the distance between words with opposite sentiment orientation. A few hundred labeled words were used for the learning process, and the learned transformation matrix was then able to convert ordinary word embeddings with previously unknown sentiment to ultra-dense embeddings which carry sentiment information, and therefore a sentiment lexicon was built in this way. Later, Rothe et al. (2016) proved that similar to operations like "*king*" $-$ "*man*" $+$ "*woman*" $\approx$ "*queen*" in the original word vector space, new operations sensitive to polarity such as $-1\times$ "*hate*" $\approx$ "*love*" can be performed as well.

2.2.2 Sentiment Domain Adaptation

In Liu (2012), it has been shown that sentiment analysis is highly sensitive to the domain from which the training data is extracted. A classifier trained using opinion documents from one domain often performs poorly on test data from another domain. The reason is that words and even language constructs used in different domains for expressing opinions can be quite different. To make matters worse, the same word in one domain may mean *positive* but in another domain may mean *negative*. For instance, the word "*sharp*" carries a *negative* sentiment under a general context, but when talking about the screen or the outlook of an electrical product, "*sharp*" is usually associated with a *positive* opinion. Thus, domain adaptation or transfer learning is needed.

Existing approaches either adapt the model, or adapt a sentiment lexicon from a *source domain* to a *target domain*. For instance, Aue and Gamon (2005) proposed to transfer sentiment classifiers to new domains in the absence of large amounts of labeled data in these domains by experimenting four different strategies. Yang et al. (2006) proposed a simple strategy based on feature selection for transfer learning of sentence level classification, in which they first used two fully labeled training sets from two domains to select features that were highly ranked in both domains; and these selected features were considered domain independent features.

Tan et al. (2007) proposed a strategy that first trains a base classifier using the labeled data from the source domain, and then uses the classifier to label some informative examples in the target domain. Based on the selected examples in the target domain, a new classifier is learned, which is finally applied to classify the test cases in the target domain. Blitzer et al. (2007) proposed a *Structural Correspondence Learning (SCL)* algorithm for domain adaptation. Pan et al. (2010) proposed a method similar to SCL at the high level. The algorithm works in the setting where there are only labeled examples in the source domain and unlabeled examples in the target domain. It bridges the gap between the domains by using a *Spectral Feature Alignment (SFA)* algorithm to align domain-specific words from different domains into unified clusters, with the help of domain independent words as the bridge. Xia and Zong (2011) found that across different domains, features of some types of *Part-of-speech (POS)* tags are usually domain-dependent, while some others are domain-free. Based on this observation, they proposed a POS-based ensemble model to integrate features with different types of POS tags to improve the classification performance. Wu and Huang (2016) proposed a new domain adaptation approach which can exploit sentiment knowledge from multiple source domains; where they first extract both global and domain-specific sentiment knowledge from the data of multiple source domains using multi-task learning, then transfer them to the target domain with the help of words' sentiment polarity relations extracted from the unlabeled target domain data. The similarities between target domains and different source domains are also incorporated into the adaptation process. Barnes et al. (2018) proposed a novel perspective that casts the domain adaptation problem as an embedding projection task. The model takes as input two mono-domain embedding spaces and learns to project them to a bi-domain space, which is jointly optimized to project across domains and to predict sentiment. Rietzler et al. (2020) performed deep transfer-learning by a two-steps fine-tuning, which consists of a self-supervised domain-specific BERT language model fine-tuning, followed by a supervised task-specific fine-tuning.

On the other hand, a lot of research has focused on adapting generic lexicons to domain-specific ones. In principle, methods described previously in Section 2.2.1 for sentiment lexicon generation are also applicable for domain adaptation. For instance, Hatzivassiloglou and McKeown (1997); Kanayama and Nasukawa (2006) applied linguistic rules such as sentiment consistency to bootstrap new word polarities in a corpus. Wu and Wen (2010) combined syntactic patterns and PMI with web search hit together to address the ambiguity of context dependent words. Lu et al. (2011) incorporated multi-dimensional information into an optimization framework to construct a domain dependent lexicon. Bollegala et al. (2011) proposed a method to automatically create a sentiment sensitive thesaurus using both labeled and unlabeled data from multiple source domains to find the association between words that express similar sentiments in different domains. The created thesaurus is then used to expand the original feature vectors to train a binary sentiment classifier. More recently, as deep learning thrives in most NLP fields, the focus of sentiment domain adaptation also shifts more to vector based (Mikolov et al., 2013a) approaches. For example Hamilton et al. (2016) induced a domain-specific lexicon through label propagation over the lexical graph. When talking about sentiment, it is believed that pre-trained word embeddings are not able to encode sentiment orientation as they are usually learned in an unsupervised manner on a general domain corpus by predicting a word given its context (or vise versa). For example, the word "*good*" and "*bad*" both share similar contexts in a general domain corpus such as *Wikipedia*, therefore their distributed word representations are similar as well. This similarity also determines that the sentiment orientations of the two words are not reflected in the learned word vectors. However, this assumption is believed to be true until Mudinas et al. (2018) discovered that the distributed word representations in fact form distinct clusters for opposite sentiments; and this behavior in general holds across different domains. In other words, in the vector space shaped by a domain specific corpus, *positive* words are closer to each other than they are to *negative* words; and the same behav-

ior is expected in other domains. The key here is that instead of learning word embeddings from a generic domain corpus, when training on different domain-specific data, distinct clusters for opposite sentiment can actually be formed in each domain-specific vector space. One explanation could be that in fact in a domain specific corpus (e.g. Amazon electronic products reviews), opinion words with opposite sentiment are less likely to appear together in the same sentence. For instance, it is unlikely that one would say "*This phone is beautiful and ugly.*". Thus, based on the cluster observation, a probabilistic word classifier can be trained on a set of seed words; and this classifier can be used to induce the generic sentiment lexicon by predicting the word polarity in a new domain given its domain-specific word embeddings.

2.2.3 Lexicon Integration with DNN Models

As an end-to-end approach, deep learning based systems lack flexibility as one cannot easily adjust the network to fix an obvious problem: e.g. when the network always predicts *positive* when seeing the word "*disappointed*", or when the network is not able to recognize the word "*dungeon*" as an indication of *negative* polarity. It could be even trickier to fix this issue in a low-resource scenario where more labeled training data is simply not available (Bao et al., 2019). An obvious way that could help the model to distinguish *positive* and *negative* words is leveraging existing language resources: sentiment lexicons. The advantage of using sentiment lexicons here is: first, as freely available language resources, it requires no extra efforts for data argumentation or feature engineering; second, by having a secondary input, the model should learn to leverage the information provided by the lexicon; compared to pure end-to-end approaches, a lexicon is easier to be maintained; for instance, the polarities of opinion words can be added, removed or updated accordingly, thus the model will become overall more robust.

Over the years, a lot of work has been done focusing on leveraging

existing sentiment lexicons to enhance the performance of deep learning based sentiment analysis systems; however, most works are performed at document and sentence level. For instance, at document level, Teng et al. (2016) proposed a weighted-sum model which consists of representing the final prediction as a weighted sum of network prediction and polarities provided by the lexicon. Zou et al. (2018) described a framework to assign higher weights to opinion words found in lexicon by transforming lexicon polarity to sentiment degree. At sentence level, Shin et al. (2017) used two convolutional neural networks to separately process sentence and lexicon inputs, and the final representation is then combined with an attention mechanism for prediction. Lei et al. (2018) described a multi-head attention network where the attention weights are jointly learned with lexicon inputs for classification. Wu et al. (2018) proposed a new labeling strategy similar to ABSA for sentence-level classification which breaks a sentence into clauses by punctuation to produce more lower-level examples; then the model processes a sentence in different levels taking into account the lexicon and other linguistic information such as POS tags; finally, the multi-level LSTM network merges all information and predicts at sentence level. Barnes (2019) explored the use of multi-task learning (MTL) for incorporating external knowledge in neural models by using MLT to enable a BiLSTM sentiment classifier to incorporate information from sentiment lexicons. Li et al. (2020b) experimented a lexicon integrated two-channel CNN–LSTM model, combining CNN and LSTM/BiLSTM branches in a parallel manner; together with a novel padding method that makes the input data sample of a consistent size and improving the proportion of sentiment information in each review. Yang et al. (2020) proposed a new sentiment analysis model named SLCABG, which is based on the sentiment lexicon and combines CNN and attention-based Bidirectional Gated Recurrent Unit (BiGRU). The SLCABG model combines the advantages of sentiment lexicon and deep learning technology, where the sentiment lexicon is first used to enhance the sentiment features in the reviews. Then the CNN and the GRU network are used to extract the main sentiment

features and context features in the reviews and the attention mechanism is used for merging and final classification. Similarly, Ren et al. (2020) proposed a lexicon-enhanced attention network (LEAN) based on bidirectional LSTM. LEAN not only can catch the sentiment words in a sentence but also concentrate on specific aspect information in a sentence. Moreover, leveraging lexicon information enhances the model's flexibility and robustness at the same time. Meanwhile, some other similar works that incorporate linguistic resources for sentiment analysis have been carried out Rouvier and Favre (2016); Qian et al. (2017).

Summary

In this section (Section 2.2), we covered general approaches for building or expanding a sentiment lexicon, and the trend has evolved from dictionary- or corpus- based approaches to learning-based approaches, and then to word vector based approaches. These approaches can also be applied for sentiment domain adaptation, which is one of the topics that we are interested in further exploring.

As mentioned previously in Section 2.1.5, deep learning based end-to-end systems lack flexibility and robustness; therefore it is reasonable to apply sentiment lexicon with a DNN model to overcome this weakness. Prior to Bao et al. (2019), existing approaches for lexicon integration in deep learning based sentiment analysis systems have been mostly carried out at document and sentence level. In addition, most of the approaches need some sort of transformation of the lexicon inputs, e.g. a convolution layer (Shin et al., 2017) or sentiment degrees (Zou et al., 2018). Thus we first proposed ATLX (Bao et al., 2019) that performs sentiment analysis at aspect level, and at the same time, the model is capable of taking directly the numerical polarities provided by lexicons as inputs and make use of that information with a less complex architecture.

With a lexicon enhanced ABSA neural system, we are interested to see whether it is possible to further improve the ATLX model with a more fine-

grained lexicon. As most works on sentiment domain adaptation measure the performance by recreating an existing domain-specific lexicon Hamilton et al. (2016); Mudinas et al. (2018), from an application point of view, it is interesting to see how much performance boost can we get from the domain-specific lexicon in a lexicon enhanced neural sentiment analysis system. In addition, we are also interested in expanding the lexicon to be aspect specific as the sentiment orientation of an opinion word is highly dependent on the target that it is describing (e.g. "*cheap price*" vs "*cheap plastic*").

2.3 Attention Regularization

Besides the lack of flexibility and robustness of a DNN model, it is less stressed that the commonly used attention mechanism is likely to over-fit and force the network to "focus" too much on a particular part of a sentence, while in some cases ignoring positions which provide key information for judging the polarity. In recent studies, both Niculae and Blondel (2017) and Zhang et al. (2019a) proposed approaches to make the attention vector more sparse; however, this would only encourage the over-fitting effect in such scenarios.

In Niculae and Blondel (2017), instead of using *softmax* or *sparesmax*, *fusemax* was proposed as a regularized attention framework to learn the attention weights. In Zhang et al. (2019a), L_{max} and *Entropy* were introduced as regularization terms to be jointly optimized within the loss function. Both approaches share the same idea of shaping the attention weights to be sharper and more sparse so that the advantage of the attention mechanism is maximized. However according to our experiments, it is possible that when applied early in the network, the overly sparse attention vector could hurt the model by not passing key information to deeper layers.

Thus in Bao et al. (2019), we proposed two regularizers to be jointly optimized within the loss function, namely a *standard deviation regularizer* and a *negative entropy regularizer*. These two regularizers aim to overcome the over-fitting effect of the attention weights by shaping it to be less sparse, i.e. instead of only a few positions have large weights and the rest being close to zero, it is preferred to have more positions with relatively higher weights. Details of our approach and experiments will be discussed in Chapter 5 and Chapter 6.

Chapter 3

OBJECTIVES

As mentioned in Section 2.1.5, deep learning based systems lack flexibility and robustness, especially when more training data are not available. Thus in this thesis, the first objective is to build a neural model for ABSA that is capable of leveraging lexicon information.

Secondly, as mentioned in 2.3, it is less stressed that the commonly used attention mechanism is likely to over-fit and force the network to ”focus” too much on a particular part of a sentence, while in some cases ignoring positions which provide key information for judging the polarity. Therefore, in this thesis, we aim to improve the attention LSTM network by regularizing the attention weights.

Thirdly, as mentioned in Section 2.2.3, from an application point of view, we are interested in seeing how much performance boost can actually be gained from a domain-specific lexicon in a lexicon enhanced neural sentiment analysis system. In addition, we would also like to explore the possibility of expanding the lexicon to be aspect specific and evaluate its performance.

Formally, the objectives of this thesis can be orchestrated as 3 main objectives followed by a few sub-objectives of each:

- Improve ABSA with lexicon enhancement in an attention LSTM neural network setting.
  - Replicate the AT-LSTM model (Wang et al., 2016c) as baseline.
  - Form a sentiment lexicon in the generic domain.
  - Experiment different approaches for merging the generic sentiment lexicon with the baseline.
- Improve the attention LSTM model for ABSA by overcoming the attention over-fit effect.
  - Experiment different approaches for attention regularization.
- Improve the lexicon enhanced neural ABSA system with sentiment induction.
  - Domain and aspect adaptation of generic sentiment lexicon.
  - Construct a gold domain specific lexicon.
  - Experiment and evaluate the performance gain of the induced lexicons in the ABSA model.

Chapter 4

THEORETICAL FRAMEWORK

In this thesis, machine learning methods serve as fundamental building blocks of our research. Specifically, we focus on improving the ABSA task using language resources with *Long Short-term Memory (LSTM)* and *Attention*, which are based on *Neural Networks (NN)* and *Recurrent Neural Networks (RNN)* in particular. In addition, *Support Vector Machine (SVM)* and distributed word representations (*Word Embeddings*) also play an important role for sentiment induction. In this chapter, we will first briefly review some of the classical machine learning methods, and then cover the more recent deep learning based methods that are adopted in our research. In addition, we will also briefly introduce *cross-validation*, the commonly used method to evaluate a machine learning model.

4.1 Classical Machine Learning Methods

Machine learning, often seen as a part of *Artificial Intelligence*, is the study of computer algorithms that improve automatically through experience and by the use of data. A machine learning algorithm is an algorithm that is able to learn from data, thus the data is also known as training data. In Mitchell (1997), the definition of learning is given: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

A task T for a machine learning algorithm is often the problems that are hard to solve with fixed programs written and designed by human beings. For instance, NLP problems such as sentiment analysis, machine translation, etc. are hard to solve with fixed programs due to the variability and diversity of languages. Other examples include regression, image classification, transcription, and so on.

The performance measure P is usually a quantitative measure, specific to the task T carried out by the system. For tasks such as classification, it is common to take the accuracy as a measure, which is the proportion of examples for which the model produces the correct output. On the other hand, it is very important to see how machine learning algorithms perform on data that they have not seen before. Therefore, a test data set that is separate from the training data set is often used to measure the performance of a system. More details about evaluation of a machine learning system will be introduced in Section 4.8.

The experience E of a machine learning algorithm can be categorized as *unsupervised* or *supervised* according to what kind of training experience is given. Unsupervised learning algorithms in a classical machine learning context usually refers to clustering, which consists of dividing the dataset into clusters of similar examples. In such a process, there are no labels or targets. Supervised learning algorithms on the other hand experience the data with labels or targets, which are essential for the algorithm

to learn. In this section, we only cover briefly some of the commonly used classical machine learning algorithms as they may appear in the literature review but are not directly related to our research. In Mitchell (1997) and Hastie et al. (2009), all concepts are explained in great depth.

4.1.1 Unsupervised Learning

K-means Clustering

K-means (MacQueen, 1967) is perhaps the most popular clustering algorithm due to its simplicity and scalability. As a centroid based algorithm, it requires a prior definition of the number of clusters K in advance, i.e. how many clusters does the user want. With the number of clusters defined, the algorithm initializes randomly K centroids and assigns the data points to their nearest centroid based on a distance measure. The distance is usually Euclidean distance, but other distances can also be applied. After grouping the data points into their nearest centroid, the algorithm computes a new centroid within each cluster to be the mean vector. Then move the centroid to the new mean position and repeat the distance based assigning process. After a number of iterations, the algorithm converges by minimizing the intra cluster average dissimilarity to the mean. More recently, Sculley (2010) proposed a mini-batch gradient descent approach to improve both the time and space complexity of the K-means algorithm.

Affinity Propagation

The biggest problem for most clustering algorithms including K-means is that they require prior definition of the number of clusters. In practice, the process of finding the optimal number of clusters often requires a lot of efforts. Frey and Dueck (2008) proposed a new exemplar based clustering algorithm named affinity propagation. Instead of moving the centroids to minimize the intra cluster distance with the means, affinity propagation

tries to maximize the total similarity by finding exemplars that best represent surrounding data points. The method requires no prior definition of the number of clusters. However, the complexity of affinity propagation in both time and space is quadratic as the operations of finding exemplars are based on a similarity matrix containing all input data in both dimensions.

Hierarchical Clustering

The hierarchical clustering method does not require manual input of the number of clusters. However, a measure of dissimilarity between clusters is needed to specify. The hierarchical clustering method produces hierarchical representations in which the clusters at each level of the hierarchy are created by merging lower level clusters. The number of clusters decreases from lower level to higher level. In a sense that at the lowest level, each cluster contains a single data point; and at the highest level, only one cluster exists containing all data points.

Strategies for hierarchical clustering divide into two basic paradigms: agglomerative (bottom-up) and divisive (top-down). Agglomerative strategies start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. The pair chosen for merging consist of the two groups with the smallest inter-group dissimilarity. Divisive methods start at the top and at each level recursively split one of the existing clusters at that level into two new clusters. The split is chosen to produce two new groups with the largest between-group dissimilarity. With both paradigms there are $N - 1$ levels in the hierarchy (Hastie et al., 2009).

Principal Components Analysis

Principal Components Analysis (PCA) is a sequence of projections of the data, mutually uncorrelated and ordered in variance (Hastie et al., 2009). It is often used to reduce feature dimensions while not losing too much

information in the original higher dimension space. It is essentially getting a sequence of best approximations of the original data by learning a projection matrix that minimizes the reconstruction error. With the learned projection parameters, one can project the data into a lower dimension.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a three-level hierarchical Bayesian model for collections of discrete data such as text corpora. It is commonly used for topic modelling where given a piece of document, the algorithm should find its topic. In LDA, each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document (Blei et al., 2003).

4.1.2 Supervised Learning

Naïve Bayes Classifier

Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayesian theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Let y be the class of a classification task, $x = [x_1, x_2, ..., x_n]$ be the feature vector. According to the Bayesian theorem:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

With the naive assumption that each feature is independent from each other, the relationship can be simplified to:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

Since the denominator is constant given the input, the estimation of a Naïve Bayes classifier is essentially:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y) \prod_{i}^{n} P(x_i|y)$$

Although the conditional independence assumption is rather over-simplified, Naïve Bayes classifier has worked well in many real-world applications such as document classification and spam filtering; as they require a relatively smaller amount of training data to estimate the needed parameters (Zhang, 2004).

Hidden Markov Model

The *Hidden Markov Model (HMM)* is based on augmenting the Markov chain. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. Formally, consider a sequence of state variables $q_1, q_2, ..., q_i$, a Markov model embodies the Markov assumption on the probabilities of this sequence: that when predicting the future, the past does not matter, only the present.

$$P(q_i = a|q_1, ..., q_{i-1}) = P(q_i = a|q_{i-1})$$

A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don't observe them directly. For example we don't normally observe part-of-speech tags in a text. Rather, we see words, and must infer the tags from the word sequence. We call the tags hidden because they are not observed. HMM allows us to talk about both

observed events and hidden events that we think of as causal factors in our probabilistic model. (Jurafsky and Martin, 2020)

Maximum Likelihood

Maximum likelihood is the general name of fitting methods that trains models by finding the model parameters that maximize the probability of some inputs mapped to the target outputs. Commonly used methods such as minimizing a sum of squared errors for regression, or minimizing the cross-entropy for classification are both examples of maximum likelihood (Hastie et al., 2009).

K Nearest Neighbors

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point (Vanderplas, 2020). The K values stands for the minimum number of votes for assigning a class.

Logistic Regression

Logistic Regression (LR) is a linear model for classification rather than regression. It is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. Let $\boldsymbol{x}$ be the input feature vector and $\hat{y}$ be the output probability of a class. First, the hypothesis function $h(\boldsymbol{x})$ (Equation 4.1) computes a linear transformation of the input features, where $\boldsymbol{\theta}$ and b

are weights and biases (parameters) of the model. Then a logistic function (Equation 4.2) is applied to convert the input to a probability.

$$h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x} + b \tag{4.1}$$

$$\hat{y} = sigmoid(h(\boldsymbol{x})) \tag{4.2}$$

By applying maximum likelihood estimation, a set of parameters $\boldsymbol{\theta}$ and b are learned through training.

Support Vector Machines (SVM)

The deep learning renaissance began when Hinton et al. (2006) outperformed the RBF kernel SVM on the MNIST dataset with a neural network. Prior to that and until today, the support vector machine has been one of the most influential models in machine learning (Boser et al., 1992; Farhat, 1992). Comparing SVM to *Logistic regression (LR)*, the hypothesis function of LR is a linear combination of weights and input features (Equation 4.1). In SVM, instead of directly applying weights on the features, a kernel function is first used to transform the input features from the original space to a new space (Equation 4.3):

$$h(\boldsymbol{x}) = \sum_i \theta_i k(\boldsymbol{x}, \boldsymbol{x}^{(i)}) + b \tag{4.3}$$

where $x^{(i)}$ is a training example, $\boldsymbol{\theta}$ is a vector of coefficients. $k(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}^{(i)})$ is the kernel function, where $\phi(\boldsymbol{x})$ is a feature function to replace $\boldsymbol{x}$. The kernel function is exactly equivalent to preprocessing the data by applying $\phi(\boldsymbol{x})$ to all inputs, then learning a linear model in the new transformed space.

The most commonly used kernel is the *Gaussian kernel* (Equation 4.4)

$$k(\boldsymbol{u}, \boldsymbol{v}) = N(\boldsymbol{u} - \boldsymbol{v}; 0, \sigma^2 \boldsymbol{I}) \tag{4.4}$$

where $N(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ is the density function of the standard normal distribution. This kernel is also known as the *radial basis function(RBF)* kernel, because its value decreases along lines in $\boldsymbol{v}$ space radiating outward from $\boldsymbol{u}$ (Goodfellow et al., 2016).

Intuitively, the Gaussian kernel can be thought of as a kind of template matching, where a training example $\boldsymbol{x}$ associated with training label y becomes a template for class y. When a test point $\boldsymbol{x}'$ is near $\boldsymbol{x}$ according to euclidean distance, the Gaussian kernel has a large response, indicating that $\boldsymbol{x}'$ is very similar to the $\boldsymbol{x}$ template. The model then puts a large weight on the associated training label y. Overall, the prediction will combine many such training labels weighted by the similarity of the corresponding training examples (Goodfellow et al., 2016).

4.2 Deep Learning and Neural Networks

Deep learning is the application of *Artificial Neural Networks (ANN)*, *Neural Networks (NN)* for short, on learning tasks using networks of multiple layers. Inspired by the structure of the biological brain, neural networks consist of a large number of information processing units (neurons) organized in layers, which work in unison. It can learn to perform tasks such as classification, regression, etc. by adjusting the connection weights between neurons, resembling the learning process of a biological brain. The neural networks were once believed to be practical only with one or two layers and a small set of data due to the limitation of computing powers back in those days. However, in recent years, the practice of deep learning (i.e. neural networks with many layers trained on large datasets) has become possible thanks to the exponential growth of computing power and digitization.

The simplest form of neural network is called a *feedforward neural network*, or *multi layer perceptron (MLP)*, which is the most basic and essential deep learning model. The goal of a feedforward network is to approximate some function f. For example, for a classifier, $y = f(x)$ maps an input x to a category y. A feedforward network defines a mapping function $y = f(x; \theta)$ and learns the value of the parameter set θ that yields the best function approximation. (Goodfellow et al., 2016; Zhang et al., 2018a)

In Figure 4.1 a simple example of a feedforward neural network is given, in which we can see three layers in total denoted by $L^{(0)}$, $L^{(1)}$ and $L^{(2)}$ with colors yellow, blue and red respectively. The layer $L^{(0)}$ is also known as the *input layer*, the layer $L^{(1)}$ is called the *hidden layer*, and the layer $L^{(2)}$ is called the *output layer*. Typically, there are only one input layer and one output layer in a neural network; while there can be more than just one hidden layer, especially in a deep learning setting.

In the input layer $L^{(0)}$, the n dimensional vector $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ represents the input features. In the output layer $L^{(2)}$, the vector $\boldsymbol{a^{(2)}}$ con-

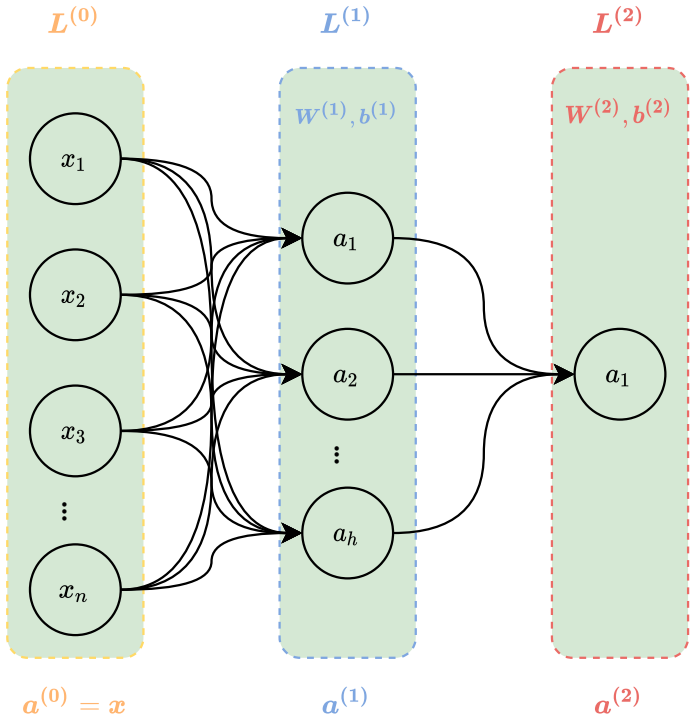

Figure 4.1: Diagram of a feedforward neural network.

sists of the output vector; in this simple example, $a^{(2)} = [a_1^2]$ in fact is a scalar as there is only one output unit in the output layer. In the hidden layer $L^{(1)}$, the h dimensional vector $\boldsymbol{a}^{(1)} = [a_1^1, a_2^1, ..., a_h^1]$ represents the hidden units of this layer. A circle in $L^{(0)}$ represents an element in the input vector, while a circle in $L^{(1)}$ and $L^{(2)}$ represents a neuron, the basic computation element of a neural network; which is also known as an activation function. A line between two neurons represents a connection for the information flow. Each connection is associated with a set of weights and biases (denoted as $W^{(1)}$, $b^{(1)}$ in $L^{(1)}$; and $W^{(2)}$, $b^{(2)}$ in $L^{(2)}$), a set of values con-

trolling the signal between two neurons. The learning of a neural network is achieved by adjusting the weights between neurons with the information flowing through them. Neurons read output from neurons in the previous layer, process the information, and then generate output to neurons in the next layer. As in Figure 4.1, the neutral network alters weights based on training examples (x_i, y_i). After the training process, it will obtain a complex form of a hypotheses function $h_{W,b}(x)$ that maps the data from input features to output predictions.

Formally, let $x \in \mathbb{R}^n$ be the input column vector of the network, the forward computation described above can be expressed as:

$$z^{(1)} = W^{(1)} \cdot x + b^{(1)}$$

$$a^{(1)} = g(z^{(1)})$$

$$z^{(2)} = W^{(2)} \cdot a^{(1)} + b^{(2)}$$

$$a^{(2)} = g(z^{(2)})$$

where $W^{(1)} \in \mathbb{R}^{h \times n}$, $b^{(1)} \in \mathbb{R}^h$, $W^{(2)} \in \mathbb{R}^h$, $b^{(2)} \in \mathbb{R}$ are weights and biases of the hidden layer and the output layer; $z^{(1)} \in \mathbb{R}^h$, $z^{(2)} \in \mathbb{R}$ are intermediate outputs of $L^{(1)}$ and $L^{(2)}$; $a^{(1)} \in \mathbb{R}^h$, $a^{(2)} \in \mathbb{R}$ are outputs of the activation function $g(z)$. An activation function is normally a non-linear differentiable function, common choices are *sigmoid function* (Equation 4.5), *hyperbolic tangent function* (Equation 4.6), or *rectified linear function* (Equation 4.7).

$$g(z) = sigmoid(z) = \frac{1}{1 + exp^{-z}} \tag{4.5}$$

$$g(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.6}$$

$$g(z) = ReLU(z) = max(0, z) \tag{4.7}$$

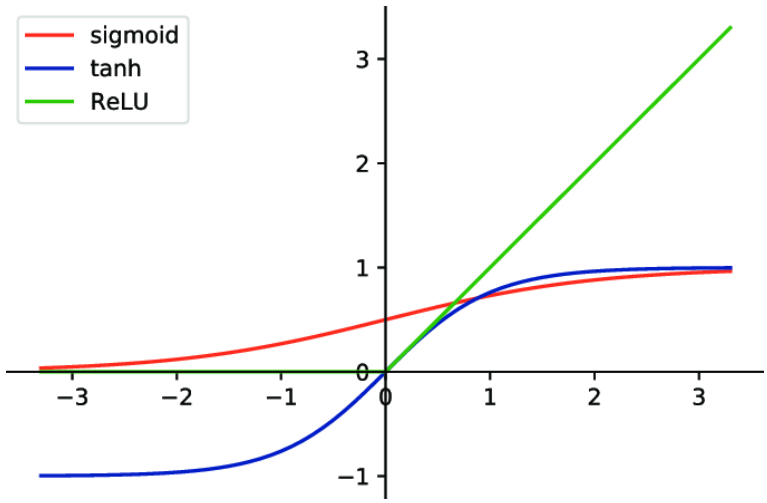

Figure 4.2: Visualization of activation functions.

As shown in Figure 4.2, the $sigmoid$ function takes a real number and squashes it to a value in the range between 0 and 1. The function has been in frequent use historically due to its nice interpretation as the firing rate of a neuron: 0 for not firing or 1 for firing. But the non-linearity of the sigmoid has recently fallen out of favour because its activations can easily saturate at either tail of 0 or 1, where gradients are almost zero and the information flow would be cut. What is more is that its output is not zero-centered, which could introduce undesirable winding dynamics in the gradient updates of the weights during training. Thus, the $tanh$ function is often preferred in practice as its output range is zero-centered ($[-1, 1]$ instead of $[0, 1]$). The $ReLU$ function has also become popular lately. Its activation is simply a threshold at zero when the input is less than 0. Compared with the $sigmoid$ function and the $tanh$ function, $ReLU$ is easy to compute, fast to converge in training and yields equal or better performance in neural networks (Glorot et al., 2011; Zhang et al., 2018a).

In the output layer $L^{(3)}$, instead of using $sigmoid$ to output a proba-

bility for binary classification, it is common practice to use the $softmax$ function (Equation 4.8) as activation for multi-class classification. It is a generalization of the logistic function that squashes a vector $\boldsymbol{z} \in \mathbb{R}^K$ of arbitrary real values to a vector $\sigma(\boldsymbol{z}) \in \mathbb{R}^K$ of real values in the range $(0, 1)$ that sum up to 1.

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \forall_i \in \{1, ..., K\} \tag{4.8}$$

To train a neural network, *Stochastic Gradient Descent (SGD)* via *Back Propagation* (Rumelhart et al., 1986) is usually employed to minimize the cross-entropy loss (Equation 4.9), which is a loss function for softmax outputs given $\boldsymbol{y} \in \mathbb{R}^i$ as the true probability distribution of the labeled classes, and $\hat{\boldsymbol{y}} \in \mathbb{R}^i$ as the predicted probability distribution of the classes (i is the number of classes). Gradients of the loss function with respect to weights from the last hidden layer to the output layer are first calculated, and then gradients of the expressions with respect to weights between upper network layers are calculated recursively by applying the chain rule in a backward manner. With those gradients, the weights between layers are adjusted accordingly. It is an iterative refinement process until certain stopping criteria are met. In practice, it is common to minimize the cost function (Equation 4.10) instead of the loss function (Equation 4.9). Compared to the loss function, a regularization term is added in the cost function: $\lambda\Omega(\boldsymbol{\theta})$, where λ is a hyperparameter, $\boldsymbol{\theta}$ contains all the parameters of the model (weights and biases), and Ω is the regularization function. The computing process that first forward computes the cost from the input, and then computes the gradient of all network parameters at each layer using back propagation are shown in Algorithm 1 and Algorithm 2 (Goodfellow et al., 2016).

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i y_i log(\hat{y}_i) \tag{4.9}$$

$$J(\boldsymbol{y}, \hat{\boldsymbol{y}}, \boldsymbol{\theta}) = L(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \lambda\Omega(\boldsymbol{\theta}) \tag{4.10}$$

Algorithm 1: Forward propagation of a feed forward neural network.

$l \leftarrow$ network depth/layers
$\boldsymbol{W}^{(i)}, i \in \{1, ..., l\} \leftarrow$ weight matrices of the network
$\boldsymbol{b}^{(i)}, i \in \{1, ..., l\} \leftarrow$ bias parameters of the model
$\boldsymbol{x} \leftarrow$ input features
$\boldsymbol{y} \leftarrow$ label
$\boldsymbol{z}^{(0)} = \boldsymbol{x}$
for $k \leftarrow 1$ **to** l **do**
  $\boldsymbol{z}^{(k)} = \boldsymbol{W}^{(k)} \boldsymbol{z}^{(k-1)} + \boldsymbol{b}^{(k)}$
  $\boldsymbol{a}^{(k)} = g(\boldsymbol{z}^{(k)})$
end
$\hat{\boldsymbol{y}} = \boldsymbol{a}^{(l)}$
$J = L(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \lambda \Omega(\boldsymbol{\theta})$

Algorithm 2: Back propagation of a feed forward neural network.

Compute the gradient on the output layer:

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\boldsymbol{y}, \hat{\boldsymbol{y}})$

for $k \leftarrow l$ **to** $l - 1, ..., 1$ **do**

Convert the gradient on the layer's output into a gradient on the pre-nonlinearity activation (element-wise multiplication if $g(z)$ is element-wise):

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{z}^{(k)}} J = \boldsymbol{g} \cdot g'(\boldsymbol{z}^{(k)})$

Compute gradients on weights and biases (including the regularization term where needed):

$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\boldsymbol{\theta})$

$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\boldsymbol{a}^{(k-1)\intercal} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\boldsymbol{\theta})$

Propagate the gradients w.r.t. the next lower-level hidden layer's activations:

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k-1)}} J = \boldsymbol{W}^{(k)\intercal} \boldsymbol{g}$

end

4.3 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) (Elman, 1990) is a class of neural networks that recursively processes the inputs and forms a circled information flow. Compared to feedforward neural networks, RNN is designed to process a sequence of inputs, where the order of the inputs actually matters. For instance, when processing a sentence, a feedforward neural network does not take into account the order of the words, which in fact is key in language processing. Thus in NLP, RNN has a natural advantage to encode sequence information in the model. It uses its internal "memory" to process a sequence of inputs, meaning that RNN performs the same task for every element of a sequence with each output being dependent on all previous computations. Intuitively, the process is similar to using "memory" to "remember" information that has been processed previously when processing the information at the current time step.

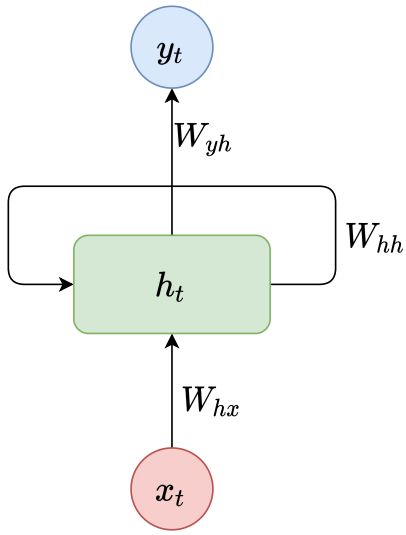

Figure 4.3: Diagram of RNN folded.

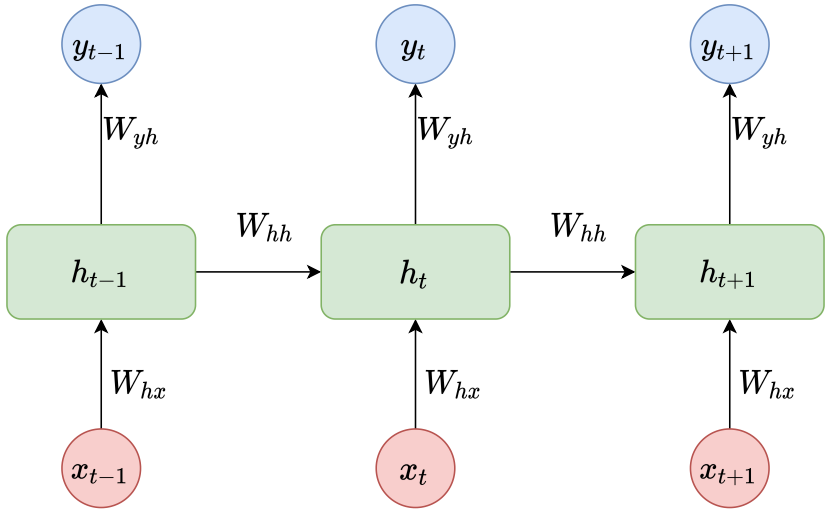

Figure 4.4: Diagram of RNN unfolded.

As shown in Figures 4.3 and 4.4, the circled RNN network (Figure 4.3) unfolds to a sequential network with different time steps (Figure 4.4). In a NLP setting, each time step corresponds to a word (a word vector to be precise) in a sentence, and the length of the sequence is equal to the length of the sentence. The unfolded network can be seen as a feedforward network with a number of layers equal to the length of the sequence, where each layer shares the exact same parameters (weights and biases).

Formally, let x_t be the input vector at time step t; h_t be the hidden state at time step t, which is computed based on the hidden state at time step $t-1$ and the input at time step t (Equation 4.11):

$$h_t = g(W_{hh}h_{t-1} + W_{hx}x_t) \tag{4.11}$$

where $g(z)$ is the activation function (e.g. $tanh$ or $ReLU$). W_{hx} is the weight matrix to be applied on the input x_t. W_{hh} is the weight matrix to be applied on the previous hidden state h_{t-1}. y_t is usually the output

probability distribution over the vocabulary set at time step t (Equation 4.12).

$$y_t = softmax(W_{yh}h_t) \tag{4.12}$$

The hidden state h_t is regarded as the memory of the network. It captures information about what happened in all previous time steps. y_t is calculated solely based on the memory h_t at time step t and the corresponding weight matrix W_{yh}. Unlike a feedforward neural network, which uses different parameters at each layer, RNN shares the same parameters (W_{hx}, W_{hh}, W_{yh}) across all time steps. This means that it performs the same task at each step, just with different inputs. This greatly reduces the total number of parameters needed to learn. Theoretically, RNN can make use of the information in arbitrarily long sequences, but in practice, the standard RNN is limited to looking back only a few steps due to the *vanishing gradient* or *exploding gradient* problem (Bengio et al., 1994; Zhang et al., 2018a). As governed by the chain rule, the derivative of each layer (from final output layer to initial input layer) is a multiplication of the derivatives from previous layers; thus when the derivative is a small number or a big number, the gradient decreases (vanishing) or increases (exploding) exponentially, making it impossible to train the model.

4.4 Long Short-term Memory (LSTM)

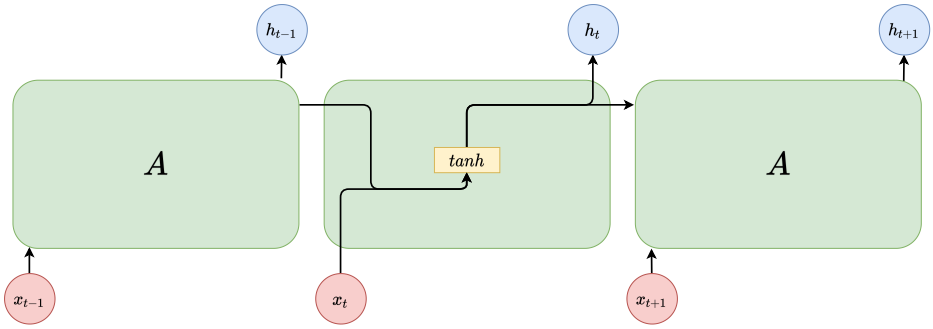

Figure 4.5: RNN with single layer.

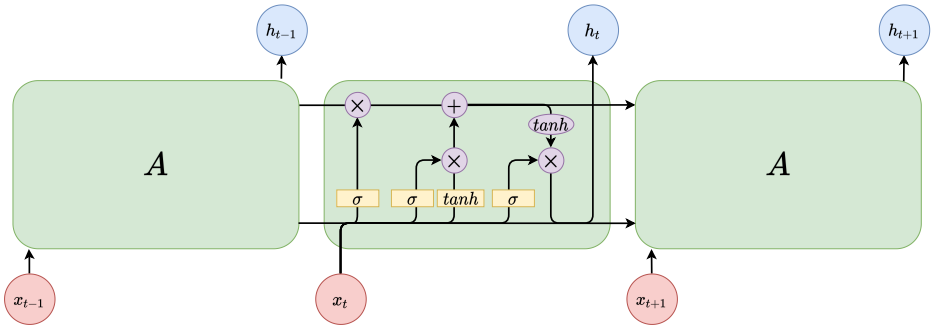

Figure 4.6: RNN with LSTM cell.

In order to overcome the vanishing/exploding gradient problem and have the RNN network to learn longer term dependencies, a special type of computing unit named Long Short-term Memory (LSTM) was proposed by Hochreiter and Schmidhuber (1997). Compared to a regular computing unit (also called computing cell), which is usually identical to a simple hidden layer of a feed forward neural network (Figure 4.5), the LSTM cell

consists of a more complicated structure that contains four different layers interacting in a carefully designed fashion (Figure 4.6).

As shown in Figure 4.7, two flows pass through the LSTM cell, namely a hidden state h and a cell state C. At each time step t, a "forget gate" is first applied to control what information is allowed to pass through, i.e. what information to dump from the cell state. The decision is made by a sigmoid function/layer σ named **forget gate**, which takes the concatenation of the input at current time step x_t and the hidden state from previous time step h_{t-1} as joint inputs, and outputs a number between $[0, 1]$ that controls the amount of information to "forget". For example, 1 means keep all and 0 means dump all. Formally, the output of the forget gate f_t is computed as (Equation 4.13):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.13}$$

Secondly, an "input gate" is adopted to control how much new information is allowed to add to the cell state. It consists of two steps: a) a sigmoid function/layer σ named **input gate** (Equation 4.14) that controls which values LSTM will update; b) a $tanh$ function/layer creates a vector of new candidate values $\tilde{C}$ (Equation 4.15) which will be used for updating the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.14}$$

$$\tilde{C}_t = ttanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.15}$$

Thirdly, with the input gate i_t and the new cell candidate $\tilde{C}_t$, together with the forget gate f_t, the old cell state from previous time step C_{t-1} is updated to be the current cell state C_t (Equation 4.16).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.16}$$

Finally, an "output gate" is applied to decide how much of the cell state can be used as output to next time step, where a sigmoid function/layer is used on x_t and h_{t-1} to first learn the **output gate** o_t (Equation 4.17). Then the output gate is applied on the current cell state C_t after a nonlinear transformation (Equation 4.18).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4.17}$$

$$h_t = o_t * tanh(C_t) \tag{4.18}$$

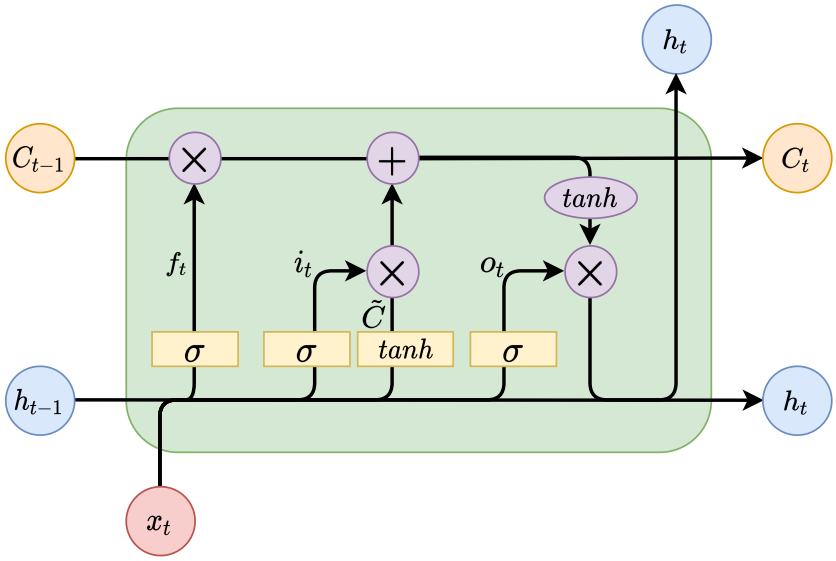

Figure 4.7: Diagram of a LSTM cell.

More recently, a lot of variations of LSTM have been proposed. For instance, Gers and Schmidhuber (2000) proposed a variant of the LSTM cell by adding "peephole connections". In Chung et al. (2014); Cho et al.

(2014), another variation named *Gated Recurrent Unit (GRU)* was proposed. It combines the "forget" and "input" gates into a single update gate, and it also merges the cell state and hidden state along with some other changes. The resulting structure is simpler than the standard LSTM, and has been growing in popularity. Koutník et al. (2014) took a completely different approach to tackle long-term dependencies with *Clockwork RNNs*. Yao et al. (2015) on the other hand proposed *Depth Gated RNNs*.

4.5 Attention Mechanism

The idea of allowing the model to "look" back at the input sequence and "focus" on different parts accordingly has been one of the most influential ideas in the NLP world in recent times. Through the years, it has inspired many great innovations such as Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019). The attention mechanism consists of learning a weight vector in the model and applying it to obtain a weighted representation of the input. It was first proposed in the sequence-to-sequence model for neural machine translation in order to overcome poor performance on translating long sentences (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015).

In fact, the attention mechanism is inspired by the visual attention mechanism found in humans. That is, the human visual attention is able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution" and then adjusting the focal point over time. In NLP, the attention mechanism allows the model to learn what to attend to based on the input text and what it has produced so far, rather than encoding the full source text into a fixed-length vector like standard RNN and LSTM (Zhang et al., 2018a).

As shown in Figure 4.8, in order to obtain a weighted representation r of the input sequence with a length of T, given $H \in \{h_1, h_2, h_3, ..., h_T\}$ as the hidden states of each time step produced by the LSTM cell, a weight vector $\alpha \in \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_T\}$ is first learned from H (Equation 4.19 and 4.20):

$$M = tanh(W_h H) \tag{4.19}$$

$$\alpha = softmax(w^\intercal M) \tag{4.20}$$

where W_h and w are parameters of the network to be learned during training. Each value of α represents the level of attention to be paid at each position when obtaining the final representation, where a higher value in-

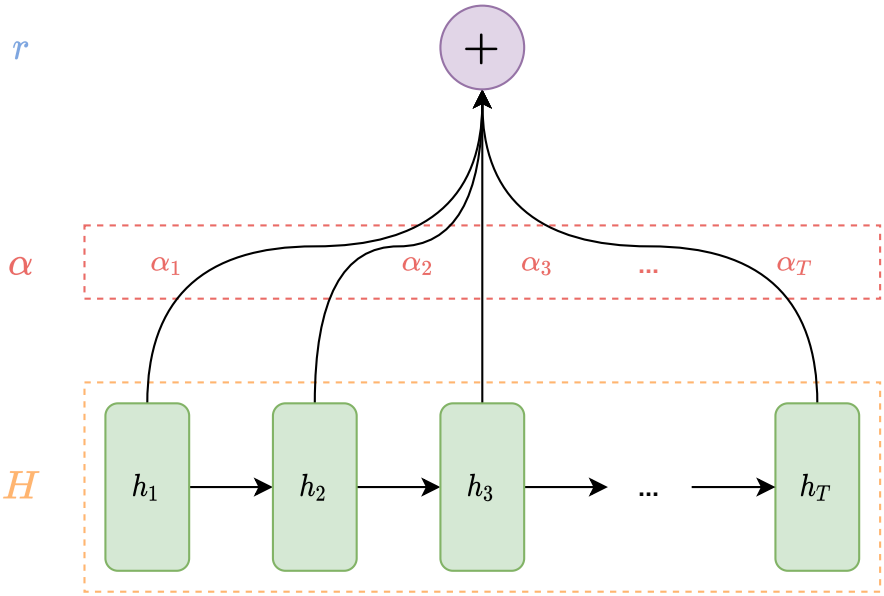

Figure 4.8: Attention mechanism in a LSTM network.

dicates more attention and vice versa. Since the outputs of the $softmax$ function are weights between $[0, 1]$ that sum up to 1, the weighted representation r is essentially a weighted sum of H (Equation 4.21) with α as weights.

$$r = H\alpha^{\intercal} \tag{4.21}$$

4.6 Word Embeddings

Distributed word representation, also known as word embeddings have been a fundamental building block of almost all deep learning based NLP applications. It is a technique for language modelling and feature learning that transforms words into vectors in a continuous space. Before word embeddings, a word is usually represented by a sparse one-hot vector or a frequency count in representations such as TF-IDF. These representations can only encode binary information of whether a word has occurred and are often high dimensional (the dimension of a one-hot vector is usually equal to the vocabulary size of a corpus); while the context information of a word and its semantic relation with others are ignored, which are keys for language understanding. The distributed word representation improves on such matter by learning a lower dimensional dense vector from the higher dimensional sparse vector through embedding techniques, where each dimension of the embedded vector represents a latent feature of a word. Therefore the embedded vectors can encode linguistic regularities, patterns and even context information.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Figure 4.9: Simplified embedding look up example.

Traditionally, word embeddings are learned as by-products of neural networks. For instance, when training a neural language model, the input one-hot word vectors are first transformed through an embedding layer which consists of a weight matrix with randomly initialized parameters (Figure 4.9). The output of the embedding layer is then fed to further layers for the language modelling task. Since the parameters in the embedding

layer form part of the model parameter set and get updated during training, when the training process is completed, the weight matrix of the embedding layer is encoded with task specific information. In the context of language modelling, the embedded vectors will contain some information of the context, and this kind of embedding vectors are also called task-specific embeddings.

Another commonly used technique to learn word embeddings is *word2vec* (Mikolov et al., 2013a,b), which consists of a computationally efficient neural network with a single linear hidden layer. As shown in Figure 4.10, two different methods can be used to learn word embeddings, namely the *Continuous Bag-of-Words (CBOW)* model and the *Skip-Gram* model. The CBOW model predicts a target word (v_t) given its context words ($v_{t-2}, v_{t-1}, v_{t+1}, v_{t+2}$), while the skip-gram model does the inverse by predicting the context words given a target word.

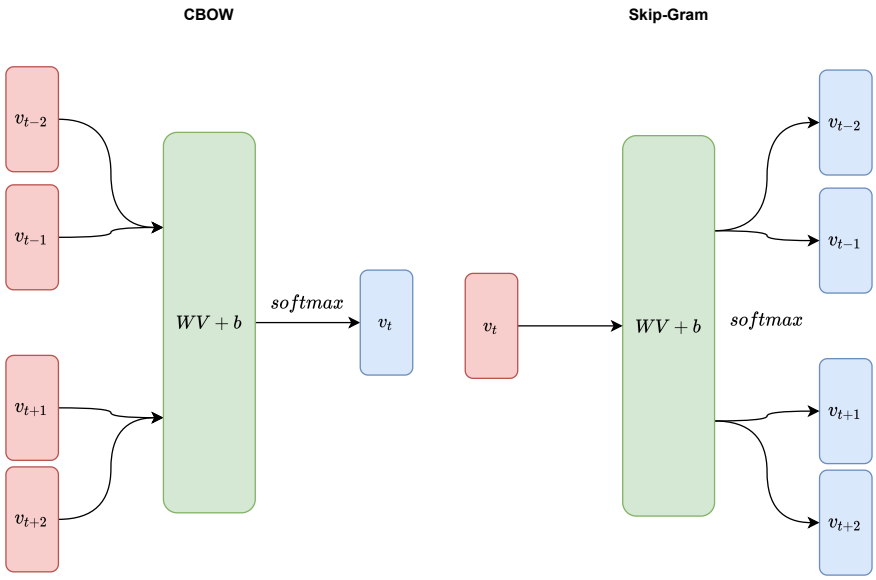

Figure 4.10: CBOW and Skip-Gram models of word2vec.

Statistically, the CBOW model smoothens over a great deal of distributional information by treating the entire context as one observation. It is effective for smaller datasets. However, the skip-gram model treats each context-target pair as a new observation and is usually better for larger datasets. Another frequently used learning approach is Global Vectors (GloVe) by Pennington et al. (2014), which is trained on the non-zero entries of a global word-word co-occurrence matrix (Zhang et al., 2018a).

As shown in Figure 4.11, one key property of word embeddings is that it allows vector space operations to coincide with linguistic relations. For instance, male-female analogies can be made by measuring the similarity between the vector $v_{king} - v_{queen}$ and the vector $v_{man} - v_{woman}$, where the two vector operations yield similar outputs. That is to say that a linguistic relation is able to be expressed in a mathematical way that $v_{king} - v_{queen} = v_{man} - v_{woman}$. Similar relations such as singular-plural, verb tense, country-capital, etc. can all be expressed in the same manner. And due to the fact that word embeddings are learned predicting a target word given its context words (or vice versa), in the embedding vector space, words with similar contexts form clusters among others, and it is independent of domain and language.

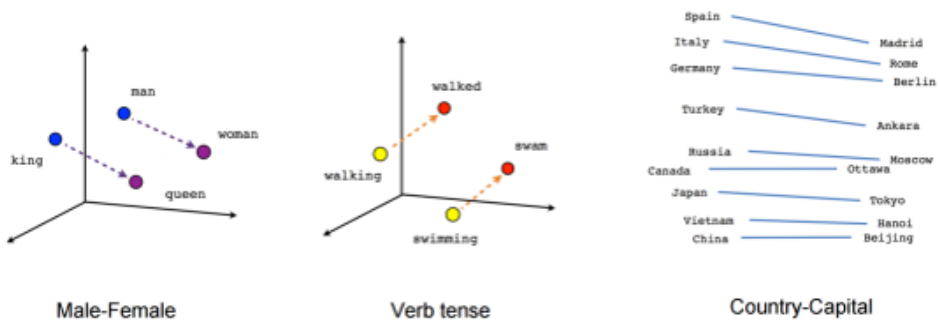

Figure 4.11: Example of word analogies in the vector space.

Sometimes, applying pre-trained word embeddings in different down-

stream tasks such as sentiment analysis is also referred as *Transfer Learning*, which means applying something learned from a different or a general task, on some other specific tasks. For instance, in computer vision, it is common to train a network on image classification where the first layers can already learn how to detect edges or smaller blocks of pixels. And then, for a different task such as image caption, instead of training a network from scratch, it is common to connect the input data to the first layers of the pre-trained model and use the outputs as processed features for new task specific layers.

4.7 Other Deep Learning Methods

Deep learning definitely has been thriving in recent years, despite the methods described in previous sections which are directly related to our research, there are a great number of methods that are important in the field. Here in this section, we briefly mention some of the relevant ones that may appear in the literature review.

Convolutional Networks

Convolutional networks LeCun et al. (1998), also known as *Convolutional Neural Networks (CNN)* are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals; and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional" indicates that the network employs a mathematical operation called convolution, which is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers (Goodfellow et al., 2016).

Transformer

Based on the idea of attention weights, Vaswani et al. (2017) proposed a pure attention model with multiple heads for machine translation. Compared to a regular attention mechanism, the Transformer model stacks multiple attention layers (also called heads) together in both the encoder and decoder, so that each layer can learn to "focus" on different parts of the input sentence. Another novel aspect of the Transformer is that it uses a positional encoding to represent the sequence information instead of a recurrent structure like LSTM.

Pre-trained Language Models

More recently, based on the Transformer design, the large pre-trained language models such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020) have become popular due to their impressive performance on many language related tasks. A language model can be understood as a model that predicts the following words given the previous words in a sentence. As both the input and the label for training a language model can be found easily in the corpus by splitting a sentence, a language model can be trained on an extremely large corpus. Moreover, models such as BERT and GPT-3 both are extremely large models with billions of parameters. With big models trained on big data, the pre-trained language models are essentially generalized language encoders that are capable of encoding linguistic and semantic relations in a high dimensional space. Thus, for any downstream NLP task, one could use the pre-trained language model to encode the input features and then connect the hidden layer outputs to a task specific layer for fine tuning. This approach is also known as *Transfer Learning*.

Deep Reinforcement Learning

Deep Reinforcement Learning (Deep RL) is a particular type of RL, with deep neural networks for state representation and/or function approximation for value function, policy, transition model, or reward function (Wang et al., 2018). Compared to other machine learning (ML) paradigms that train a model, RL trains an ”agent” that interacts with the environment over time. Instead of giving instant feedback from the loss function in a ML setting, RL agents make a series of decisions that finally lead to a terminal state, and then return the discounted, accumulated reward given by the reward function. The most famous example of RL perhaps is the AlphaGo developed by Deep Mind that plays the game of Go.

General Adversarial Network

General Adversarial Network (GAN) (Goodfellow et al., 2014) consists of two neural networks that contest with each other in a game (in the form of a zero-sum game, where one agent's gain is another agent's loss). The two networks are: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. In practice, the generative model after training is usually more interesting for application as it learns to generate data that is very similar to the original training distribution. Examples include generating arts and up-scaling low resolution pictures to high resolution pictures.

4.8 Evaluation of a ML System

4.8.1 Training and Test Errors

The central challenge in machine learning is that a trained model must perform well not only on data that have been seen, but also on data that are previously unseen. In other words, the key that defines the success of a model is its ability to generalize.

Typically, when training a machine learning model, we have access to a dataset, named training set. We can compute some error measure on the training set, called the training error, and we train the model by reducing this training error. So far, it is simply an optimization problem. What separates machine learning from optimization is that we want the generalization error, also called the test error, to be low as well. The generalization error is defined as the expected value of the error on a new input. Here the expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.

However in reality, we usually only have access to the training set. In order to effectively measure the test error, some assumptions based on the statistical learning theory are made. That is, the training and test data are generated by a probability distribution over datasets called the data-generating process. We typically make a set of assumptions known collectively as the *i.i.d.* assumptions. These assumptions are that the examples in each dataset are independent from each other, and that the training set and test set are identically distributed, drawn from the same probability distribution as each other. This assumption enables us to describe the data-generating process with a probability distribution over a single example. The same distribution is then used to generate every train example and every test example. We call that shared underlying distribution the data-generating distribution (Goodfellow et al., 2016).

Based on these assumptions, we can split the complete dataset into a training set and test set to evaluate the error on both (the test set here is also

referred to as holdout test set). Typically when training a machine learning algorithm, the factors that determine a good model are:

1. A small training error.

2. A small gap between training and test error.

These two factors also correspond to two central challenges in machine learning: under-fitting and over-fitting. Under-fitting occurs when the model is not able to fit well on the training set, i.e. having a training error that is sufficiently low. On the contrary, over-fitting occurs when the gap between training error and test error is too big, i.e. only the training error is low. In both cases, the model fails to generalize.

4.8.2 Validation Sets

Usually, a machine learning model involves some hyperparameters (e.g. learning rate for gradient descent, λ in the regularization term, number of layers in a neural network, etc.), which are keys to the performance. As mentioned in the previous section (Section 4.8.1), only having a low training error is no guarantee of having an overall good model; but if we select the hyperparameters using the test error, it will be against our idea of evaluating the model on data that has never seen.

Hence, a validation set comes in place, which is a subset of the training set. Specifically, we split the training data into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is the validation set, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly.

However, dividing the dataset into a fixed training set and a fixed validation set can be problematic if it results in the validation set being small. A small validation set implies statistical uncertainty around the estimated average error. To overcome this, alternative procedures enable one to use

all the examples in the estimation of the mean validation error, at the price of increased computational cost. These procedures are based on the idea of repeating the training and validation computation on different randomly sampled subsets or splits of the original training set. The commonly used method is called the k-fold cross-validation procedure, in which a partition of the dataset is formed by splitting it into k non overlapping subsets. The validation error may then be estimated by taking the average validation error across k trials. On trial i, the i-th subset of the data is used as the validation set, and the rest of the data is used as the training set.

Chapter 5

METHODOLOGY

Previously in Section 1.2, 1.5.1 and 2.1.5, we discussed that as an end-to-end approach, deep learning based systems lack flexibility and robustness because one cannot easily adjust the network to fix an obvious problem: e.g. when the model always predicts *positive* when seeing the word "*disappointed*", or the network is not able to recognize the word "*dungeon*" as an indication of *negative* polarity. It will be even harder to fix it when more labeled training data are simply not available. Thus in Chapter 3, our first objective in this thesis is to improve deep learning based ABSA systems with lexicon enchantment. In this chapter, we describe the details of our approach (named ATLX) for merging a sentiment lexicon with the neural ABSA baseline (Section 5.2). The experiment details of ATLX will be covered in Section 6.1 of Chapter 6.

Later, as described in Section 2.3, during the development of ATLX we noticed that the commonly used attention mechanism is likely to over-fit and force the network to "focus" too much on a particular part of a sentence, while in some cases ignoring positions which provide key information for judging the polarity. Recent studies (Niculae and Blondel, 2017; Zhang et al., 2019a) proposed approaches to make the attention vector more sparse; however, this would only encourage the over-fitting effect

in such scenarios. Thus, in this chapter, we also describe the details of our approach to regularize the attention weights in order to overcome the over-fitting effect (Section 5.3), which corresponds to our second objective described in Chapter 3. Experiments and discussions on attention regularization will be covered in Section 6.2 of Chapter 6.

Finally, as mentioned in Section 2.2.3, with the ATLX model which is enhanced by a simple generic domain lexicon, it is interesting to see whether it is possible to further improve the ATLX model with a more fine-grained lexicon (domain-specific and aspect-specific). In addition, most works on sentiment domain adaptation measure the performance by recreating an existing domain-specific lexicon Hamilton et al. (2016); Mudinas et al. (2018). From an application point of view, it is also interesting to see how much performance boost can be obtained from the domain-specific/aspect-specific lexicon in a lexicon enhanced neural sentiment analysis system. Thus, regarding the third objective described in Chapter 3, in this chapter, we will describe our detailed approach of domain/aspect adaptation and how we constructed a gold lexicon for comparison and evaluation (Section 5.4). Related experiments will be described in Section 6.3 of Chapter 6.

The ATLX experiments and attention regularization experiments are conducted on SemEval 14, Task 4, restaurant domain dataset. The domain/aspect adaptation experiments are conducted on SemEval 15, Task 12, laptop dataset. Details of the datasets will be described in Chapter 6.

5.1 Baseline AT-LSTM

In this thesis, we replicate the AT-LSTM model proposed by Wang et al. (2016c) as our baseline system as it was the state of the art system for ABSA when this research started. Compared to a traditional LSTM network Hochreiter and Schmidhuber (1997), AT-LSTM is able to learn the attention vector and at the same time to take into account the aspect embeddings; thus the network is able to assign higher weights to more relevant parts of a given sentence with respect to a specific aspect. For instance, in the case of "*Staffs are not that friendly, but the taste covers all.*", given the aspect *service*, the network should pay more attention to the first clause.

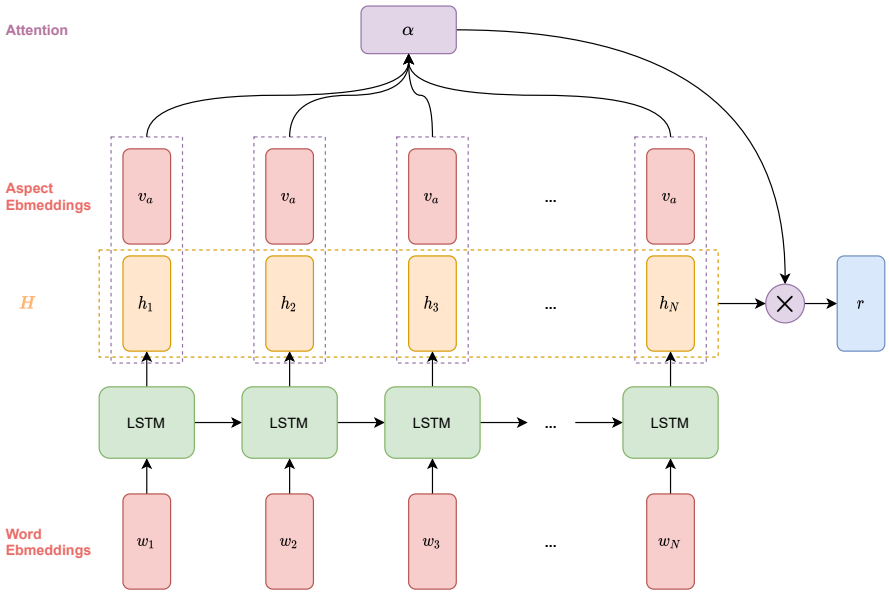

Figure 5.1: AT-LSTM model architecture.

As shown in Figure 5.1, the AT-LSTM model consists of an attention mechanism on top of a LSTM network, where the attention weights are

learned through a concatenation of the hidden states and the aspect embedding vector. The learned attention vector is then applied to the hidden states to produce a weighted representation of the whole sentence.

Formally, given a sentence S, let $\{w_1, w_2, ..., w_N\}$ be the word vectors of each word in S where N is the length of the sentence. $v_a \in \mathbb{R}^{d_a}$ represents the aspect embeddings where d_a is its dimension. Let $H \in \mathbb{R}^{d \times N}$ be the matrix of the hidden states $\{h_1, h_2, ..., h_N \in \mathbb{R}^d\}$ produced by the LSTM network where d is the number of neurons of the LSTM cell. Thus the attention vector α is computed as (Equation 5.1 and 5.2):

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix}) \tag{5.1}$$

$$\alpha = softmax(w^\intercal M) \tag{5.2}$$

and the weighted sentence representation r is computed as (Equation 5.3):

$$r = H\alpha^\intercal \tag{5.3}$$

where, $M \in \mathbb{R}^{(d+d_a) \times N}$, $\alpha \in \mathbb{R}^N$, $r \in \mathbb{R}^d$, $W_h \in \mathbb{R}^{d \times d}$, $W_v \in \mathbb{R}^{d_a \times d_a}$, $w \in \mathbb{R}^{d+d_a}$. α is a vector consisting of attention weights and r is the weighted representation of the input sentence with respect to the input aspect. $v_a \otimes e_N = [v_a, v_a, ..., v_a]$, is the operator that repeatedly concatenates v_a for N times. Then, the final representation h^* is obtained by (Equation 5.4):

$$h^* = tanh(W_p r + W_x h_N) \tag{5.4}$$

and fed to the output $softmax$ layer for prediction (Equation 5.5):

$$\hat{y} = softmax(W_s h^* + b_s) \tag{5.5}$$

where, $h^* \in \mathbb{R}^d$, W_p and W_x are projection parameters to be learned during training. W_s and b_s are weights and biases in the output layer. The prediction $\hat{y}$ is then plugged into the cross-entropy loss function for optimization,

and L_2 regularization is applied (Equation 5.6):

$$loss = -\sum_{i} y_i log(\hat{y}_i) + \lambda \|\mathbf{\Theta}\|_2^2 \tag{5.6}$$

where i is the number of classes (ternary classification in our experiments). λ is the hyperparameter for L_2 regularization. And $\mathbf{\Theta}$ is the parameter set of the network to be regularized.

5.2 ATLX

5.2.1 Lexicon Build

To effectively merge a sentiment lexicon into the baseline model, we first build our lexicon by merging 4 existing lexicons to one. Namely, *MPQA*[1], *Opinion Lexicon*[2], *Opener*[3] and *Vader*[4]. *SentiWordNet* was in the initial design but has been removed from the experiments as it introduced unnecessary noise: e.g. "*highly*" is annotated as *negative*. There is no specific reason for us to select any particular lexicon; as all four lexicons are open source, easily accessible, and domain independent, we select them out of convenience.

After gathering the resources, we have to standardize the polarities in these lexicons as they are not annotated with the same standard. Specifically, for lexicons with categorical labels such as *negative*, *weakneg*, *neutral*, *both*, *positive*, we convert them into numerical values as {*-1.0, -0.5, 0.0, 0.0, 1.0*} respectively. On the other hand, regarding lexicons with real number annotations, for each lexicon, we adopt the annotated value normalized by the maximum absolute polarity value in that lexicon. Namely, let $\boldsymbol{p} \in \{p_1, p_2, ..., p_n\}$ be the set of unique numerical polarities of a given lexicon, the normalized polarity p_i is computed as (Equation 5.7):

$$p_i = \frac{p_i}{max(|\boldsymbol{p}|)} \quad \forall_i \in \{1, 2, ..., n\} \tag{5.7}$$

Finally, the union U of all lexicons is taken where each word $w_l \in U$ has an associated vector $v_l \in \mathbb{R}^n$ that represents the polarities given by each lexicon (n here is the number of lexicons). Average values across all available lexicons are taken for missing values. For example, the lexical

[1]https://bit.ly/2Ia4u74
[2]https://bit.ly/36JNmPN
[3]https://bit.ly/3iIrnv1
[4]https://bit.ly/3jJ95uH

Word	MPQA	Opener	OL	Vader
adorable	1.0	1.0	1.0	0.55
accomplished	0.74	0.74	1.0	0.48
bravo	1.0	1.0	1.0	1.0
broke	-1.0	-1.0	-1.0	-0.45
complete	0.0	0.0	0.0	0.0
costly	-1.0	-1.0	-1.0	-1.0

Table 5.1: Example of the merged lexicon U.

features of the word "*adorable*" are represented in the vector $[1.0, 1.0, 1.0, 0.55]$, whose values are taken from *MPQA* (1.0), *Opener* (1.0), *Opinion Lexicon* (1.0) and *Vader* (0.55) respectively. For words outside of U, a zero vector of dimension n is supplied. As an example, Table 5.1 shows a small portion of the merged lexicon.

5.2.2 Lexicon Integration

As shown in Figure 5.2, on top of the baseline model, a new set of inputs consisting of lexical features are introduced, where each vector is the lexical features of a word given by the lexicon union set U. To merge them into the baseline system, we first perform a linear transformation on the input lexical features V_l in order to preserve the original sentiment distribution and have compatible dimensions for further computations. Later, the attention vector α learned as in the baseline is applied to the transformed lexical features L to produce a weighted representation of the lexical features l. Finally, all weighted representations l and r are added together with the last hidden state h_N after being projected by some network parameters to make the final prediction.

Formally, let $S \in \{w_1, w_2, ..., w_N\}$ be the input sentence, $V_l \in \{v_{l1}, v_{l2}, ..., v_{lN}\}$ be the lexical features of each word in S, $v_a \in \mathbb{R}^{d_a}$ be the aspect embeddings. Let $H \in \mathbb{R}^{d \times N}$ be the matrix of the hidden states

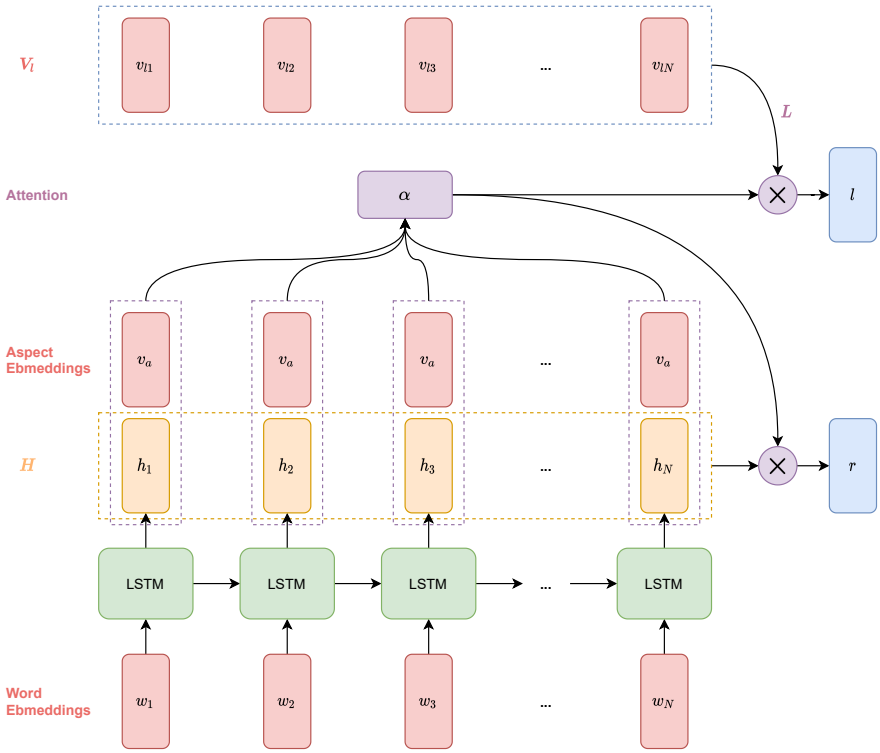

Figure 5.2: ATLX model architecture

$\{h_1, h_2, ..., h_N \in \mathbb{R}^d\}$ produced by the LSTM network. Same as the baseline (Equation 5.1, 5.2 and 5.3), the attention vector α and the weighted sentence representation r is computed as:

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix})$$

$$\alpha = softmax(w^\intercal M) \tag{5.8}$$

$$r = H\alpha^\intercal$$

where $M \in \mathbb{R}^{(d+d_a)\times N}, \alpha \in \mathbb{R}^N, r \in \mathbb{R}^d, W_h \in \mathbb{R}^{d\times d}, W_v \in \mathbb{R}^{d_a\times d_a}, w \in \mathbb{R}^{d+d_a}$. $v_a \otimes e_N = [v_a, v_a, ..., v_a]$ represents the operation that repeatedly concatenates v_a for N times.

Regarding the lexical inputs, let $V_l \in \mathbb{R}^{n\times N}$ be the lexical feature matrix of the sentence, V_l then is transformed linearly (Equation 5.9) by:

$$L = W_l \cdot V_l \tag{5.9}$$

where $L \in \mathbb{R}^{d\times N}, W_l \in \mathbb{R}^{d\times n}$. Later, the attention vector α learned from the concatenation of H and $v_a \otimes e_N$ is applied on L to obtain a weighted representation of the lexical features (Equation 5.10):

$$l = L \cdot \alpha^\intercal \tag{5.10}$$

where $l \in \mathbb{R}^d, \alpha \in \mathbb{R}^N$. Finally the mixed final representation of all inputs h^* is updated and passed to the output layer by:

$$h^* = tanh(W_p r + W_x h_N + W_o l) \tag{5.11}$$

$$\hat{y} = softmax(W_s h^* + b_s) \tag{5.12}$$

where $W_o \in \mathbb{R}^{d\times d}$ is a projection parameter as W_p and W_x; W_s and b_s are weights and biases in the output layer. The same loss function as the baseline is used to train the model:

$$loss = -\sum_i y_i log(\hat{y}_i) + \lambda \|\mathbf{\Theta}\|_2^2 \tag{5.13}$$

where i is the number of classes (ternary classification in our experiments). λ is the hyperparameter for L_2 regularization. And $\mathbf{\Theta}$ is the parameter set of the network to be regularized; compared to the baseline, new parameters W_l and W_o are added to $\mathbf{\Theta}$.

5.3 Attention Regularization

Since the attention vector is learned purely based on the training examples, it is possible that it is over-fitted in some cases, causing the network to ignore other relevant positions. A graphical representation of this effect is shown in Figure 6.14: the attention weights in ATLX are less sparse across the sentence, while the ones in the baseline are focusing only on the last parts of the sentence (details will be discussed in Section 6.2 of Chapter 6). In addition, we observe that the distribution of all the attention weights in ATLX has a lower variance[5] than in AT-LSTM (baseline). Note that the attention weights sum up to one, so when weights are closer to mean (not zero), the standard deviation is smaller; on the other hand, when most weights are close to zero and the rest few weights are close to one, the standard deviation is larger.

Thus we propose a simple attention regularizer to further validate our hypothesis, which consists of adding into the loss function a parameterized standard deviation or negative entropy term for regularizing the attention weights. The idea is to avoid the attention vector being overly sparse by having heavier weights in few positions; instead, it is preferred to have higher weight values for more positions, i.e. to have an attention vector with more spread out weights. Formally, the attention regularized loss function is defined as:

$$loss = -\sum_{i} y_i log(\hat{y}_i) + \lambda \|\boldsymbol{\Theta}\|_2^2 + \epsilon\, \Omega(\boldsymbol{\alpha}) \tag{5.14}$$

Compared to the loss function in AT-LSTM and ATLX (Equation 5.6 and 5.13), a second regularization term $\epsilon\, \Omega(\boldsymbol{\alpha})$ is added, where ϵ is the hyper-parameter for the attention regularizer (always positive); Ω stands for the regularization function defined in Equation 5.15 or Equation 5.16; and α

[5]Standard deviation of the attention weights distribution in the test set: AT-LSTM: 0.0354 $>$ ATLX: 0.0219

is the attention vector, i.e. the distribution of attention weights. Note that during implementation, the attention weights for batch padding positions must be excluded from α when computing the regularization term.

Regarding Ω itself, we experiment two different regularizers in our experiments: one uses the standard deviation of α defined in equation 5.15; and another one uses the negative entropy of α defined in equation 5.16.

Standard Deviation Regularizer

$$\Omega(\alpha) = \sigma(\alpha) = \sqrt{\frac{1}{N}\sum_{i}^{N}(\alpha_i - \mu)^2} \tag{5.15}$$

Negative Entropy Regularizer

$$ent(\alpha) = -\sum_{i}^{N}\alpha_i log(\alpha_i)$$

$$\Omega(\alpha) = -ent(\alpha) \tag{5.16}$$

5.4 Sentiment Induction

5.4.1 Sentiment Domain Adaptation

As most works on sentiment domain adaptation measure the performance by recreating an existing domain-specific lexicon (Hamilton et al., 2016; Mudinas et al., 2018), from an application point of view, we ask the question: how much improvement can we get from the domain-specific lexicon in a lexicon enhanced neural sentiment analysis system? And what is its limit? This is particularly interesting since we already have a model that works with a generic sentiment lexicon: ATLX.

In our experiments, we take the approach by Mudinas et al. (2018) to perform sentiment domain adaptation, who discovered that the distributed word representations in fact form distinct clusters for opposite sentiments; and this behavior in general holds across different domains. In other words, in the vector space shaped by a domain specific corpus, *positive* words are closer to each other than they are to *negative* words; and the same behavior is expected in other domains. Thus a probabilistic word classifier can be trained on a set of seed words (a number of predefined words which have consistent sentiment behavior in different domains, e.g. "*good*" and "*bad*)"; and this classifier can be used to induce the generic sentiment lexicon by predicting the word polarity in a new domain given its domain-specific word embeddings.

Specifically, we use the domain-specific word embeddings[6] learned from Amazon electronics review corpus; and a set of seed words (listed in Table 5.2) to form a set of training examples. Each example is composed by (x, y) pairs where $x \in \mathbb{R}^{500}$ is the 500 dimensional domain-specific word vector, and y is the seed word polarity as label. Next, we train a SVM classifier (Pedregosa et al., 2011) with *rbf* kernel and $C = 10$ as a regularization parameter. Finally, we use the trained classifier to predict

[6]Available at https://bit.ly/2U9X5aP

the polarity of generic lexicon words (U described in 5.2.1). When predicting, a confidence threshold $t = 0.7$ is applied to reduce noise; i.e. the polarity of the generic lexicon is updated only when $p \geq t$ where p is the maximum predicted probability of the classifier.

We use this approach to convert U from a generic domain lexicon into a domain-specific one (Amazon electronic reviews). Then we apply it in ATLX, and compare with applying a gold domain specific lexicon constructed by ourselves (Section 5.4.2). We evaluate the performance gain of each lexicon when applied in the ATLX model to understand the limit of domain adaptation.

In addition, compared to the binary classification originally applied in Mudinas et al. (2018) using only *positive* and *negative* seed words, we find that the binary classification would misclassify obvious *neutral* words, even when a 0.7 confidence threshold is applied. E.g. "*really*", "*very*" and "*thought*" are predicted to be *negative*, *positive* and *negative* respectively. Thus, to further reduce noise, we introduce an additional set of 35 *neutral* seed words (Table 5.2) to perform ternary classification instead of binary.

5.4.2 Gold Lexicon

To better interpret the experimental results and understand the limit of domain adaptation, we find the intersection I (839 elements) between the set of generic lexicon entries G (13,297 elements) and the set of the corpus vocabulary V (2,965 elements), where $I = G \cap V$. Then we label I to be the gold lexicon of the electronics review domain, where polarities: *positive*, *neutral* and *negative* are annotated as numerical values: 1, 0 and -1 respectively. Three principles are defined as annotation guidelines:

1. **Domain first**: prioritize the most common meaning of the word in the current domain. E.g. in the electronics or laptops review domain, "*fallout*" or "*excel*" are *neutral* proper nouns referring to a video game and a software; however, under generic context "*fallout*" and

Positive: 31 words	Negative: 34 words	Neutral: 35 words
amazing awesome	awful bad bland	absolutely actual
beneficial best correct	bore worst damages	actually air anyway
delightful excellent	disappointed disgusting	baby basically else
fortunate gains genius	down evil failure hate	entirely exact exactly
gifted good happy	hated hates horrible	expression eyebrows
improved improving	inferior lifeless	idea imagination
incredible interesting	litigation loss losses	information judgement
love loved lovely	nasty negative negligent	know likely much
loves nice perfect	poor sad shallow	opinion particular
pleasant positive	simplistic terrible	particularly perhaps
profit success	unfortunate unhappy	point seem should so
successful superior	unpleasant volatile	think thinking to
unforgettable fantastic	disappointing wrong	difference nature
		intention such

Table 5.2: Seed words and word counts used for domain adaptation.

"*excel*" are marked as *negative* and *positive*. Similarly, nouns such as "*brightness*", "*durability*" and "*security*" have *positive* sentiment under generic context, but here they in fact refer to *neutral* product aspects.

2. **Neutral adverbs**: adverbs in general should be neutral especially when they can be used to modify both *positive* and *negative* words. For example "*definitely*", "*fairly*" and "*truly*" can all express opposite sentiment depends on the word that follows ("*definitely* ***great***" vs "*definitely* ***garbage***").

3. **Neutral ambiguity**: ambiguous context dependent words should be *neutral* in the lexicon, in order to avoid feeding confusing information to the model. For instance, "*cheap price*" carries *positive* sen-

timent while "*cheap plastic*" is definitely *negative*. Other examples are: "*black screen*" vs "*black macbook*"; "*loud speaker*" vs "*loud click*"; "*low price*" vs "*low grade*".

5.4.3 Sentiment Aspect Adaptation

To deal with the aspect-dependent problem (e.g. "*cheap price*" vs "*cheap plastic*"), we adopt a similar approach to the domain adaptation method described in Section 5.4.1. More specifically, we build a set of training data using the same seed words shown in Table 5.2: the domain specific word embeddings of each word is merged with the aspect embeddings of a aspect word to be input features; and the seed words labels are served as classes. Then the same SVM classifier as for domain adaptation (Section 5.4.1) is trained and used to update the generic lexicon U given its word embeddings and aspect embeddings as joint inputs.

Formally, let A be the set of 9 aspect words in which each word is $A = \{$*"connectivity", "design", "general", "miscellaneous", "performance", "portability", "price", "quality", "usability"*$\}$. Let v_a^j be the word vector of an aspect word $j \in A$, where all word vectors are learned from the Amazon electronics review corpus, same as the domain adaptation method. Let S be the set of seed words in Table 5.2, and v_s^i be the domain specific word embeddings of a seed word $i \in S$. y_i be the label of the word i from S, namely *positive*, *neutral* or *negative*. Thus for each training example (x_{ij}, y_i), we have

$$x_{ij} = v_s^i \oplus v_a^j \quad \forall_j \in A$$

where $\oplus$ is an operation of concatenation, summation or mean of two vectors v_s^i and v_a^j. This is equivalent to a Cartesian product between S and A, and for each element in the output, we concatenate (or sum, or average) their corresponding domain-specific word vectors as input features.

Then, these training examples are used to train a SVM classifier same as the domain adaptation method. And finally, the trained classifier is used

to predict the polarity of a tuple consisting of the domain-specific word vector of a given word in U, and the aspect vector of any aspect from A. When the predicted probability is larger than the threshold t, the polarity of that word-aspect pair is modified. The final aspect-specific lexicon is essentially a dictionary with keys as the Cartesian product of U and A. And when used in the ABSA system, the polarity of a word is given by the expanded lexicon based on the input word and its associated aspect. Same as the domain adaptation method described in Section 5.4.1, we train a SVM classifier (Pedregosa et al., 2011) with *rbf* kernel and $C = 10$ as a regularization parameter; and the threshold $t = 0.7$ is used.

Chapter 6

EXPERIMENTS

In Chapter 3, we introduce the main objectives of this thesis. In Section 5.2, 5.3 and 5.4 of Chapter 5, we describe our approaches to complete these objectives. And in this chapter, we will describe three main experiments in details that correspond to the objectives and methods. The three main experiments consist of: merging lexicon into the AT-LSTM baseline model (ATLX), regularizing the attention vector, and sentiment induction applied in the lexicon enhanced ATLX model.

In addition, based on the outcomes of the experiments, we will discuss the possible reasons behind the results and the implications that come along. Moreover, some support experiments will also be covered to better illustrate the ideas.

6.1 AT-LSTM with Lexicon Enhancement (ATLX)

6.1.1 Datasets

Same as Wang et al. (2016c), we experiment on SemEval 2014 Task 4, restaurant domain dataset. The data consists of reviews of restaurants with predefined aspects: {*food, price, service, ambience, miscellaneous*} and associated polarities: {*positive, neutral, negative*}. The objective is to predict the polarity given a sentence and an aspect. For instance, given a review sentence "*The restaurant was too expensive.*", the model should identify the *negative* polarity associated with the aspect *price*. In total, there are 3,518 training examples and 973 test examples in the corpus. Table 6.1 shows the distribution of aspects per label for both training and test data.

Polarity	Positive		Neutral		Negative	
Aspect\Split	Train	Test	Train	Test	Train	Test
food	867	302	209	69	90	31
price	179	51	115	28	10	1
service	324	101	218	63	20	3
ambience	263	76	98	21	23	8
miscellaneous	546	127	119	41	357	51
TOTAL	2179	657	839	222	500	94

Table 6.1: Distribution of aspects by label and train/test split in the SemEval 2014 Task4, restaurant domain dataset.

In addition, we also reproduce our experiments on the SemEval 2015 Task 12, laptop domain dataset. The dataset consists of reviews of laptops with annotated entity-attribute pairs such as: {*LAPTOP#GENERAL, KEYBOARD#QUALITY, LAPTOP#PRICE, ...*} and associated polarities: {*positive, neutral, negative*}. In order to have comparable results with the SemEval 2014 dataset, we simplify the attribute annotations to: {*general,*

Polarity	Positive		Neutral		Negative	
Aspect\Split	Train	Test	Train	Test	Train	Test
connectivity	17	6	0	3	15	15
design	150	71	33	16	67	39
general	401	197	10	15	168	79
miscellaneous	71	43	12	5	35	21
performance	164	88	9	6	114	77
portability	36	5	0	1	8	2
price	41	38	22	17	25	5
quality	115	61	10	5	289	65
usability	108	32	10	11	44	26
TOTAL	1103	541	106	79	765	329

Table 6.2: Distribution of aspects by label and train/test split in the SemEval 2015 Task12, laptop domain dataset.

performance, design, usability, portability, price, quality, miscellaneous, connectivity$\}$ and use them as aspects. Together, there are 1,973 training examples and 949 test examples in the corpus. Table 6.3 shows a small portion of examples from both datasets.

In addition, we use pre-trained word embeddings to initialize the parameters in the embedding layer of our model. Namely, the 300 dimensional Glove[1] vectors trained on 840B tokens are used for the ATLX model.

6.1.2 Lexicons

As shown in Table 6.4, we merge four existing and online available lexicons into one. The merged lexicon U as described in Section 5.2.1 is used for our experiments. After the union, the following post-process is carried out: $\{bar, try, too\}$ are removed from U since they are unreasonably an-

[1]https://stanford.io/2FeYJnn

SemEval 14 Task 4, Restaurant Domain

Label	Aspect	Sentence
negative	*quality*	“the battery has never worked well .”
negative	*performance*	“the battery has never worked well .”
negative	*quality*	“the battery gets so hot it is scary .”
positive	*general*	“this computer is absolutely amazing ! ! !”
neutral	*general*	“overall , it ’s ok .”

SemEval 15 Task 12, Laptop Domain

Label	Aspect	Sentence
positive	*service*	“good , fast service .”
positive	*miscellaneous*	“you can not go wrong at the red eye grill .”
negative	*miscellaneous*	“this restaurant used to be pretty decent .”
neutral	*miscellaneous*	“was there friday night .”
positive	*food*	“great wine , great food .”

Table 6.3: Examples from the SemEval 14, Task 4 and the SemEval 15, Task 12 datasets.

	Positive	Neutral	Negative	In corpus
MPQA	2298	440	4148	908
OL	2004	3	4780	732
Opener	2298	440	4147	908
Vader	3333	0	4170	656
Merged U	5129	404	7764	1234

Table 6.4: Lexicon statistics of *positive*, *neutral*, *negative* words and number of words covered in corpus.

notated as negative by *MPQA* and *Opener*; $\{n't, not\}$ are added to U with -1 polarity for negation as we have observed cases in early experiments where the model struggles to identify negation after lexicon integration.

6.1.3 ATLX Variants

In order to effectively merge lexicon information to the baseline system, apart from the ATLX model described in Section 5.2.2, we have designed a set of variants as well. Namely, a variety of ways slightly different from ATLX to merge lexicon information into the system.

Variant 1

Recall that in ATLX (Equation 5.10, Section 5.2.2), the lexical representation l is obtained by applying the attention weights α on the transformed lexical features L:

$$l = L \cdot \alpha^{\intercal}$$

Here, instead of applying the attention vector α, a linear transformation is adopted to obtain l (Equation 6.1):

$$l = L \cdot w_{v1} \tag{6.1}$$

where $L \in \mathbb{R}^{d \times N}$, $w_v \in \mathbb{R}^N$, and $l \in \mathbb{R}^d$.

Variant 2

Recall that in ATLX (Equation 5.8, Section 5.2.2), the attention vector α is computed with the concatenation of transformed hidden states H and the repeated aspect vectors v_a as input:

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix})$$

$$\alpha = softmax(w^\intercal M)$$

Here, we add a third input to compute α, which is the lexical features L projected by some network parameter W_{v2}:

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \\ W_{v2} L \end{bmatrix}) \tag{6.2}$$

$$\alpha = softmax(w^\intercal M)$$

where $L \in \mathbb{R}^{d \times N}$, $W_{v2} \in \mathbb{R}^{d \times d}$, and $M \in \mathbb{R}^{(d+d_a+d) \times N}$.

Variant 3

Recall that in ATLX (Equation 5.11, Section 5.2.2), the final representation h^* is composed by the summation of three lower level representations: r, h_N and l:

$$h^* = tanh(W_p r + W_x h_N + W_o l)$$

Here instead of summation, a concatenation of three elements are made to form the final representation:

$$h^* = tanh(\begin{bmatrix} W_p r \\ W_x h_N \\ W_o l \end{bmatrix}) \tag{6.3}$$

where $r \in \mathbb{R}^d$, $l \in \mathbb{R}^d$, $h_N \in \mathbb{R}^d$, and $h^* \in \mathbb{R}^{3d}$.

Variant 4

Similar to Variant 3, compared to ATLX where the final representation is obtained through the summation of three lower level representations, here we use a different approach to compute h^*. Inspired by the attention mechanism, we would like to have a second attention mechanism here to weight the lower level representations when aggregating the final representation. This way, the model would be able to weight different information sources accordingly as lexical features are not always available (words outside of the lexicon are treated as *neutral* as described in Section 5.2.1).

Formally, let H^* be the concatenation of $W_p r + W_x h_N$ and $W_o l$ (Equation 6.4), a new attention vector β (Equation 6.5) is learned and applied back to H^* to obtain the final representation (Equation 6.6):

$$H^* = tanh(\big[W_p r + W_x h_N, W_o l\big]) \tag{6.4}$$

$$\beta = softmax(w_b^\intercal H) \tag{6.5}$$

$$h^* = H\beta^\intercal \tag{6.6}$$

where $H^* \in \mathbb{R}^{d \times 2}$, $w_b \in \mathbb{R}^d$, $\beta \in \mathbb{R}^2$, $h^* \in \mathbb{R}^d$.

6.1.4 Evaluation

In our experiments, we use cross validation (CV) to evaluate the performance of each model. Specifically, the training set is randomly shuffled and split into 6 folds with a fixed random seed. According to the code[2] released by Wang et al. (2016c), a development set containing 528 examples is used in the implementation of AT-LSTM, which is roughly $\frac{1}{6}$ of the training corpus. In order to remain faithful to the original implementation, we thus evaluate our model with a cross validation of 6 folds.

	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157
ATLX	**75.64**	1.275	**82.62**	0.498
Variant1	75.59	1.349	80.97	0.683
Variant2	75.56	1.465	82.12	1.380
Variant3	74.36	1.291	80.49	1.680
Variant4	73.39	2.544	79.48	1.976

Table 6.5: Mean accuracy and standard deviation (σ) of cross validation results on six folds of development sets and one holdout test set of the SemEval14, restaurant dataset. Note that in our replicated baseline system, the cross validation performance on the test set ranges from 80.06 to 83.45; in Wang et al. (2016c), 83.1 was reported.

	SemEval14 Restaurant				SemEval15 Laptop			
	CV	σ^{CV}	Test	σ^{Test}	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157	82.48	2.154	74.06	0.624
ATLX	**75.64**	1.275	**82.62**	0.498	**83.39**	2.640	**75.92**	1.497

Table 6.6: Mean accuracy and standard deviation (σ) of cross validation results on six folds of development sets and one holdout test set. Evaluated on the SemEval14, restaurant dataset and the SemEval15, laptop dataset.

Table 6.5 shows the evaluation results of the baseline system, ATLX and four variants of ATLX on the SemEval14 restaurant dataset. Compared to the baseline system on both datasets, ATLX improves substantially on both CV and test sets. Meanwhile, the four variants of ATLX cannot achieve a superior performance compared to ATLX, and some even decrease compared to the baseline. For instance, both variant 1 and variant 2 improve slightly on the CV sets compared to the baseline; however, only variant 2 improves on the test set as well while variant 1 suffers a drop back. On the other hand, both variant 3 and variant 4 show a inferior performance compared to the baseline on both CV sets and test set, where variant 4 suffers the largest decrease.

To further validate the effectiveness of ATLX, we conduct similar experiments on the SemEval15 laptop dataset. More specifically, we apply both the baseline and the ATLX model on the SemEval15 dataset and see if a similar improvement can be observed. Table 6.6 shows the evaluation results of the two models on both datasets. From the table, we can see that similar to the SemEval14 dataset, compared to the baseline, ATLX improves on both the CV sets and the test set of the SemEval15 dataset as well; showing that our proposed method is effective.

It is worth mentioning that the results on the SemEval15 dataset have higher variance than the SemEval14 dataset (σ^{CV} of the baseline and ATLX on the SemEval15 dataset are both above 2.0, compared to the ones in SemEval14 which are both below 1.5); and the variance improvements of the proposed methods are only observed in the SemEval14 dataset. Given the fact that both datasets are not large in terms of scale under modern deep learning standards, and the SemEval15 dataset is even smaller than the SemEval14 dataset, it is hard to draw a strong conclusion here.

[2]https://bit.ly/2I9H4yx

6.1.5 Discussion

Qualitative Analysis - ATLX Improvements

In previous sections, we described a simple yet effective approach for an attention LSTM network to leverage natural polarities in numeric form provided by lexical resources for aspect-level sentiment analysis. As a result, the overall performance of the ATLX model is enhanced compared to the baseline; and more importantly, by leveraging lexical features independent from the training data, the model becomes more robust and flexible.

For instance in Figure 6.1, although the baseline is able to pay relatively high attention to the word "*disappointed*" and "*dungeon*", it is not able to recognize these words as clear indicators of *negative* polarity; while ATLX is able to correctly predict *negative* for both examples. It is also interesting to see that in the second example, the attention shifts to the word "*dungeon*" in ATLX compared to the baseline, suggesting that the model is able to take advantage of the extra information provided by the lexicon.

More similar examples can be observed as well. For instance in Figure 6.2, the baseline is consistent about the opinion word "*disappointed*", similar to previous examples, and it recognises the word as a *positive* indication; thus predicts the negation clause "*not be disappointed*" as *negative*. Although the attention weights are still similarly distributed (more focused on the ending part of the sentence) after the lexicon is introduced, the ATLX model is able to understand "*disappointed*" as a *negative* opinion word just like the example in Figure 6.1.

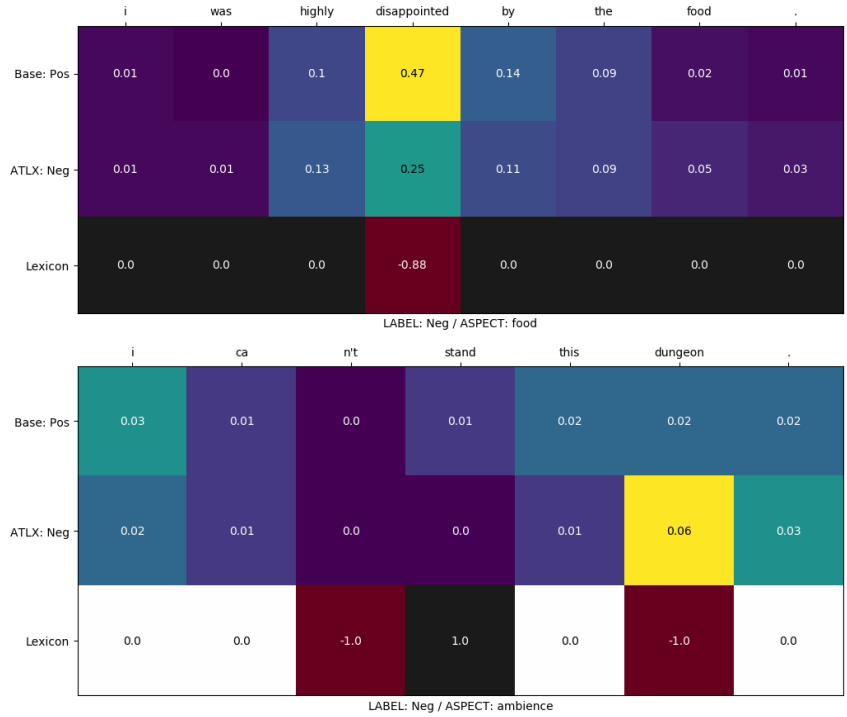

Figure 6.1: Baseline ("Base") and ATLX comparison (1/12); baseline predicts *positive* ("Pos") for both examples, while the gold labels are *negative* ("Neg") for all. In some of the following plots, the *neutral* is annotated as ("Neu"). In the rows annotated as "Base" and "ATLX", the numbers represent the attention weights of each model when predicting. Note that they do not sum up to 1 in the Figure because predictions are done in a batch with padding positions in the end which are not shown in the Figure. The rows annotated as "Lexicon" indicate the average polarity per word given by U as described in Section 5.2.1

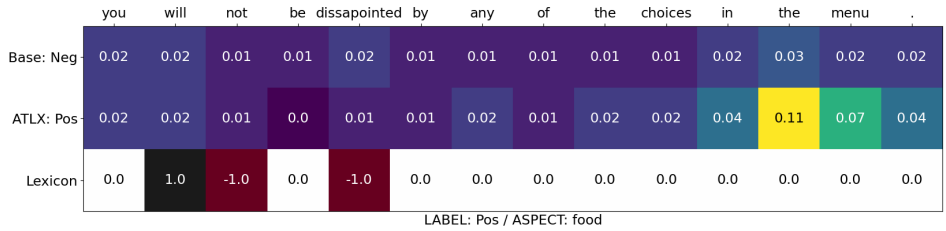

Figure 6.2: Baseline and ATLX comparison (2/12).

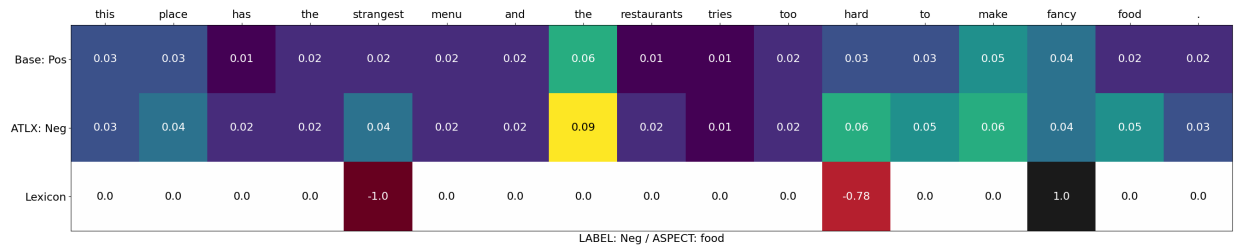

Figure 6.3: Baseline and ATLX comparison (3/12).

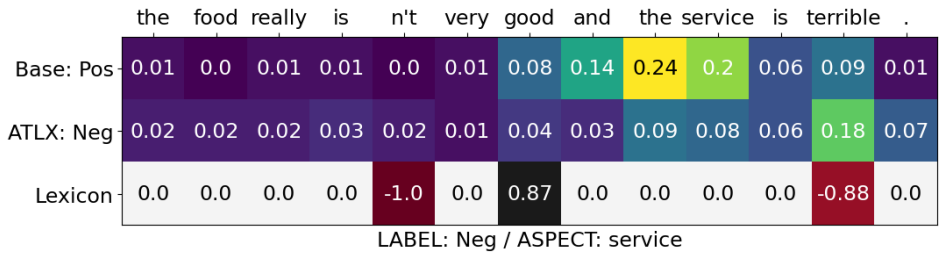

Figure 6.4: Baseline and ATLX comparison (4/12).

In Figure 6.3, compared to the ATLX model, the attention weights in the baseline model obviously fail to take into account some of the key elements for polarity judgement. For example in the baseline, given the aspect "*food*", the attention weights for the word "*strangest*" and the clause "*too hard to make*" are clearly lower than they are in the ATLX model. However in the ATLX model, after introducing the lexicon, both the word "*strangest*" and the word "*hard*" receive a significantly higher attention; which helps the model to correctly predict the *negative* class.

In Figure 6.4, we can see another case in which after introducing the lexicon, the model is able to attend more to a keyword and makes the correct prediction. Specifically, given the aspect *service*, the baseline is not able to predict the correct *negative* label although the clause "*the service is terrible*" has been given relatively higher weights. On the other hand, the weight of the opinion word "*terrible*" is doubled in ATLX with the polarity of the word "*terrible*" fed to the model.

In Figure 6.5 we can see a rather simple case, in which the baseline predicts incorrectly the *neutral* label. However, in ATLX, although the distribution of the attention weights is similar to the baseline, the model now can correctly predict *positive* given the aspect *food*.

In these cases, compared to the baseline, ATLX is not only able to leverage the newly introduced lexical features, but also to better distribute the attention weights accordingly. However, in some cases, without having

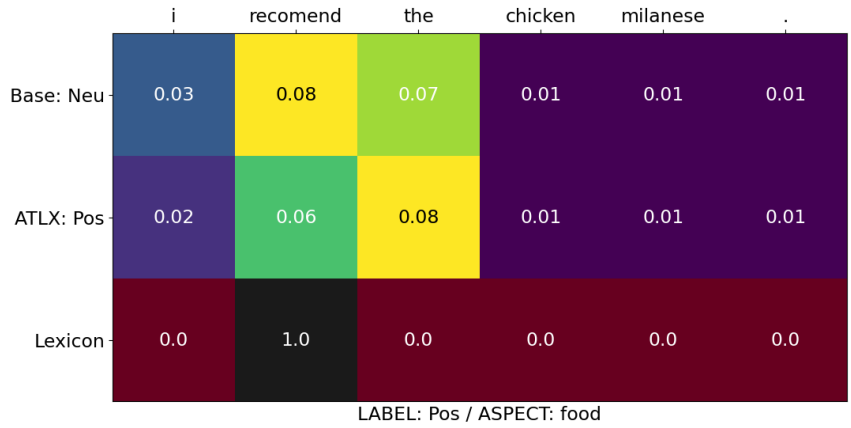

Figure 6.5: Baseline and ATLX comparison (5/12). Baseline predicts *neutral* ("Neu")

any valuable lexical features, the attention weights of the ATLX model are still able to be more reasonable.

For instance in Figure 6.6, all lexical features are 0 which means no external polarity information is available. However, there is a clear difference between the attention weight distribution of the baseline and ATLX. In the baseline, the model "focuses" more on the beginning part of the sentence; while in ATLX, the word "*varied*" is highlighted, which is the key for polarity judgment in this case.

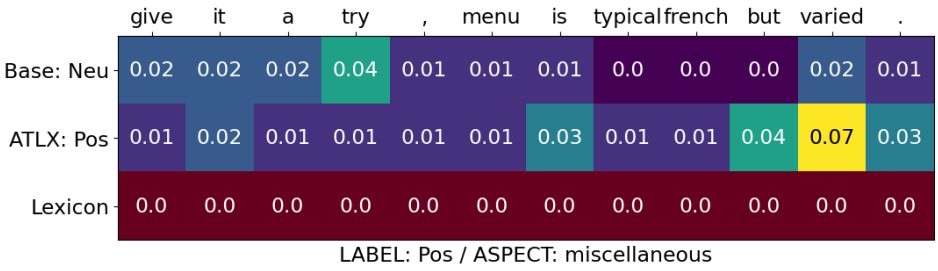

Figure 6.6: Baseline and ATLX comparison (6/12).

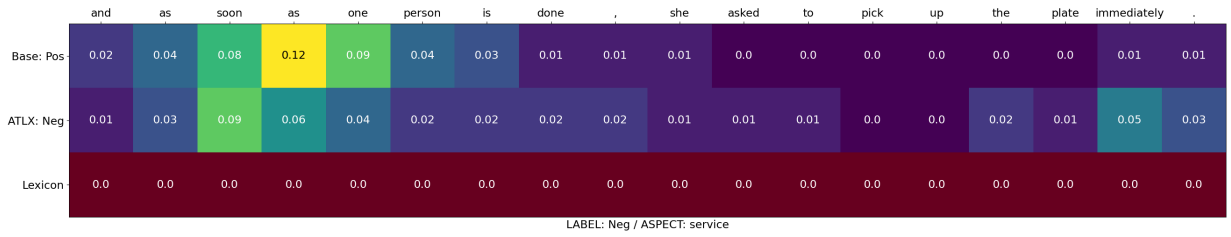

Figure 6.7: Baseline and ATLX comparison (7/12).

Similarly, in Figure 6.7, given the aspect *service*, the baseline model almost only "focuses" on the first clause, and completely ignores the key word "*immediately*" at the end of the sentence. In this case, the word "*immediately*" directly ties to the *service* as it refers to the waiter/waitress picking up the plate too soon. Compared to the baseline, the attention weight of the word "*immediately*" increases significantly even though all lexical features are 0, making the attention distribution more reasonable than it is in the baseline.

Qualitative Analysis - ATLX Trade-offs

Although the general performance of ATLX is better than the baseline, there are also cases where the lexicon enhanced model performs worse than the baseline. By adding lexical features in the system, it is inevitable to introduce noise, and such noise may confuse the model.

For example in Figure 6.8, both the baseline and ATLX are able to pay relatively high attention to the second clause: "*definitely the place to be*"; however, ATLX is not able to identify the *positive* polarity given the *miscellaneous* aspect. It is worth mentioning that the polarities of all three non-neutral words given by the lexicon seem more reasonable to be *neutral*. Such noise from the lexicon can produce a negative effect on the ATLX model.

Similarly, in Figure 6.9, given the aspect *miscellaneous*, the ATLX model fails to identify the *neutral* polarity. Compared to the baseline, ATLX "focuses" more on the word "*promptly*", which carries a *positive* sentiment according to the lexicon. Under this context, it is reasonable that "*seated promptly*" refers to good fast service because there was no need to wait. However, it is indeed *neutral* regarding the aspect *miscellaneous*. Nevertheless, the words "*close*" and "*dance*" are marked as *negative* and *positive* by the lexicon, which are disputable.

A slightly more complex example can be found in Figure 6.10, where a *comparative opinion* is expressed on top of a *positive* opinion. In fact

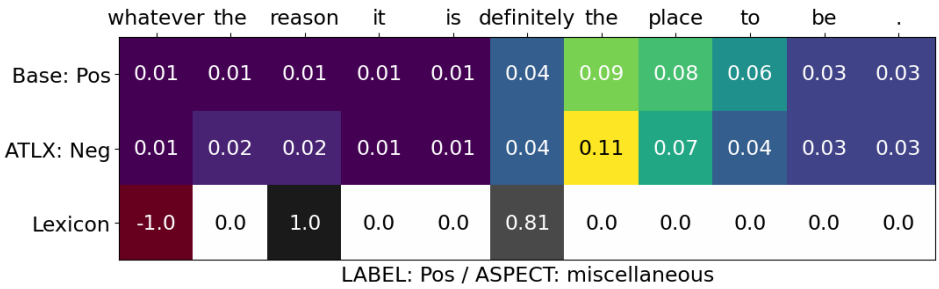

Figure 6.8: Baseline and ATLX comparison (8/12).

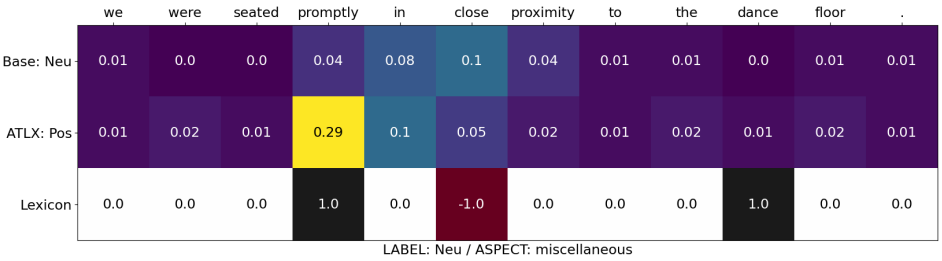

Figure 6.9: Baseline and ATLX comparison (9/12).

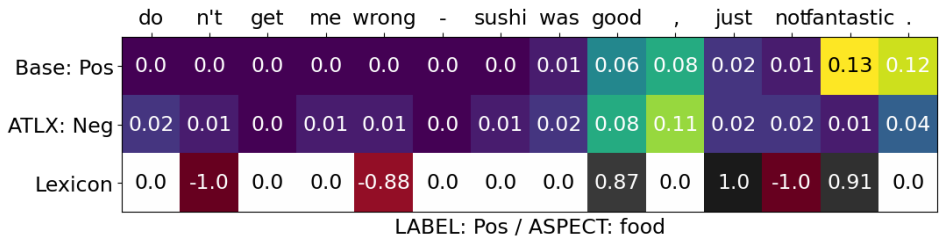

Figure 6.10: Baseline and ATLX comparison (10/12).

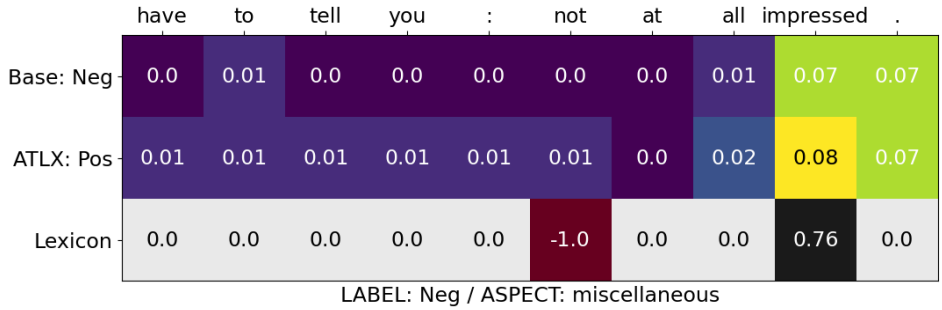

Figure 6.11: Baseline and ATLX comparison (11/12).

comparative opinions are studied as a sub-field of sentiment analysis due to their complex structure (Liu, 2012). In this case, the baseline model predicts correctly and the ATLX model seems mazed by the number of *positive* and *negative* opinion words marked by the lexicon.

In some other cases, the ATLX model shows inconsistency in terms of treating negation expressions. For example in Figure 6.11, the ATLX model correctly "focuses" on the word "*impressed*" with *positive* polarity given by the lexicon; however, the negation coming before it is completely ignored. It has been shown that ATLX is usually capable of handling negations (e.g. Figure 6.2, 6.4). However, this case suggests that the handling of negation in ATLX is not consistent.

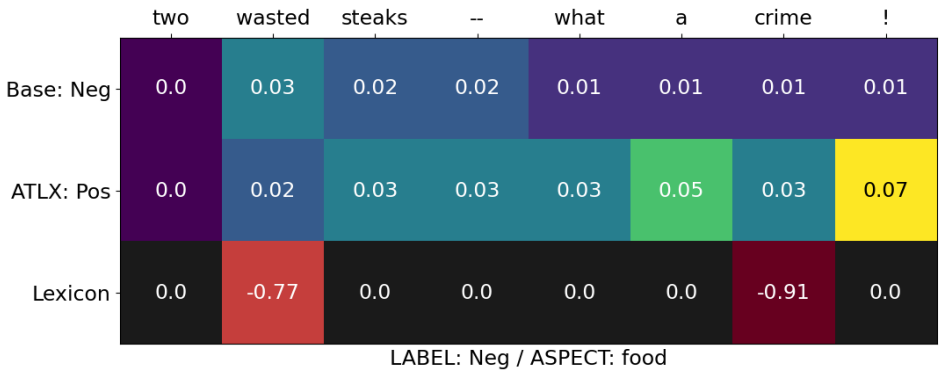

Figure 6.12: Baseline and ATLX comparison (12/12).

Last but not least, in Figure 6.12, we can see that the ATLX has a higher "focus" on the exclamation mark in the end, and despite the two *negative* opinion words marked by the lexicon, the ATLX model still predicts *positive* given the aspect *food*.

Lexicon Integration

Regarding the proposed approach for merging lexical features into the system (ATLX in Section 5.2), it is worth mentioning that the computation of the attention vector α does not take lexical features V_l into account. Although it is natural to think that adding V_l as input for computing α would be a good option, the results of ATLX* in Table 6.7 suggest otherwise.

In order to understand better where the improvement of ATLX comes from, whether it comes from the availability of lexical features or it is also related to how we introduce lexical features to the system, we conduct a support experiment that aims at verifying the impact of the lexicon (base$^{\text{LX}}$); it consists of naively concatenating input word vectors with their associated lexical vectors and feed the extended embeddings to the baseline. As shown in Table 6.7, by comparing baseline with base$^{\text{LX}}$, we see

	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157
ATLX	**75.64**	1.275	**82.62**	0.498
ATLX*	74.99	1.638	82.03	1.409
Base$^{\text{LX}}$	71.98	1.588	79.24	2.322

Table 6.7: ATLX support experiments on SemEval14, restaurant domain dataset.

that the simple merge of lexical features with the network without carefully designed mechanisms, the model is not able to leverage new information; and as a consequence, the overall performance is decreased.

Regarding the ATLX variants described in Section 6.1.3, apparently none of them achieves a superior performance compared to not only ATLX but also the baseline. As deep neural networks are still considered some kind of black-box model, it is hard to reason why one solution performs better than another. Here by comparing the difference with the ATLX model, we try to point out some potential insights learned from these experiments.

Compared to ATLX, variant 1 uses a linear transformation to process the lexical features L instead of applying the attention vector on L (Equation 6.1). First, a linear transformation seems to be incapable of efficiently passing lower level information to higher level layers, especially as hidden layers of neural networks. Second, by applying the attention vector α on L instead of putting L as input to learn α, when training the network and updating the parameters, the lexical features still have impact on how α will change and thus the attention framework is capable of taking into account lexical features as well. Consequently, we observe the impact on attention vectors in ATLX compared to the baseline, which allows it to attend more on key opinion words with sentiment information from the lexical features. In addition, the fact that the attention vector is learned to attend to both the input sentence and the lexical features, ensures that

when putting them together at later steps, there will be no conflict between these two components and it allows us to obtain a final representation more smoothly.

In variant 2, we add the linearly transformed lexical features L into the input for computing the attention vector α (Equation 6.2). The results in Table 6.5 show that variant 2 does improve on both the CV sets and the test set compared to the baseline; however, the improvement is not as large as ATLX. One possible reason is that by adding L into the equation, we have also introduced more model parameters that need to be learned; while the dataset is limited in size and cannot help to train a better model. Meanwhile, the model becomes redundant when trying to obtain the final representation of all inputs as: $h^* = tanh(W_p r + W_x h_N + W_o l)$, where both l and r are products of the attention vector α.

Both variant 3 and variant 4 suffer a performance decrease compared to the baseline. Similarly, instead of summing the lower level representations, they try to concatenate the lower level representations (Equation 6.3), or using a weighted sum to combine them (Equation 6.6). Concatenation as a commonly used approach to combine the outputs of two hidden layers has been widely used in deep neural networks. However in Table 6.5, the results suggest that summation yields better results. Similar results can be observed for variant 4, where using weighted sum for combining the sentence representation and lexicon representation does not yield a better performance.

Lexicon Size

To further explore the impact of adding lexical features into the system, we conduct another support experiment focusing on the changes caused by the size of the lexicon.

As described in Section 5.2.1, neutral polarity is supplied for words outside the lexicon. Let $u \in U$ be the subset of lexicon entries where $u = U \cap V$ and V is the vocabulary of the corpus. As shown in Table 6.4,

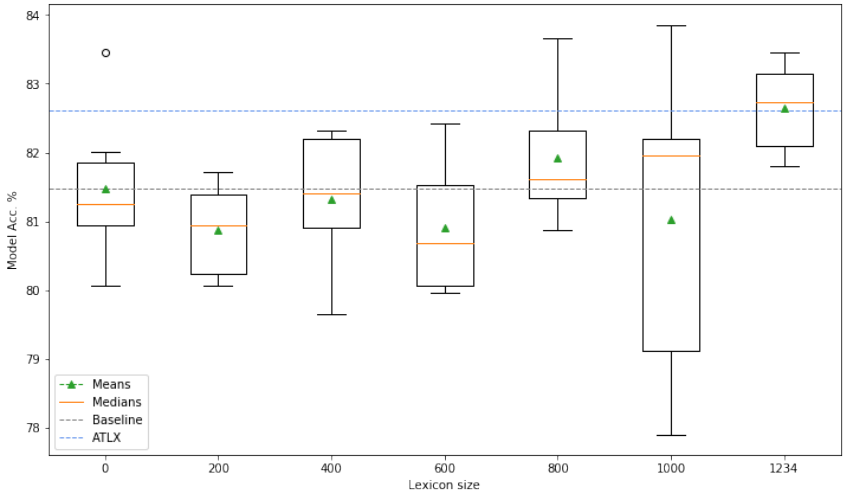

Figure 6.13: ATLX cross validation results on test set with increasing lexicon size on SemEval14, restaurant domain dataset.

the size of u in our experiment is $1,234$. In order to experiment the impact caused by the size of the lexicon, we randomly shuffle u and perform the same cross validation evaluation on ATLX with an increasing size of u by a step of 200. Figure 6.13 shows the cross validation performance on the test set.

In general, we can see that larger size of u tends to yield better overall performance but with an exception of size $1,000$; where the performance becomes more variant.

Lexicon Dimensions

As shown in Table 6.8, the dimension of the lexical feature affects the performance of the model to some extent. The best performance comes from $n = 3$, i.e. when using only 3 columns of the merged lexicon U, which is the result reported as ATLX in Table 6.5 and others that follow.

	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157
ATLX$^{\text{n=1}}$	75.47	2.422	81.91	0.407
ATLX$^{\text{n=2}}$	75.19	1.531	82.10	1.253
ATLX$^{\text{n=3}}$	**75.64**	1.275	**82.62**	0.498
ATLX$^{\text{n=4}}$	75.50	2.034	82.60	0.800

Table 6.8: ATLX lexicon dimension experiments on SemEval14, restaurant domain dataset.

Although the difference between ATLX$^{\text{n=3}}$ and ATLX$^{\text{n=4}}$ is negligible and the performance seems linear with respect to n, it would be safer to select n through tuning.

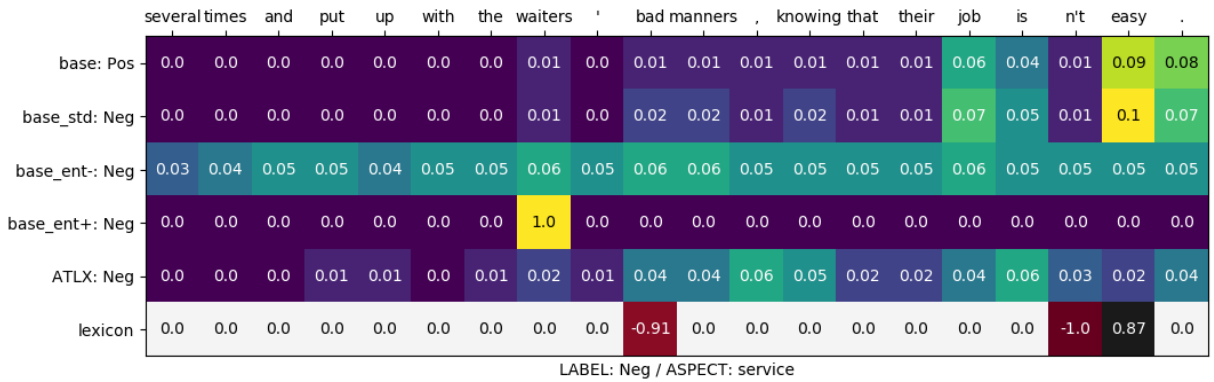

Figure 6.14: Comparison of attention weights between baseline (Base), baseline with standard deviation regularizer (Base^{std}), baseline with negative entropy regularizer ($\text{Base}^{\text{ent-}}$), baseline with positive entropy regularizer ($\text{Base}^{\text{ent+}}$) and ATLX. Baseline predicts *positive* while all other models correctly predict *negative*. The row annotated as "Lexicon" indicates the average polarity given by U. Note that only ATLX takes into account lexical features, the rest do not.

6.2 Attention Regularization

6.2.1 Evaluation

As described in Section 5.3, in Figure 6.14, we can observe that in the baseline system, before adding lexical features, the attention weights are more sparse (i.e. large weights in few positions, small weights close to zero in many positions), and mostly focusing only on the last parts of the sentence. However in the ATLX system, the attention weights are less sparse across the sentence. This sparseness could hurt the model by not passing key information to deeper layers. In this case, the baseline is not able to pay attention to "*bad manners*", while the ATLX model can.

Since the attention vector is purely learned on the training data, we believe it could be over-fitting. Thus we design two regularizers (Section 5.3) and try to overcome the over-fitting effect. Namely, a parameterized standard deviation regularizer, and a negative entropy regularizer. The idea is to avoid the attention vector being overly sparse by having heavy weights in few positions; instead, it is preferred to have higher weights for more positions, i.e. to have an attention vector with more spread out weights.

Table 6.9 shows the evaluation results of applying these two regularizers in both the baseline and the ATLX model. Compared to the baseline system on both datasets, by adding attention regularization to the baseline system without introducing lexical features, both the standard deviation regularizer (base^{std}) and the negative entropy regularizer ($\text{base}^{\text{ent-}}$) are able to contribute positively, where $\text{base}^{\text{ent-}}$ yields the largest improvement. But this is only observed on the test sets of both datasets, the performance on the CV sets of SemEval15 is generally worse than the baseline. However, by combining attention regularization and lexical features together, the model is able to achieve the highest test accuracy in all experiments conducted on both datasets.

SemEval14 Restaurant

	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157
Base$^{\text{std}}$	74.67	1.688	81.57	0.915
Base$^{\text{ent-}}$	**75.93**	1.467	82.24	0.863
ATLX	75.64	1.275	82.62	0.498
ATLX$^{\text{std}}$	75.64	1.275	82.68	0.559
ATLX$^{\text{ent-}}$	75.53	1.265	**82.86**	1.115

SemEval15 Laptop

	CV	σ^{CV}	Test	σ^{Test}
Baseline	82.48	2.154	74.06	0.624
Base$^{\text{std}}$	81.45	1.572	74.53	1.845
Base$^{\text{ent-}}$	81.91	1.194	75.80	0.763
ATLX	**83.39**	2.640	75.92	1.497
ATLX$^{\text{std}}$	82.36	2.082	74.75	2.560
ATLX$^{\text{ent-}}$	82.87	1.696	**75.94**	1.582

Table 6.9: Comparison between main experiments and attention regularizers. Mean accuracy and standard deviation of cross validation results on six folds of development sets and one holdout test set. Evaluated on SemEval14 and SemEval15 dataset.

6.2.2 Discussion

Attention Regularizers

As shown in Figure 6.14, when comparing ATLX with the baseline, we find that although the lexicon only provides non-neutral polarity information for three words, the attention weights of ATLX are less sparse and more spread out than it is in the baseline. On the other hand, this effect is general as the standard deviation of the attention weights distribution in the test set in ATLX (0.0219) is notably lower compared to the baseline (0.0354).

Thus it makes us think that the attention weights might be over-fitting in some cases as they are purely learned on training examples. This could cause that by giving too much weight to particular words in a sentence, the network ignores other positions which could provide key information for higher level classification. For instance, the example in Figure 6.14 shows that the baseline, which predicts *positive*, is "focusing" on the last parts of the sentence, mostly the word "*easy*"; while ignoring the "*bad manners*" coming before, which is key for judging the polarity of the sentence given the aspect *service*. In contrast, the same baseline model trained with attention regularized by standard deviation is able to correctly predict *negative* just by "focusing" a little bit more on the "*bad manners*" part.

However, the hard regularization by standard deviation might not be ideal as the optimal minimum value of the regularizer will imply that all words in the sentence have homogeneous weights, which is the opposite of what the attention mechanism is able to gain.

Regarding the negative entropy regularizer, taking into account that the attention weights are outputs of $softmax$ which is normalized to sum up to 1 [3], although the minimum value of this term would also imply homogeneous weight of $\frac{1}{N}$, it is interesting to see that with an almost evenly

[3] As explained in the caption of Figure 6.1, in all Figures the attention weights do not sum up to 1 because they are predicted in a batch with padding positions in the end, which are not included in the Figures.

SemEval14 Restaurant

	CV	σ^{CV}	Test	σ^{Test}
Baseline	75.27	1.420	81.48	1.157
Base$^{\text{ent-}}$	**75.93**	1.467	82.24	0.863
Base$^{\text{ent+}}$	75.36	1.405	81.81	0.854
ATLX	75.64	1.275	82.62	0.498
ATLX$^{\text{ent-}}$	75.53	1.265	**82.86**	1.115
ATLX$^{\text{ent+}}$	75.42	1.298	82.32	0.501

SemEval15 Laptop

	CV	σ^{CV}	Test	σ^{Test}
Baseline	82.48	2.154	74.06	0.624
Base$^{\text{ent-}}$	81.91	1.194	75.80	0.763
Base$^{\text{ent+}}$	81.71	1.133	74.99	1.859
ATLX	**83.39**	2.640	75.92	1.497
ATLX$^{\text{ent-}}$	82.87	1.696	**75.94**	1.582
ATLX$^{\text{ent+}}$	82.06	2.080	75.48	1.309

Table 6.10: Comparison between positive entropy regularizer and negative entropy regularizer. Mean accuracy and standard deviation of cross validation results on six folds of development sets and one holdout test set. Evaluated on SemEval14 and SemEval15 dataset.

distributed α, the model remains sensitive to few positions with relatively higher weights. For example in Figure 6.14, the same sentence with negative entropy regularization demonstrates that although most positions are closely weighted, the model is still able to differentiate key positions even with a weight difference of 0.01 and correctly predict *negative* given the *service* aspect.

Positive Entropy Regularizer

According to Zhang et al. (2019a), the attention mechanism is designed to allow the network to "focus" on different segments given different scenarios; however it lacks control to enforce that the attention weights are sparse and sharp so that the effect of the attention mechanism is maximized. Although we agree that a sharper and more sparse attention distribution could improve the performance of the model, we also argue that at least in the sentiment analysis domain, where the attention is usually applied early in the model, the attention weights could be over-fitting by being sparse, thus causing the model to ignore other relevant positions and underperform.

Here we use the positive entropy regularizer defined in Equation 6.7 as the attention regularization term in the loss function defined in Equation 5.14. The objective is to minimize the loss while preventing $\Omega(\alpha)$ from being large, so that the attention weights are regularized to be sparse. Note that during implementation, when using positive entropy regularizer, the padding positions for α are included in the computation of the regularization term. If excluded, the padding positions would have higher attention weights than the sentence words.

$$R(\alpha) = ent(\alpha) = -\sum_{i}^{N} \alpha_i log(\alpha_i) \tag{6.7}$$

As shown in Table 6.10, the superscript $^{\text{ent+}}$ denotes experiments applying the positive entropy regularizer. We can see that base$^{\text{ent+}}$ and ATLX$^{\text{ent+}}$ improve on both CV and test sets on the SemEval14 dataset, however the margin is very small. On the SemEval15 dataset, they can only improve on the test set. More importantly, all results with the positive entropy regularizer are inferior compared with the ones with the negative entropy regularizer. This aligns with our hypothesis that the overly sparse attention vector can cause over-fitting and hurt the model performance.

Qualitatively, the example in Figure 6.14 shows that base$^{\text{ent+}}$ does produce a sparse and sharp attention weight; and by shifting the "focus" to the

word "*waiters*", it also corrects the mistake made by the baseline. However, as shown in Figure 6.15, both baseline and base^{ent-} are able to correctly predict *positive* given the aspect *food*, while the base^{ent+} model predicts *negative*, ignoring the word "*delish*".

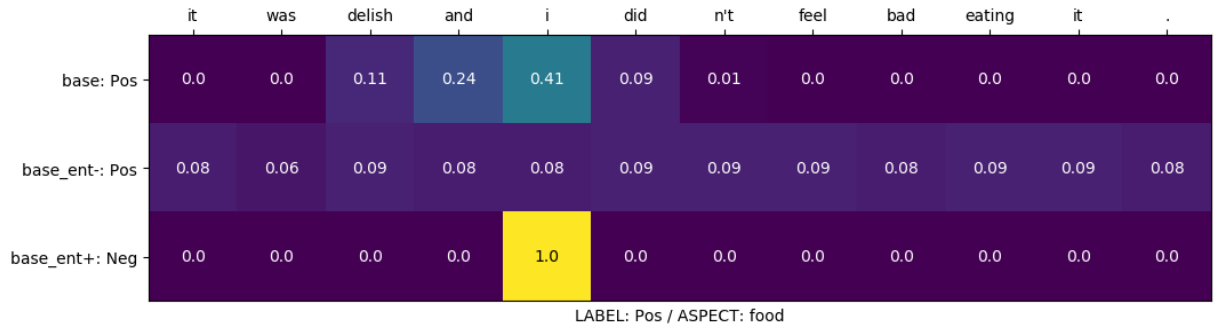

Figure 6.15: Comparison of attention weights between baseline, base^{ent-} and base^{ent+}.

6.3 Sentiment Induction Applied

So far, the experiments on ATLX have been using sentiment lexicons in a generic domain; to further improve the system, the idea of domain adaptation comes naturally. Currently, most works on sentiment domain adaptation measure the performance by recreating an existing domain-specific lexicon (Hamilton et al., 2016; Mudinas et al., 2018). It is less frequent to see how much improvement can actually be gained in an applied case. Thus from an application point of view, we ask the question: how much improvement can we get from the domain-specific lexicon in a lexicon enhanced neural sentiment analysis system? And what is its limit?

To answer these questions, we first apply the method described in Section 5.4.1 to adapt our generic sentiment lexicon described in Section 5.2.1 to a domain specific lexicon (electronics review). Then we experiment with the adapted lexicons by applying them in the ATLX model and test the model performance on the SemEval 2015 laptop domain dataset. To better understand the quality of the adapted lexicon and the performance gain it is able to obtain, we also compare the adapted lexicon with the gold lexicon that we constructed (Section 5.4.2). Details of the experimental results will be described in Section 6.3.2.

On the other hand, as described in Section 5.4.3, we would like to expand the domain adaptation method and apply it to aspect adaptation. In other words, expand the existing generic lexicon to be aspect specific. Since there is no gold lexicon to evaluate and compare, we test the quality of the aspect adapted lexicon by applying it in the ATLX model and measuring the performance differences. Details of the experiment will be described in Section 6.3.2.

6.3.1 Dataset

We experiment the domain adaptation performance on the SemEval 2015 Task 12, laptop dataset, same as the ATLX experiments described in Sec-

tion 6.1.1. The dataset consists of reviews of laptops with annotated entity-attribute pairs such as: {*LAPTOP # GENERAL, KEYBOARD # QUALITY, LAPTOP # PRICE, ...*} and associated polarities: {*positive, neutral, negative*}. To be compatible with the ATLX system, we simplify the entity-attribute annotations by keeping only the attributes as aspects: {*general, performance, design, usability, portability, price, quality, miscellaneous, connectivity*}. Together, there are 1,973 training examples and 949 test examples in the corpus.

6.3.2 Evaluation

Same as the ATLX experiments, a cross validation of 6 folds is performed and the average accuracy on both development sets and test set is recorded together with the variance. We also compare the adapted lexicons with the gold lexicon by measuring their accuracy and f-score in both binary and ternary scenarios, where *neutral* is excluded from binary but included in ternary. Table 6.11 shows the evaluation results.

Table 6.12 shows the performance of the aspect adapted lexicons when applied in the ATLX model, together with the model performance of the domain adapted lexicons and other variations. The subscripts $_{\text{add}}$, $_{\text{avg}}$, $_{\text{concat}}$ correspond to the $\oplus$ operation described in Section 5.4.3; where each of them stands for summation, mean and concatenation of the domain specific word vector v_s^i and the domain specific aspect vector v_a^j respectively.

6.3.3 Discussion

Sentiment Domain Adaptation

As shown in Table 6.11, we observe that in the laptops review domain, the generic lexicon improves performance compared to no lexicon applied. Moreover, the accuracy keeps increasing on the test set as the lexicon gets more similar to the gold one, i.e. performance on No Lexicon, Generic,

(a) DAL ATLX Performance

	CV	σ^{CV}	Test	σ^{Test}
No lexicon	82.48	2.15	74.06	0.62
Generic	**83.39**	2.64	75.92	1.50
DAL_{bin}	82.63	1.38	76.24	1.12
DAL_{ter}	82.02	1.29	**77.08**	0.61
Gold	82.47	1.71	77.21	1.20

(b) DAL Lexicon Evaluation

	Binary		*Ternary*	
	ACC.	F1	ACC.	F1
No lexicon	-	-	-	-
Generic	**96.58**	**0.97**	**77.00**	0.74
DAL_{bin}	89.93	0.90	64.60	0.58
DAL_{ter}	80.22	0.88	75.45	**0.75**
Gold	100.0	1.00	100.0	1.00

Table 6.11: (a) ATLX model performance (average cross validation accuracy and variance) with Domain Adapted Lexicons (DAL) on SemEval15 Task 12, laptop dataset. **(b)** Accuracy and f-score of DALs measured against the gold lexicon, where *binary* excludes *neutral* and *ternary* not. The subscripts $_{\text{bin}}$ and $_{\text{ter}}$ refer to binary classification and ternary classification respectively.

AAL ATLX performance

	CV	σ^{CV}	Test	σ^{Test}
No lexicon	82.48	2.15	74.06	0.62
Generic	**83.39**	2.64	75.92	1.50
DAL_{bin}	82.63	1.38	76.24	1.12
DAL_{ter}	82.02	1.29	**77.08**	0.61
AAL_{add}	82.47	1.56	74.61	1.27
AAL_{avg}	82.78	0.83	74.57	1.67
AAL_{concat}	82.32	0.87	75.24	1.19
Gold	82.47	1.71	77.21	1.20

Table 6.12: ATLX model performance (average cross validation accuracy and variance) with Aspect Adapted Lexicons (AAL) on SemEval15 Task 12, laptop dataset.

DAL_{bin}, and DAL_{ter} gets increasingly similar to Gold. However, the performance on the CV sets does not have a clear pattern, in particular, the generic lexicon outperforms all domain specific lexicons including the gold one; nevertheless, it's worth noticing that the generic lexicon does cause a larger variance on the CV sets; and the size of the dataset is rather small to obtain an overall robust performance.

Regarding the domain adaptation method, when applied in ATLX, the domain adapted lexicon (DAL) achieves comparable results compared to the gold lexicon, especially after *neutral* seeds are used for ternary classification. However, the gold lexicon only improves the performance on the test set by 1.29% but also decreases on the dev set by 0.92%; indicating the performance ceiling of the best possible domain adaptation method on this dataset.

Looking at the lexicon evaluation, a good score compared to the gold lexicon does not necessarily translate to good performance when applied in the model. In addition, as shown in Figure 6.16, both the Gold and the DAL_{ter} lexicon have noticeably more *neutral* words than others; suggesting

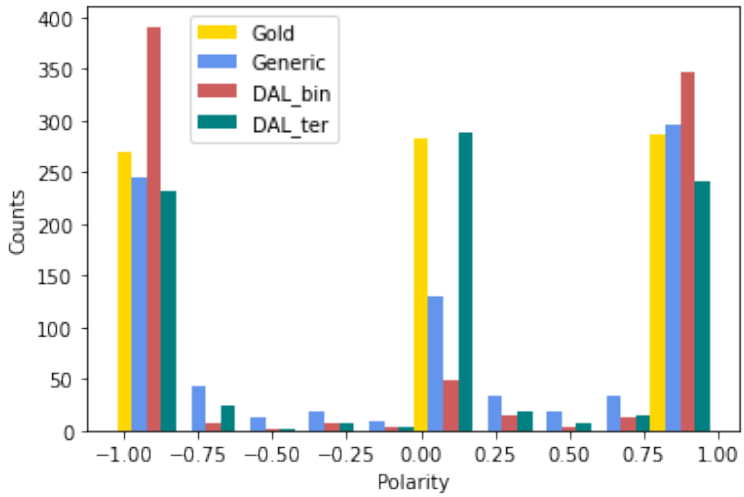

Figure 6.16: Polarity distribution of different lexicons.

that bias exists in generic lexicon and it is important to include *neutral* for sentiment domain adaptation.

In addition, as no automatic domain adaptation method can avoid introducing noise, we wonder how lexicon noise affects the model performance. Figure 6.17 shows the cross validation results of the ATLX model with respect to the increasing size of noisy lexicon entries, where the noise is added by flipping the polarity in the gold lexicon to be a random choice between any opposite polarities but the annotated one (e.g. the polarity of *good* will be changed to be a random choice between *neutral* or *negative*). We find that the model is sensitive to lexicon noise as a significant performance decrease is observed since 20% noise level. Interestingly, the model seems capable of ignoring noisy lexical information because when the noise level keeps increasing, the performance remains close to when no lexicon is applied.

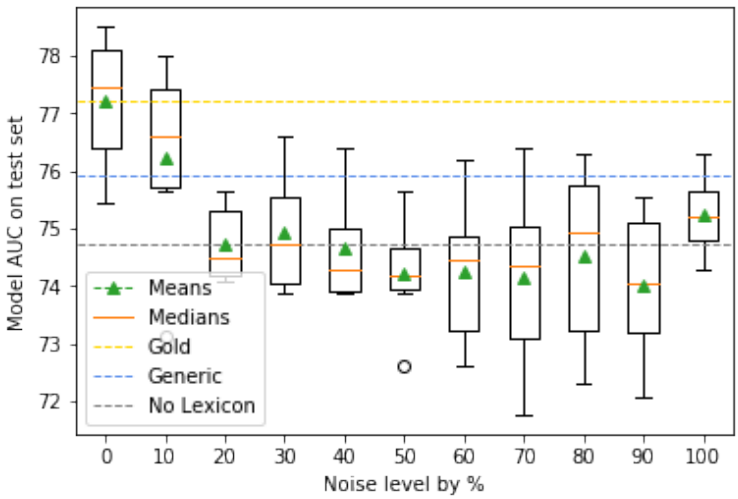

Figure 6.17: Model performance in accuracy by increasing size of noise in lexicon.

Sentiment Aspect Adaptation

As shown in Table 6.12, we create three aspect adapted lexicons with different ways of combining the word vector and the aspect vector. First we can see that all three methods do not make a huge difference in terms of model performance. Secondly, when compared to the baseline (no lexicon applied), there is only a marginal improvement with an exception of $\text{AAL}_{\text{concat}}$ that suffers a decrease on the CV sets. Thirdly, when compared to the ATLX model with generic lexicon, none of the three aspect adapted lexicons are able to achieve superior performance. Thus we can conclude that the aspect adaptation method does not provide the model extra useful information to make a better prediction.

The reason for that is most likely due to the limitation in terms of size and variability of the training data ensembled from the seed words. More importantly, the aspect specific polarity difference cannot be properly reflected in the ensembled training data. In other words, the idea is to have the classifier to be able to disambiguate aspect-dependent opinion words such as "***cheap*** *price*" vs "***cheap*** *design*" and "***low*** *price*" vs "***low*** *quality*". However the size of the seed words limits the chance of these important examples to appear in the training data. On the other hand, due to the selected seed words themselves, they are more likely to be homogeneously positive or negative on all aspects, making for the classifier even harder to learn the difference between aspect-dependent opinion words.

On the other hand, this kind of ambiguity is not usually associated with the aspect directly, it is mostly based on the context. For example, consider the case "***black*** *screen*" vs "***black*** *macbook*"; where "*black screen*" is indirectly associated with the aspect *quality*, and "*black macbook*" is indirectly associated with the aspect *design*. Thus it is hard to connect ambiguous opinion words with the aspects directly. For similar examples consider: "*battery lasts* ***long***" vs "*takes a* ***long*** *time to load*"; "*the laptop will* ***burn*** *my lags*" vs "***burn*** *3 dvds*".

Chapter 7

CONCLUSION

In this chapter, we review the motivations that drive the research in this thesis and summarize the findings of our work. Specifically, by addressing the research questions and objectives of this thesis, we can conclude that, in this work, we propose ATLX, a simple yet effective approach to merge the lexical features given by sentiment lexicon with an attention based LSTM neural network for ABSA. We experiment our approach on two different datasets from different domains and the results prove the effectiveness of our approach.

In addition, we find that the commonly used attention mechanism is likely to over-fit, especially when applied early in the network for a task such as ABSA. This over-fitting effect hurts the performance by binding the model from key elements for polarity judgement. The effect is shown by comparing the difference between ATLX and the baseline model. Moreover, the effect is also shown when we experiment with two attention regularizers that try to overcome this over-fitting effect. The experimental results show that regularizing the attention vector from being sparse can lead to performance improvement. However, both proposed regularizers are not ideal as the optimum of both regularizers suggests uniformly distributed weights, which is the opposite of what the attention mechanism is

able to gain.

Lastly, we try to improve the ATLX system by adapting the generic lexicon to a domain-specific one, or even an aspect-specific one. To do that we test the performance gain of sentiment domain adaptation in our system, as most existing researches measure sentiment domain adaptation by recreating an existing domain-specific lexicon. The experimental results suggest that when applied, the improvement from domain adaptation is limited and a good evaluation on lexicon recreation does not necessarily translate to model performance gain.

7.1 ATLX

In this thesis, we start by addressing that the deep learning based ABSA system lacks robustness and flexibility. For instance in Section 6.1.5 of Chapter 6, we see that the baseline AT-LSTM model is not able to recognize words with a clear polarity indication such as "*dungeon*", "*disappointed*" and "*terrible*", thus making the model incapable of predicting correctly sentiment labels. To fix these issues, one can sometimes gather more training data and retrain the model; however, this option is not always possible. In addition, from an application point of view, it makes a lot of sense to have an easy way to tell the model the polarities of these words so that the model can leverage this information and make a better prediction. Therefore the idea of leveraging existing sentiment lexicons to improve the baseline model comes naturally. Sentiment lexicons as freely available language resources are easy to access; besides, they are also easier to maintain compared to gathering data and retraining the model in a practical scenario. Thus we ask the question, how to improve a neural ABSA system by using sentiment lexicons? And compared to existing approaches, is there a simpler yet effective way of doing it?

To answer these questions, we design a set of approaches and test their performance with both the SemEval 2014, restaurant dataset and the Se-

mEval 2015, laptop dataset. Among these designed approaches, namely the ATLX model (Section 5.2 of Chapter 5) and its variants (Section 6.1.3 of Chapter 6), ATLX yields largest improvements on both datasets of different domains. ATLX consists of applying the attention vector, learned from the concatenation of LSTM outputs and aspect embeddings, directly on the lexical features given by the sentiment lexicon, which is composed by four existing generic domain lexicons; and then summing all lower level representations that are projected by model parameters to form the final representation for sentiment prediction.

In Table 6.6 the mean accuracy and standard deviation of the cross-validation results on 6 folds of development sets and one holdout test set are recorded. Compared to the baseline, ATLX improves on both the development sets and the test set of two datasets from different domains. The experimental results show the effectiveness of our proposed approach, and part of the outcomes has been published in Bao et al. (2019).

7.2 Attention Regularization

As we look deeper into the difference between ATLX and the baseline, we notice that ATLX is not only able to leverage the newly introduced lexical features, but also able to better distribute the attention weights accordingly. In some cases, even without having any valuable lexical features, the attention weights of the ATLX model are still more reasonably shaped. Details of these observations are discussed in Section 6.1.5 of Chapter 6.

Inspired by these observations, we notice that the attention vector could be over-fitting by "focusing" too much on particular parts of the sentence, and overlooking other positions which are key for judging the polarity. In order to verify this hypothesis, we conduct experiments with two attention regularizers, namely the standard deviation regularizer and the negative entropy regularizer described in Section 5.3 of Chapter 5.

From experiments conducted on both datasets (Table 6.9), both regular-

izers improve on the test sets where the negative entropy regularizer yields the largest improvement. Analysis shown in Figure 6.14 also suggests that by shifting the attention weight a bit more on the overlooked part of the sentence, the model is able to correctly predict the sentiment. Compared to other attention regularization methods that try to make the attention vector more sparse (Niculae and Blondel, 2017; Zhang et al., 2019a), we demonstrate that when applied early in the model, for instance in the ABSA task, the sparseness of the attention vector could hurt the model; and by regularizing the attention vector, we are able to overcome some of this negative effect. The research on attention regularization has been published in Bao et al. (2019).

7.3 Sentiment Induction

With the ATLX model that is able to leverage sentiment information provided by the lexicon, we try to further improve the system by applying sentiment domain adaptation and sentiment aspect adaptation. To do that we adopt a state of the art sentiment domain adaptation method and compare the performance gain of different lexicons in the ATLX system. We test the performance ceiling of the domain adaptation method by annotating a gold domain-specific lexicon and compare it with automatically adapted ones. In addition, we also apply a similar method to create an aspect specific lexicon and test its performance. However, as described in Section 6.3.3, due to the fact that it is hard to reflect aspect dependent cases in the training data ensembled from a limited number of seed words, the aspect adaptation does not yield any improvement on the model performance.

Regarding the domain adaptation experiments, as shown in Table 6.11, we find that in general, domain-specific lexicons do improve the model performance compared to a generic one; however, the performance ceiling suggested by the gold lexicon is rather low; and ternary classification is more reasonable than binary as model performance is sensitive to noise.

Moreover, compared to common methods that evaluate the domain adaptation method by recreating an existing lexicon, in our experiments we see that a good score on the lexicon evaluation does not necessarily translate to a final performance gain; thus evaluation in an applied scenario is indeed necessary and valuable.

7.4 Future Works

In this thesis, perhaps the outcomes of the attention regularization experiments are the least expected. As described in Section 6.2.2, in Figure 6.14, the negative entropy regularizer applied in the baseline system leads to an almost evenly distributed attention vector. However, this kind of distribution does not hurt the model at all, in fact improvements compared to the baseline are generally observed for the negative entropy regularizer (Table 6.9).

As a fundamental building block of many state of the art models (e.g. Transformer, BERT), the attention framework is designed mimicking the attention of a human being when processing information. In other words, it supposes to discard irrelevant information and "focus" on particular key points. And one would expect having almost evenly distributed attention weights could hurt the model. However, the results in our experiments suggest otherwise. Similarly, the interpretability of attention is questioned in Serrano and Smith (2019) as well. In our case, one possible explanation is that, in our model, the attention is applied early in the network, thus filtering more information (i.e. the attention weight distribution being overly sparse) could hurt the performance by passing too little information to deeper layers of the network.

Due to limited time, in this doctoral research project, we were not able to follow this hypothesis. Thus future works could focus on the relation between the sparsity of the attention distribution and the position that the attention mechanism is applied in the network. In addition, it is also in-

teresting to see whether more recent multi-head attention models such as Transformer (Vaswani et al., 2017), or other network architectures such as Bi-LSTM could suffer from similar attention over-fitting issues.

Appendix A

EXPERIMENT SETTINGS

A.1 Parameter Settings

ϵ $\Omega(\alpha)$	Value
ϵ base$^{\text{std}}$	1e-3
ϵ base$^{\text{ent-}}$	0.5
ϵ base$^{\text{ent+}}$	0.025
ϵ ATLX$^{\text{std}}$	1e-4
ϵ ATLX$^{\text{ent-}}$	0.006
ϵ ATLX$^{\text{ent+}}$	0.001

Table A.1: Attention regularization parameter settings.

In the ATLX experiments, apart from the newly introduced parameter, namely the ϵ parameter for attention regularization, we follow what is described in the paper by Wang et al. (2016c) and their released code.

More specifically, we set batch size as 25; aspect embedding dimension d_a equals to 300, same as Glove vector dimension; number of LSTM cell d as 300; number of LSTM layers as 1; dropout with 0.5 keep probability

is applied to h^*; AdaGrad optimizer is used with initial accumulate value equals to $1e-10$; learning rate is set to 0.01; L_2 regularization parameter λ is set to 0.001; network parameters are initialized from a random uniform distribution with min and max values as -0.01 and 0.01; all network parameters except word embeddings are included in the L_2 regularizer. The hyperparameter ϵ for attention regularization is shown in Table A.1.

A.2 Resources

Language Resources

- MPQA: https://bit.ly/2Ia4u74
- Opinion Lexicon: https://bit.ly/36JNmPN
- Opener: https://bit.ly/3iIrnv1
- Vader: https://bit.ly/3jJ95uH
- Glove: https://stanford.io/2FeYJnn

Implementation Code

- AT-LSTM code by Wang et al. (2016c): https://bit.ly/2I9H4yx
- ATLX and attention regularization code: https://github.com/LingxB/atlx
- Domain and aspect adaptation code: https://github.com/LingxB/dalx

Other resources

- ATLX and related experiments are implemented with TensorFlow
- SVM related experiments are implemented with Scikit-Learn

Bibliography

A M Turing, I. B. (1950). I. — Computing Machinery and Intelligence. *Mind Association Computing Machinery and Intelligence Author(s): A. M. Turing Source: Mind*, 59(236):433–460.

Abbasi, A., Chen, H., and Salem, A. (2008). Sentiment analysis in multiple languages: Feature selection for opinion classification in Web forums. *ACM Transactions on Information Systems*, 26(3).

Agarwal, B., Mittal, N., Bansal, P., and Garg, S. (2015). Sentiment analysis using common-sense and context information. *Computational Intelligence and Neuroscience*, 2015.

Akhtar, M. S., Kumar, A., Ghosal, D., Ekbal, A., and Bhattacharyya, P. (2017). A Multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 540–546.

Asur, S. and Huberman, B. A. (2010). Predicting the future with social media. In *Proceedings - 2010 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2010*, volume 1, pages 492–499.

Aue, A. and Gamon, M. (2005). Customizing Sentiment Classifiers to

New Domains: A Case Study. In *Recent Advances in Natural Language Processing (RANLP)*.

Bahdanau, D., Cho, K. H., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Bao, L., Lambert, P., and Badia, T. (2019). Attention and lexicon regularized LSTM for aspect-based sentiment analysis. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Student Research Workshop*, pages 253–259.

Bar-Haim, R., Dinur, E., Feldman, R., Fresko, M., and Goldstein, G. (2011). Identifying and following expert investors in stock microblogs. In *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1310–1319.

Barnes, J. (2019). LTG-Oslo hierarchical multi-task network: The importance of negation for document-level sentiment in Spanish. In *CEUR Workshop Proceedings*, volume 2421, pages 378–389.

Barnes, J., Klinger, R., and im Walde, S. S. (2018). Projecting embeddings for domain adaptation: Joint modeling of sentiment analysis in diverse domains.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bespalov, D., Bai, B., Qi, Y., and Shokoufandeh, A. (2011). Sentiment classification based on supervised latent n-gram analysis. In *International Conference on Information and Knowledge Management, Proceedings*, pages 375–382.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Boiy, E. and Marie-Francine, M. (2008). A machine learning approach to sentiment analysis in multilingual Web text. *Information Retrieval*, 12(5):526–558.

Bollegala, D., Weir, D., and Carroll, J. (2011). Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 132–141.

Bollen, J., Mao, H., and Zeng, X.-J. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Boser, B., Guyon, I., of the 5th, V. V. P., and 2003, U. (1992). A training algorithm for optimal margin classifiers. *Gautampendse.Com*, pages 144–152.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Caetano, J. A., Lima, H. S., Santos, M. F., and Marques-Neto, H. T. (2018). Using sentiment analysis to define twitter political users' classes and their homophily during the 2016 American presidential election. *Journal of Internet Services and Applications*, 9(1).

Cambria, E. (2016). Affective Computing and Sentiment Analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Ceron, A., Curini, L., Iacus, S. M., and Porro, G. (2014). Every tweet counts? how sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to italy and france. *New Media & Society*, 16(2):340–358.

Chen, B., Zhu, L., Kifer, D., and Lee, D. (2010). What is an opinion about? Exploring political standpoints using opinion scoring model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 2, pages 1007–1012.

Chen, H., Sun, M., Tu, C., Lin, Y., and Liu, Z. (2016). Neural sentiment classification with user and product attention. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1650–1659.

Chen, P., Sun, Z., Bing, L., and Yang, W. (2017). Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 452–461.

Cheng, K., Li, J., Tang, J., and Liu, H. (2017). Unsupervised sentiment analysis with signed social networks. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 3429–3435.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Ap-

proaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

Cimiano, P. and Völker, J. (2005). Towards large-scale, open-domain and ontology-based named entity classification. Technical report.

Cui, H., Mittal, V., and Datar, M. (2006). Comparative experiments on sentiment classification for online product reviews. In *Proceedings of the National Conference on Artificial Intelligence*, volume 2, pages 1265–1270.

Das, S. R., Chen, M. Y., Agarwal, V., Brooks, C., Chan, Y.-S., Gibson, D., Friesen, G., Leinweber, D., Martinez-Jerez, A., Raghubir, P., Ra-Jagopalan, S., Ranade, A., Rubinstein, M., Tufano, P., Uppal, R., Vaithyanathan, S., and Wilensky, R. (2004). Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web. Technical report.

Dasgupta, S. and Ng, V. (2009). Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 701–709.

Dave, K., Lawrence, S., and Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on World Wide Web, WWW 2003*, pages 519–528.

Davidov, D., Tsur, O., and Rappoport, A. (2010). Enhanced sentiment learning using twitter hashtags and smileys. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 241–249.

Dayalani, G. G., Quadri, S., and Patil, B. K. (2014). Emoticon-based unsupervised sentiment classifier for polarity analysis in tweets. *International Journal of Engineering Research and General Science*, 2(6).

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 4171–4186.

Di Caro, L. and Grella, M. (2013). Sentiment analysis via dependency parsing. *Computer Standards and Interfaces*, 35(5):442–453.

Ding, X., Liu, B., and Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. In *WSDM'08 - Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–239.

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., and Xu, K. (2014). Adaptive Recursive Neural Network for target-dependent Twitter sentiment classification. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 2, pages 49–54. Association for Computational Linguistics.

Dos Santos, C. N. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *COLING 2014 - 25th International Conference on Computational Linguistics, Proceedings of COLING 2014: Technical Papers*, pages 69–78.

Dou, Z. Y. (2017). Capturing user and product information for document level sentiment analysis with deep memory network. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 521–526.

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

Esuli, A. and Sebastiani, F. (2005). Determining the semantic orientation of terms through gloss classification. In *International Conference on Information and Knowledge Management, Proceedings*, pages 617–624.

Fan, Z.-P., Che, Y.-J., and Chen, Z.-Y. (2017). Product sales forecasting using online reviews and historical sales data: A method combining the bass model and sentiment analysis. *Journal of Business Research*, 74:90–100.

Farhat, N. H. (1992). Photonit neural networks and learning mathines the role of electron-trapping materials. *IEEE Expert-Intelligent Systems and their Applications*, 7(5):63–72.

Feldman, R., Rosenfeld, B., Bar-Haim, R., and Fresko, M. (2011). The stock sonar - Sentiment analysis of stocks based on a hybrid approach. *Proceedings of the National Conference on Artificial Intelligence*, 2(January 2011):1642–1647.

Fernández-Gavilanes, M., Juncal-Martínez, J., García-Méndez, S., Costa-Montenegro, E., and González-Castaño, F. J. (2018). Creating emoji lexica from unsupervised sentiment analysis of their descriptions. *Expert Systems with Applications*, 103:74–91.

Frey, B. J. and Dueck, D. (2008). Clustering by passing messages between data points".

Gamon, M., Aue, A., Corston-Oliver, S., and Ringger, E. (2005). Pulse: Mining customer opinions from free text. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3646 LNCS, pages 121–132.

Gandhi, H. and Attar, V. (2020). Extracting Aspect Terms using CRF and Bi-LSTM Models. In *Procedia Computer Science*, volume 167, pages 2486–2495.

García-Díaz, J. A., Cánovas-García, M., and Valencia-García, R. (2020). Ontology-driven aspect-based sentiment analysis classification: An infodemiological case study regarding infectious diseases in Latin America. *Future Generation Computer Systems*, 112:641–657.

Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194 vol.3.

Giannakopoulos, A., Musat, C., Hossmann, A., and Baeriswyl, M. (2017). Unsupervised aspect term extraction with B-LSTM & CRF using automatically labelled datasets.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 513–520.

Goldberg, A. B. and Zhu, X. (2006). Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of TextGraphs: The 1st Workshop on Graph-Based Methods for Natural Language Processing*, pages 45–52.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 2047–2052.

Groh, G. and Hauffa, J. (2011). Characterizing Social Relations Via NLP-based Sentiment Analysis. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 502–505.

Guan, Q., Huang, Y., Zhong, Z., Zheng, Z., Zheng, L., and Yang, Y. (2018). Diagnose like a Radiologist: Attention Guided Convolutional Neural Network for Thorax Disease Classification.

Guan, Z., Chen, L., Zhao, W., Zheng, Y., Tan, S., and Cai, D. (2016). Weakly-supervised deep learning for customer review sentiment classification. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2016-Janua, pages 3719–3725.

Guggilla, C., Miller, T., and Gurevych, I. (2016). CNN-and LSTM-based Claim Classification in Online User Comments. In *Proceedings of the International Conference on Computational Linguistics (COLING 2016)*, pages 2740–2751.

Hamilton, W. L., Clark, K., Leskovec, J., and Jurafsky, D. (2016). Inducing domain-specific sentiment lexicons from unlabeled corpora. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 595–605.

Hassan, A., Qazvinian, V., and Radev, D. (2010). What's with the Attitude? Identifying sentences with attitude in online discussions. In *EMNLP 2010 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, number 11, pages 1245–1255. Association for Computational Linguistics.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. pages 174–181.

Hatzivassiloglou, V. and Wiebe, J. M. (2000). Effects of adjective orientation and gradability on sentence subjectivity. pages 299–305.

Hayashi, T. and Fujita, H. (2019). Word embeddings-based sentence-level sentiment analysis considering word importance. *Acta Polytechnica Hungarica*, 16(7):7–24.

He, R., Lee, W. S., Ng, H. T., and Dahlmeier, D. (2018). Effective Attention Modeling for Aspect-Level Sentiment Classification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1121–1131.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hobbs, Jerry; Riloff, E. (2010). Information Extraction. In *Handbook of Natural Language Processing*, pages 129–135.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hong, Y. and Skiena, S. (2010). The wisdom of bookies? Sentiment analysis versus the NFL point spread. *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 251–254.

Hu, M. and Liu, B. (2004a). Mining and summarizing customer reviews. In *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Hu, M. and Liu, B. (2004b). Mining opinion features in customer reviews. In *Proceedings of the National Conference on Artificial Intelligence*, pages 755–760.

Hu, X., Tang, J., Gao, H., and Liu, H. (2013). Unsupervised sentiment analysis with emotional signals. In *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, pages 607–617.

Huang, J., Meng, Y., Guo, F., Ji, H., and Han, J. (2020). Weakly-Supervised Aspect-Based Sentiment Analysis via Joint Aspect-Sentiment Topic Embedding.

Huang, M., Qian, Q., and Zhu, X. (2017). Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems*, 35(3).

Ilyas, S. H. W., Soomro, Z. T., Anwar, A., Shahzad, H., and Yaqub, U. (2020). Analyzing brexit's impact using sentiment analysis and topic modeling on twitter discussion. In *The 21st Annual International Conference on Digital Government Research*, dg.o '20, page 1–6, New York, NY, USA. Association for Computing Machinery.

Jain, A., Pal Nandi, B., Gupta, C., and Tayal, D. K. (2020). SentiNSetPSO: large-sized document-level sentiment analysis using Neutrosophic Set and particle swarm optimization. *Soft Computing*, 24(1):3–15.

Jakob, N. and Gurevych, I. (2010). Extracting opinion targets in a single- and cross-domain setting with Conditional Random Fields. In *EMNLP 2010 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, number 11, pages 1035–1045. Association for Computational Linguistics.

Jebbara, S. and Cimiano, P. (2016). Aspect-based sentiment analysis using a two-step neural network architecture. In *Communications in Computer and Information Science*, volume 641, pages 153–167.

Jiang, L., Yu, M., Zhou, M., Liu, X., and Zhao, T. (2011). Target-dependent Twitter sentiment classification. In *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 151–160.

Jiang, M., Lan, M., and Wu, Y. (2018). ECNU at SemEval-2017 Task 5: An Ensemble of Regression Algorithms with Effective Features for Fine-Grained Sentiment Analysis in Financial Domain. pages 888–893.

Jiménez-Zafra, S. M., Martín-Valdivia, M. T., Martínez-Cámara, E., and Ureña-López, L. A. (2016). Combining resources to improve unsupervised sentiment analysis at aspect-level. *Journal of Information Science*, 42(2):213–229.

Jin, W. and Ho, H. H. (2009). A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, pages 465–472.

Jin, W., Ho, H. H., and Srihari, R. K. (2009). Opinionminer: A novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 1195–1204, New York, NY, USA. Association for Computing Machinery.

Joachims, T. (1998). Making Large-Scale SVM Learning Practical.

Johnson, R. and Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Proceedings of the Conference, pages 103–112.

Joshi, M., Das, D., Gimpel, K., and Smith, N. A. (2010). Movie reviews and revenues: An experiment in text regression. *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, (May 2014):293–296.

Jurafsky, D. and Martin, J. (2020). Speech and Language Processing. In *Speech and Language Processing.*, volume 3, pages 441–458.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 655–665. Association for Computational Linguistics.

Kamps, J., Marx, M., Mokken, R. J., and De Rijke, M. (2004). Using WordNet to measure semantic orientations of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation, LREC 2004*, pages 1115–1118.

Kanayama, H. and Nasukawa, T. (2006). Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *COLING/ACL 2006 - EMNLP 2006: 2006 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 355–363. Association for Computational Linguistics.

Karimi, A., Rossi, L., Prati, A., and Full, K. (2020). Adversarial Training for Aspect-Based Sentiment Analysis with BERT.

Kennedy, A. and Inkpen, D. (2006). Sentiment classification of movie reviews using contextual valence shifters. In *Computational Intelligence*, volume 22, pages 110–125.

Kim, J., Li, J. J., and Lee, J. H. (2009). Discovering the discriminative views: Measuring term weights for sentiment analysis. In *ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf.*, pages 253–261.

Kim, S.-M. and Hovy, E. (2004). Determining the sentiment of opinions. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1367–es.

Kim, S.-M. and Hovy, E. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. pages 1–8.

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

Kobayashi, N., Inui, K., and Matsumoto, Y. (2007). Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL 2007 - Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1065–1074.

Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork RNN. In *31st International Conference on Machine Learning, ICML 2014*, volume 5, pages 3881–3889.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Lafferty, J., Mccallum, A., and Pereira, F. (2001). Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data Abstract. (June):282–289.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 260–270.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *31st International Conference on Machine Learning, ICML 2014*, volume 4, pages 2931–2939.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323.

Lei, T., Barzilay, R., and Jaakkola, T. (2016). Rationalizing neural predictions. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 107–117.

Lei, Z., Yang, Y., and Yang, M. (2018). Sentiment lexicon enhanced attention-based LSTM for sentiment classification. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 8105–8106.

Li, C., Guo, X., and Mei, Q. (2017a). Deep memory networks for attitude identification. In *WSDM 2017 - Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 671–680.

Li, F., Han, C., Huang, M., Zhu, X., Xia, Y. J., Zhang, S., and Yu, H. (2010). Structure-aware review mining and summarization. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 653–661.

Li, J., Sun, A., Han, J., and Li, C. (2020a). A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Li, J., Yang, H., and Zong, C. (2018). Document-level Multi-aspect Sentiment Classification by Jointly Modeling Users, Aspects, and Overall Ratings. Technical report.

Li, T., Zhang, Y., and Sindhwani, V. (2009). A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf.*, pages 244–252.

Li, W., Zhu, L., Shi, Y., Guo, K., and Cambria, E. (2020b). User reviews: Sentiment analysis using lexicon integrated two-channel cnn-lstm family models. *Applied Soft Computing*, 94:106435.

Li, Z., Zhang, Y., Wei, Y., Wu, Y., and Yang, Q. (2017b). End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 2237–2243.

Lin, S., Su, W., Chien, P., Tsai, M., and Wang, C. (2020). Self-attentive sentimental sentence embedding for sentiment analysis. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1678–1682.

Liu, B. (2010). Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition*, pages 627–666.

Liu, B. (2011). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data Second Edition*.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Liu, B., Hu, M., and Cheng, J. (2005). Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of the 14th International Conference on World Wide Web*, pages 342–351.

Liu, F., Zheng, J., Zheng, L., and Chen, C. (2020a). Combining attention-based bidirectional gated recurrent neural network and two-dimensional convolutional neural network for document-level sentiment classification. *Neurocomputing*, 371:39 – 50.

Liu, F., Zheng, L., and Zheng, J. (2020b). Hienn-dwe: A hierarchical neural network with dynamic word embeddings for document level sentiment classification. *Neurocomputing*, 403:21 – 32.

Liu, J. and Seneff, S. (2009). Review sentiment scoring via a parse-and-paraphrase paradigm. In *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009*, pages 161–169.

Liu, J. and Yue, Z. (2017). Attention modeling for targeted sentiment. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, volume 2, pages 572–577.

Liu, Q., Zhang, H., Zeng, Y., Huang, Z., and Wu, Z. (2018). Content attention model for aspect based sentiment analysis. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1023–1032, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Liu, Y., Huang, X., An, A., and Yu, X. (2007). ARSA: A sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07*, pages 607–614.

Lu, Y., Castellanos, M., Dayal, U., and Zhai, C. (2011). Automatic construction of a context-aware sentiment lexicon: an optimization approach. *Proceedings of the 20th international conference on World wide web - WWW '11*, 92(June):347.

Lu, Y., Zhai, C. X., and Sundaresan, N. (2009). Rated aspect summarization of short comments. In *WWW'09 - Proceedings of the 18th International World Wide Web Conference*, pages 131–140.

Lutz, B., Pröllochs, N., and Neumann, D. (2018). Sentence-level sentiment analysis of financial news using distributed text representations and multi-instance learning.

Ma, D., Li, S., Wu, F., Xie, X., and Wang, H. (2020). Exploring sequence-to-sequence learning in aspect term extraction. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 3538–3547.

Ma, D., Li, S., Zhang, X., and Wang, H. (2017). Interactive attention networks for aspect-level sentiment classification.

Ma, Y., Kim, J. J., Bigot, B., and Khan, T. M. (2016). Feature-enriched word embeddings for named entity recognition in open-domain conversations. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2016-May, pages 6055–6059.

Ma, Y., Peng, H., and Cambria, E. (2018). Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 5876–5883.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *ACL-HLT*

2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, volume 1, pages 142–150.

MacQueen, J. (1967). SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS. *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1 (Univ. of Calif. Press, 1967).*, 25(8):281—-297.

Martineau, J., Martineau, J., Finin, T., Finin, T., Fink, C., Fink, C., Piatko, C., Piatko, C., Mayfield, J., Mayfield, J., Syed, Z., Syed, Z., Others, and Others (2008). Delta TFIDF: An Improved Feature Space for Sentiment Analysis. *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM*, 29(May):490–497.

McDonald, R., Hannan, K., Neylon, T., Wells, M., and Reynar, J. (2007). Structured models for fine-to-coarse sentiment analysis. In *ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 432–439. Association for Computational Linguistics.

McGlohon, M., Glance, N., and Reiter, Z. (2010). Star quality: Aggregating reviews to rank products and merchants. In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 114–121.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*.

Mikolov, T., Yih, W. T., and Zweig, G. (2013b). Linguistic regularities in continuous spaceword representations. In *NAACL HLT 2013 - 2013*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference, pages 746–751. Association for Computational Linguistics.

Miller, M., Sathi, C., Wiesenthal, D., Leskovec, J., and Potts, C. (2011). Sentiment Flow Through Hyperlink Networks. *Fifth International AAAI Conference on Weblogs and Social Media*, pages 550–553.

Mishra, A., Dey, K., and Bhattacharyya, P. (2017). Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 1, pages 377–387.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Moghaddam, S. and Ester, M. (2011). Ilda: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 665–674, New York, NY, USA. Association for Computing Machinery.

Mohammad, S., Dunne, C., and Dorr, B. (2009). Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009*, pages 599–608.

Mohammad, S. and Yang, T. (2011). Tracking Sentiment in Mail: How Genders Differ on Emotional Axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis, ACL-HLT 2011, pages 70–79*, pages 70–79.

Mohammad, S. M. (2012). From once upon a time to happily ever after: Tracking emotions in mail and books. In *Decision Support Systems*, volume 53, pages 730–741. Association for Computational Linguistics.

Mooney, R. J. and Bunescu, R. (2005). Mining knowledge from text using information extraction. *ACM SIGKDD Explorations Newsletter*, 7(1):3–10.

Moraes, R., Valiati, J. F., and Wilson, P. (2013). Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621 – 633.

Morinaga, S., Yamanishi, K., Tateishi, K., and Fukushima, T. (2002). Mining product reputations on the Web. pages 341–349, Edmonton, Alberta, Canada. ACM.

Mudinas, A., Zhang, D., and Levene, M. (2018). Bootstrap Domain-Specific Sentiment Classifiers from Unlabeled Corpora. *Transactions of the Association for Computational Linguistics*, 6:269–285.

Mullen, T. and Collier, N. (2004). Sentiment Analysis using Support Vector Machines with Diverse Information Sources. In *Proceedings of EMNLP 2004*, pages 412–418.

Muthukumar, P. and Zhong, J. (2021). A Stochastic Time Series Model for Predicting Financial Trends using NLP.

Nakagawa, T., Inui, K., and Kurohashi, S. (2010). Dependency tree-based sentiment classification using CRFs with hidden variables. In *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 786–794.

Nasukawa, T. and Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd International Conference on Knowledge Capture, K-CAP 2003*, pages 70–77.

Ng, V., Dasgupta, S., and Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 611–618.

Niculae, V. and Blondel, M. (2017). A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 3339–3349.

Nigam, K., Thrun, S., Mitchell, T. O. M., and McCallum, A. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*, 34:103–134.

O'Connor, B., Balasubramanyan, R., Routledge, B. R., and Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 122–129.

Pagolu, V. S., Reddy, K. N., Panda, G., and Majhi, B. (2017). Sentiment analysis of Twitter data for predicting stock market movements. In *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, pages 1345–1350. SCOPES.

Paltoglou, G. and Thelwall, M. (2012). Twitter, myspace, digg: Unsupervised sentiment analysis in social media. *ACM Trans. Intell. Syst. Technol.*, 3(4).

Pan, S. J., Ni, X., Sun, J. T., Yang, Q., and Chen, Z. (2010). Cross-domain sentiment classification via spectral feature alignment. In *Proceedings*

of the 19th International Conference on World Wide Web, WWW '10, pages 751–760.

Pang, B. and Lee, L. (2004). A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.

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

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? pages 79–86.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1532–1543.

Pokharel, B. P. (2020). Twitter Sentiment Analysis During Covid-19 Outbreak in Nepal. *SSRN Electronic Journal*.

Polanyi, L. and Zaenen, A. (2005). Contextual valence shifters. In *AAAI Spring Symposium - Technical Report*, volume SS-04-07, pages 106–111.

Popescu, A. M. and Etzioni, O. (2005). Extracting product features and opinions from reviews. In *HLT/EMNLP 2005 - Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 339–346.

Poria, S., Cambria, E., and Gelbukh, A. (2016). Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Pröllochs, N., Feuerriegel, S., and Neumann, D. (2019). Learning interpretable negation rules via weak supervision at document level: A reinforcement learning approach. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 407–413.

Qian, Q., Huang, M., Lei, J., and Zhu, X. (2017). Linguistically regularized LSTM for sentiment classification. In *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, volume 1, pages 1679–1689.

Qian, Q., Tian, B., Huang, M., Liu, Y., Zhu, X., and Zhu, X. (2015). Learning tag embeddings and tag-specific composition functions in recursive neural network. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, volume 1, pages 1365–1374.

Qiu, L., Zhang, W., Hu, C., and Zhao, K. (2009). SELC: A self-supervised model for sentiment classification. In *International Conference on Information and Knowledge Management, Proceedings*, pages 929–936.

Qu, L., Ifrim, G., and Weikum, G. (2010). The bag-of-opinions method for review rating prediction from sparse text patterns. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 913–921.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286.

Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment.

Ramachandran, P., Bello, I., Parmar, N., Levskaya, A., Vaswani, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems*, volume 32.

Rao, G., Huang, W., Feng, Z., and Cong, Q. (2018). Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49 – 57.

Ren, R., Wu, D. D., and Liu, T. (2019). Forecasting stock market movement direction using sentiment analysis and support vector machine. *IEEE Systems Journal*, 13(1):760–770.

Ren, Z., Zeng, G., Chen, L., Zhang, Q., Zhang, C., and Pan, D. (2020). A Lexicon-Enhanced Attention Network for Aspect-Level Sentiment Analysis.

Rhanoui, M., Mikram, M., Yousfi, S., and Barzali, S. (2019). A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Machine Learning and Knowledge Extraction*, 1(3):832–847.

Rietzler, A., Stabinger, S., Opitz, P., and Engl, S. (2020). Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification. In *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*, pages 4933–4941.

Riloff, E. and Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. *Proceedings of the National Conference on Artificial Intelligence*, pages 474–479.

Riloff, E., Patwardhan, S., and Wiebe, J. (2006). Feature subsumption for opinion analysis. In *COLING/ACL 2006 - EMNLP 2006: 2006 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 440–448.

Riloff, E. and Wiebe, J. (2003). Learning extraction patterns for subjective expressions. pages 105–112.

Rothe, S., Ebert, S., and Schütze, H. (2016). Ultradense word embeddings by orthogonal transformation. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 767–777.

Rothe, S. and Schütze, H. (2016). Word embedding calculus in meaningful ultradense subspaces. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Short Papers*, pages 512–517.

Rouvier, M. and Favre, B. (2016). SENSEI-LIF at SemEval-2016 Task 4 : Polarity embedding fusion for robust sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 207–213.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2016). A hierarchical model of reviews for aspect-based sentiment analysis. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 999–1005.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Saad, S. E. and Yang, J. (2019). Twitter Sentiment Analysis Based on Ordinal Regression. *IEEE Access*, 7:163677–163685.

Sadikov, E., Parameswaran, A., and Venetis, P. (2009). Blogs as predictors of movie success. *Third International AAAI Conference on Weblogs and Social Media*, pages 304–307.

Saji, T. and Whig, P. (2020). Unsupervised Sentiment Analysis Using Small Recurrent Language Models. *arXiv*, 8:171–181.

Sakunkoo, P. and Sakunkoo, N. (2009). Analysis of Social Influence in Online Book Reviews. *AAAI's ICWSM 2009*, pages 1968–1970.

Schmitt, M., Steinheber, S., Schreiber, K., and Roth, B. (2020). Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 1109–1114.

Scholkopf, B. and Smola, A. J. (2018). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series.

Sculley, D. (2010). Web-Scale K-Means Clustering. In *Proceedings of the 19th international conference on World wide web*.

Serrano, S. and Smith, N. A. (2019). Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Shin, B., Lee, T., and Choi, J. D. (2017). Lexicon Integrated CNN Models with Attention for Sentiment Analysis. In *ACL*, pages 149–158.

Snyder, B. and Barzilay, R. (2007). Multiple aspect ranking using the good grief algorithm. In *NAACL HLT 2007 - Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 300–307.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL 2012 - 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Proceedings of the Conference*, pages 1201–1211. Association for Computational Linguistics.

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

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1631–1642. Association for Computational Linguistics.

Sun, C., Huang, L., and Qiu, X. (2019). Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 380–385.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence

learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 4, pages 3104–3112.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. Technical report.

Täckström, O. and McDonald, R. (2011). Semi-supervised latent variable models for sentence-level sentiment analysis. In *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 569–574.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-Term memory networks. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, volume 1, pages 1556–1566.

Tan, S., Wu, G., Tang, H., and Cheng, X. (2007). A novel scheme for domain-transfer problem in the context of sentiment analysis. In *International Conference on Information and Knowledge Management, Proceedings*, pages 979–982.

Tang, D., Qin, B., Feng, X., and Liu, T. (2016a). Effective LSTMs for target-dependent sentiment classification. In *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, pages 3298–3307.

Tang, D., Qin, B., and Liu, T. (2015a). Document modeling with gated recurrent neural network for sentiment classification. In *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural*

Language Processing, pages 1422–1432. Association for Computational Linguistics.

Tang, D., Qin, B., and Liu, T. (2015b). Learning semantic representations of users and products for document level sentiment classification. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, volume 1, pages 1014–1023.

Tang, D., Qin, B., and Liu, T. (2016b). Aspect level sentiment classification with deep memory network. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 214–224.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 1555–1565. Association for Computational Linguistics.

Tay, Y., Tuan, L. A., and Hui, S. C. (2017). Dyadic memory networks for aspect-based sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 107–116, New York, NY, USA. Association for Computing Machinery.

Teng, Z., Vo, D. T., and Zhang, Y. (2016). Context-sensitive lexicon features for neural sentiment analysis. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1629–1638.

Titov, I. and McDonald, R. (2008). Modeling online reviews with multi-

grain topic models. In *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, pages 111–120.

Titov, I. and Mcdonald, R. (2008). Modeling Online Reviews with Multi-grain Topic Models. Technical report.

Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welpe, I. M. (2010). Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Turney, P. D. (2002). Thumbs up or thumbs down? In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, number July, page 417.

Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Vanderplas, J. (2020). 1.6. Nearest Neighbors — scikit-learn 0.24.1 documentation.

Vashishtha, S. and Susan, S. (2019). Fuzzy rule based unsupervised sentiment analysis from social media posts. *Expert Systems with Applications*, 138.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009.

Vo, D. T. and Zhang, Y. (2015). Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2015-Janua, pages 1347–1353.

Wan, X. (2008). Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *EMNLP 2008 - 2008 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference: A Meeting of SIGDAT, a Special Interest Group of the ACL*, pages 553–561.

Wang, H., Liu, B., Li, C., Yang, Y., and Li, T. (2020). Learning with noisy labels for sentence-level sentiment classification. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 6286–6292.

Wang, J., Yu, L. C., Lai, K. R., and Zhang, X. (2016a). Dimensional sentiment analysis using a regional CNN-LSTM model. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Short Papers*, pages 225–230.

Wang, W. Y., Li, J., and He, X. (2018). Deep reinforcement learning for NLP. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference Tutorial Abstracts*, pages 19–21.

Wang, X., Jiang, W., and Luo, Z. (2016b). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, pages 2428–2437.

Wang, X., Liu, Y., Sun, C., Wang, B., and Wang, X. (2015). Predicting polarities of tweets by composing word embeddings with long short-Term memory. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation*

of Natural Language Processing, Proceedings of the Conference, volume 1, pages 1343–1353.

Wang, X., Wei, F., Liu, X., Zhou, M., and Zhang, M. (2011). Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *International Conference on Information and Knowledge Management, Proceedings*, pages 1031–1040.

Wang, Y., Huang, M., Zhao, L., and Zhu, X. (2016c). Attention-based LSTM for Aspect-level Sentiment Classification. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 606–615.

Wei, W. and Gulla, J. A. (2010). Sentiment learning on product reviews via Sentiment Ontology Tree. In *ACL 2010 - 48th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 404–413. Association for Computational Linguistics.

Weichselbraun, A., Gindl, S., and Scharl, A. (2013). Extracting and grounding contextualized sentiment lexicons. *IEEE Intelligent Systems*, 28(2):39–46.

Wen, J., Zhang, G., Zhang, H., Yin, W., and Ma, J. (2020). Speculative text mining for document-level sentiment classification. *Neurocomputing*, 412:52 – 62.

Wiebe, J., Bruce, R., Martin, M., Wilson, T., and Bell, M. (2004). Learning subjective language. *Computational Linguistics*, 30(3):277–308.

Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language.

Wiebe, J. M. (2000). Learning Subjective Adjectives from Corpora. *Proceedings of the National Conference on Artificial Intelligence*, pages 735–741.

Wiebe, J. M., Bruce, R. F., and O'Hara, T. P. (1999). Development and use of a gold-standard data set for subjectivity classifications. pages 246–253.

Wilson, T., Wiebe, J., and Hwa, R. (2004). Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of the National Conference on Artificial Intelligence*, pages 761–767.

Wilson, T., Wiebe, J., and Hwa, R. (2006). Recognizing strong and weak opinion clauses. In *Computational Intelligence*, volume 22, pages 73–99.

Wu, F. and Huang, Y. (2016). Sentiment domain adaptation with multiple sources. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, volume 1, pages 301–310.

Wu, O., Yang, T., Li, M., and Li, M. (2018). Rho-hot Lexicon Embedding-based Two-level LSTM for Sentiment Analysis.

Wu, Q., Tan, S., and Cheng, X. (2009). Graph ranking for sentiment transfer. In *ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf.*, pages 317–320.

Wu, Y. and Wen, M. (2010). Disambiguating dynamic sentiment ambiguous adjectives. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 1191–1199.

Xia, R. and Zong, C. (2011). A POS-based Ensemble Model for Cross-domain Sentiment Classification. *in Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 614–622.

Xing, B., Liao, L., Song, D., Wang, J., Zhang, F., Wang, Z., and Huang, H. (2019). Earlier attention? Aspect-aware LSTM for aspect-based sentiment analysis. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, volume 2019-Augus, pages 5313–5319.

Xu, H., Liu, B., Shu, L., and Yu, P. S. (2019). BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 2324–2335.

Xu, J., Chen, D., Qiu, X., and Huang, X. (2016). Cached long short-term memory neural networks for document-level sentiment classification. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1660–1669.

Yadav, S. and Chakraborty, T. (2020). Unsupervised sentiment analysis for codemixed data.

Yang, H., Si, L., and Callan, J. (2006). Knowledge transfer and opinion detection in the trec2006 blog track. In *NIST Special Publication*.

Yang, L., Li, Y., Wang, J., and R. Simon Sherratt (2020). Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning. *IEEE Access*.

Yang, M., Tu, W., Wang, J., Xu, F., and Chen, I. (2017). Attention-Based LSTM for Target-Dependent Sentiment Classification. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17) Attention-Based*, pages 5013–5014.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *2016*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, pages 1480–1489.

Yano, T. and Smith, N. A. (2010). What's worthy of comment? Content and comment volume in political blogs. In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 359–362.

Yao, K., Cohn, T., Vylomova, K., Duh, K., and Dyer, C. (2015). Depth-Gated LSTM.

Yessenalina, A., Yue, Y., and Cardie, C. (2010). Multi-level structured models for document-level sentiment classification. In *EMNLP 2010 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, number 11, pages 1046–1056. Association for Computational Linguistics.

Yin, Y., Song, Y., and Zhang, M. (2017). Document-level multi-aspect sentiment classification as machine comprehension. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 2044–2054.

You, Q., Luo, J., Jin, H., and Yang, J. (2016). Cross-modality consistent regression for joint visual-textual sentiment analysis of social multimedia. In *WSDM 2016 - Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 13–22.

Yu, H. and Hatzivassiloglou, V. (2003). Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136.

Yu, J. and Jiang, J. (2016). Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 236–246.

Zeng, Z., Zhou, W., Liu, X., Lin, Z., Song, Y., Kuo, M. D., and Chiu, W. H. K. (2020). A Variational Approach to Unsupervised Sentiment Analysis Ziqian.

Zhai, S. and Zhang, Z. (2016). Semisupervised autoencoder for sentiment analysis. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 1394–1400.

Zhang, H. (2004). The Optimality of Naive Bayes Naive Bayes and Augmented Naive Bayes. *Aa*, 1(2):3.

Zhang, J., Zhao, Y., Li, H., and Zong, C. (2019a). Attention with sparsity regularization for neural machine translation and summarization. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 27(3):507–518.

Zhang, L., Ghosh, R., Dekhil, M., Hsu, M., and Liu, B. (2011). Combining lexicon-based and learning-based methods for twitter sentiment analysis. *HP Laboratories Technical Report*, (89).

Zhang, L. and Liu, B. (2015). Aspect and Entity Extraction for Opinion Mining. Technical report.

Zhang, L., Wang, S., and Liu, B. (2018a). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4).

Zhang, M., Zhang, Y., and Vo, D. T. (2016). Gated neural networks for targeted sentiment analysis. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 3087–3093.

Zhang, W. and Skiena, S. (2010). Trading strategies to exploit blog and news sentiment. In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 375–378.

Zhang, Y., Ji, D.-H., Su, Y., and Wu, H. (2013). Joint naïve bayes and lda for unsupervised sentiment analysis. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G., editors, *Advances in Knowledge Discovery and Data Mining*, pages 402–413, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhang, Y., Song, D., Li, X., and Zhang, P. (2018b). Unsupervised sentiment analysis of twitter posts using density matrix representation. In Pasi, G., Piwowarski, B., Azzopardi, L., and Hanbury, A., editors, *Advances in Information Retrieval*, pages 316–329, Cham. Springer International Publishing.

Zhang, Y., Zhang, Z., Miao, D., and Wang, J. (2019b). Three-way enhanced convolutional neural networks for sentence-level sentiment classification. *Information Sciences*, 477:55–64.

Zhang, Z., Zou, Y., and Gan, C. (2018c). Textual sentiment analysis via three different attention convolutional neural networks and cross-modality consistent regression. *Neurocomputing*, 275:1407 – 1415.

Zhao, Z., Lu, H., Cai, D., He, X., and Zhuang, Y. (2017). Microblog sentiment classification via recurrent random walk network learning. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 3532–3538.

Zhou, X., Tao, X., Yong, J., and Yang, Z. (2013). Sentiment analysis on tweets for social events. *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2013*, (June):557–562.

Zhou, X., Wan, X., and Xiao, J. (2016). Attention-based LSTM network for cross-lingual sentiment classification. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 247–256.

Zhu, J., Wang, H., Tsou, B. K., and Zhu, M. (2009). Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 1799–1802, New York, NY, USA. Association for Computing Machinery.

Zhu, P. and Qian, T. (2018). Enhanced Aspect Level Sentiment Classification with Auxiliary Memory. In *COLING*, pages 1077–1087.

Zou, Y., Gui, T., Zhang, Q., and Huang, X. (2018). A Lexicon-Based Supervised Attention Model for Neural Sentiment Analysis. *Coling-2018*, pages 868–877.