Towards Emotion Intelligence in Neural Dialogue Systems

by

Chenyang Huang

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

Dialogue systems, also known as Conversational Agent (CA), are designed to mimic coherent conversations with humans. Most conversational agents are specialized for a specific domain such as travel booking and are typically finite state-based or template-based. Open domain dialogue systems have seen a growing interest in recent years thanks to neural dialogue generation systems, based on deep learning models.

These systems basically learn to predict the words and the sentence to respond based on the previous utterances. However, while such a system can generate grammatically correct and human-like answers, the responses are often generic and non-committal instead of being specific and emotionally intelligent. In this work, the objective is to tackle two main problems that are essential towards building emotionally intelligent chatbots: "How to detect the emotions expressed by the human accurately?" and "How can a chatbot express an emotion?"

We propose a Neural Network model which is dedicated for emotion recognition. It combines multiple recent advances in semantic and emotional feature representations. Our experiments show that the proposed model outperforms the current state-of-the-art models by a large margin. We then develop a hierarchical variant of the model and get outstanding result on the SemEval2019-Task3 shared task.

In order to design a more reliable/generalized emotion detection system, we also collect a dataset from scratch which is more than 10 times larger than the current largest emotion dataset that is publicly available.

In order to generate specific emotions in open-domain dialogue environment, we propose a total of seven models that are all based on a widely used neural dialogue generation framework. The results indicate that all the models are able to tackle the task equally well, and we compare the models in terms of accuracy and computation costs.

We also reflect on the problems of anticipating expressed emotions but an interlocutor and the problem of determining the appropriate emotion to express in a generated response.

Preface

There are three papers that are related to manuscript (sorted by time):

1. **Chenyang Huang**, Amine Trabelsi, Osmar R. Zaïane, "ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and BERT", the 13th International Workshop on Semantic Evaluation collocated with NAACL, Minneapolis, USA, June 2-7, 2019. (Selected as the only oral presentation for the task.) [34]

2. **Chenyang Huang**, Osmar R. Zaïane. "Generating Responses Expressing Emotion in an Open-domain Dialogue System", CONVERSATIONS: 2nd International Workshop on Chatbot Research, In conjunction with the International Conference on Internet Science , St. Petersburg, Russia, October 26, 2018 [Chapter in Springer LNCS 11551 Internet Science, Svetlana S. *et al.* Niedermayer (Eds.), pp 100-112, 2019] [33]

3. **Chenyang Huang**, Osmar R. Zaïane, Amine Trabelsi, and Nouha Dziri. "Automatic Dialogue Generation with Expressed Emotions", the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), New Orleans, USA, June 1-6, 2018. [32]

In the list above, the 1st paper is covered by Section 2.9.1. The 2nd and the 3rd paper are covered by Chapter 3. I contribute most of the ideas, experiments and writing to the three papers, however, Prof. Zaïane and Dr. Trabelsi also have very constructive inputs. Without them, none of the papers would have been published. In addition I also contribute to the paper of "Basic and depression specific emotion identification in Tweets: multi-label classification experiments" [20], in which I am responsible for the Neural Network models. These models are later becoming the baseline models in Chapter 2.

We are currently trying to publish the LDET dataset which is introduced in Section 2.7.2 in the form of a conference paper. The results in Section 2.8 and content in Chapter 4 may also be submitted as conference papers in the future.

To my parents

For supporting me unconditionally.

To study without thinking is futile.

To think without studying is dangerous.

– Confucius, Chinese philosopher.

Acknowledgements

The support and impact from my supervisor Prof. Osmar R. Zaïane of the department of computing science at the University of Alberta are beyond words. Without his constructive opinions and patience, my “MSc thesis program” in the department as long as my research career would have been terminated long ago. I very appreciate the help from Prof. Jia You, he reached out and got me through the toughest semester in the department. I want to thank Dr. Amine Trabelsi, who helps me with finishing this thesis. Last but not least, thanks to my friends Xuebin Qin, Chao Gao, and Nawshad Farruque for helping me survive the dark time.

Contents

List of Tables

List of Figures

Acronyms

ANA	Automated Nursing Agent. 2, 41
BOW	Bag-of-Words. ix, xii, 11, 12, 14
BPEmb	Byte-pair Embeddings. 15, 16
CA	Conversational Agent. ix, 1, 40–42, 73, 77, 78
CBOW	Continuous Bag-of-Words Model. 15
CE	Cross Entropy. 20
CNN	Convolutional Neural Network. 47
CVAE	Conditional Variational Autoencoders. 44
DL	Deep Learning. 1, 13, 20, 42, 44, 48, 72
EQ	Emotional Quotient. 1, 2, 41, 42, 79
ERM	Empirical Risk Minimization. 33, 48
FC	Fully Connected. 23, 47, 48
GloVe	Global Vectors for Word Representation. 15, 22
IPA	Intelligent Personal Assistant. 41
IQ	Intelligence Quotient. 1, 2, 41, 42
LBL	Log-bilinear Language Model. 15
LDET	Large Dataset of Emotional Tweets. v, xi, 26, 27, 29–31

LM Language Model. 43, 44

LSTM Long short-term memory. 17–20, 22, 46, 50, 53, 67

ML Machine Learning. 4, 8, 10, 42

MLP Multilayer Perceptrons. 16

NLG Natural Language Understanding. 44

NLP Natural Language Processing. 11, 13–16, 21, 43, 47

NN Neural Network. v, 10, 13, 14, 16, 43–45, 53, 79

NNLM Neural Net Language Model. 14

RBF Radial Basis Function. 13

RL Reinforcement Learning. 1, 73

RNN Recurrent Neural Networks. xii, 16–18, 45, 47, 49, 50

RNNLM Recurrent Neural Network Language Modeling. 15

Seq2Seq Sequence-to-sequence. x, 5, 44, 45, 48–50, 53, 60, 61, 66

SeqGAN Sequence Generative Adversarial Nets. 44

SVM Support Vector Machine. 13

Chapter 1

Introduction

1.1 Motivation

Dialogue systems, also known as Conversational Agent (CA), are designed to mimic coherent conversations with humans. They have been adopted in many downstream applications such as technical support, entertainment, and personal assistant. Early systems, such as, Eliza [95], Parry[12], and Alice[92] were able to converse with a human in term of text-based conversations using hand-crafted rules.

Dialog systems can be generally divided into open-domain systems and task-oriented systems based on the nature of the conversation. Open-domain systems usually aim at engaging users and providing mental support, on the other hand, task-oriented systems are designed to assist user to complete a particular task such as booking a ticket or answering general questions such as “what is the weather”. With the recent development of Deep Learning (DL), building open-domain social chatbots has received much more attention from the research community.

A recent technical report that describes the design of the popular Chinese social textual chatbot XiaoIce [104] states that a mature chatbot should be improved in three aspects: Intelligence Quotient (IQ), Emotional Quotient (EQ), and the persona of the agent. In the past few years, there have been attempts to keep user engaged [47] using Reinforcement Learning (RL) by making the proposed system take into consideration the dialogue context[77]. Moreover, there have also been some attempts to avoid generating dull, short responses [45, 48]. These attempts are focusing on improving the IQ of dialogue systems. However, the research for improving EQ has received less attention.

In some applications such as digital companion or medical technical support, the ability of sensing emotions and respond with “safe” emotions is without doubt important. In this research, we are studying the problem of how to improve emotion intelligence for chatbots. For example, the Automated Nursing Agent (ANA) [17, 69] is a conversational agent that is designed to help the elderly live at home. The current system is able to answer specific questions according to users’ personalized knowledge base. The current “skills” of ANA are mostly developed through rule-based algorithms and domain focused, thus lacking the ability of chitchatting. Considering that the goal of ANA is becoming both personal assistant and digital companion, ANA should also be designed as an open-domain conversation agent.

1.2 Problem Statement

The concept of emotional intelligence in chatbot systems is not well-defined at the time of writing this dissertation. In this section, we firstly give informal definitions for some of the potential problems that are under the scope of the “Emotional Intelligence in Chatbots”. The proposed problems will be later formally defined and tackled in Chapter 2, 3, 4.

In this research, we only consider text-based communication between humans and chatbots. Figure 1.1 illustrates an example of a conversation session between a human and a chatbot system. Based on the dynamics of the expressed emotions, four main questions could emerge.

1. How to recognize the emotions expressed by humans? Given a set of possible emotions and an utterance from the user, we need to build a model that is able to detect which emotion is expressed.

2. Could a chatbot express a specific emotion? Given an emotion to be expressed in response to a user utterance, a chatbot system needs to produce

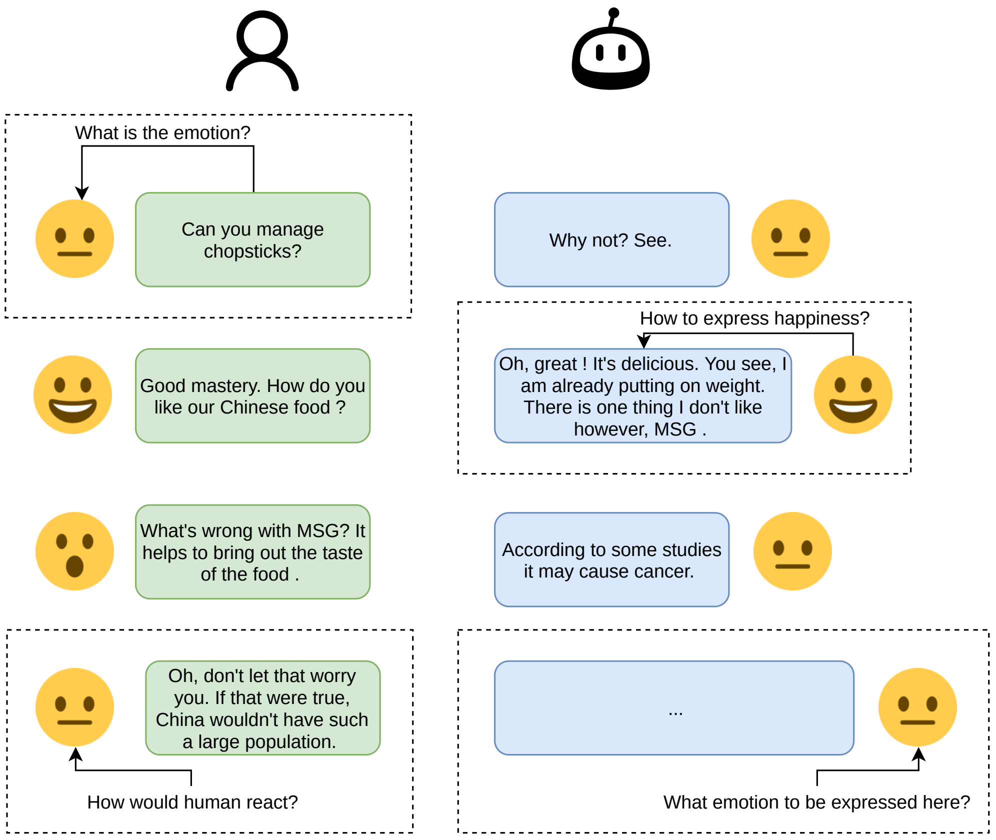

What is the emotion?
Can you manage chopsticks?
Why not? See.
How to express happiness?
Oh, great ! It's delicious. You see, I am already putting on weight. There is one thing I don't like however, MSG .
Good mastery. How do you like our Chinese food ?
What's wrong with MSG? It helps to bring out the taste of the food .
According to some studies it may cause cancer.
Oh, don't let that worry you. If that were true, China wouldn't have such a large population.
...
How would human react?
What emotion to be expressed here?

Figure 1.1: An example of dialogue exchanges with emotions

a coherent response, which is relevant with regard to the user utterance, and where the reply message should convey the specified emotion.

3. Could a chatbot anticipate the emotions which a user may express? A EQ enabled chatbot system should also have the quality of empathy. Given the dialogue history and user's personality. The chatbot should be able to anticipate the reactions of the users. In the scenario of emotions, before passing a message to user, the chatbot should have a general idea of what user might react so that it could make an adjustment to the message.

4. How a chatbot knows what emotions to express? An empathetic chatbot should be "smart" enough to know what is the best emotion to express while communicating with humans. For example, if it detects a strong signal of *sadness* from a human. Should the chatbot try to be positive to cheer the human up or express *sympathy*?

There are other questions that could potentially be raised. However, in this thesis, we focused on addressing the four proposed questions and specifically proposed solutions to the first two.

1.3 Thesis Statement

In this research, our objective is to improve the EQ of open-domain dialogue systems. The main hypotheses of this thesis is the following:

> *A dialogue agent can recognize and predict the emotion expressed in a dialogue utterance*

> *We can make a conversational agent's response express a specific emotion.*

In other words, we assume that human emotions which expressed through text are following specific patterns. Given enough data, one should be able to design a Machine Learning (ML) system to learn such patterns. This thesis is focusing on how to design such a system using state-of-art techniques.

1.4 Thesis Contributions

The major contributions of our research are as follows:

- We proposed a neural model which is dedicated for emotion detection tasks (Section 2.6), our experiments show that the proposed model outperforms the current state-of-art models by a large margin.
- In order to design a more reliable/generalized emotion classification system, we also collect a dataset from scratch which is more than 10 times larger than the current available emotion dataset (Section 2.7.2).
- We propose a total of seven models to express specific emotions in Seq2Seq based open-domain dialogue system. The experiments show that all the models are able to tackle the task.

- We make our dataset and source code public to the research community[1].

1.5 Thesis Organization

The remainder of the manuscript contains three chapters, Chapters 2, 3, and 4, each dedicated to answer one of the three questions brought up in Section 1.2. A last chapter is devoted to the perspectives and conclusions.

Chapter 2: Emotion Detection

Emotion mining from text serves as a building block component for this research. This chapter will first cover some of the widely accepted emotion models introduced by the field of psychology. We treat the problem of emotion detection as a special case of the text classification task. Therefore, we further introduce most of the popular text classification methods. Based on the existing works, we proposed a model which is dedicated for emotion detection task. In our experiments, using the existing datasets and our newly collected dataset, we demonstrate the significant improvements brought by our proposed method.

Chapter 3: Response with Emotions

In this chapter, we aim at the tackling the task of *teaching* an a neural dialogue system to express emotion in an open-domain environment. This chapter starts with one widely-applied neural dialogue generation framework – Seq2Seq [87] . Using Seq2Seq as the backbone of for the open-domain dialogue generation system, we proposed seven different models that are all able to automatically generate responses while conditioned on a particular emotion to be expressed.

Chapter 4: Future Work and Conclusion

Compared to the previous chapters which tackle the first two problems in the *Problem Statement* (Section 1.2) respectively, the last two problems are much less studied and potentially more complicated. In this chapter, we first give

[1]`https://github.com/chenyangh/DialogueGenerationWithEmotion`

the formal definitions of the last two problems and state the challenges for these. In addition, we design an experiment and show the preliminary results on how the task of *anticipating humans' emotion* can be achieved. Last but not least, we conclude the entire manuscript at the end this chapter.

Chapter 2

Emotion Detection

2.1 Introduction

Emotion mining from text is the task of recognizing human emotions. In [78], the concept of textual emotion mining is further divided into three fine-grained tasks: emotion detection, emotion polarity classification, and emotion classification. Emotion polarity classification is the task that tells whether a document is "positive" or "negative" in terms of sentiments and is thus often referred to as *Sentiment Analysis*. On the other hand, given an emotion category (discussed in Section 2.2), emotion detection consists in classifying documents based on whether any of the specified emotions in a category exist or not. Emotion classification involves telling what exact emotions are contained in the document. In this manuscript, we do not discriminate between the terms "detection" and "classification" in the domain of emotion mining from text, as the former one can be viewed as a sub-task of the latter one by simply adding "No emotion" into the emotion category. Therefore, we only use the term "emotion detection" for simplicity.

Emotion detection serves as the basis across the whole research in this manuscript. There are three aspects of research that are related to emotion mining from text. Firstly, what are "emotions"? We need to understand the "emotions" in terms of computational objects rather than psychology concepts. The second aspect is how to gather the "emotions" from text. Last but not least, how can we recognize the emotions from an unseen text? Essentially, the second part provides labeled datasets for the third part to treat the whole problem as a supervised Machine Learning (ML).

To illustrate this, we will first introduce some of the widely accepted emotion models. In Section 2.3, we introduce most of the popular algorithms for text classification. In addition, we propose a model that is designed for emotion detection. In order to show the performance of the proposed model, we test it not only on the existing datasets, but also on a newly collected dataset.

2.2 Emotion Models

In the area of psychology research, emotions can be identified and grouped based on their *types* or *intensity* [73]. Existing emotion models can be divided into two classes: categorical and dimensional [8].

In categorical models, emotions are viewed as fixed atomic units, often called **basic emotions**. The most well-known categorical model might be Ekman's 6 basic emotions [18]. Plutchik [67] presented a wheel of emotions, which consists of four pairs of opposing pairs as shown in Figure 2.1.

Emotion can play a role in medical informatics tasks like detecting depression or suicidal intent. Detecting emotions expressed toward characters in novels might play a role in understanding how different social groups were viewed by society at different times.

There are two widely-held families of theories of emotion. In one family, emotions are viewed as fixed atomic units, limited in number, and from which others are generated, often called **basic emotions** (Tomkins 1962, Plutchik 1962). Perhaps most well-known of this family of theories are the 6 emotions proposed by (Ekman, 1999) as a set of emotions that is likely to be universally present in all cultures: *surprise, happiness, anger, fear, disgust, sadness*. Another atomic theory is the (Plutchik, 1980) wheel of emotion, consisting of 8 basic emotions in four opposing pairs: *joy–sadness*, *anger–fear*, *trust–disgust*, and *anticipation–surprise*, together with the emotions derived from them, shown in Fig. 20.11.

basic emotions

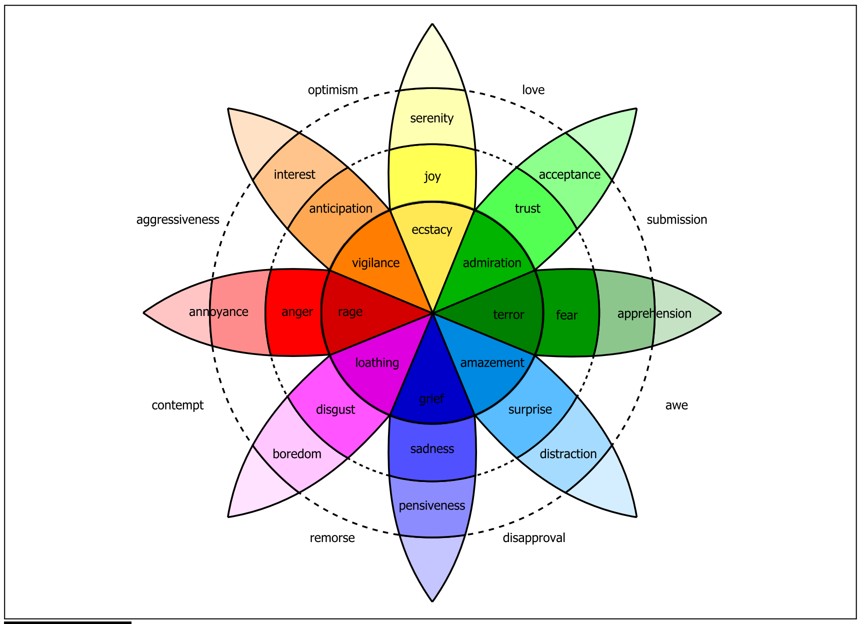


Figure 20.11 Plutchik wheel of emotion.

Figure 2.1: Plutchik's wheel of emotion [37]

The second class of emotion theories views emotion as a space in 2 or 3 dimensions (Russell, 1980). Most models include the two dimensions **valence** and **arousal**, and many add a third, **dominance**. These can be defined as:

valence: the pleasantness of the stimulus
arousal: the intensity of emotion provoked by the stimulus
dominance: the degree of control exerted by the stimulus

Another class of emotion models represent emotions into a space of two or three dimensions [72]. Most models include the dimensions **valence** and **arousal**. Many others add a third one, **dominance** [37].

Practical lexicons have been built for both kinds of theories of emotion.

20.5.1 Lexicons for emotion and other affective states

While semi-supervised algorithms are the norm in sentiment and polarity, the most common way to build emotional lexicons is to have humans label the words. This

In this research, we put attention on the models that contains basic emotions. The reasons for this are twofold: it is easy to collect data with basic emotions rather than dimensional annotations; the categorical emotions are easier to represent.

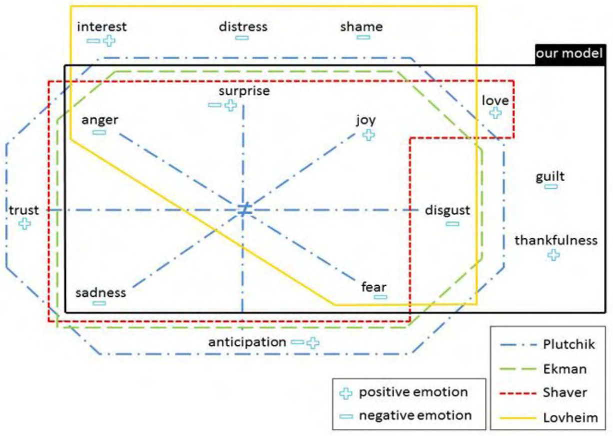


Figure 2.2: Comparison of most of the existing categorical emotion models, in which "our model" refers to the model proposed by Yadollahi, Shahraki, and Zaïane [96]

As mentioned above, P. Ekman, one of the earliest emotion theorists, suggested 6 basic emotions in 1972 [18] which are *anger, disgust, fear, joy, sadness* and *surprise.* The following work by [80], [41] suggest removing *disgust* and adding *love.* A. G. Shahraki *et al.* [79, 96] combine the aforementioned emotion models by involving two additional emotions (*guilt* and *thankfulness*) but keeping *disgust.* Figure 2.2 shows the comparison, in which "our model" represent Shahraki's 9 emotion model. The 9 emotions are shown in Figure 2.3.

Figure 2.3: Shahraki's 9 emotion model

2.3 Text Classification

Text Classification is a supervised Machine Learning task. It takes as input (X, Y) pairs, where X is a document to be classified and Y is the corresponding label/labels for X. Denote $C = \{c_1, c_2, \cdots\}$ as the set of the categories of the classification target. In practice, Y is often represented as a binary-value vector based on the occurrence of the labels. Define $Y \triangleq [y_1, y_2, \cdots, y_{|C|}]$, $y_i \triangleq \mathbb{1}(c_i \in Y)$, $|C|$ is number of the categories. For general multi-class (single label) classification problem, $\sum Y = 1$, while in multi-label classification setup, y can have arbitrary number of "1"s in its value ($\sum Y \geq 1$). Document X can be described as sequence of words/tokens: $X = [x_1, x_2, \cdots, x_n]$, where x_i is the ith word/token and n is the length of the document. In this literature, we are focusing on single-label multi-class text classification, therefore we are assuming that there exist one and only one emotion for each document X. A *classifier* is a function that can estimate the probabilities of each class given any X. Denote such a function as f, $\hat{Y} = f(X)$. The function is usually learned through a particular distance measurement $\mathcal{L}(Y, \hat{Y})$. In the following sections, we first introduce two categories of classifiers based on whether it is a Neural Network (NN) based model or not.

NN text classifiers have shown their effectiveness in many literature studies [14, 39, 50, 97, 102], however we will still cover some of the former popular text classifiers for the purpose of showing the superior performance of NN models in emotion mining from text.

2.4 None-neural Text Classification

In this section, we choose three popular algorithms that are able to do classification over text. A text classifier usually does not take as input words/tokens directly, therefore before jumping into the details of the algorithms, we need to first introduce the methods for representing words/tokens.

2.4.1 Bag-

Bag-of-Words (
guage Processir
would be able t
representation i
6 words/tokens
and ".". A *bag*
an arbitrary orc

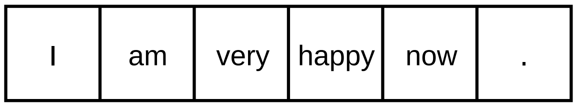


Figure 2.4: An example of BOW feature that consists of 6 words/tokens

Since tokens, e.g. numbers, punctuation, can be regraded as special words, we will use words to represent both words and tokens for brevity.

For a sentence of arbitrary length n, it will be represented in the space of $\mathbb{R}$ by counting the occurrence of the words. E.g., the sentence "I am very very happy." will be represented as "112101" according to the above "bag" (Figure 2.4) if the indices start from 1. The sentence "Hello" will be represented as "000000" since "hello" in not in the "bag". Capitalization is usually not considered, all words can be converted into lower case only.

BOW representation is sparse, the space depends on the amount of words $|V|$ which has to be decided initially. BOW does not consider the order of word, thus "I am" and "Am I" will have the same vector representation. Common words, such as 'a', 'the' are occurring frequently in documents and might not contain important significance in semantic meaning. Those words are referred to as *stop words* and often removed before converting a document into BOW representation.

2.4.2 Naïve Bayes

Naïve Bayes is a simple, yet effective machine learning algorithm that can be applied on supervised learning tasks. Following the annotations, for any given document X, it learns the probability of $\mathcal{P}(Y|X)$.

The Equation 2.1 describes the Bayesian Rule.

$$\mathcal{P}(Y|x_1, \ldots, x_n) = \frac{\mathcal{P}(Y)\mathcal{P}\left(x_1, \ldots x_n|Y\right)}{\mathcal{P}\left(x_1, \ldots, x_n\right)} \tag{2.1}$$

The term "naïve" implies the assumption of independence between every pair of features given the value of the class variable, which can be formalized as following:

$$\mathcal{P}\left(x_i|Y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\right) = \mathcal{P}\left(x_i|Y\right). \tag{2.2}$$

By applying chain rule, and the assumption in Equation 2.2, Equation 2.1 can be derived into the following:

$$\mathcal{P}(Y|x_1, \ldots, x_n) = \mathcal{P}\left(Y\right)\prod_{i=1}^{n}\mathcal{P}\left(x_i|Y\right). \tag{2.3}$$

Using Naïve Bayes for text classification has been proven to be effective even compared to the early stage of deep learning models [39].

2.4.3 Random Forest

Random forests algorithm is an ensemble learning method that can be used in multi-label classification tasks. Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [6]. The details of the algorithm is not discussed in this thesis because it is not relative to the main topic of this research.

A document X, by using BOW representation, can be represented by a vector $\mathbb{R}^{|V|}$. We denote this vector as $\text{BOW}(X)$. Each document X also has a label in space $\mathbb{Z}_2^{|C|} = 0, 1$, where $|C|$ is the number of labels. Random forest can be directly used for multi-label text classification. A random forest classifier can learn the mapping from BOW to $\mathbb{Z}_2^{|C|}$ directly.

2.4.4 SVM

Support Vector Machine (SVM) is a well-known single-label/multi-class classification algorithm. Given two vectors $\mathbf{x}$ and $\mathbf{y}$ of the same size, a function k is called kernel if it satisfies $k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$, where φ is another function. Kernel plays an import role in SVM algorithm. Linear kernel and Radial Basis Function (RBF) kernel are commonly used. Linear kernel is defined as,

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathsf{T}}\mathbf{y} + c)^d \tag{2.4}$$

Where c is a constant. RBF kernel is defined as,

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{2.5}$$

Where σ is a free parameter.

Linear kernel has been shown to be a degenerate version of RBF kernel [38].

Since SVM is usually used for single single-label classification, to adopt it in multi-label classification task, a trick named *one-vs-all* has to be used. For $|C|$ labels, $|C|$ SVM classifiers are needed in the fashion of *one-vs-all*. Each SVM is trained to fit a single label by assuming all other labels are negative. SVM_i is trained with the label $\text{MASK}_i(\mathbb{Z}_2^{|C|})$, where MASK is an algorithm that simply mask the labels that is not at the ith position as negative. E.g, for label “11001” will be converted to “01000” by MASK_2.

2.5 Neural Models for Text Classification

Deep Learning (DL) models have shown remarkable power in computer vision [28, 42] and speech recognition [27] in recent years. DL is usually referred to as hierarchical learning which consists of multiple layers of Neural Network (NN). In NLP, NN models are usually shallow compared to that in other areas, thus we prefer using the term *neural network* rather than *deep learning*. In text classification, there have been many research that has been proven to be very successful [39, 43, 100]. According to [53, 88], NN models have the following advantages over the traditional models:

- NN models require less formal statistical training to develop,
- NN models can implicitly detect complex nonlinear relationships between independent and dependent variables,
- NN models have the ability to detect all possible interactions between predictor variables,
- NN models do not impose any restrictions on the input variables.

These advantages make NN models perform very well on unseen data, i.e. they generalize well.

2.5.1 Word Embedding Feature

Similar to Section 2.4, before introducing the NN classification models, we need to explain its feature representation. As explained, BOW features are one-hot sparse vectors, which simply represents the word count. Therefore, BOW features do not capture any semantic meanings. For example, *good* and *nice* have similarity in terms of meaning but their BOW features only depend on their indices when creating the dictionary of vocabularies. The difference between *good* and *nice* is the same as that between *good* and any other words in the dictionary. Another disadvantage of using BOW features is that the features are sparse. Typically, a BOW feature is of dimension $\mathbb{R}^{|V|}$, where $|V|$ is the number of words that a model is taking into consideration. A BOW model of dimension 5000 has only 5000 words. On the other hand, unsupervisedly learned word embeddings representaions are dense and fixed-size regardless of the amount of tokens. Such representations have seen tremendous success in numerous NLP tasks in recent years [25, 55, 57, 65]. We will briefly introduce the word embedding features in the following part of this section.

Before Word2Vec In 2003, Bengio et al. firstly proposed the Neural Net Language Model (NNLM) [4] for language modeling and found that the feature extracted from the first intermediate layer is able to capture semantic similarity among words. Afterwards, there are plenty of related works that aim at improving NNLM, e.g. Log-bilinear Language Model (LBL) [59], Collobert and Weston [13] (C&W), and Recurrent Neural Network Language Modeling (RNNLM) [54]. There are two major problems with these models: 1, the captured feature can not capture long term dependency; 2, the training takes too long.

The Word2Vec In 2013, Mikolov, *et al.* proposed Continuous Bag-of-Words Model (CBOW) and Skip-gram model [57] which are able to produce dense vector representations for words efficiently. They further improved the work by subsampling the frequent words and negative sampling [56]. They also publish their work through an application named Word2Vec [1]. Although Word2Vec is only the name of the released tool that contains the pre-trained vectors, it is widely used to refer to the models themselves. In short, CBOW algorithm can be described as *predicting the word given its context* while skip-gram is *predicting the context given a word*. By learning from large corpora, features generated by these algorithms are able to capture the precise syntactic and semantic relationships among words. Furthermore, the generated word embedding features are dense.

After Word2Vec After the great success of Word2Vec, using pre-trained word vectors became the standard for many downstream NLP tasks. But there are some other following works that perform well. Two other widely used pre-trained word embeddings are Global Vectors for Word Representation (GloVe) [65] and fastText[36]. The core of GloVe is a modified loss function that is able to put more concentration on the co-occurrence of words compared to what is been done in Word2Vec. On the other hand, perhaps because one of the authors of fastText is Mikolov again after he joined Facebook, fastText can be considered as an extension of Word2Vec. Instead of words, fastText uses sub-words (charecter level n-grams) to train a CBOW model. Byte-pair Embeddings (BPEmb) [29] is a fully sub-words embedding which is developped upon BPEmb. The difference of usign sub-words between fastText and BPEmb

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

is that fastText has 1 million tokens where BPEmb has much less (options from 1,000 to 200,000). It is possible to load the full BPEmb model for fully vocabulary coverage, but we can not do so on fastText directly.

2.5.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN), a variant of NN, is a popular set of models that have shown great promise in many NLP tasks. The equation of RNN is shown as 2.6. The basic RNN is also often referred to as vanilla Recurrent Neural Networks (RNN).

$$h_t = f\left(Ux_t + Wh_{t-1}\right) \tag{2.6}$$

In the equation, x_t is the input at time step t, h_t is interpreted as hidden state. f is usually a nonlinear activation function such as hyperbolic tangent (Tanh) or ReLu [62]. Where U and W are trainable matrices. Siegelmann and Sontag [82] show that there exist finite RNNs that are Turing complete, and can therefore implement any algorithm. Unlike normal Multilayer Perceptrons (MLP), implementing RNN usually requires loops. Figure 2.5 shows the structure of RNN with a feedback loop to itself, Figure 2.6 is an unfold visualization of RNN.

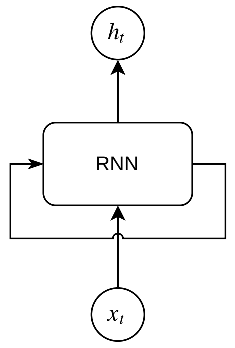


Figure 2.5: Vanilla Recurrent Neural Network

Despite the theoretical capability for RNN models, Pascanu, Mikolov, and Bengio [63] show that when training vanilla RNN, the gradient of the loss function decays exponentially with time (vanishing gradient problem). Therefore,

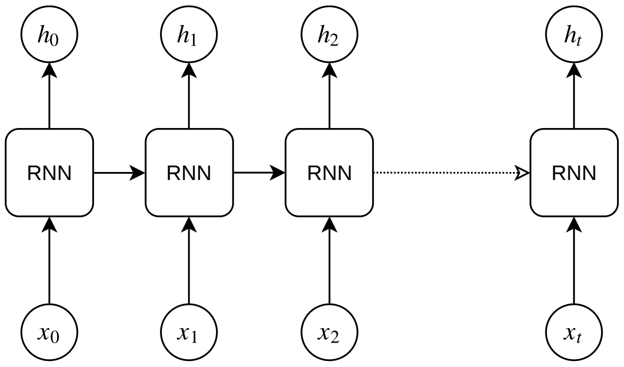


Figure 2.6: Unfold of Vanilla RNN

training a vanilla RNN that is capable of capturing long-term dependency is hard.

2.5.3 Long Short-term Memory

Hochreiter and Schmidhuber firstly proposed Long short-term memory (LSTM) in 1997 [30]. LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it is output, and when it is forgotten. This architecture lets them learn longer-term dependencies.

$$i_t = \sigma(x_t U^i + h_{t-1} W^i + b_i) \tag{2.7}$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f + b_f) \tag{2.8}$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o + b_o) \tag{2.9}$$

$$l_t = \tanh(x_t U^l + h_{t-1} W^l + b_l) \tag{2.10}$$

$$c_t = f_t * c_{t-1} + i_t * l_t \tag{2.11}$$

$$h_t = o_t * \tanh(c_t) \tag{2.12}$$

LSTM has three gates: input gate i_t, forget gate f_t and output gate o_t. All gates are generated by a *sigmoid* function over the ensemble of input x_t and the preceding hidden state h_{t-1}. In order to generate the hidden state at current step t, it first generates a temporary result l_t by a *tanh* non-linearity over the ensemble of input x_t and the preceding hidden state h_{t-1}, then combines

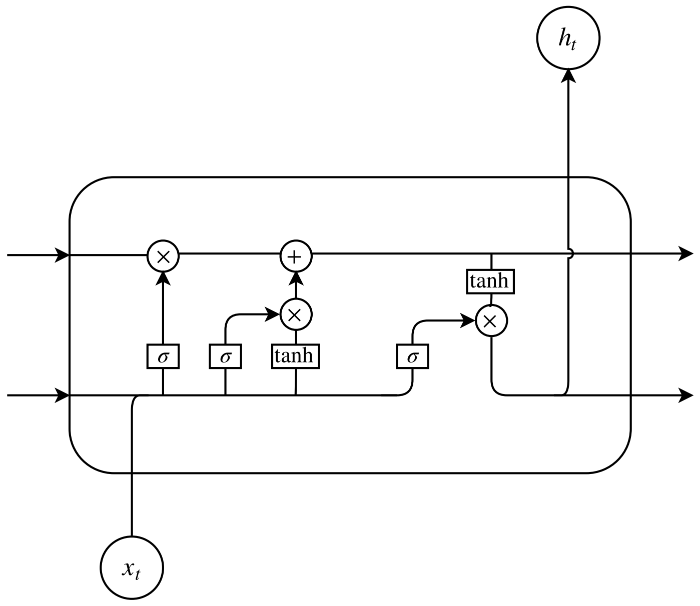


Figure 2.7: A graph explanation of LSTM block

Gated Recurrent Unit (GRU), introduced by Chung et al. in 2014 [10], is also showing promising empirical performance, but it is not outperforming LSTM. For research purpose, we only use LSTM as the choice of RNN block in this thesis.

2.5.4 Self-attention

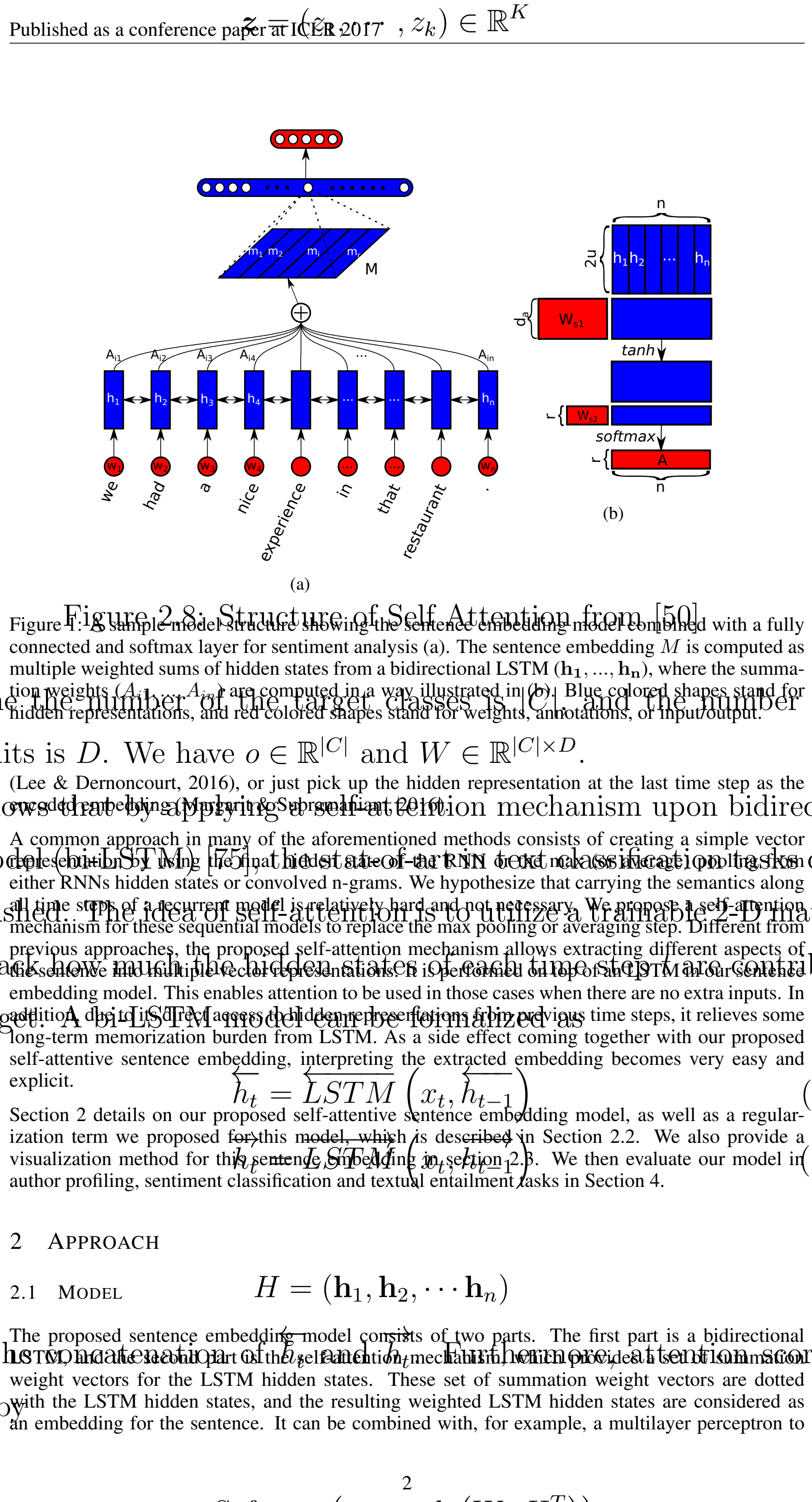
When applying RNN for classification problems, the common practice is using the last hidden state h_t and project it to a target space by a linear layer and apply *Softmax* regularization to represent the likelihood of each class. The resulting distribution can be denoted as the following equation,

$$\hat{Y} = \text{Softmax}(Wh_t + b). \tag{2.13}$$

The **Softmax** function is a mapping of $\mathbb{R}^K \to \mathbb{R}^K$ which is defined as following:

$$\text{Softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad \text{for } i = 1, \cdots, K \tag{2.14}$$

$$\boldsymbol{z} = (z_1, \cdots, z_k) \in \mathbb{R}^K \tag{2.15}$$



Published as a conference paper at ICLR 2017

M
m_1 m_2 m_i m_r
A_{i1} A_{i2} A_{i3} A_{i4} ... A_{in}
h_1 h_2 h_3 h_4 ... ... h_n
w_1 w_2 w_3 w_4 ... ... w_n
we had a nice experience in that restaurant .
(a)
n
2u
h_1 h_2 ... h_n
d_a
W_{s1}
tanh
r
W_{s2}
softmax
A
(b)

Figure 1: A sample model structure showing the sentence embedding model combined with a fully connected and softmax layer for sentiment analysis (a). The sentence embedding M is computed as multiple weighted sums of hidden states from a bidirectional LSTM ($\mathbf{h_1}, ..., \mathbf{h_n}$), where the summation weights ($A_{i1}, ..., A_{in}$) are computed in a way illustrated in (b). Blue colored shapes stand for hidden representations, and red colored shapes stand for weights, annotations, or input/output.

(Lee & Dernoncourt, 2016), or just pick up the hidden representation at the last time step as the encoded embedding (Margarit & Subramaniam, 2016).

A common approach in many of the aforementioned methods consists of creating a simple vector representation by using the final hidden state of the RNN or the max (or average) pooling from either RNNs hidden states or convolved n-grams. We hypothesize that carrying the semantics along all time steps of a recurrent model is relatively hard and not necessary. We propose a self-attention mechanism for these sequential models to replace the max pooling or averaging step. Different from previous approaches, the proposed self-attention mechanism allows extracting different aspects of the sentence into multiple vector representations. It is performed on top of an LSTM in our sentence embedding model. This enables attention to be used in those cases when there are no extra inputs. In addition, due to its direct access to hidden representations from previous time steps, it relieves some long-term memorization burden from LSTM. As a side effect coming together with our proposed self-attentive sentence embedding, interpreting the extracted embedding becomes very easy and explicit.

Section 2 details on our proposed self-attentive sentence embedding model, as well as a regularization term we proposed for this model, which is described in Section 2.2. We also provide a visualization method for this sentence embedding in section 2.3. We then evaluate our model in author profiling, sentiment classification and textual entailment tasks in Section 4.

2 APPROACH

2.1 MODEL

The proposed sentence embedding model consists of two parts. The first part is a bidirectional LSTM, and the second part is the self-attention mechanism, which provides a set of summation weight vectors for the LSTM hidden states. These set of summation weight vectors are dotted with the LSTM hidden states, and the resulting weighted LSTM hidden states are considered as an embedding for the sentence. It can be combined with, for example, a multilayer perceptron to

2

Figure 2.8: Structure of Self Attention from [50]

Assume the number of the target classes is $|C|$, and the number of the hidden units is D. We have $o \in \mathbb{R}^{|C|}$ and $W \in \mathbb{R}^{|C| \times D}$.

[50] shows that by applying a self-attention mechanism upon bidirectional LSTM model (bi-LSTM) [75], the state-of-art in text classification tasks can be further pushed. The idea of self-attention is to utilize a trainable 2-D matrix to keep in track how much the hidden states of each time step t are contributing to the target. A bi-LSTM model can be formalized as

$$\overleftarrow{h_t} = \overleftarrow{LSTM}\left(x_t, \overleftarrow{h_{t-1}}\right) \tag{2.16a}$$

$$\overrightarrow{h_t} = \overrightarrow{LSTM}\left(x_t, \overrightarrow{h_{t-1}}\right) \tag{2.16b}$$

$$H = (\mathbf{h}_1, \mathbf{h}_2, \cdots \mathbf{h}_n) \tag{2.17}$$

$\mathbf{h}_t$ is the concatenation of $\overleftarrow{h_t}$ and $\overrightarrow{h_t}$. Furthermore, attention score $\boldsymbol{\alpha}$ is obtained by

$$\boldsymbol{\alpha} = \text{Softmax}\left(w_{s2} \tanh\left(W_{s1} H^T\right)\right). \tag{2.18}$$

W_{s1} is a weight matrix with a shape of d_a-by-$2u$, where u is the dimension of the LSTM of each direction. and w_{s2} is a vector of size d_a, where d_a is a hyper-parameter that represents the dimension of the attention layer.

The final output of a bi-LSTM model with self-attentive module attached to its end is

$$\sum_{i=1}^{n} \boldsymbol{\alpha}_i H_i. \tag{2.19}$$

Therefore, for the classification problem, the probability distribution of the estimated labels is generated by the following:

$$\hat{Y} = \text{Softmax}(W \sum_{i=1}^{n} \boldsymbol{\alpha}_i H_i + b). \tag{2.20}$$

Another use of the self-attention module is that we can use the attention scores to visualize how the trained model is interpreting the documents. More specifically, it is able to show how much each token in a document is contributing to the positive label.

2.5.5 Training Target

Equation 2.20 gives an estimated probability distribution over the classification categories $\hat{Y}$, the final discrete prediction can be given by

$$\arg\max_i \hat{Y}(i), \tag{2.21}$$

where $\hat{Y}(i) \triangleq \hat{y}_i$ and $\hat{Y} = [\hat{y}_1, \cdots, \hat{y}_n]$.

The distance between the ground truth Y and the estimated distribution $\hat{Y}$ is used for training and often referred to as *loss*. In the field of Deep Learning, Cross Entropy (CE) loss is widely used and it is defined as following,

$$H(Y, \hat{Y}) = -\sum_{i=1}^{|C|} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right]. \tag{2.22}$$

2.5.6 Transfer Learning

Transfer learning refers to the process/method of reusing a model which is trained on another task on the target task. As mentioned in Section 2.5.1, using pre-trained word vectors for NLP tasks have become a standard and there is no doubt of its outstanding performance. Using pre-trained word vectors is a classical process of transfer learning as the vectors are trained on the task of language modeling. In this research, we find two other recent models that could be helpful in the task of emotion detection.

ELMo

Embeddings from Language Models (ELMo) is firstly proposed by Peters et al. in early 2018 [66]. Similar to Word2Vec or GloVe, it uses unsupervised data to train a language model which is then applied on downstream NLP applications. It uses a stacked bidirectional LSTM model as backbone and is essentially a character-level language model which dramatically reduces the amount of unseen words. Unlike traditional word vector models, to use ELMo, the pre-trained model has to first take as input the document in order to extract the corresponding hidden states which are treated as the vector representations of the words. This means that, the same word in different context may have different vector representations. Hence, ELMo representations are usually generated on the fly.

DeepMoji

Pre-trained word vectors are normally obtained in the process of training general purpose language models. DeepMoji [21], on the other hand, is trained specifically for detecting sentiment, emotion and sarcasm. In short, DeepMoji uses the emojis in tweets as self-annotated labels, and train an LSTM based model to predict the emojis given tweets. After having a proper model (good accuracy and not overfitting), the hidden states of the LSTM will be extracted given any text. Compared to ELMo, DeepMoji is a model trained through supervised learning. DeepMoji also has a relatively simpler structure than ELMo.

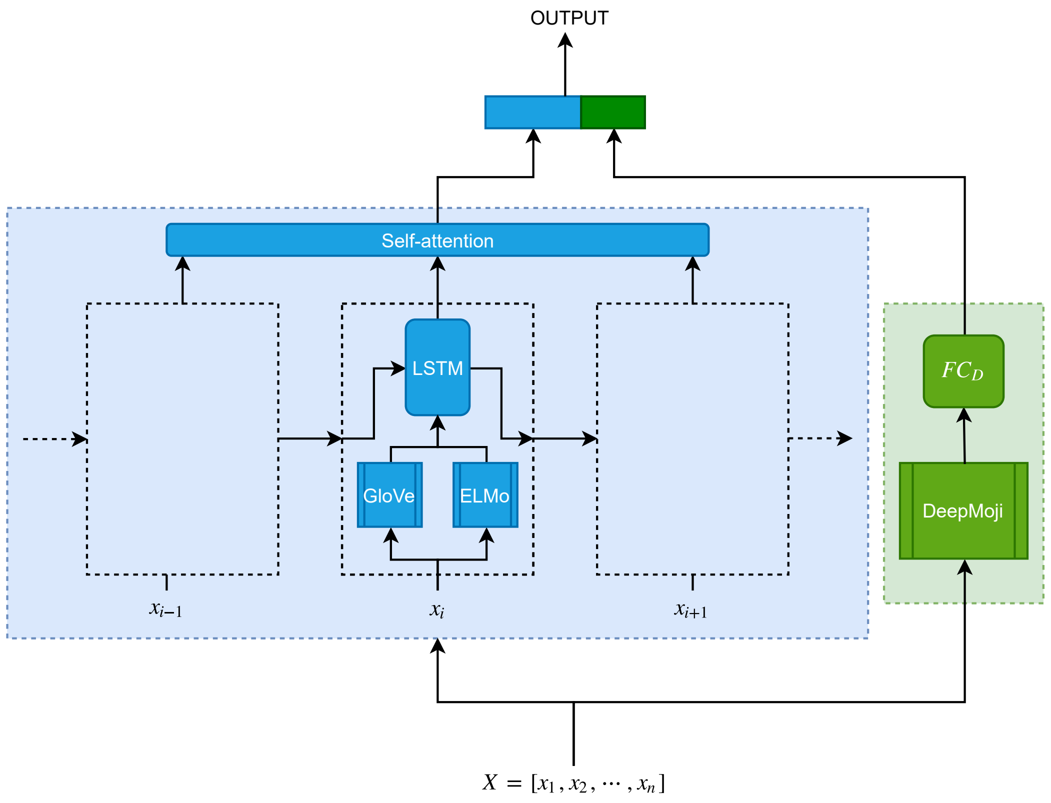


Figure 2.9: The proposed model for emotion detection

2.6 Proposed Model

Based on the specific task and recent advances (Section 2.5.4, Section 2.5.6), we propose a model that is dedicated for emotion detection. The structure of the proposed model is demonstrated in Figure 2.9.

As indicated by the different colours, the model can be seen as a composition of two parts. The left (blue) part of the model is an LSTM based model which aims at capturing the semantic meaning of the input. The right (green) part is mainly a fully connected layer which transfers the emotional information that is captured by DeepMoji to the proposed model.

In the LSTM part of the model, GloVe and ELMo together are regarded for word representations. Following the annotations, we have a sequence of words as input: $X = [x_1, x_2, \cdots x_n]$. Denote a GloVe model which takes as input a word x_i as GloVe(x_i). The result will be a vector in the space of $\mathbb{R}^{D_g}$. Where D_g is the dimension of the pre-trained GloVe model. On the other hand, a pre-trained ELMo model needs to take as input the whole sequence X and resulting n vectors. Denote the list of vectors as $\mathbf{h}^{ELMo}$, so that $\mathbf{h}^{ELMo} = [\mathrm{h}_1^{ELMo}, \mathrm{h}_2^{ELMo}, \cdots, \mathrm{h}_n^{ELMo}] = \mathrm{ELMo}(X)$. We use bi-direction LSTM the variant because it is generally used with the self-attention module [50]. Therefore, the Equation 2.16a and Equation 2.16b are rewritten as following:

$$\overleftarrow{h_t} = \overleftarrow{LSTM}\left([\mathrm{GloVe}(x_t); \mathrm{h}_t^{ELMo}], \overleftarrow{h_{t-1}}\right) \tag{2.23a}$$

$$\overrightarrow{h_t} = \overrightarrow{LSTM}\left([\mathrm{GloVe}(x_t); \mathrm{h}_t^{ELMo}], \overrightarrow{h_{t-1}}\right) \tag{2.23b}$$

The bi-directional LSTM is followed by a self-attention layer which is described by Equation 2.19. Denote Equation 2.19 as $\mathrm{LSTM}_{\mathrm{out}}$. The $\mathrm{LSTM}_{\mathrm{out}}$ aims at capturing the general semantic meaning of the input sequence X.

In this research, we are particularly interested in the domain of emotion. It is intuitive to include a module which is designed to capture the emotional representation of sentences. In this research, we use the aforementioned DeepMoji pre-trained model to explicitly add the emotion representation into our proposed model. Denote the pre-trained model as D, the output $D(X)$ is the emotion representation of the input sentence X. The $D(X)$ outputs vectors in $\mathbb{R}^{2304}$ which is quite high dimensional vectors considering the $\mathrm{LSTM}_{\mathrm{out}}$ is only in the space of $\mathbb{R}^{800}$ (800 is a manually-set hyper-parameter). We add another Fully Connected (FC) layer FC_D to map the emotion representation into a lower dimensional space (our choice is $\mathbb{R}^{300}$). Note that the parameters in the DeepMoji model are set to be fixed during the training whereas the ones for the FC layer is set to trainable. The output of the FC layer can be regarded as the final emotional representation, we denote it as $\mathrm{DeepMoji}_{out}$ ($\mathrm{DeepMoji}_{out} = FC_D(D(X))$). We concatenate $\mathrm{LSTM}_{\mathrm{out}}$ and $\mathrm{DeepMoji}_{out}$ to form a hybrid representation of input X which captures both the semantic and emotional information. We finally add another FC layer FC_{out} to project this representation into the space of the emotion categories.

To sum up, it is a joint model of DeepMoji and LSTM model with self-attention. On a lower level, the word representation for LSTM is a combination of GloVe and ELMo. The DeepMoji model serves as a feature extractor which output is further fed into a fully connected layer.

2.7 Datasets

In this section, we introduce the existing datasets and a newly collected dataset for emotion classification.

2.7.1 Existing datasets

Human annotation is one straightforward approach to obtain labeled datasets but it is costly. Only very small amount of manually labeled categorical dataset is available at the time of writing this paper. As an alternative, distant supervision [58] has been investigated in many emotion detection research [60, 94] and proven to be efficient. Generally, they harvest tweets with emotion-carrying hashtags which are used as a surrogate of emotion labels. Table 2.1 shows the hashtages used by Shahraki and Zaïane [79].

Emotion	**List of Hashtags**
anger	#anger, #angry, #rage
fear	#fear
joy	#happy
love	#love
sadness	#sad
surprise	#surprise
thankfulness	#thankful
disgust	#disgust, #disgusted, #disgusting
guilt	#guilty, #sorry

Table 2.1: Hashtags used to extract tweets by [79]

There are categorical datasets for emotion classification in text. For example, *Cleaned Balanced Emotional Tweets (CBET)* dataset [79, 96], *Twitter Emotion Corpus (TEC)* [60] and the *International Survey on Emotion Antecedents and Reactions (ISEAR)* [74]. There are some other dimensional datasets but they can not directly fit into this task. Table 2.2 shows more details about the aforementioned datasets.

Among the three datasets, ISEAR and CBET are balanced in terms of labels, where as TEC is not. Figure 2.10 shows their label distributions. In

Dataset	# of categories	# of instances	Emotions
ISEAR	7	7,666	joy, fear, anger, sadness, disgust, shame, guilt
TEC	6	21,051	anger, disgust, fear, joy, sadness, surprise
CBET	9	81,163	anger, surprise, joy, love, sadness, fear, disgust, guilt , thankfulness
LDET	6	861,662	joy, fear, anger, sadness, disgust, surprise

Table 2.2: Statistics of the four datasets: ISEAR, TEDC, CBET, and LDET (will be introduced in Section 2.7.2)

TEC, the most frequent emotion *joy* is almost 10 times as much as the least occurring emotion *disgust*. ISEAR and TEC are single labeled, but CBET contains instances that have more than one emotion. Since in this research we only consider the problem of single label text classification, the multi-labeled instances are ignored. In the end, we used 76,860 instances from CBET.

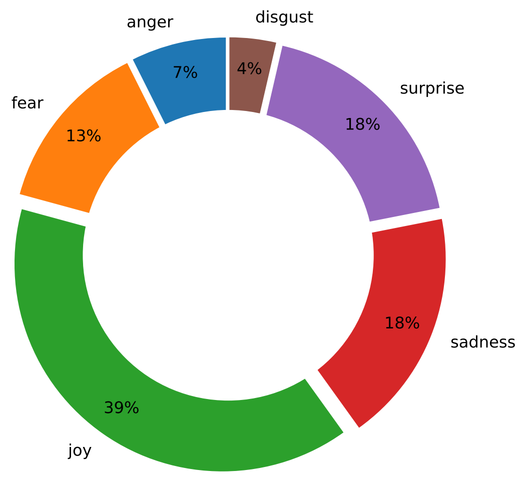


Figure 2.10: Label distribution of TEC dataset

2.7.2 Newly Collected Emotion Detection Dataset

The current public emotion classification datasets that are mentioned in Section 2.7.1 are small. Therefore, we follow the work in [1] to collect a larger dataset from scratch. We choose Ekman's 6 basic emotions as the emotion categories [18]. We first acquire 4 billion raw tweets from arxive.org [2]. After filtering out the tweets that are not English or are reposts. We use 45 hashtags as keywords to find related tweets. The chosen hashtags are synonyms of the 6 emotions which are listed in Table 2.3.

Emotion	**List of Hashtags**
anger	#anger, #angry, #rage, #pain
fear	#fear, #anxiety, #horror, #horrific
joy	#happy, #joy, #like, #happiness, #smile, #peace, #pleased, #satisfied, #satisfying
sadness	#sad, #sadness, #depression, #depressed, #alone
surprise	#surprise, #amazing, #awesome, #fascinate, #fascinating, #incredible, #marvelous, #prodigious, #shocking, #stunning, #surprising, #unbelievable
disgust	#disgust, #disgusted, #disgusting, #antipathy, #distaste, #hatred, #loathing

Table 2.3: Hashtags used to extract tweets for our larger dataset

After cleaning up duplicates and removing the tweets which are shorter than 3 words, we have 861,662 tweets. We refer to this dataset as Large Dataset of Emotional Tweets (LDET) in the following sections. The statistics of the LDET is listed in Table 2.2, from which we can see it is more than 10 times larger than the CBET dataset. Figure 2.11 shows the label distribution of the dataset. In this manuscript, we will refer to this newly collected dataset as LDET for simplicity. Same as the TEC dataset, emotion *joy* is still the most frequent emotion whereas *disgust* and *anger* are the least occurred emotions. The similarity between TEC and LDET is expected because both of them are collected from raw tweets using hashtags.

2.8 Experiments and Results

In this section, we first introduce our methodologies and then demonstrate the results.

[2]https://archive.org/details/twitterstream

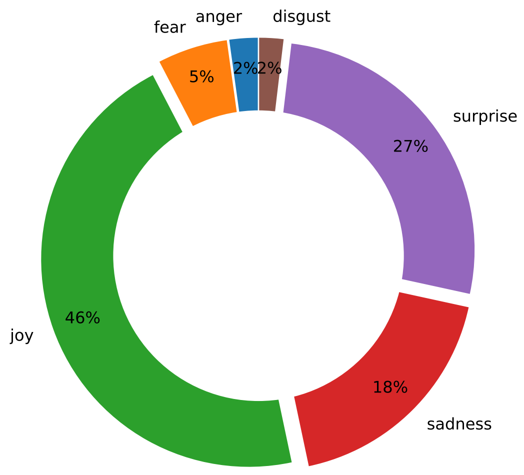


Figure 2.11: Label distribution of LDET

2.8.1 Models

We first show the best results for all the models that are mentioned above. The models to be compared are listed in Table 2.4, where SA-LSTM stands for bi-directinal LSTM with a self-attention layer.

Table 2.4: Models that will be compared in emotion detection task

Method	Type
Naïve Bayes	Non-neural
Random Forest	Non-neural
SVM	Non-neural
LSTM	Neural
SA-LSTM	Neural
Proposed	Neural

2.8.2 Metrics

The prediction of a class c_i can be seen as a binary classification by regarding the predictions of any other classes $c_j(c_j \in C, j \neq i)$ as one negative class. Therefore, for each of the class the corresponding predictions can be divided into four groups as described in Table 2.5.

Two terms are further defined according to this table,

Table 2.5: True conditions of predictions in binary classification

		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted positive	True positive (TP)	False positive (FP)
	Predicted negative	False negative (FN)	True negative (TN)

$$\mathrm{P}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \tag{2.24}$$

$$\mathrm{R}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \tag{2.25}$$

Where P_i is referred to as the *precision* of the class c_i, R_i is the *recall.* By taking the harmonic average of the precision and recall, we have the F1 score defined as the following.

$$\mathrm{F1}_i = \left(\frac{\mathrm{P}_i^{-1} + \mathrm{R}_i^{-1}}{2}\right)^{-1} \tag{2.26}$$

Precision, recall, and F1 scores are widely used as metrics for imbalanced binary classification problems. In the case of multi class classification, there are two widely used methods to summarize the scores for all the classes: macro average and micro average. Macro average simply takes the average of the corresponding metrics for each class. E.g. Macro-averaged precision $= \frac{1}{|C|}\sum_i^{|C|} P_i$. On the other hand, micro average aims at bringing everything together internally.

$$\text{Micro-averaged P} = \frac{\sum_i^{|C|} \mathrm{TP}_i}{\sum_i^{|C|} \mathrm{TP}_i + \sum_i^{|C|} \mathrm{FP}_i} \tag{2.27}$$

$$\text{Micro-averaged R} = \frac{\sum_i^{|C|} \mathrm{TP}_i}{\sum_i^{|C|} \mathrm{TP}_i + \sum_i^{|C|} \mathrm{FN}_i} \tag{2.28}$$

Similar to Equation 2.26, the Micro-averaged F1 is obtained by the harmonic average of micro-averaged precision and micro-averaged recall. In single label classification, it turns out $\sum_i^{|C|} \mathrm{FP}_i = \sum_i^{|C|} \mathrm{FN}_i$, hence micro-averaged precision, recall, and F1 have the same value.

Metric	Abbreviation
Macro-averaged precision	Macro P.
Macro-averaged recall	Macro R.
Macro-averaged F1	Macro F1
Micro-averaged F1	Micro F1

Table 2.6: Abbreviations of metrics

Considering TEC and LDET are imbalanced in terms of labels, we use both macro and micro averaged scores as the metrics. More specifically, we show the macro averaged precision, macro averaged recall, macro averaged F1-score and micro F1-score. For simplicity, we use the abbreviations that are listed in Table 2.6.

2.8.3 Experiments

All the models are experimented with the four datasets as in Table 2.2. For each of the dataset, we first hold out 10% as the final testset. The remaining 90% is used for training. To ensure the stability, we apply 5-fold cross validation on every model. That being said, each model is trained five rounds, each time the remained dataset is further divided into a training and validation set using the ratio of 4:1. Each round of the training, the model performs an inference on the held-out testset, and the final results is generated by doing majority voting on the five inference results. For the Non-neural methods in Table 2.4, the validation set is not used directly. Whereas the neural models use the loss on the validation set as the criterion for early stopping to prevent the models from overfitting. [68].

For the non-neural models, we use the implementations provided by Scikit-learn toolkit [64]. The size of the bag is set to 5,000. For *Random Forest*, the number of trees are 100. For *SVM*, we use the linear kernel as in Equation 2.4. We use

Table 2.7 2.8 2.9 2.10 show the results of the six models on the ISEAR, TEC, CBET and LDET datasets respectively. From the results, one can conclude with certainty that deep learning models perform much better than the three traditional text classification methods. Comparing the LSTM and

	Macro P.	Macro R.	Macro F1	Micro F1
Naïve Bayes	0.5351	0.536	0.5338	0.5359
Random Forest	0.546	0.5467	0.5363	0.5463
SVM	0.501	0.5061	0.5004	0.5059
LSTM	0.6167	0.6132	0.6068	0.6128
SA-LSTM	0.642	0.6261	0.6296	0.6258
Proposed Model	**0.6514**	**0.6523**	**0.6451**	**0.6519**

Table 2.7: Results on ISEAR dataset

	Macro P.	Macro R.	Macro F1	Micro F1
Naïve Bayes	0.5477	0.4841	0.5008	0.6035
Random Forest	0.5831	0.4024	0.4333	0.5741
SVM	0.5369	0.4969	0.5079	0.5935
LSTM	0.5904	0.5345	0.5555	0.6505
SA-LSTM	0.6207	0.5568	0.5759	0.6543
Proposed Model	**0.6522**	**0.5817**	**0.6034**	**0.6781**

Table 2.8: Results on TEC dataset

	Macro P.	Macro R.	Macro F1	Micro F1
Naïve Bayes	0.5277	0.5245	0.5218	0.5218
Random Forest	0.5188	0.5128	0.5111	0.5218
SVM	0.5288	0.5258	0.5263	0.5258
LSTM	0.5985	0.5993	0.5977	0.5993
SA-LSTM	0.6102	0.6107	0.6092	0.6107
Proposed Model	**0.6312**	**0.6324**	**0.6302**	**0.6324**

Table 2.9: Results on CBET dataset

	Macro P.	Macro R.	Macro F1	Micro F1
Naïve Bayes	0.5037	0.5142	0.5049	0.6313
Random Forest	0.5813	0.4568	0.4889	0.661
SVM	0.6414	0.4701	0.5072	0.6684
LSTM	0.6655	0.5914	0.6206	0.7385
SA-LSTM	0.6565	0.5954	0.6194	0.7376
Proposed Model	**0.6958**	**0.6054**	**0.6392**	**0.7523**

Table 2.10: Results on LDET dataset

SA-LSTM models, we find self-attention mechanism helps improving the performance of LSTM model on ISEAR, TEC, and CBET datasets but not on LDET. Our proposed model out performs all other models by a large margin on all the four datasets.

To better compare the performance of the models, we further take the average of the Macro F1 and the Micro F1 scores of the 4 datasets and then compare them in the bar chart as shown in Figure 2.12. In the bar chart, the vertical black lines indicate the standard deviation. It is clear that the models can be categorized into two groups: Naïve Bayes, Random Forest and SVM can be regarded as one group, whereas LSTM, SA-LSTM and the proposed model form the other. In our experiments, we find Naïve Bayes is computationally the cheapest among all the models and SVM is the most expensive model.

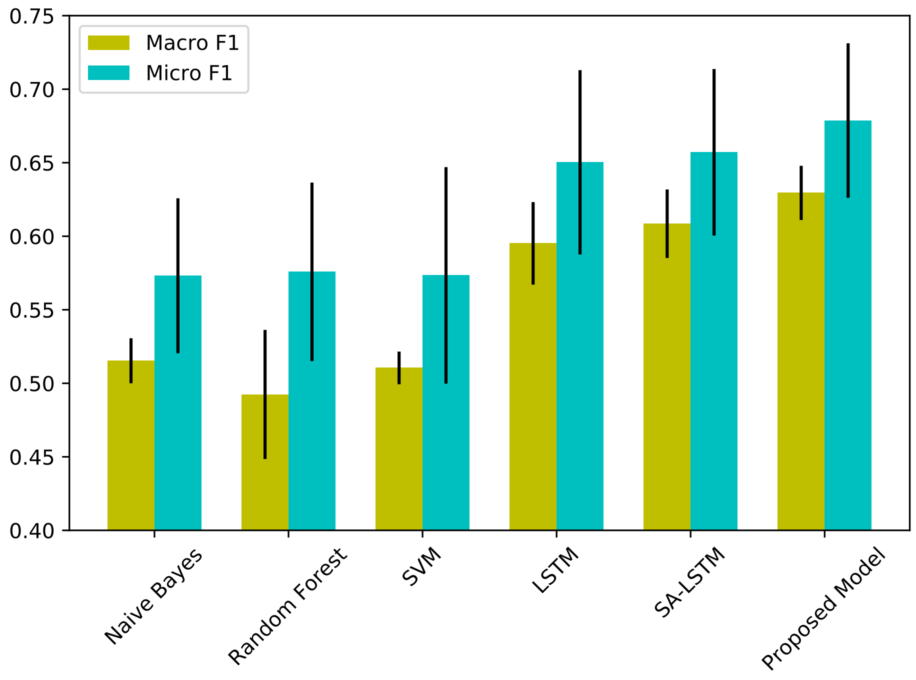


Figure 2.12: Comparison of Macro F1 and Micro F1 of the six models

2.9 Emotion Detection in Context

In the previous sections, we demonstrated the emotion classification in utterance level, i.e. each prediction is only subjected to an individual sentence.

In this section, we tackle the problem of detecting emotions in a contextual setup.

A conversations between two people can be seen as a sequence of utterances: $\boldsymbol{u} = u_1, u_2, \cdots u_n$, where n is the number of total utterances. Correspondingly, there are emotional labels that are assigned to each utterance: $e_1, e_2, \cdots, e_n$. Given the utterances as contextual information, the target is to fit a model towards maximizing the likelihood:

$$\mathcal{P}(e_i|u_1, u_2, \cdots, u_i), i \in \{1, 2 \cdots n\}. \tag{2.29}$$

The Equation 2.29 suggests that the estimation of each label e_i is conditioned on all of the previous utterances. In human conversations, it is obvious that we can not assume that every utterance carries the same emotion. Therefore, an optimal model should be able to take into consideration the emotions of each utterance independently.

For a conversation with n utterances, the emotion of each utterance can be estimated according to its context. Therefore, assume every utterance is labeled with one of the emotion categories, there can be n possible sub-tasks. To differentiate the tasks, we denote a task as *k-contextual emotion detection* when the emotion of the kth utterance is estimated given the $k-1$ previous occurred utterances as context.

2.9.1 SemEval 2019: Task3

The International Workshop on Semantic Evaluation (SemEval) hosts shared tasks in the domain of semantic understanding every year since 2012. SemEval 2019 is the 13th workshop. We attended the Task 3 [9] (*EmoContext: Contextual Emotion Detection in Text*) in SemEval 2019 and ranked at the 5th position among all the 165 participants [32].

Based on our definition in Section 2.9, this task is a problem of 3-contextual emotion detection. There are four emotion categories: *happy*, *sad*, *angry*, and *others*. The entire dataset provided by the organizer was split into three parts, *train*, *dev*, *test*. Table 2.11 gives the label distribution of each split. It is noticeable that the label distribution of the *train* set is very different from that of *dev* and *test* set. It violates one of the basic assumptions of many machine learning systems, where the training and testing sets are usually assumed to have the same data distribution.

Importance Weighting

Importance Weighting [86] is used when label distributions between the training and test sets are generally different, which is the case of the competition datasets (Table 2.11). It corresponds to weighting the samples according to their importance when calculating the loss.

A supervised deep learning model can be regarded as a parametrized function $f(\boldsymbol{x};\boldsymbol{\theta})$. The back-propagation learning algorithm through a differentiable loss is a method of Empirical Risk Minimization (ERM). Denote $(\boldsymbol{x}_i^{tr}, y_i^{tr})$, $i \in [1 \dots n_{tr}]$ are pairs of training samples, testing samples are $(\boldsymbol{x}^{te}, y^{te})$, $i \in [1 \dots n_{te}]$.

The ratio $\mathcal{P}(\boldsymbol{x})^{te}/\mathcal{P}(\boldsymbol{x})^{tr}$ is referred to as the *importance* of a sample $\boldsymbol{x}$. When the label distribution of training data and testing data are different: $\mathcal{P}(\boldsymbol{x}^{te}) \neq \mathcal{P}(\boldsymbol{x}^{tr})$, the training of the model $f_{\boldsymbol{\theta}}$ is then called under *covariate shift*. In such situation, the parameter $\hat{\boldsymbol{\theta}}$ should be estimated through *importance-weighted ERM*:

$$\arg\min_{\boldsymbol{\theta}} \Big[\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \frac{\mathcal{P}(\boldsymbol{x}^{te})}{\mathcal{P}(\boldsymbol{x}^{tr})} \text{loss}(y_i^{tr}, f(\boldsymbol{x}_i^{tr};\boldsymbol{\theta})\Big]. \tag{2.30}$$

	happy	angry	sad	others	size
Train	14.07%	18.26%	18.11%	49.56%	30160
Dev	5.15%	5.44%	4.54%	84.86%	2755
Test	4.28%	5.57%	4.45%	85.70%	5509

Table 2.11: Label distribution of train, dev, and test set

Proposed system

Hierarchical RNN for context One of the building component of our proposed model (see Figure 2.13) is the Hierarchical or Context recurrent encoder-decoder (HRED) [84]. HRED architecture is used for encoding dialogue context in the task of multi-turn dialogue generation [76]. It has been proven to be effective in capturing the context information of dialogue exchanges. It contains two types of recurrent neural net (RNN) units: *encoder*

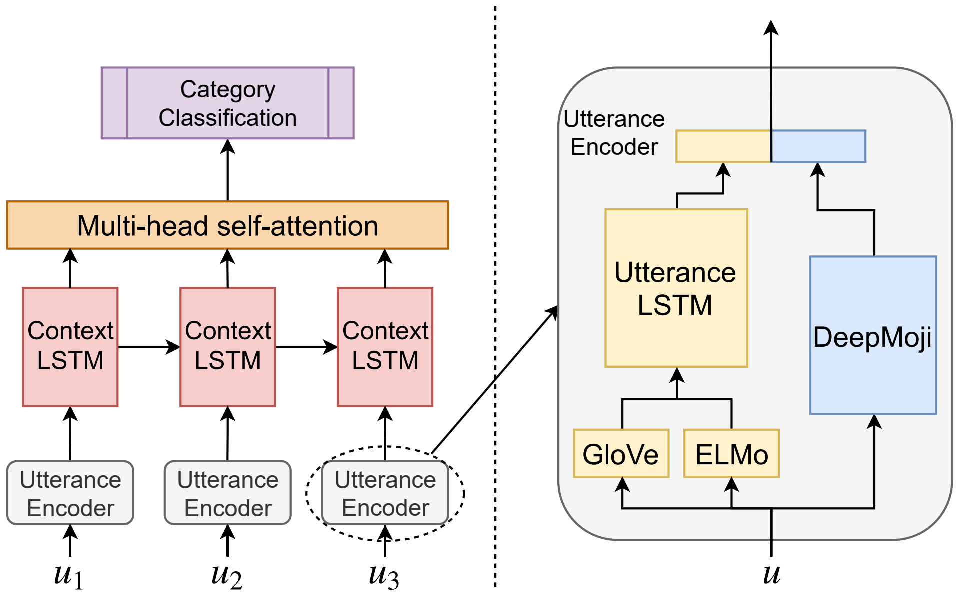


Figure 2.13: Overview of the proposed CSLD mod

Self-attentive LSTM (SL) Let $\boldsymbol{x}$ be the concatenation of u_1 ,u_2, and u_3. Hereby, $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]$, where x_i is the ith word in the combined sequence. Denote the pre-trained GloVe model as G. As GloVe model can be directly used by looking up the word x_i, we can use $G(x_i)$ to represent its output. On the contrary, ELMo embedding is not just dependent on the word x_i, but on all the words of the input sequence. When taking as input the entire sequence $\boldsymbol{x}$, n vectors can be extracted from the pre-trained ElMo model. Denote the vectors as $\boldsymbol{E} = [E_1, E_2, \cdots, E_n]$. E_i contains both contextual and semantic information of word x_i. We use a two-layer bidirectional LSTM as the encoder of the sequence $\boldsymbol{x}$. For simplicity, we denote it as $LSTM^e$. In order to better represent the information of x_i, we use the concatenation of $G(x_i)$ and E_i as the feature embedding of x_i. Therefore, we have the following recurrent progress:

$$h_t^e = LSTM^e([G(x_t); E_t], h_{t-1}^e). \tag{2.31}$$

h_t^e is the hidden state of encoder LSTM at time step t, and $h_0^e = \mathbf{0}$. Let $\boldsymbol{h}_{\boldsymbol{x}}^e = [h_t^e, h_t^e, \cdots, h_t^e]$ be the n hidden states of encoder given the input $\boldsymbol{x}$. Self-attention mechanism has been proven to be effective in helping RNN dealing with dependency problems [50]. We use the multi-head version of the self-attention [90] and set the number of channels for each head as 1. Denote the self-attention module as SA, it takes as input all the hidden states of the LSTM and summarizes them into a single vector. This process is represented as $h_{\boldsymbol{x}}^{sa} = SA(\boldsymbol{h}_{\boldsymbol{x}}^e)$. To predict the model, we append a fully connected (FC) layer to project $h_{\boldsymbol{x}}^{sa}$ on to the space of emotions. Denote the FC layer as *output*. Let $o_{\boldsymbol{x}}^{SL} = output(h_{\boldsymbol{x}}^{sa})$, then the estimated label of $\boldsymbol{x}$ is the $\arg\max_i(o_{\boldsymbol{x}}^{SL})$, where i is ith value in the vector $o_{\boldsymbol{x}}^{SL}$.

SL with DeepMoji (SLD) As shown in Sectin 2.8, in the task of utterance level emotion detection, using DeepMoji to add the emotion representation along side the semantic features can improve the performance on emotion detection as in Section 2.6. SLD is the combination of SA and DeepMoji. An SLD model without the output layer is in fact the utterance encoder of the proposed CSLD, which is illustrated in the right side of Figure 2.13. Denote the DeepMoji model as D, when taking as input $\boldsymbol{x}$, the output is represented as $h_{\boldsymbol{x}}^d = D(\boldsymbol{x})$. We concatenate $h_{\boldsymbol{x}}^d$ and $h_{\boldsymbol{x}}^{sa}$ as the feature representation of sequence of $\boldsymbol{x}$. Same as SL, an FC layer is added in order to predict the label: $o_{\boldsymbol{x}}^{SLD} = output([h_{\boldsymbol{x}}^{sa}; h_{\boldsymbol{x}}^d])$.

The model SLD without the output layer can be represented as the *utterance encoder* as in the Figure 2.13. SLD is in fact a variant of the proposed model in Section 2.6. The difference between the SLD model and the proposed model as tin Figure 2.9 comes in two folds: there there is no FC layer after the DeepMoji module; the parameters in DeepMoji module in previously proposed model is fixed whereas the ones in SLD is set to trainable.

Contextual SLD (CSLD) Unlike SL and SLD, the input of CSLD is not the concatenation of u_1, u_2, and u_3. Instead, each utterance is firstly fed into the *utterance encoder* respectively, which results in one dense representation for each utterance. The dense representations are further fed into other module to capture the contextual information.

Following the previous annotations, an utterance u_i is firstly encoded as $h^{sa}_{u_i}$ and $h^{d}_{u_i}$. We use another two layer bidirectional LSTM as the context RNN, denoted as $LSTM^c$. Its hidden states are iterated through:

$$h^c_t = LSTM^c([h^{sa}_{u_t}; h^d_{u_t}], h^c_{t-1}), \tag{2.32}$$

where $h^c_0 = \mathbf{0}$. The three hidden states $\boldsymbol{h^c} = [h^c_1, h^c_2, h^c_3]$, are fed as the input to a self-attention layer. The resulting vector $SA(\boldsymbol{h^c})$ is also projected to the label space by an FC layer.

An overview of the proposed system is shown in Figure 2.13.

2.9.2 Experiments

	F1	Happy	Angry	Sad	Harm. Mean
SL	Dev	0.6430	0.7530	0.7180	0.7016
	Test	0.6400	0.7190	0.7300	0.6939
SL w/o IW	Dev	0.6170	0.6860	0.6710	0.6566
	Test	0.6240	0.6670	0.6840	0.6573
SLD	Dev	0.6470	0.7610	0.7360	0.7112
	Test	0.6350	0.7180	0.7360	0.6934
SLD w/o IW	Dev	0.6350	0.6950	0.6890	0.6719
	Test	0.6220	0.6610	0.6920	0.6571
CSLD	Dev	0.7460	0.7590	0.8100	**0.7706**
	Test	0.7220	0.7660	0.8180	**0.7666**
CSLD w/o IW	Dev	0.6650	0.6930	0.7850	0.7108
	Test	0.6870	0.7170	0.7720	0.7237

Table 2.12: Macro-F1 scores and its harmonic means of the proposed models, w/o IW stands for training without using the importance weight calcuated by Equation 2.30.

Data preprocessing From the training data we notice that emojis are playing an important role in expressing emotions. We first use *ekphrasis* package [3] to clean up the utterances. *ekphrasis* corrects misspellings, handles textual emotions (e.g. ':)))'), and normalizes tokens (hashtags, numbers, user mentions etc.). In order to keep the semantic meanings of the emojis, we use the *emojis* package[3] to first convert them into their textual aliases and then replace the ":" and "_" with spaces.

Environment and hyper-parameters We use PyTorch 1.0 for the deep learning framework, and our code in Python 3.6 can be accessed in GitHub[4]. For fair comparisons, we use the same parameter settings for the common modules that are shared by the SL, SLD, and CSLD. The dimension of *encoder* LSTM is set to 1500 per direction; the dimension of *context* LSTM is set to 800 per direction. We use Adam optimizer with initial learning rate as 5e-4 and a decay ratio of 0.2 after each epoch. The parameters of DeepMoji are set to trainable.

According to the description in [11], the label distribution for *dev* and *test* sets are roughly 4% for each of the emotions. However, from the *dev* set (Table 2.11) we know that the proportions of each of the emotion categories are better described as %5 each, thereby we use %5 as the empirical estimation of distribution $\mathcal{P}(\boldsymbol{x}^{te})$. We did not use the exact proportion of *dev* set as the estimation to prevent the overfitting towards *dev* set. The sample distribution of the *train* set is used as $\mathcal{P}(\boldsymbol{x}^{tr})$. We use *Cross Entropy* loss for all the aforementioned models, and the loss of the training samples are weighted according to Equation 2.30.

Results and analysis We run 9-fold cross validation on the *train* set. Each iteration, 1 fold is used to prevent the models from overfitting while the remaining folds are used for training. Therefore, every model is trained 9 times to ensure stability. The inferences over *dev* and *test* sets are performed on each

[3]https://pypi.org/project/emoji

[4]https://github.com/chenyangh/SemEval2019Task3

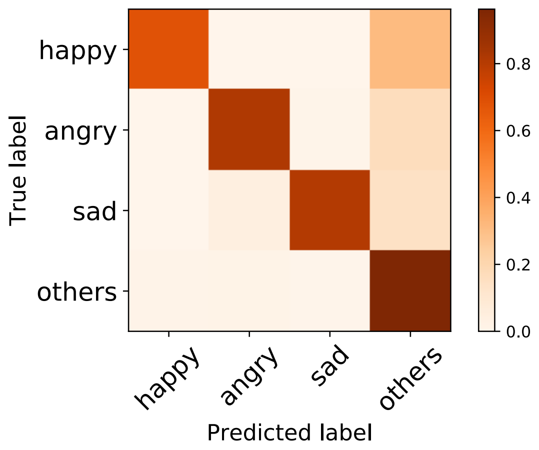


Figure 2.14: Heat map of the predictions

iteration. We use the majority voting strategy to merge the results from the 9 iterations. It has to be mentioned that the *dev* set is not the development set we used in training in order to prevent overfitting. The results are shown in Table 2.12. It shows that the proposed CSLD model performs the best. It is also obvious that the *importance weight* plays an important role for this task. The performance of SLD and SL are very close to each other, on the *dev* set, SLD performs better than SL but they have almost the same overall scores on the *test* set. The Macro-F1 scores of each emotion category are very different from each other: the classification accuracy for emotion *Sad* is the highest in most of the cases, while the emotion *Happy* is the least accurately classified by all the models. We also noticed that the performance on the *dev* set is generally slightly better than that on the *test* set.

Figure 2.14 shows the heat map of our final submission when it is compared to the ground truth. From a confusion matrix of our final submission, we notice that there are barely miss-classifications among the three categories (*Angry, Sad*, and *Happy*). For example, the emotion Sad is rarely miss-classified as "Happy" or "Angry". Most of the errors correspond to classifying the emotional utterances in the *Others* category. We think, as future improvement, the models need to first focus on the binary classification "Others" versus "Not-Others", then the "Not-Others" are classified in their respective emotion.

2.10 Conclusion and Perspective

In this chapter, we first review the concepts and theories of emotions. We formalise the task of emotion mining from text as text classification and proposed a model that is able to achieve the state-of-the-art performance on utterance level emotion classification. We also show our work on the SemEval-2019 task 3, where we use a model that derived from the model in Section 2.6. The empirical results show that the proposed structure, which combines three pre-trained models is both effective and precise in the tasks of emotion classification.

In the Table 2.12, we also notice significant performance boost by the use of *importance weight*. This observation raises a very important problem in emotion detection. In theory, most of the machine learning models require consistency in the training and testing datasets. However, this assumption usually does not hold in real-world applications. Covariate shift and other non-stationary environments are common especially in the task of emotion detection. For example, many datasets for training emotional classifiers are extracted from tweets where people tend to express opinions differently than in their real life. Another critical problem is the detection of neutral emotions. Let alone there are only a few datasets [61] that contain the label *neutral* The frequency of people expressing emotions is unknown and might be subjective to their personalities. If we assume *non-emotion* equals neutral. Given the emotional tweets, we can assume the detection of neutral emotions as the problem of anomaly detection [15, 23, 44].

Chapter 3

Response with Emotions

In this chapter, we aim at tackling the second question that is raised in Section 1.2: *How can a chatbot express a specific emotion?*.

This chapter is organized as follows. Section 3.1 describes the general framework that is widely applied in chit-chat systems. Section 3.2 formally defines the task of *expressing specific emotion*. We explain a open-domain neural dialogue generation model in Section 3.3. In Section 3.4, we demonstrate 7 models that are able to handle this task. Lastly, we show our perspectives on this task and conclude this chapter in Section 3.5.

The work presented in this chapter on generating a response while expressing a given emotion, would logically rely on the contributions on emotion detection as a multi-label emotion classification, presented previously. However, the work presented in this current chapter was in effect chronologically conducted before Chapter 2 and was relying on existing emotion detection methods. Therefore, here, we only use the CBET dataset (Section 2.7.1) with SA-LSTM (Section 2.8.1).

3.1 Motivation

3.1.1 Conversational Agent

Conversational Agent (CA) or *dialogue systems* are computer programs which are intended to converse with a human with a coherent structure [16]. Dialogue systems can use text, speech, gestures, graphics, etc. for the communication on both input and output. In this research, we only consider the conversations that are using text as the channel of information. Note that speech and text can be transformed to each other using speech-to-text or text-to-speech converters.

Based on the purpose of the conversations, CAs can be categorized into two groups: task-specific systems and chit-chat systems. Task-specific (or goal-oriented) agents are usually designed to help users with completing some specific tasks. For examples, many companies deploy goal-oriented conversational agents on their websites to serve as customer service. Some popular Intelligent Personal Assistant (IPA) agents, such as Siri [83], Amazon Alexa [2], are mostly designed as task-specific systems which are able to complete a narrowed range of tasks (booking, making phone calls, navigating, etc.). On the other hand, chit-chat systems (also known as chatbots, chatterbot, open-domain or open-ended dialogue systems) are designed to carry on extended conversations with the goal of mimicking the unstructured conversational or 'chats' characteristic of human-human interaction rather than focusing particular tasks such as booking plane flights [37, p. 423].

Chit-chat systems are commonly used for entertainment purpose. The Automated Nursing Agent (ANA) is an undergoing project that aims at building a personal companion for the elderly. ANA is a sophisticated system, containing task-specific modules which not only should be able to answer general queries such as weather, nutrition, and medicine, but also should be able to extend the chats to open-ended scenarios (i.e. casual chit-chatting). When designing CA for the elderly, it is very important to not only keep tracking the emotions which are expressed by the elderly but also respond with proper (or safe) emotions. For example, if the elderly says: "I just lost my old good friend." It would be much more appropriate if we can force the system to respond with sympathy: "I am sorry to hear that." rather than happiness (e.g. "I am glad to hear that.").

3.1.2 Empathetic Dialogue System

In Chapter 1, we make a brief statement that Intelligence Quotient (IQ) and Emotional Quotient (EQ) play essential roles in building conversational agents.

We give a detailed explanation as follows.

IQ is referring to the cores that are derived from several standardized tests designed to evaluate the intelligent level of humans. In the context of building a CA , IQ refers to having some particular skills (include memory modeling, image, natural language understanding, reasoning, generation, and prediction [104]) that would help users in solving sophisticated problems.

EQ, on the other hand, is formally defined on Collins Dictionary as "a (notional) measure of a person's adequacy in such areas as self-awareness, empathy, and dealing sensitively with other people" [19]. Empathy can be briefly described as the ability to understand the feeling of others. An empathetic dialogue system needs to identify the user's emotions from the conversation, detect how emotions evolve over time, understand the user's emotional needs and react properly. [104].

At the current stage, there have been many studies in detecting emotions (see Chapter 2). Understanding user's emotional needs is a subjective task. There are not many related studies which are under the scope of computer science. The personalities, context of the dialogue, specific background knowledge would all have impacts on what is the need of the user emotionally. We will discuss this problem from the perspective of Machine Learning (ML) in Chapter 4. In this Chapter, however, we tackle the problem of how a CA expresses emotions given that we have already known what emotions to express.

3.2 Task Definition

In Section 2.5 and 2.9, we apply Deep Learning models to the task of emotion classification and achieved very promising results. Given a document X and its corresponding label Y, a text classifier is a discriminative model [5] which essentially learns a function that estimates the probability of $\mathcal{P}(Y|X)$ directly. In the context of dialogue generation, the label Y is a sequence of words. In this section, we will give the formal definition of the task of expressing specific emotions in dialogue generation.

3.2.1 Language Model

Language modeling is the task of assigning a probability to a sentence in a given language [24, p. 105]. For example, what is the probability of seeing the sentences “How are you doing today?” in the English language.

Besides evaluating the probability of a sentence, a Language Model (LM) can also estimate the probability of the next word given a sequence of words. For instance, a LM can assign probabilities to all possible words after the sequence “How are you”. The word “doing” might have a great chance since ”How are you doing” is a common English phrase.

Formally, let $\boldsymbol{w} = [w_1, w_2, \cdots, w_n]$ be a sentence of length n. A Language Model yields the following joint probability:

$$\mathcal{P}(w_1, w_2, \cdots, w_n). \tag{3.1}$$

In the case of prediction the word w_t given $w_1, w_2, \cdots, w_{t-1}$, it gives the following:

$$\mathcal{P}(w_1, w_2, \cdots, w_t) = \mathcal{P}(w_t|w_1, w_2, \cdots, w_{t-1})\mathcal{P}(w_1, w_2, \cdots, w_{t-1}). \tag{3.2}$$

After applying the chain rule of probability, Equation 3.1 and 3.2 have the following relation:

$$\begin{aligned}\mathcal{P}(\boldsymbol{w}) &= \mathcal{P}(w_1, w_2, \cdots, w_n) \\ &= \mathcal{P}(w_1)\mathcal{P}(w_2|w_1)\mathcal{P}(w_3|w_1, w_2)\cdots\mathcal{P}(w_n|w_1, w_2, \cdots w_{n-1}) \\ &= \prod_{t=1}^{n} \mathcal{P}(w_t|w_1, w_2, \cdots w_{t-1}),\end{aligned} \tag{3.3}$$

where $\mathcal{P}(w_1|w_0) = \mathcal{P}(w_1)$, since we can regard w_0 as a constant value which indicates the signal of starting. Learning an LM is a unsupervised learning task which can be trained on any unlabeled corpus. In the applications of NN, LM is often used to learn word representations (see Section 2.5.1) which, in turn, can be applied on further downstream NLP tasks (pre-trained models such as ELMo [66] and BERT [14] have shown great success on this track). LM can also be used to generate fake documents, such as GPT-2 [70].

3.2.2 Conditional Language Model

Compared to LM, a conditional language model assigns probabilities to a sequence of words $\boldsymbol{w}$, given some conditioning context $\boldsymbol{x}$. Similar to Section 3.2.1, we have the following conditional probability $\boldsymbol{w}$ given $\boldsymbol{x}$:

$$\mathcal{P}(\boldsymbol{w}|\boldsymbol{x}) = \prod_{t=1}^{n} \mathcal{P}(w_t|\boldsymbol{x}, w_1, w_2, \cdots w_{t-1}), \tag{3.4}$$

where $\mathcal{P}(w_1|\boldsymbol{x}, w_0) = \mathcal{P}(w_1|\boldsymbol{x})$. Conditional language models has many real-word applications such as machine translation [7, 71, 87], document summarization [71], question answering [93, 98], dialogue generation [46, 91], etc.

3.2.3 Open-domain Dialogue Generation Models

Open-domain dialogue (or response) generation is an essential task in Natural Language Understanding (NLG). With the recent development in Deep Learning, neural conversational models have attracted increasing attention in the past years [47, 48, 91]. The task itself can be regarded as a problem of conditional language modeling, where in Equation 3.4 the condition context $\boldsymbol{x}$ is the history of the whole conversation session or just a single input from the user and the generated sequence $\boldsymbol{w}$ is the response by the dialogue generation model given the context $\boldsymbol{x}$.

There have been several types of unique NN frameworks that are able to handle the task of conditional language modeling, for example, Sequence-to-sequence (Seq2Seq) [87, 91], Conditional Variational Autoencoders (CVAE) [81, 101], Sequence Generative Adversarial Nets (SeqGAN) [89, 99], etc. Theoretically, any conditional language model can be used in the task of open-domain dialogue generation, because eventually what we need for this task to learn a model that generates a sequence Y given the input X from user. In this research, however, we chose Seq2Seq as the backbone of the dialogue generation model. Therefore, we will introduce the general framework of the Seq2Seq

(Section 3.3) and a dedicated attention mechanism which is commonly used to improve Seq2Seq (Section 3.3.3).

3.2.4 Emotion as Additional Condition

As described in Section 3.1.2, in this chapter we tackle the problem of expressing a given emotion in the context of open-domain dialogue generation setup. Assume we are able to build a conditional language model for the task of open-ended dialogue generation. The next question is how to make it express an emotion.

Let e be the emotion we want to express, following the above annotation, we need to train a conditional language model that uses both the context X and emotion e as condition. Therefore, the task becomes modeling the following conditional probability distribution:

$$\mathcal{P}(Y|X, e), \tag{3.5}$$

where context X, depending on the task, may be the entire conversation history or just a single input. The question of "How to express a given emotion in the open-domain dialogue generation setup? " then could be transformed into "How do we design and train a special conditional language model that take as input an additional condition rather than the context sequence X?"

3.3 The Seq2Seq Model

As mentioned above, we use Seq2Seq as the backbone of this research, therefore it is important to first explain the model in detail. In Section 3.3.1 we introduce the overall structure of the Seq2Seq, in Section 3.3.3 we demonstrate the widely used mechanism that is used to improve the performance of the RNN based Seq2Seq model.

3.3.1 The Main Structure

The Sequence-to-sequence (Seq2Seq) model is originally proposed by Sutskever, Vinyals, and Le in 2014 [87]. It is one of the first attempts that use NN for

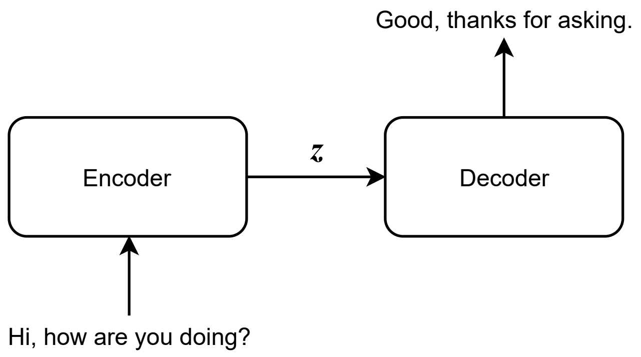


Figure 3.1: Structure of sequence-to-sequence model

Consider a single turn conversations, for example one says "Hi, how are you doing?" and another responses with "Good, thanks for asking.". In the following context, for simplicity, we will refer the first utterance ("How are you doing?") as the *source sentence* and the following utterance ("Good, thanks for asking") as the *target sentence*. As illustrated in Figure 3.1, an encoder will first take as input the source sentence. this processing is referred as "encoding" and the source sentences will be encoded into a vector or matrix representation, denote it as $\boldsymbol{z}$. The decoder will then try to output the target sentence given $\boldsymbol{z}$.

Denote X as the source sentence, and Y as the target sentence. A single turn conversational dataset can be represented as $\{X, Y\}^k$, where k is the number of pairs in the dataset. Both X and Y are sequence of words. Let $X = [x_1, x_2, \cdots, x_m]$ and $Y = [y_1, y_2, \cdots, y_n]$.

Note that Y and y_i have different meanings than that in the task of text classification (see Section 2.3). In this chapter, Y is a sequence of words and each word y_i can be regarded as label (classification over candidate words, we will explain this in Section 3.3). In the task of text classification, Y represents label corresponding to the sequence X and it is represented in the form binary vectors such that $y_i \in \{0, 1\}$ is the decomposition of Y.

The procedure of encoding and decoding can be formulated as

$$\boldsymbol{z} = \text{Encoder}(X), \tag{3.6}$$

$$Y = \text{Decoder}(\boldsymbol{z}). \tag{3.7}$$

This encoder-decoder structure becomes the foundation of many other models in NLP such as Convolutional Neural Network (CNN) based model [22] and fully attention based model [90]. However, in this research we follow the original research in [87] and use RNN based models as both the encoder and decoder.

In section 2.5.2 we have explained the formula of vanilla RNN. On a high level of abstraction, vanilla RNN can be formulated as following,

$$h_t = RNN(M(x_t), h_{t-1}), \tag{3.8}$$

where h_0 is usually initialized as $\mathbf{0}$. In current NLP research, the use of pre-trained word embedding is widely applied (see Section 2.5.1). Therefore, x_t in the source sentence X is often firstly transformed to its corresponding pre-trained embeddings. Here we use M to represent this procedure. In Figure 3.2, we show complete computation graph, where "E" represents an RNN based encoder and "D" represents an RNN based decoder. The source sentence X is thereby encoded as $\boldsymbol{z} = h^E_m$, where we use the h^E_i to denote the hidden states of the encoder RNN at time step i, and similarly h^D_i is the corresponding state for the decoder.

The decoder is another RNN based model which uses $h_0 = h^E_m$ as the initial hidden state, so that the information of sequence X is fed into the decoder. Therefore, the dimension of the RNN based encoder and decoder models are usually set to the same, otherwise a Fully Connected layer can be used as a bridge between them. To begin with, there is firstly a synthetic token for the decoder to be started with. Denote this token as "<s>", and we can regard this token as y_0 which will be embedded with a random initialized vector.

To avoid ambiguity, we use RNN^D to represent the decoder RNN, and its recursive iteration can be formulated as the following.

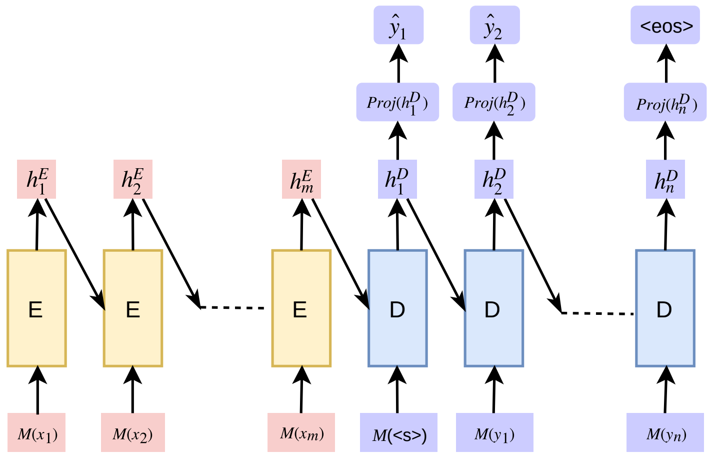


Figure 3.2: Structure of sequence-to-sequence model

$$h_t^D = RNN^D(M(y_{t-1}), h_{t-1}^D), \tag{3.9}$$

$$h_0^D = h_m^E. \tag{3.10}$$

In order to generate the target sequence Y, we apply a Fully Connected layer upon the hidden state of each time step t and project it to the space of the vocabulary $\mathbb{R}^{|V|}$ (see Section 2.4.1). We denote this FC layer as $Proj$. We use Softmax function (See Equation 2.14) to normalize $Proj(h_t^D)$ so that it could represent the probability distribution of next generated token. Denote the token that is estimated by the model as $\hat{y}_t$, by choosing the most likely one, we have:

$$\hat{y}_t = \arg\max\Big(\text{Softmax}\big(Proj(h_t^D)\big)\Big). \tag{3.11}$$

As shown in the top-left corner of Figure 3.2, there is another token "<eos>" which indicate the end-of-sentence literately. "<eos>" token can be seen as y_{n+1}.

3.3.2 Loss Function of Seq2Seq

We have introduced the concept of Empirical Risk Minimization (ERM) in Section 2.9.1, as a DL model, Seq2Seq is also trained through ERM. Therefore, we have also to define how to calculate the loss which measures the distance between the target sequence Y and its approximation $\hat{Y}$ which is generated by the Seq2Seq model. In Section 2.5.5, we show how the loss is calculated in the case of text classification (see Equation 2.22). In the task of sequence generation the target Y is a sequence of discrete words, the target Y is a sequence of discrete words and each word y_i is estimated by according to Equation 3.11. Therefore, the estimation of each word can be regarded as a sub-task of text classification where the target category is the set of candidate words V. In order to generate correctly according to the training dataset, we need to take into consideration the joint probability of each predicted word. This is also known as the conditional language model as we described in Section 3.2.2. Following the annotation, we have y_i as the ground truth of each word in the sequence Y, and $\hat{y}_i$ is the corresponding word in the predicted sequence $\hat{Y}$. Similar to the general text classification, y_i can be represented as one-hot binary vectors $[b_1, \cdots, b_{|V|}]$, $\sum_i b_i = 1,\ b_i \in \{0, 1\}$. The probability distribution of $\hat{y}_i$ is estimated through $\text{Softmax}\big(Proj(h_t^D)\big)$, we denote it as $[\hat{b}_1, \cdots, \hat{b}_{|V|}]$, $\sum_i \hat{b}_i = 1,\ \hat{b}_i \in [0, 1]$. The loss over the target sequence Y and its approximation is thus evaluated as follows,

$$\mathcal{L}(Y, \hat{Y}) = \sum_{i}^{m} H(y_i, \hat{y}_i) \tag{3.12}$$

$$H(y, \hat{y}) = -\sum_{i=1}^{|V|} \Big[b_i \log \hat{b}_i + (1 - b_i) \log(1 - \hat{b}_i) \Big]. \tag{3.13}$$

By minimizing the loss $\mathcal{L}$, a Seq2Seq model is able to learn the conditional language model of $\mathcal{P}(Y|X)$. Therefore, when X and Y represents two utterances of one conversation exchange, a properly trained model would generate a response $\hat{Y}$ given any X.

3.3.3 Seq2Seq Attention

Similar to Section 2.5.4 where we apply self-attention on RNN based models to achieve better results, in the research of [52, 71], Rush, Chopra, and Weston and Luong, Pham, and Manning propose several variants of attention mechanisms that improve the performance of the original Seq2Seq significantly. In this research, we also use Seq2Seq models to improve our end-to-end neural dialogue model. In order to avoid ambiguity with the self-attention module, we refer this type of mechanisms as *Seq2Seq-attention*.

As shown in Equation 3.10, the work of Sutskever, Vinyals, and Le [87] and ȅxtcitecho2014learning, only the last hidden state of the encoder is used for decoding ($\boldsymbol{z} = h_m^E$). In addition to that, the vector $\boldsymbol{z}$ is only used once while initialize the hidden state the decoder, which potentially make the decoder lost the track of long term dependency on the encoded sequence X. On the other hand, *Seq2Seq-attention* allows the decoder to look up the entire information in source sequence X throughout the phase of decoding. In this section we introduce the *global attention* which is proposed in [52].

Same as the emotion classification, we choose LSTM as the RNN based model for this neural dialogue generation task. In the following context, we use $LSTM^E$ and $LSTM^D$ to represent the encoder and decoder RNN models respectively.

Similarly, a LSTM encoder can be represented as

$$\begin{aligned} h_t^E, c_t^E &= \mathrm{LSTM}^E(M(x_i), [h_{t-1}^E; c_{t-1}^E]) \\ h_0^{En} &= c_0^{En} = \mathbf{0}. \end{aligned} \tag{3.14}$$

In the above formula, the notation of square brackets with vectors that are separated by semicolons refers to the concatenation of vectors. h_t^E are c_t^E both hidden states for the encoder LSTM (see Section 2.5.3). Compared to the vanilla RNN, LSTM has additional gates which help retaining more long range dependency, in addition, it also has an extra *cell state* c_t at each time step. It has to be mentioned that the notation c_t is *not* the same as the *context vector* in [52]. *Context vector* is the source-side information that helps predict the current target word y_t. In this research, we use CTX_t to avoid ambiguity with the *cell state* in LSTM.. In Equation 3.11, h_t is directly used to predict the probability distribution of the target token y_t. However, with attention mechanism, h_t^D will be firstly updated to its corresponding *attention vector* $\tilde{h}_t^D$ according to:

$$\tilde{h}_t^D = \tanh\left(W_c\left[CTX_t; h_t^D\right]\right) \tag{3.15}$$

where W_c is a trainable matrix.

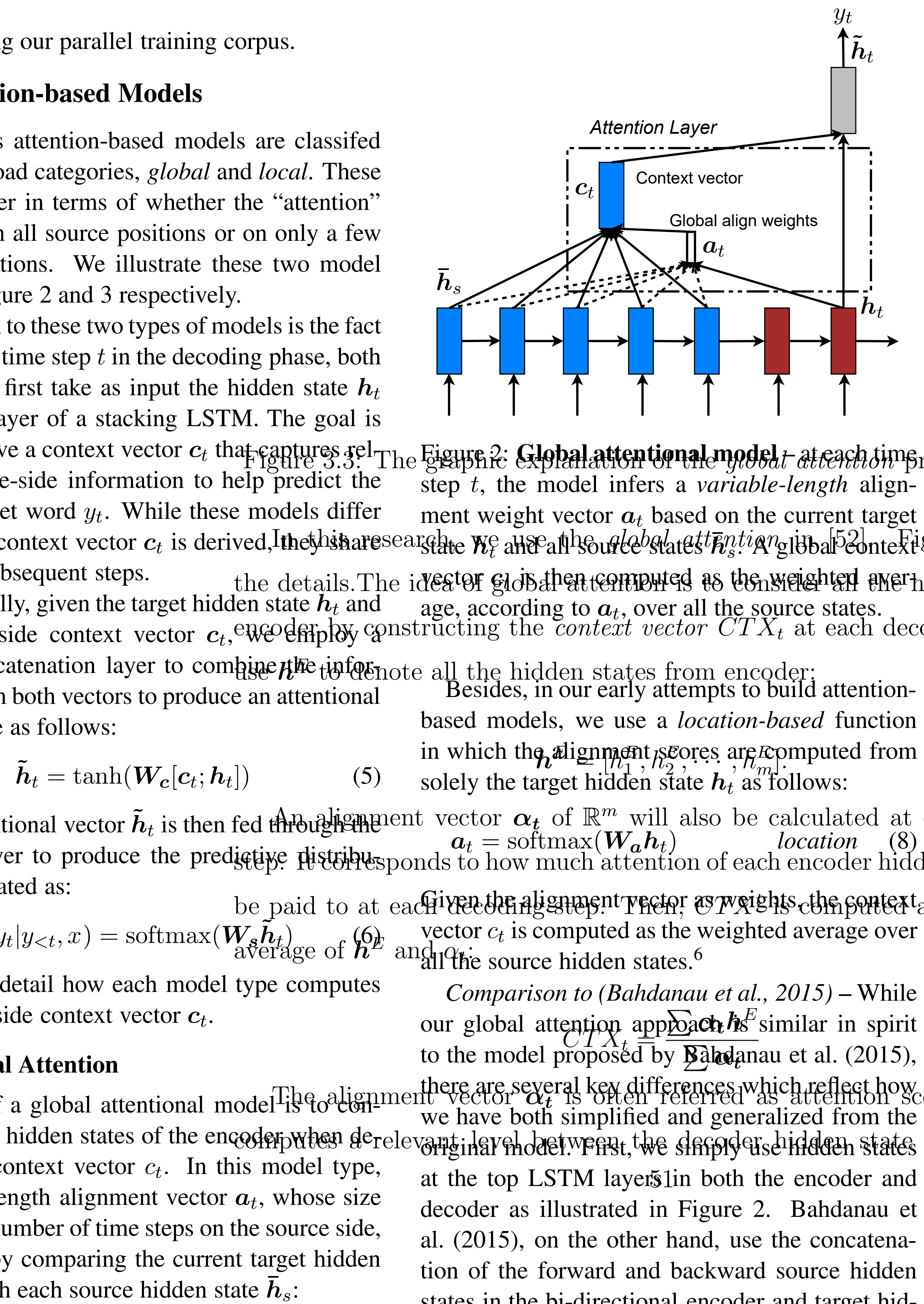


Figure 3.3: The graphic explanation of the *global attention* proposed in [52]

g our parallel training corpus.

ion-based Models

s attention-based models are classifed oad categories, *global* and *local*. These er in terms of whether the "attention" n all source positions or on only a few tions. We illustrate these two model ure 2 and 3 respectively.

to these two types of models is the fact time step t in the decoding phase, both first take as input the hidden state $\boldsymbol{h}_t$ ayer of a stacking LSTM. The goal is ve a context vector $\boldsymbol{c}_t$ that captures rel- e-side information to help predict the et word y_t. While these models differ context vector $\boldsymbol{c}_t$ is derived, they share bsequent steps.

lly, given the target hidden state $\boldsymbol{h}_t$ and side context vector $\boldsymbol{c}_t$, we employ a catenation layer to combine the infor- n both vectors to produce an attentional e as follows:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{5}$$

tional vector $\tilde{\boldsymbol{h}}_t$ is then fed through the ver to produce the predictive distribu- ated as:

$$_t|y_{<t}, x) = \text{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t) \tag{6}$$

detail how each model type computes side context vector $\boldsymbol{c}_t$.

l Attention

f a global attentional model is to con- hidden states of the encoder when de- context vector c_t. In this model type, ength alignment vector $\boldsymbol{a}_t$, whose size umber of time steps on the source side, y comparing the current target hidden h each source hidden state $\bar{\boldsymbol{h}}_s$:

Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

Besides, in our early attempts to build attention-based models, we use a *location-based* function in which the alignment scores are computed from solely the target hidden state $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{W_a}\boldsymbol{h}_t) \quad \textit{location} \tag{8}$$

Given the alignment vector as weights, the context vector c_t is computed as the weighted average over all the source hidden states.[6]

Comparison to (Bahdanau et al., 2015) – While our global attention approach is similar in spirit to the model proposed by Bahdanau et al. (2015), there are several key differences which reflect how we have both simplified and generalized from the original model. First, we simply use hidden states at the top LSTM layers in both the encoder and decoder as illustrated in Figure 2. Bahdanau et al. (2015), on the other hand, use the concatenation of the forward and backward source hidden states in the bi-directional encoder and target hid-

In this research, we use the *global attention* in [52]. Figure 3.3 shows the details.The idea of global attention is to consider all the hidden from the encoder by constructing the *context vector* CTX_t at each decoding step. We use $\boldsymbol{h}^E$ to denote all the hidden states from encoder:

$$\boldsymbol{h}^E = [h_1^E, h_2^E, \dots, h_m^E]. \tag{3.16}$$

An alignment vector $\boldsymbol{\alpha_t}$ of $\mathbb{R}^m$ will also be calculated at every decoding step. It corresponds to how much attention of each encoder hidden state should be paid to at each decoding step. Then, CTX is computed as the weighted average of $\boldsymbol{h}^E$ and α_t:

$$CTX_t = \frac{\sum \boldsymbol{\alpha_t}\boldsymbol{h}^E}{\sum \boldsymbol{\alpha_t}} \tag{3.17}$$

The alignment vector $\boldsymbol{\alpha_t}$ is often referred as attention score as well. It computes a relevant level between the decoder hidden state h_t and each of the encoder hidden states in $\boldsymbol{h}^E$ according to a **score** function. The *relevance scores* will be further normalized using Softmax to get the attention scores $\boldsymbol{\alpha_t} = [\alpha_1, \alpha_2, \cdots, \alpha_m]$, where m is length of the source sequence. Let $\boldsymbol{\alpha_t}(i) \triangleq \alpha_i$. $\boldsymbol{\alpha_t}$ will be calculated as:

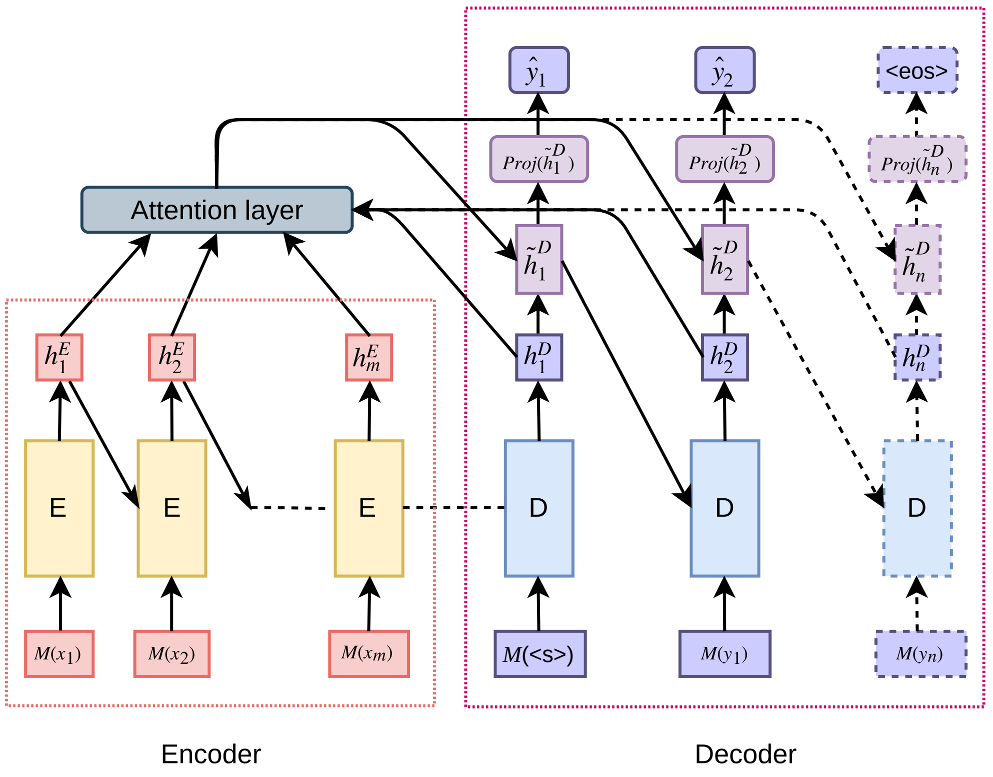


Figure 3.4: Seq2seq model with attention mechanism

$$\text{score}\left(h_t^D, h_i^E\right) = \begin{cases} {h_t^D}^\top h_i^E & \textit{dot} \\ {h_t^D}^\top W_a h_i^E & \textit{general} \\ v_a^\top \tanh\left(W_a\left[h_t^D; h_i^E\right]\right) & \textit{concat} \end{cases} \tag{3.19}$$

where W_a and v_a are trainable matrix and vector respectively. $\top$ is the transpose operation. In this research, we apply the *general* attention score on all our proposed models.

So far we have explained how to calculate the context augmented hidden state $\tilde{h}_t^D$ based on the decoder hidden state of time step t and all the encoder hidden states. The recursive decoding steps in Equation 3.9 is thus modified as:

$$\begin{aligned} h_t^D, c_t^D &= \text{LSTM}^D\Big(M(y_i), \big[\tilde{h}_{t-1}^D; c_{t-1}^D\big]\Big) \\ \tilde{h}_0^D &= h_m^E,\ c_0^D = c_m^E \end{aligned} \tag{3.20}$$

In the above equation, the hidden state of the LSTM model is using the updated value of $\tilde{h}_{t-1}^D$ instead of h_{t-1}^D. Therefore, the probability distribution of the target token $\hat{y}$ is estimated through:

$$\hat{y}_t = \arg\max\Big(\text{Softmax}\big(Proj(\tilde{h}_t^D)\big)\Big). \tag{3.21}$$

The computation pipeline can be regarded as $h_t^D \rightarrow \boldsymbol{\alpha}_t \rightarrow CTX_t \rightarrow \tilde{h}_{t-1}^D$. We show a detailed graphical explanation of the Seq2Seq with attention model in Figure 3.4.

3.4 Proposed models

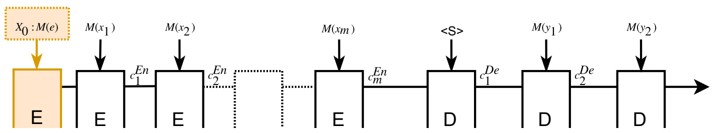
In Section 3.2, we have explained how the task of "expressing specific emotions" can be regarded an a special case of conditional language modeling. In Section 3.3, we explain in detail the Seq2Seq model which is known as a start-of-the-art NN conditional language model. In this section, we will connect the dots and tackle the problem of making Seq2Seq expressing a specific emotion e while generating a coherent response Y to any given source sequence X.

This section is organized according to two of our published papers [34] and [33]. The first paper tries to tackle the problem using three basic models whereas the the second paper extends the work with other four models. Therefore, we will construct this section as follows: in Section 3.4.1, we will cover three models presented in the first paper as mention above and show some examples and results. In Section 3.4.2, we will firstly explain other four models that are also able to handle this task. Then, we will compare all of the seven models and show our perspectives on this task.

3.4.1 Three Basic models

Enc-bef and Enc-aft Our first attempts are inspired by Google's multilingual neural machine translation system [35]. Generating different types of emotional responses can be an analogy to translating the same sentence into different languages. The implementation is straight forward; we make each emotion a single token and concatenate it with the input X so that our model has the target of minimizing $-\log P(Y|X')$, where $X' = \text{Concat}(e, X)$.

This approach reduces the two individual inputs into one so that they can be trained on normal Seq2Seq models. Further more, we consider the concate



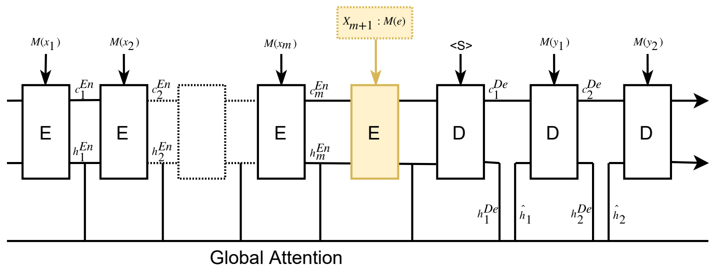


Figure 3.6: Graphic explanation of the Enc-aft model

Both of the methods are embedding the desired emotion into an encoder. We name them *Enc-bef* and *Enc-aft*, respectively as shown in the Figure 3.5 and 3.6. e is the emotion of the generated response and is obtained from Y by an emotion mining classifier. Both models require to change the length of the source sequence X from m to $m+1$.

Dec-rep Li et al. proposed a modified Seq2Seq model that allows models to learn the speaking styles of different people from a movie script corpus. Our third approach adapts their idea but instead of embedding people/speaker into the decoder, we feed the emotion vectors v_e during the decoding. Equation (5

D

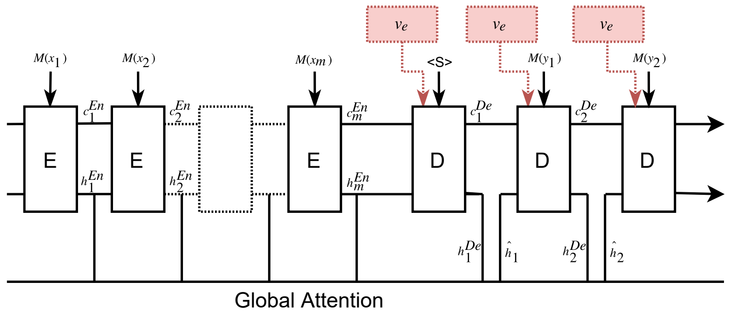


Figure 3.7: Graphic explanation of the Dec-rep model

Datasets To train the dialogue models, we use the OpenSubtitles dataset [51]. Precisely, we use the pre-processed data by [46] and further removed duplicates. The total amount of utterances is 11.3 million, each utterance has a minimal length of 6 words.

Since there is no existing dialogue data set labeled with emotions, we trained our own emotion classifier (Chapter 2) to tag the corpus. We use the CBET dataset[1] [79, 96], it contains 9 emotions and 81k instances. Each

[1]`http://webdocs.cs.ualberta.ca/~zaiane/data/CBET`

instance is labeled with up to two emotions. The emotions are *anger, surprise, joy, love, sadness, fear, disgust, guilt*, and *thankfulness*. We train a bidirectional LSTM [26] model and achieve an F1-score of 52.9% with precision 68.4% and recall 49.1% on the 9 emotions. To tag the target utterances with higher confidence, we use a threshold to separate those utterances that do not express emotion. 34.01% are thus labeled as Non-emotion. "Non-emotion" is treated as a special emotion when training the dialogue models, but it is not considered in the evaluation.

source	what is she doing here ?
target	how do i open this god forsaken window !
anger *Enc-bef*	she 's going to kill herself
anger *Enc-aft*	she 's going to kill you
anger *Dec-rep*	she 's supposed to be in the house
fear *Enc-bef*	i 'm afraid i can 't tell you
fear *Enc-aft*	she 's trying to kill herself
fear *Dec-rep*	i 'm afraid she 's not here
love *Enc-bef*	she wants to see you in the bedroom
love *Enc-aft*	she 's in love with you
love *Dec-rep*	she wants to be with you
disgust *Enc-bef*	she 's not allowed to leave
disgust *Enc-aft*	she 's going to be sick
disgust *Dec-rep*	she 's the one who raped me

Table 3.1: Examples of generated responses by the three models with emotions *anger*, *fear*, *love* and *disgust*.

Experiments setup With the purpose of comparison, the parameters of the three models are set to be the same. The dimensions of LSTM hidden units are set to 600. Adam optimizer [40] with learning rate of 0.0001 is used. The size of the vocabulary space is set to 25,000, which is the same as that in [46]. We also use *FastText* [36] pre-trained word embedding which is shared by the LSTMs in both encoder and decoder and set to trainable. We held out 50k samples from the whole dataset as test set. 95% of the remaining is used to

source	i didn 't realize you were here
target	maybe i should leave so you can continue
joy *Enc-bef*	i 'm here to make a phone call
joy *Enc-aft*	i 'm so happy for you
joy *Dec-rep*	i was just in the garden house
sadness *Enc-bef*	i thought you were gonna be here
sadness *Enc-aft*	she 's trying to kill herself
sadness *Dec-rep*	i thought i 'd be here
guilt *Enc-bef*	i 'm sorry i didn 't
guilt *Enc-aft*	i 'm sorry i didn 't know you were here
guilt *Dec-rep*	i 'm sorry i didn 't hear you
surprise *Enc-bef*	i 'm here to find out
surprise *Enc-aft*	i thought you were going to be here
surprise *Dec-rep*	i thought you might be here

Table 3.2: Examples of generated responses by the three models with emotions *joy*, *sadness*, *guilt* and *surprise*.

train the dialogue models, and 5% of it is used for evaluation and preventing overfitting.

Accuracy of expressed emotions In this research, we tackle the problem of training a generative model that can respond while expressing a specific emotion. Unlike the work by [46], expensive human evaluation is not needed. Instead, we evaluate the output using an emotion mining classifier to see whether the intended emotion is among the detected ones. For each input utterance, we let the model generate responses for each of the 9 emotions and check, using the emotion classifier, which emotion is indeed expressed in the output. Hence, the emotions' accuracies of the generated responses are estimated by the emotion classifier. Different from the procedure of tagging, where we put a threshold to enforce a higher precision, the most possible emotion is chosen in the evaluation. The results are shown in Table 3.3.

Emotion	*Enc-bef*	*Enc-aft*	*Dec-rep*
anger	60.34%	62.44%	68.24%
fear	89.34%	86.46%	87.52%
joy	45.76%	41.36%	48.53%
love	56.96%	55.32%	59.13%
sadness	94.16%	93.93%	94.22%
surprise	84.46%	85.11%	87.22%
thankfulness	87.89%	89.51%	91.06%
disgust	78.06%	76.94%	79.01%
guilt	93.25%	92.16%	91.22%
Average	76.69%	75.91%	78.46%

Table 3.3: Per class accuracy of generated response

Tables 3.1 and 3.2 display examples of generated responses, according to different emotions, given a source utterance extracted from the test set. In the two tables, "source" represents human's message which is given to the chatbot, and "target" shows the corresponding actual response in the test set. We can observe that the generated text is: (1) related to the source text; (2) expresses the desired emotions. For instance, when responding to "What is she doing here?", the generated text employs "she" rather than "he". The models are also able to express the emotion of *fear* by generating the word "afraid". When instructed to respond to the previous utterance "I didn't realize you were here", and to express *guilt*, all the models are able to generate "I am sorry". In terms of semantics, while the source is mentioning "here", the *Dec-rep* model is able to answer with "I was just in the garden" which remains coherent with the location context.

(3) We also notice that the "target" sentences are not very intuitive, or even out of context. This also indicates that the opensubtitles dataset is noisy and perhaps is not the best choice for training a dialogue generation model.

Since increasing the diversity is not the target of this work, our models also suffer from this common problem of Seq2Seq models. Similar to generating "I don't know" regardless of source sentences, in Seq2Seq models [47, 76, 85], our model tends to generate "I <unk>l be back in a minute" for emotion *anger*. The diversity of words that are used for each emotion are low, e.g., generations for emotion *fear* often have the word "gun" and the responses of emotion "sadness" often start with "I don't want ". This is clearly a side effect from our training data.

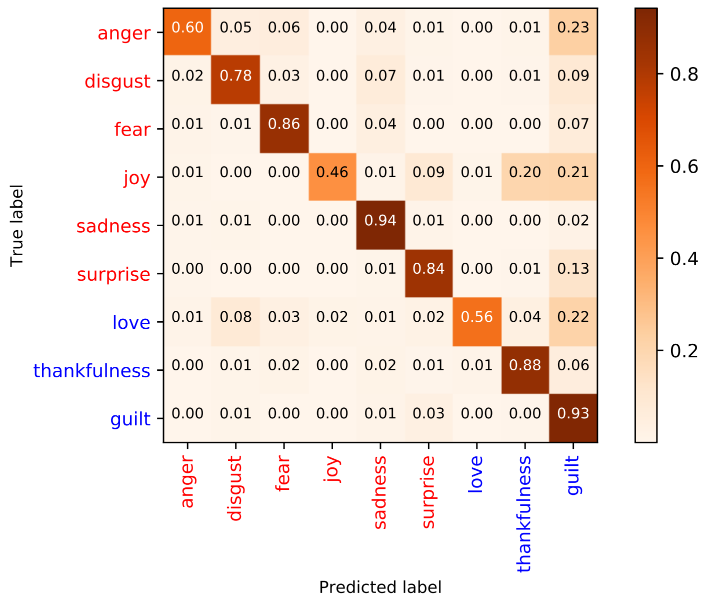


Figure 3.8: Confusion matrix of model *Enc-bef*

Results analysis From Table 3.3, we can observe that *Dec-rep* has better overall average accuracies than *Enc-bef* and *Enc-aft*. The average accuracies of *Enc-bef* and *Enc-aft* are very close. However, we notice some discrepancies in the individual emotions' accuracies. For instance, *fear* is better captured by *Enc-bef*, while *anger* has a much better accuracy for *Dec-rep*.

To further inspect the results, we also show the normalized confusion matrix of each model respectively, as in Figure 3.8, 3.9 and 3.10. We can notice obvious dark colored diagonals for the three figures. This indicates that all the three proposed models, indeed, have the ability to generate responses with given emotions. From these figures, we find that models tend to generate the responses with *guilt* regardless of the desired emotion. All the three models tend to generate *thankfulness* while they were instructed to express *joy*.

The patterns of confusion matrices of model *Enc-aft*, *Enc-bef* and *Dec-rep* are close to each other. However, *Dec-rep* model has a slightly better overall

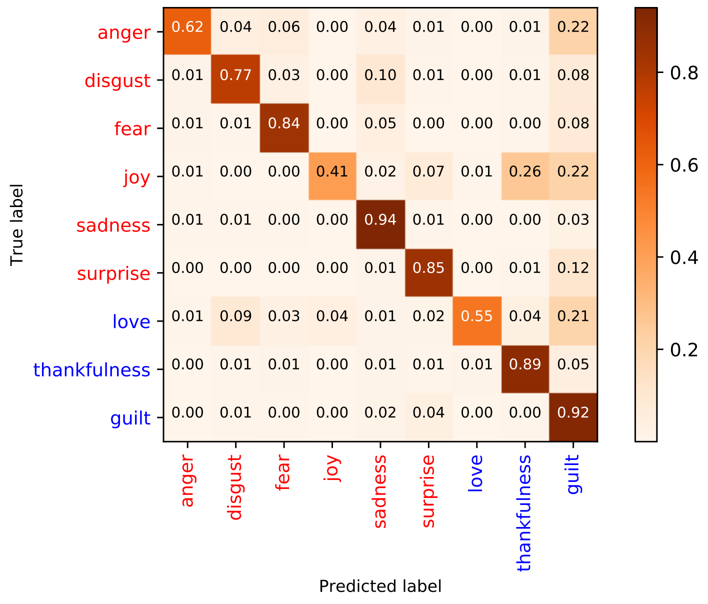


Figure 3.9: Confusion matrix of model *Enc-aft*

performance.

3.4.2 Four Extended models

As mentioned above in Section 3.3, general Seq2Seq models are learning the probability of $\mathcal{P}(Y|X)$. While in the task of controlling responses by an instructed emotion, each (X, Y) pair is assigned with an additional desired response emotion e. We have also explained that the goal of designing any models for this task should take the emotion e as an additional input so that it can learning the conditional probability distribution of $\mathcal{P}(Y|X, e)$.

We propose three models (*Enc-bef*, *Enc-aft* and *Dec-rep*) in Section 3.4.1. *Enc-bef* and *Enc-aft* are models that inject an emotion e in the encoder by putting special tokens before or after the input sequence X. The *Dec-rep* model, on the other hand, puts e at each decoding step, which is similar to the method in [46]. In this section, we present four additional models and compare them together and consider the three models in Section 3.4.1 as baseline models.

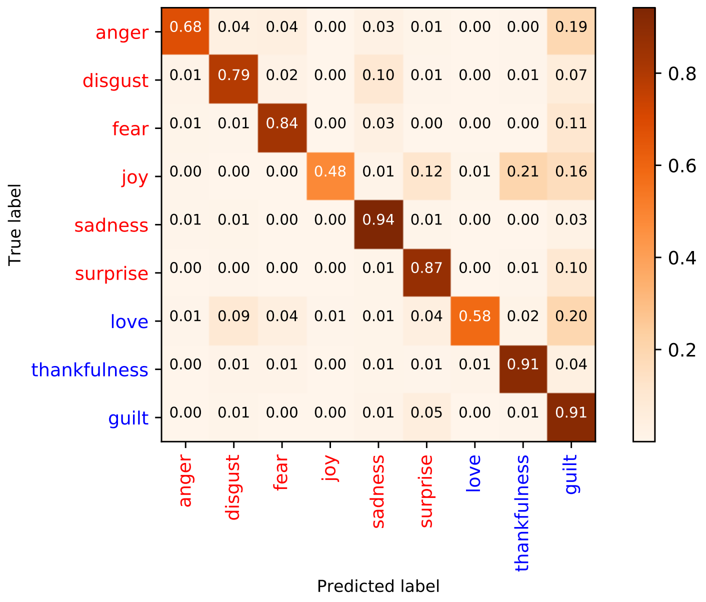


Figure 3.10: Confusion matrix of model *Dec-rep*

D

ce

tw

ar

fe

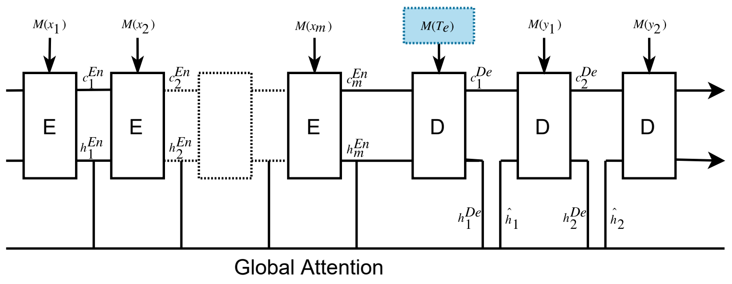


Figure 3.11: Graphic explanation of the Dec-start model

In this Dec-start model, we simply substitute the start token <s> with an emotion token T_e to generate h_1^D and the rest of the recursive computation will be the same as the normal Seq2Seq model with global attention mechanism. The generation of h_1^D is shown in Equation (3.25).

$$h_1^D = LSTM^D\Big(M(T_e), \big[h_m^E, c_m^E\big]\Big) \tag{3.25}$$

Dec-trans As an alternative, we multiply the h_t^D with another matrix to transform the hidden state of time t with respect to the emotion to be expressed. Denote the transforming matrix as $Trans_e$, then the recursive decoding

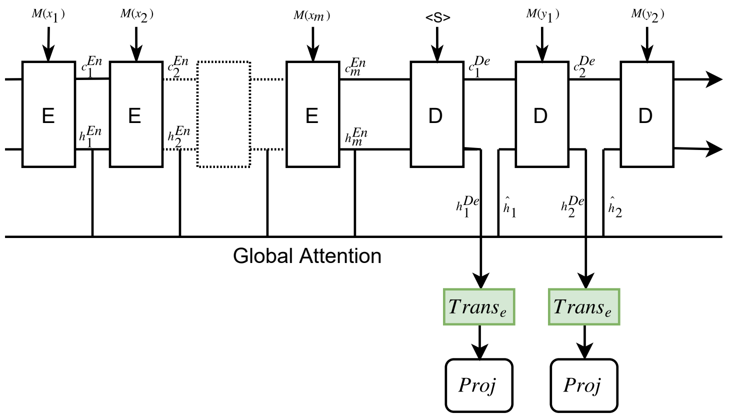


Figure 3.12: Graphic explanation of the Dec-trans model

Figure 3.12 shows the model.

Dec-proj The work in [103] also proposed an external memory which maps the hidden state h_t^D into a slightly different vocabulary space for each of the emotions. By taking a step forward, we propose a *Dec-proj* model which will make h_t^D to totally independent vocabulary spaces. This is done by making unique projection layer $Proj_e$ for each of the emotion. The model is shown in Fig 3.13. Equation 3.21 is thus changed to the following:

$$\hat{y}_t = \arg\max\Big(\text{softmax}\big(Proj_e(\tilde{h}_t^D)\big)\Big) \tag{3.28}$$

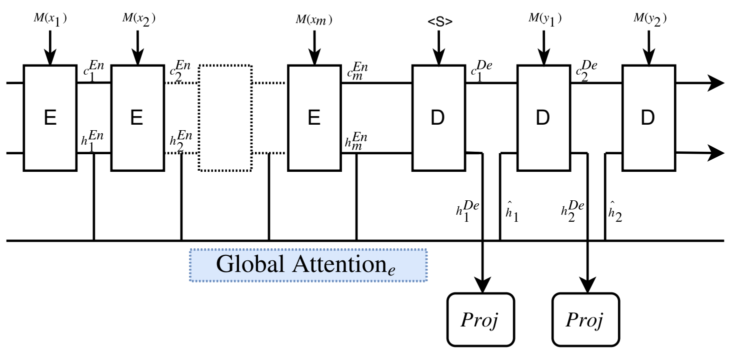


Figure 3.13: Graphic explanation of the Dec-proj model

Enc-att The attention mechanism has been proven to be very powerful in many sequence to sequence tasks. The methods proposed in [90] further outperforme many tasks by using an attention only encoder-decoder model. Luong, Pham, and Manning [52] proposed three methods to calculate the attention score. The one we chose in Equation 3.18 is referred to as *general* score in their paper. It is a parameterized method compared to *dot* score. Since the general attention has individual parameters, making different attention layer for different emotion is possible as well. The parameter of the *attention layer* with *general* scores has two parameters: W_c in the Equation 3.15 which softly combines the CTX_t and h_t^D; W_a in the Equation 3.19. Therefore, given emotion e, the *Dec-att* model changes Equation 3.15 to

$$\tilde{h}_t^D = \tanh\left(W_c^e\left[CTX_t; h_t^D\right]\right) \tag{3.29}$$

And also change the computation of the attention scores in Equation 3.19 to

$$\operatorname{score}\left(h_t^D, h_i^E\right) = \begin{cases} {h_t^D}^\top h_i^E & \textit{dot} \\ {h_t^D}^\top W_a^e h_i^E & \textit{general} \\ v_a^{e\top} \tanh\left(W_a^e\left[h_t^D; h_i^E\right]\right) & \textit{concat} \end{cases} \tag{3.30}$$

Compared to the original equation, for each of the given emotion e, the trainable matrix W_a is replace with W_a^e, W_c is changed to W_c^e, and v_a is substituted with v_a^e. Figure 3.14 show the model on an abstract level, it

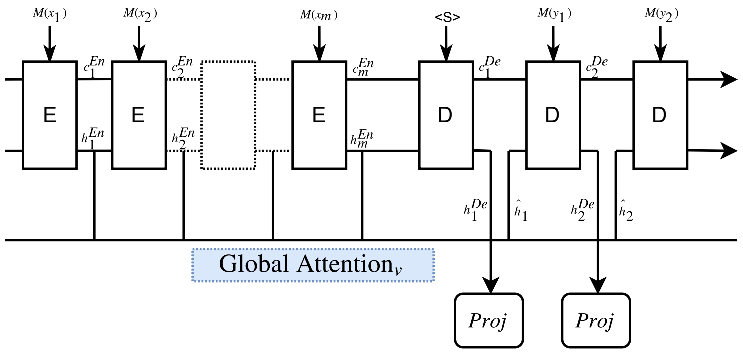


Figure 3.14: Graphic explanation of the Dec-att model

Results analysis In the work of [33], we propose four models that are able to automatically generate a response while conveying a given emotion. We compare our models with the baseline models in [34] in terms of both performance and efficiency. Our *Enc-att* model outperforms the strongest baseline and we show how it works using attention heatmaps. *Dec-rep* and *Enc-att* turn out to be both effective and efficient.

However, in this work, we did not experiment with any combinations of the models. It is shown in [103] that the combination of external and internal memory outperforms each of the single model. We think the combination of *Dec-rep* and *Enc-att* has a potential to give a better result. One major limitation of this work is that we heavily rely on the accuracy of the emotion mining classifier and assume it is of acceptable accuracy. Moreover, the main effort of this research lies on generating responses accurately and efficiently but without focusing on properties like grammar, relevance and diversity.

The *estimated accuracy* scores of the 7 models are shown in Table 3.4. Moreover, we draw the confusion matrices of the 4 extended models to show the misclassification errors (Figure. 3.15, 3.16, 3.18, 3.17).

From the table, we can see that despite the fact that the *Enc-att* model only achieves 38.71% accuracy for the emotion *joy*, it still outperforms the

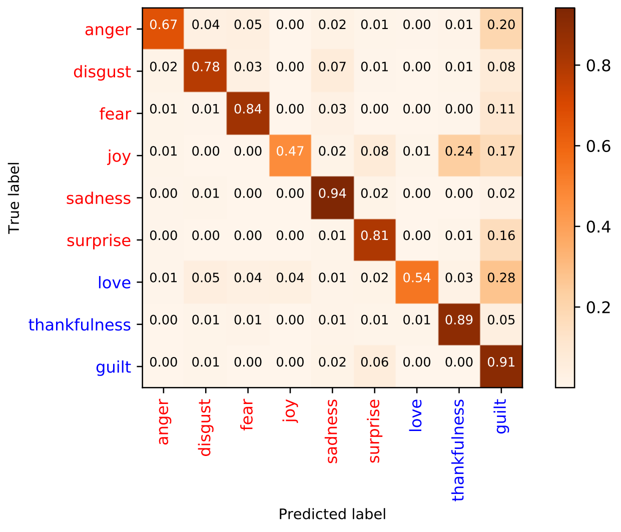


Figure 3.15: Confusion matrix of model *Dec-start*

Table 3.4: Per class accuracy of generated response

Emotion	*Enc-bef*	*Enc-aft*	*Dec-rep*	***Dec-start***	***Dec-trans***	***Dec-proj***	***Enc-att***
anger	60.18%	62.30%	67.95%	66.81%	64.27%	**78.48**%	65.09%
disgust	77.98%	76.79%	79.02%	78.42%	78.33%	**86.43**%	78.29%
fear	**86.40**%	84.17%	83.52%	84.10%	77.15%	73.70%	86.00%
joy	45.69%	41.15%	48.30%	47.42%	49.69%	**59.12**%	38.71%
sadness	94.19%	93.98%	94.21%	94.18%	88.42%	89.83%	**95.09**%
surprise	84.47%	85.09%	87.21%	80.55%	83.61%	80.56%	**92.54**%
love	56.38%	54.69%	58.32%	54.25%	62.82%	**85.14**%	64.56%
thankfulness	87.69%	89.31%	**90.83**%	89.44%	82.03%	61.80%	89.11%
guilt	93.19%	92.17%	91.20%	90.68%	86.64%	50.92%	**94.40**%
Average	76.24%	75.52%	77.84%	76.21%	74.77%	74.00%	**78.20**%

others on most of the emotions. From Fig. 3.17, one can observe a significant mismatch between *joy* and *thankfulness*. Instead of expressing *joy*, *Enc-att* conveys *thankfulness* which could also be considered reasonable. However, *love* is also confused with *guilt*. Note that the measured accuracy is also subject to the accuracy of the emotion tagger used.

It is also noticeable that the performance of model *Dec-start* is close to that of models *Enc-bef* and *Enc-aft*. This is expected considering the models are simply injecting the information of emotions by only one special token. The highlighted numbers in Table 3.4 show the best accuracy of each emotion. Model *Dec-rep*, *Dec-proj* and *Enc-att* have at least 2 best scores whereas the others almost have none.

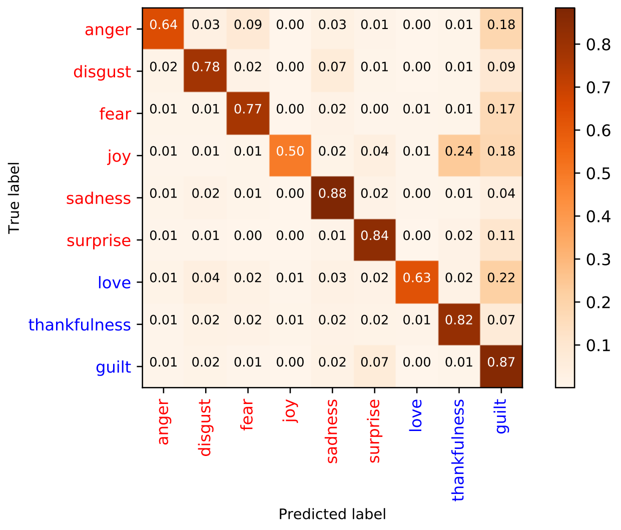


Figure 3.16: Confusion matrix of model *Dec-trans*

To compare the extended emotion model with Ekman's basic emotions, we highlight the two group of emotions in the figures of confusion matrices and Table 3.4. The emotions in red are the six basic emotions, the blue ones are those added by [96].

Enc-att model visualization To show how the *Enc-att* model works, we chose an example utterance and show how the attention scores vary with respect to responses with different emotions. The attention is visualized by heatmaps in Fig. 3.19. To respond to the utterance "You scared me today at the hotel", the model focused on "scared" when expressing "fear". When conveying "guilt", except for focusing on the pronoun "you", it focused on "me" and "today" and show a strong preference to using the word "sorry". When responding with "joy", it focused on the word "hotel". Interestingly, to response to the utterance with "sadness, the model did no pay attention to any words except for the pronoun, but it did try to answer with the phrase "little bit more".

Parameter cost Apart from the performance of the models, another important comparison of deep learning models is their sizes. Considering that all the 7 models are based on the basic Seq2Seq with attention model. We

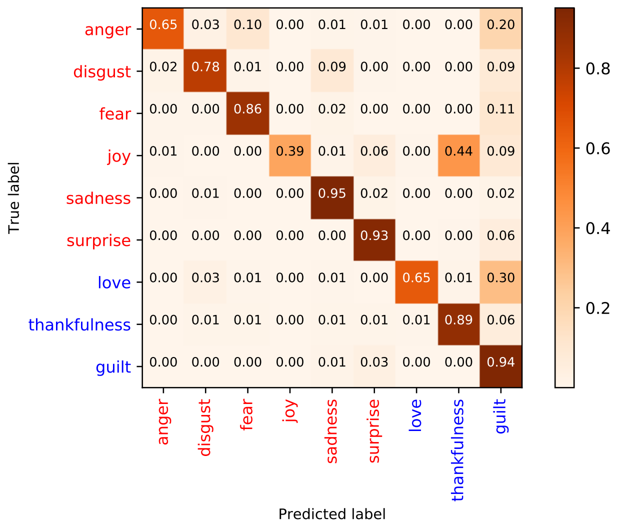


Figure 3.17: Confusion matrix of model *Enc-att*

Table 3.5: Comparison of the models in terms of additional space for required parameters

Model	Additional para. in symbols	Additoinal para. in our exp.
Enc-bef	0	0
Enc-aft	0	0
Enc-att	$m \times D \times S$	180,000
Dec-rep	$D \times S$	6,000
Dec-start	0	0
Dec-trans	$D \times D \times S$	3,600,000
Dec-proj	$\|V\| \times D \times S$	150,000,000

only need to compare the additional parameters that are needed. Let's denote the size of vocabulary space as $|V|$, the length of source sentences as m, the dimension of the decoder LSTM as D, and the number of emotions as S. The comparison of the models in terms of these parameters is shown in Table 3.5. It has to be mentioned that S in our experiments is 10: 9 emotions plus an *non-emotion* category.

From the above table, *Dec-proj* is the least cost efficient model. *Dec-rep* and *Enc-att* are both outperforming models considering their performance.

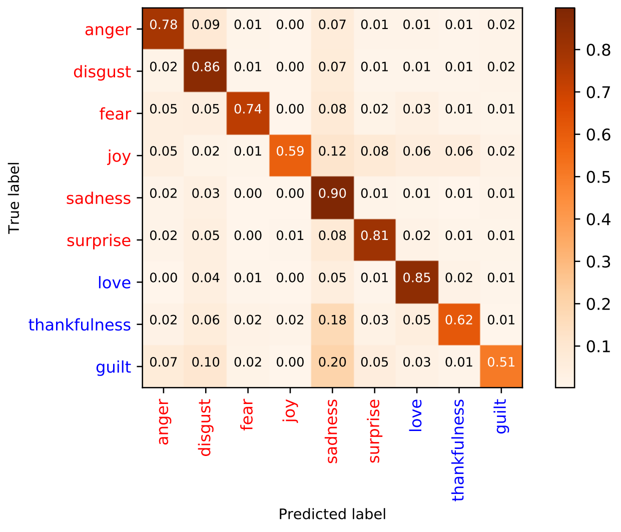
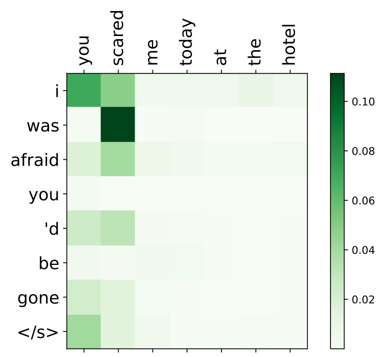


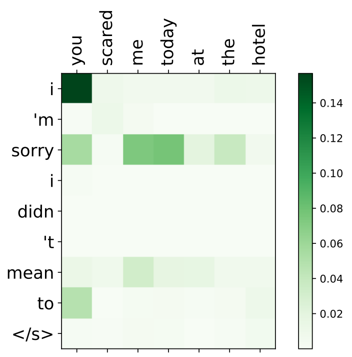
Figure 3.18: Confusion matrix of model *Dec-proj*

3.5 Conclusion

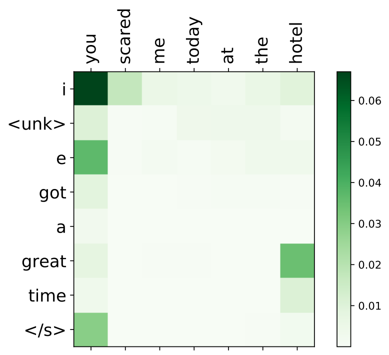
In this chapter, we introduced 7 models that are able to express different types of emotions. Emotional intelligence is the ability to monitor interlocutor's emotions and in turn appropriately express emotions in response. In our case, monitoring emotions in utterances is done using an emotion mining classifier. We assume that given some mapping rules, we can decide to express a specific emotion in the response. For instance if the message expresses sadness, the response could express compassion or surprise depending upon the context. The work presented herein focuses solely on generating a response that expresses a given desired emotion, and assumes the emotion to be expressed is given via these mapping rules. However, one could automatically learn the emotion to express given the emotion in the message directly from the data by changing the input message-response pairs (X, Y) into $((X, e_X), (Y, e_Y))$ where e_X is the emotion in the message and e_Y is the emotion in the response. In this chapter, we show that it is indeed possible to generate fluent responses that express a desired emotion. We present seven models to do so. Despite the differences among the models, they are all trained towards minimizing $-\log \mathcal{P}(Y|X, e)$ and all converge. The expression of some emotions (*guilt*, *sadness* and *thankfulness*) even reach accuracies over the 90%.

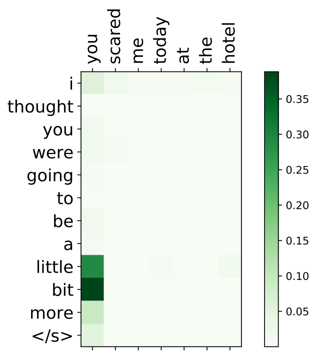


(a) fear



(b) guilt





13/05/2019 attentionLSTMSeq2Seq.html

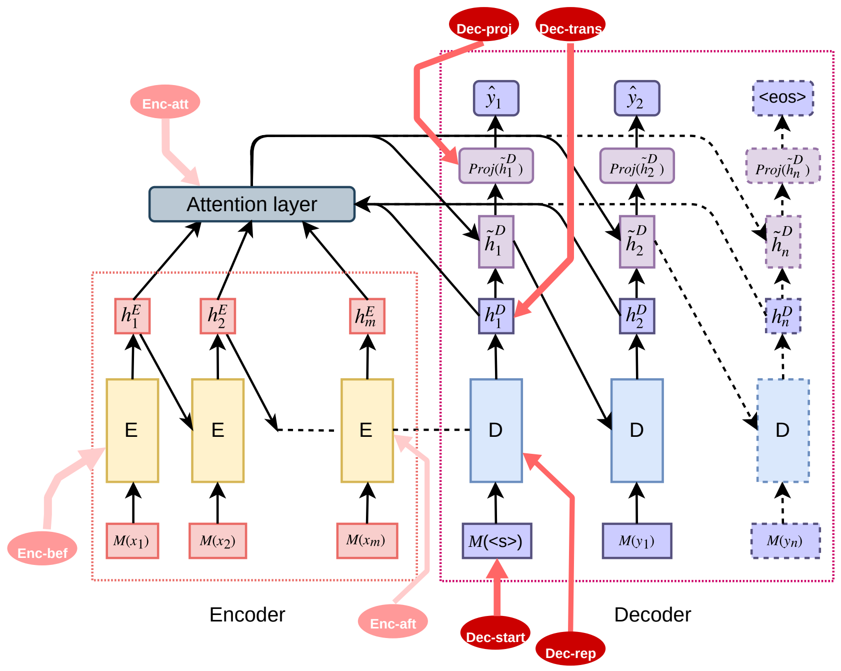


Figure 3.20: A summarization of the 7 proposed models

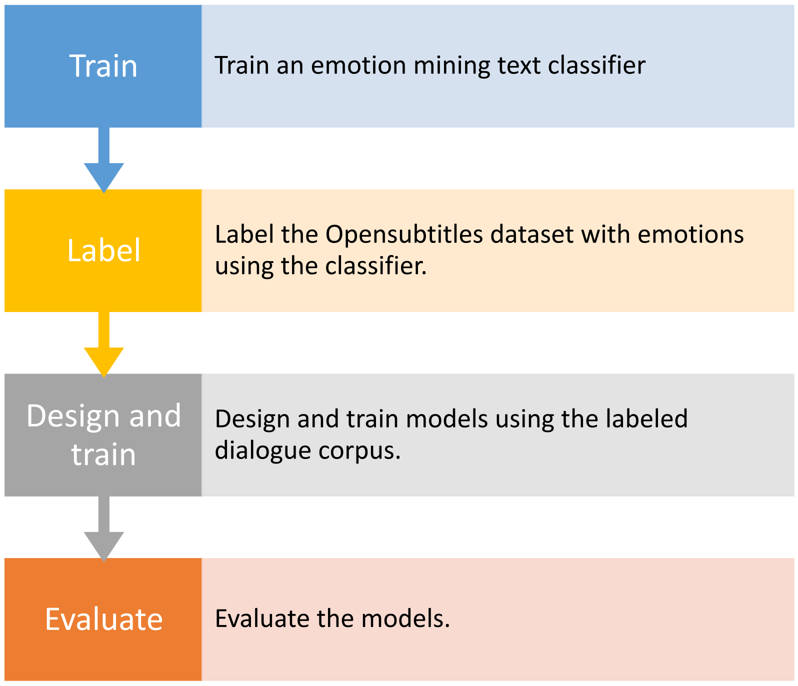

Train
Train an emotion mining text classifier
Label
Label the Opensubtitles dataset with emotions using the classifier.
Design and train
Design and train models using the labeled dialogue corpus.
Evaluate
Evaluate the models.

Figure 3.21: The pipeline of the models in this chapter

In our early experiments, we tagged each of the target utterance with the most possible emotion regardless of its confidence, wrongly assuming that all target utterances have a significant emotion. Although, our generative models can still be forced to produce the desired emotions, the quality of the generated sentences in terms of expressed emotions is below what is presented in Table 3.3 where the utterances without emotions (below a certain threshold) were labeled by "Non-Emotion". This shows the importance of learning to express emotions only from the utterances that indeed strongly convey measurable emotions. The other sentences are still kept to contribute in building the language model. We believe that adding reasoning to the mix can further enhance the emotional intelligence of a conversational agent.

The significance of the improvement by simply adding a threshold suggests that performance of the proposed models is largely relying on the liability of the text classifier.

Furthermore, this framework can be summarized as an approach to generate text with different styles. For example, given information of movies, one can generate reviews with different sentiments based on the idea of this research.

Although the emotion to express must be given at inference time, some simple rules can be added to make it more real word like. Imagine a scenario, a user is expressing *sadness* frequently, we can force the model to response with *love* in order to comfort the human user.

Chapter 4

Future Work and Conclusion

In Chapter 2, we study the problem of detecting emotions and achieved favorable results with the help of DL models. We then tackle the problem of how to express a specific emotion in conversations in Chapter 3.

In this chapter, we first show some other potential research problems that could be conducted under the scope of "Emotion Intelligence of chatbots" in the future. In addition, we conclude the entire manuscript at the end of this chapter.

4.1 Other Potential Research

In Chapter 3, we only tackle the problem of single turn conversations. Each conversation session only contains one source sequence X and one target sequence Y. However, in order to anticipate a user's emotional reaction, we need to consider a more general situation. In Equation 4.1, we consider the emotions for both source utterances and target utterances are also in the context of a conversation session. Denote S be a the single session, we have

$$\begin{aligned} S = \big(X_1, e(X_1), Y_1, e(Y_1)\big), \big(X_2, e(X_2), Y_2, e(Y_2)\big), \cdots, \\ \big(X_N, e(X_N), Y_N, e(Y_N)\big) \end{aligned} \tag{4.1}$$

In the representation above, N is the number of exchanges for a dialogue session. Function $e : X \to y$ generate the emotion y expressed in a sequence of words X. e may represent either a emotion classifier, human annotation, or an empty element $\emptyset$ (meaning that the emotion of the utterance X is unknown).

4.1.1 Anticipate Human's Emotion

Recall the third question brought up in the *Introduction* of this manuscript: "Could a chatbot anticipate the emotions which user may express?" Here, we will try to answer this question with our preliminary experiments, but we first need a formal definition of this task.

Problem Definition In the Equation 4.1, there are two speakers who produce two sequences of utterances $(X_1, X_2, \cdots, X_N)$ and $(Y_1, Y_2, \cdots, Y_N)$ respectively. Assume X_i is spoken by the human and Y_i is the Conversational Agent's corresponding response. Since we are trying to anticipate the human's emotion, we need to predict $e(X_i)$ given the other information. Given the fact that the number of dialogue exchanges may vary for each session. We put a constraint on the depth of the conversation, and define a *K-turn emotion anticipation* problem as following: given

$$A_{K-1} = X_1, Y_1, X_2, Y_2, \cdots, X_{K-1}, Y_{K-1} \tag{4.2}$$

as context, the target of such task is to anticipate the emotion of human's emotional reaction $e(X_K)$ (we consider $e(X_K)$ as the ground truth here) without knowing the actual response X_K. The task can be also represented as learning the following conditional probability distribution:

$$P(e(X_K)|A_{K-1}). \tag{4.3}$$

Challenges To the best of our knowledge, we are among the first to address this task. We found that the task potentially has the following challenges:

- Human's emotion reaction may be subjective to one's personality, the mood (the emotional state while the conversation is occurring), age, education, and so on. Such dynamics may be very difficult to be quantified and hard to be learned by existing models. Applying Reinforcement Learning is a possible avenue for investigation.

- There are not too many public high-quality datasets which include both conversation and corresponding emotions.

Experiments We perform one preliminary experiment based on an existing dataset and the proposed model which we have which has achieved good performances in the task of *detecting emotions in conversation* (Section 2.9). More specifically, we apply the SemEval2019-Task3 dataset and the proposed CSLD model. However, in order to eliminate the affect of *covariate shift* (Section 2.9.1), we combine the *train*, *dev*, and *test* datasets and randomly split

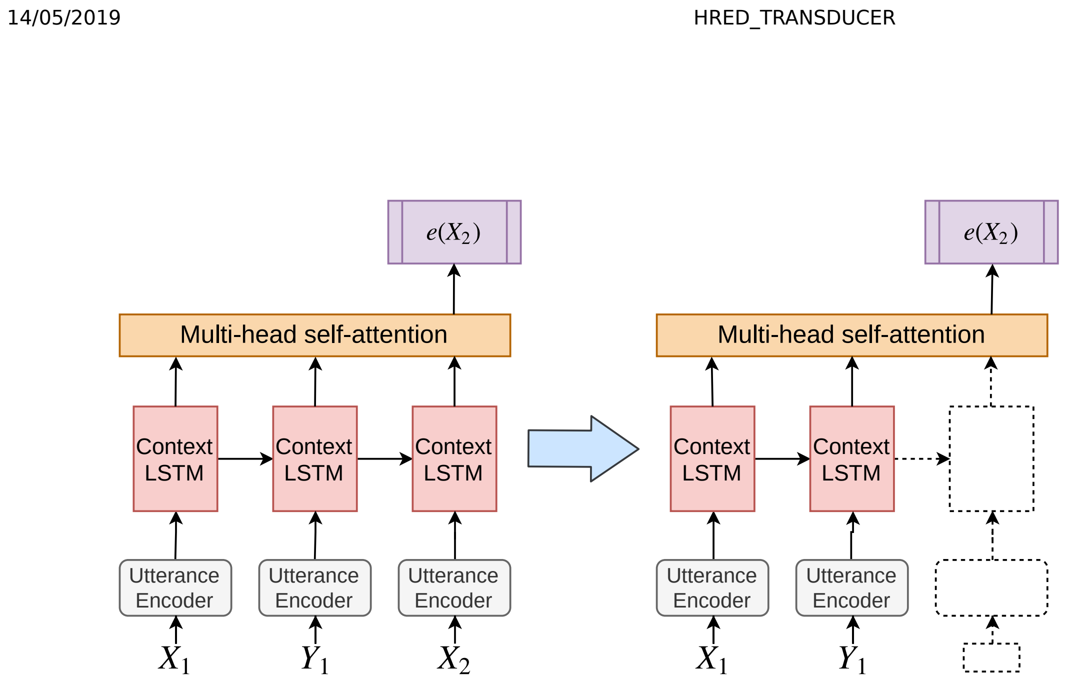


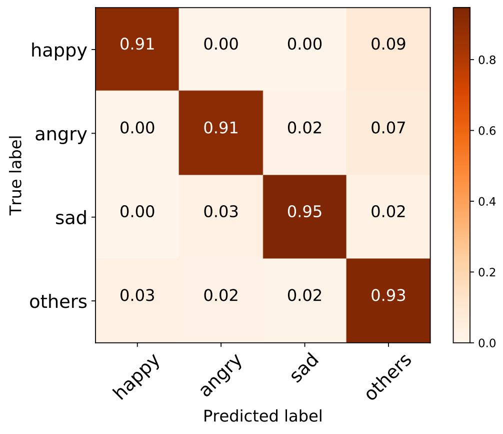
Figure 4.1: Illustration of applying the proposed CSLD model on the task of *2-turn emotion anticipation*.

The experiment is shown in Figure 4.1, the left side shows the task of emotion recognition in conversation using CSLD model which take as input the context (X_1, Y_1, X_2) to predict $e(X_2)$. While on the right side, it tries to predict emotion of $e(X_2)$ without X_2 as input, which is a task of *2-turn emotion anticipation*. In this experiment, although we do not explicitly take as input the emotions of the utterances in the context A_{K-1}, CSLD does have a DeepMoji module which can be regarded as an estimator of $e(X_i)$ and $e(Y_i)$.

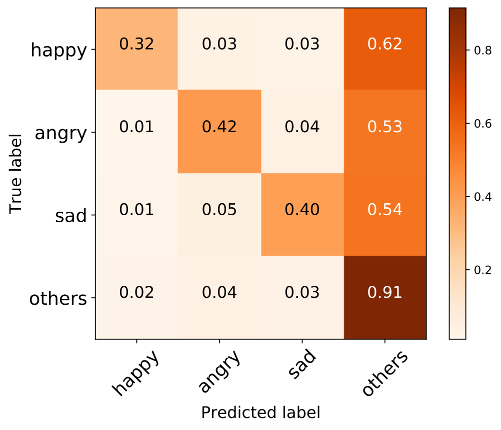
For simplicity, we refer the two tasks in Figure 4.1 as *detection* and *anticipation* respectively. The classification performance is shown in Table 4.1, and the misclassification is visualized as two confusion matrices in Figure 4.2.

Task	Macro P.	Macro R.	Macro F1	Micro F1
Detection	0.9100	0.9257	0.9176	0.9297
Anticipation	0.6926	0.5131	0.5625	0.6872

Table 4.1: Comparison of the performance of the emotion detection task and the emotion anticipation task on SemEval2019-Task3 dataset.



(a) Emotion detection task



(b) Emotion anticipation task

Figure 4.2: Comparison of the confusion matrices of the detection task and the anticipation task

We compare the results of both tasks to show how much accuracy can be lost if the actual user response X_K is missing from the input information. From the results that are shown in Figure 4.2 and Table 4.1, two conclusions can be drawn confidently:

- The accuracy drop is significant.
- Figure 4.2 shows that the results of emotion anticipation has a strong pattern and definitely better than an complete random generator.

We also observe that, without the input X_K, there are still barely any misclassifications among the three emotion categories (happy, angry, sad). The error mostly occurs between each of the emotion category and the "others".

The preliminary results indicate that:

Anticipate human's emotion is possible but may not be as accurate as emotion recognition.

However, we only test it on the *2-turn emotion anticipation* task without having additional input such as the personality and other information of the human speaker.

There are some other datasets that can be applied to this task. For example, *Emotionlines* [31] dataset has two sub-datasets: *Friends* (collected from the TV serious of the same name) and *EmotionPush* (collected from Facebook messages). They both contain human annotated emotions on the conversation exchanges. *DailyDialog* dataset [49] is another human annotated dataset of conversations with annotated emotions. In our early experiments, the annotations in *DailyDialog* (from websites for English learners) is of higher quality than that of *Emotionlines* datasets. We can make subsamples of each of the three datasets according to the length of the context K, and perform similar experiments as shown above. However, there are more emotion categories on the three datasets and the label distribution are extremely imbalanced. Meanwhile, the speakers of each utterance in the *Friends* dataset is known. Therefore, it is possible to gather more information for this datasets since the TV series is very popular and each character has a very significant personality. Hence, there can be more experiments on this route, but there will be more problems to conquer than what is shown in these preliminary experiments.

4.1.2 Respond to human with appropriate emotions

In Chapter 3, we tackle the problem of expressing any given emotion. Various models are proposed and the experiments suggest that it is possible to generate coherent responses with the emotion that we ask the model to generate. However, what is the emotion that a chatbot is supposed to express given the context in a dialogue session? This is an interesting yet undiscovered topic. In this section, we will provide a formal definition of this task and discuss potential challenges.

Task definition Consider a dialogue session

$$(X_1, e(X_1), Y_1, e(Y_1)), \cdots, (X_{K-1}, e(X_{K-1}), Y_{K-1}, e(Y_{K-1})),$$
$$(X_K, e(X_K), Y_K, e(Y_K))$$

where X_i is generated by a human and Y_i is by a CA. Assume that the CA wants to take the control of the conversation so that by responding Y_{K-1} to the human, it would not cause unpleasant which would be expressed in the following utterance X_K or its corresponding emotion $e(X_K)$.

To tackle this problem, we need another function which defines the degree of appropriateness. Denote the function as $\mathcal{R}$, it is measured by given X_K and $e(X_K)$ as input. The target of this task would be

$$\max \mathcal{R}(X_K, e(X_K)), \tag{4.4}$$

if the higher score of the function $\mathcal{R}$ indicates that the CA responds more appropriately.

However, both X_K and $e(X_K)$ are unknown by the time we need to make a decision on what $e(Y_{K-1})$ and Y_{K-1} should be. Therefore, approximations on X_K and $e(X_K)$ has to be made. Under this scenario, we have

$$A'_{K-1} = X_1, Y_1, X_2, Y_2, \cdots, X_{K-1} \tag{4.5}$$

as the context of this task (Y_{K-1} is an action to take). We further assume f and g are two the approximation functions for X_K and $e(X_K)$ respectively.

We thus have,

$$f([A'_{K-1}; e(Y_{K-1}); Y_{K-1}]) \sim X_K \tag{4.6}$$
$$g([A'_{K-1}; e(Y_{K-1}); Y_{K-1}]) \sim e(X_K) \tag{4.7}$$

Hence, the task of "responding to human with appropriate emotions" can be represented by the following optimization problem:

$$\underset{e(Y_{K-1}),Y_{K-1}}{\arg\max}\ \mathcal{R}\Big(f\big([A'_{K-1};e(Y_{K-1});Y_{K-1}]\big),g\big([A'_{K-1};e(Y_{K-1});Y_{K-1}]\big)\Big) \tag{4.8}$$

Challenges We find the problem given by Equation 4.8 very challenging:

- The assumption of the two approximators p and q is too strong. Although in the Section 4.1.1, we show that estimating $e(X_K)$ is possible given context A_{K-1}, the accuracy is low.
- The searching space of Y_{K-1} is very large because it could be any sequence of words.
- The appropriateness function $\mathcal{R}$ plays a very important rule here but its definition could be very subjective and might involve the knowledge from psychology.

4.2 Conclusion of the Manuscript

In this manuscript, we explore the task of building an empathetic CA. In Chapter 1, we describe the motivation of the research in this manuscript and briefly explain the questions we are trying to solve. We first work on detecting the emotions on the utterance level and extend the proposed model to conversation level. We advance the state-of-the-art of emotion detection on multiple datasets in Chapter 2.

Knowing the expressed in utterances, we move one step forward by designing several models that can express given emotions in generated response. In Chapter 3, we explore the problem by proposing seven models and extensive experiments. Additionally, we show some interesting results, make concrete analysis, and state our perspectives on the question.

In Chapter 4, we explore two more questions that are much less studied in the literature compared to that in the previous chapters. The two questions are: "Can we anticipate a human's emotion?" and "What are the appropriate emotions for a Conversational Agent to express?" We give formal definitions and state our insights by describing the potential challenges. In addition, we provide one experiment on the first question as a proof of concept and show that it can be done, but additional work is required to improve the accuracy.

There may be many other potential works that can be done for improving EQ of dialogue agents. We are among the first to tackle the problem with purely NN models. We believe that empathy is one of the essential requirements for developing any useful intelligent agent to interact with humans.

References

[1] Muhammad Abdul-Mageed and Lyle H. Ungar. "EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 718–728. DOI: `10.18653/v1/P17-1067`. 25

[2] *Amazon Alexa*. Apr. 2019. URL: `https://en.wikipedia.org/wiki/Amazon_Alexa`. 41

[3] Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. "DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis." In: *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. 2017, pp. 747–754. DOI: `10.18653/v1/S17-2126`. 37

[4] Yoshua Bengio et al. "A Neural Probabilistic Language Model." In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155. 14

[5] JM Bernardo et al. "Generative or discriminative? getting the best of both worlds." In: *Bayesian statistics* 8.3 (2007), pp. 3–24. 42

[6] Leo Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: `10.1023/A:1010933404324`. 12

[7] Peter F. Brown et al. "The Mathematics of Statistical Machine Translation: Parameter Estimation." In: *Computational Linguistics* 19.2 (1993), pp. 263–311. 44

[8] Rafael A. Calvo and Sunghwan Mac Kim. "Emotions in Text: Dimensional and Categorical Models." In: *Computational Intelligence* 29.3 (2013), pp. 527–543. DOI: `10.1111/j.1467-8640.2012.00456.x`. 8

[9] Ankush Chatterjee et al. "SemEval-2019 Task 3: EmoContext: Contextual Emotion Detection in Text." In: *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Minneapolis, Minnesota, 2019. 32

[10] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." In: *CoRR* abs/1412.3555 (2014). arXiv: `1412.3555`. 18

[11] CodaLab. *SemEval19 Task 3: EmoContext.* `https://competitions.codalab.org/competitions/19790#learn_the_details-data-set-format`. Feb. 2019. 37

[12] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. "Artificial Paranoia." In: *Artif. Intell.* 2.1 (1971), pp. 1–25. DOI: `10.1016/0004-3702(71)90002-6`. 1

[13] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: deep neural networks with multitask learning." In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008.* 2008, pp. 160–167. DOI: `10.1145/1390156.1390177`. 15

[14] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. 10, 44

[15] Terrance DeVries and Graham W. Taylor. "Learning Confidence for Out-of-Distribution Detection in Neural Networks." In: *CoRR* abs/1802.04865 (2018). arXiv: `1802.04865`. 39

[16] *Dialogue system.* Feb. 2019. URL: `https://en.wikipedia.org/wiki/Dialogue_system`. 40

[17] Amir Dinevari and Osmar Zaïane. "Automated nursing agent: A software agent for at-home elderly care." In: *Proceedings of the Eighth International Conference on eHealth, Telemedicine, and Social Medicine.* 2016. 2

[18] Paul Ekman, Wallace V Friesen, and Phoebe Ellsworth. *Emotion in the Human Face: Guide-lines for Research and an Integration of Findings.* Pergamon, 1972. 8, 9, 26

[19] *EQ definition and meaning.* URL: `https://www.collinsdictionary.com/dictionary/english/eq`. 42

[20] Nawshad Farruque et al. "Basic and depression specific emotion identification in Tweets: multi-label classification experiments." In: The 20th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), 2019. v

[21] Bjarke Felbo et al. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017.* 2017, pp. 1615–1625. 21

[22] Jonas Gehring et al. "Convolutional Sequence to Sequence Learning." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* 2017, pp. 1243–1252. 47

[23] Izhak Golan and Ran El-Yaniv. “Deep Anomaly Detection Using Geometric Transformations.” In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 2018, pp. 9781–9791. 39

[24] Yoav Goldberg. *Neural Network Methods for Natural Language Processing.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. DOI: 10.2200/S00762ED1V01Y201703HLT037. 43

[25] Edouard Grave et al. “Bag of Tricks for Efficient Text Classification.” In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers.* 2017, pp. 427–431. 14

[26] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. “Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition.” In: *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II.* 2005, pp. 799–804. DOI: 10.1007/11550907_126. 56

[27] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. “Speech recognition with deep recurrent neural networks.” In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013.* 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947. 13

[28] Kaiming He et al. “Deep Residual Learning for Image Recognition.” In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. 13

[29] Benjamin Heinzerling and Michael Strube. “BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages.” In: (2018). 15

[30] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. 17

[31] Chao-Chun Hsu and Lun-Wei Ku. “SocialNLP 2018 EmotionX Challenge Overview: Recognizing Emotions in Dialogues.” In: *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, SocialNLP@ACL 2018, Melbourne, Australia, July 20, 2018.* 2018, pp. 27–31. 76

[32] Chenyang Huang, Amine Trabelsi, and Osmar R. Zaïane. “ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and BERT.” In: *CoRR* abs/1904.00132 (2019). arXiv: 1904.00132. iv, 32

[33] Chenyang Huang and Osmar R. Zaïane. "Generating Responses Expressing Emotion in an Open-Domain Dialogue System." In: *Internet Science*. Springer International Publishing, 2019, pp. 100–112. DOI: `10.1007/978-3-030-17705-8_9`. iv, 53, 64

[34] Chenyang Huang et al. "Automatic Dialogue Generation with Expressed Emotions." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*. 2018, pp. 49–54. iv, 53, 61, 64

[35] Melvin Johnson et al. "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation." In: *TACL* 5 (2017), pp. 339–351. 54

[36] Armand Joulin et al. "FastText.zip: Compressing text classification models." In: *CoRR* abs/1612.03651 (2016). arXiv: `1612.03651`. 15, 56

[37] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. ISBN: 9780135041963. 8, 41

[38] S. Sathiya Keerthi and Chih-Jen Lin. "Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel." In: *Neural Computation* 15.7 (2003), pp. 1667–1689. DOI: `10.1162/089976603321891855`. 13

[39] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1746–1751. 10, 12, 13

[40] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014). 56

[41] A.C. Krendl and T.F. Heatherton. "Social Emotions: Imaging Social Emotions." In: *Reference Module in Neuroscience and Biobehavioral Psychology*. Elsevier, 2017. ISBN: 978-0-12-809324-5. DOI: `https://doi.org/10.1016/B978-0-12-809324-5.02489-5`. 9

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114. 13

[43] Siwei Lai et al. "Recurrent Convolutional Neural Networks for Text Classification." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* 2015, pp. 2267–2273. 13

[44] Kimin Lee et al. "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples." In: (2018). 39

[45] Jiwei Li, Will Monroe, and Dan Jurafsky. "Data Distillation for Controlling Specificity in Dialogue Generation." In: *CoRR* abs/1702.06703 (2017). arXiv: `1702.06703`. 2

[46] Jiwei Li et al. "A Persona-Based Neural Conversation Model." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers.* 2016. 44, 55–57, 60

[47] Jiwei Li et al. "Deep Reinforcement Learning for Dialogue Generation." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.* Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1192–1202. DOI: `10.18653/v1/D16-1127`. 1, 44, 58

[48] Jiwei Li et al. "Adversarial Learning for Neural Dialogue Generation." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2157–2169. DOI: `10.18653/v1/D17-1230`. 2, 44

[49] Yanran Li et al. "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset." In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers.* 2017, pp. 986–995. 76

[50] Zhouhan Lin et al. "A Structured Self-Attentive Sentence Embedding." In: (2017). 10, 19, 23, 35

[51] Pierre Lison and Jörg Tiedemann. "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles." In: 2016. 55

[52] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015.* 2015, pp. 1412–1421. 49–52, 63

[53] Jahnavi Mahanta. "Introduction to Neural Networks, Advantages and Applications." In: *Towards Data Science* (2017). 13

[54] Tomas Mikolov et al. "RNNLM - Recurrent Neural Network Language Modeling Toolkit." In: Dec. 2011. 15

[55] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* 2013, pp. 3111–3119. 14

[56] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* 2013, pp. 3111–3119. 15

[57] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space." In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.* 2013. 14, 15

[58] Mike Mintz et al. "Distant supervision for relation extraction without labeled data." In: *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore.* 2009, pp. 1003–1011. 24

[59] Andriy Mnih and Geoffrey E. Hinton. "Three new graphical models for statistical language modelling." In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007.* 2007, pp. 641–648. DOI: `10.1145/1273496.1273577`. 15

[60] Saif Mohammad. "#Emotional Tweets." In: *Proceedings of the First Joint Conference on Lexical and Computational Semantics, *SEM 2012, June 7-8, 2012, Montréal, Canada.* 2012, pp. 246–255. 24

[61] Saif Mohammad et al. "SemEval-2018 Task 1: Affect in Tweets." In: *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018.* 2018, pp. 1–17. 39

[62] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel.* 2010, pp. 807–814. 16

[63] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013.* 2013, pp. 1310–1318. 16

[64] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. 29

[65] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1532–1543. 14, 15

[66] Matthew E. Peters et al. "Deep Contextualized Word Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. 2018, pp. 2227–2237. 21, 44

[67] Robert Plutchik. "Emotions and psychotherapy: A psychoevolutionary perspective." In: *Emotion, psychopathology, and psychotherapy*. Elsevier, 1990, pp. 3–41. 8

[68] Lutz Prechelt. "Early Stopping-But When?" In: *Neural Networks: Tricks of the Trade*. 1996, pp. 55–69. DOI: `10.1007/3-540-49430-8\_3`. 29

[69] Kevin Quinn and Osmar R. Zaïane. "Identifying Questions & Requests in Conversation." In: *International C* Conference on Computer Science & Software Engineering, C3S2E '14, Montreal, QC, Canada - August 03 - 05, 2014*. 2014, 10:1–10:6. DOI: `10.1145/2641483.2641534`. 2

[70] Alec Radford et al. "Language Models are Unsupervised Multitask Learners." In: (2019). 44

[71] Alexander M. Rush, Sumit Chopra, and Jason Weston. "A Neural Attention Model for Abstractive Sentence Summarization." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 2015, pp. 379–389. 44, 49

[72] James A Russell. "A circumplex model of affect." In: *Journal of personality and social psychology* 39.6 (1980), p. 1161. 8

[73] Kashfia Sailunaz et al. "Emotion detection from text and speech: a survey." In: *Social Netw. Analys. Mining* 8.1 (2018), 28:1–28:26. DOI: `10.1007/s13278-018-0505-2`. 8

[74] Klaus R Scherer and Harald G Wallbott. "Evidence for universality and cultural variation of differential emotion response patterning." In: *Journal of personality and social psychology* 66.2 (1994), p. 310. 24

[75] Mike Schuster and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." In: *IEEE Trans. Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: `10.1109/78.650093`. 19

[76] Iulian Vlad Serban et al. "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.* 2016, pp. 3776–3784. 34, 58

[77] Iulian Vlad Serban et al. "A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 2017, pp. 3295–3301. 1

[78] Ameneh Gholipour Shahraki. "Emotion Mining from Text." PhD thesis. University of Alberta, 2015. 7

[79] Ameneh Gholipour Shahraki and Osmar R Zaïane. "Lexical and Learning-based Emotion Mining from Text." In: *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing.* 2017. 9, 24, 55

[80] Peter Shaver et al. "Emotion knowledge: further exploration of a prototype approach." In: *Journal of personality and social psychology* 52 6 (1987), pp. 1061–86. 9

[81] Xiaoyu Shen et al. "Improving Variational Encoder-Decoders in Dialogue Generation." In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 2018, pp. 5456–5463. 44

[82] Hava T. Siegelmann and Eduardo D. Sontag. "On the Computational Power of Neural Nets." In: *Journal of computer and system sciences* 50.1 (1995), pp. 132–150. DOI: `10.1006/jcss.1995.1013`. 16

[83] *Siri.* Apr. 2019. URL: `https://en.wikipedia.org/wiki/Siri`. 41

[84] Alessandro Sordoni et al. "A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion." In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015.* 2015, pp. 553–562. DOI: `10.1145/2806416.2806493`. 34

[85] Alessandro Sordoni et al. "A Neural Network Approach to Context-Sensitive Generation of Conversational Responses." In: (2015), pp. 196–205. 58

[86] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments - Introduction to Covariate Shift Adaptation.* Adaptive computation and machine learning. MIT Press, 2012. ISBN: 978-0-262-01709-1. 33

[87] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks.” In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada.* 2014, pp. 3104–3112. 5, 44, 45, 47, 50

[88] Jack V. Tu. “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes.” In: *Journal of Clinical Epidemiology* 49.11 (1996), pp. 1225–1231. ISSN: 0895-4356. DOI: `https://doi.org/10.1016/S0895-4356(96)00002-9`. 13

[89] Yi-Lin Tuan and Hung-yi Lee. “Improving Conditional Sequence Generative Adversarial Networks by Stepwise Evaluation.” In: *IEEE/ACM Trans. Audio, Speech & Language Processing* 27.4 (2019), pp. 788–798. DOI: `10.1109/TASLP.2019.2896437`. 44

[90] Ashish Vaswani et al. “Attention is All you Need.” In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 6000–6010. 35, 47, 63

[91] Oriol Vinyals and Quoc V. Le. “A Neural Conversational Model.” In: *CoRR* abs/1506.05869 (2015). arXiv: `1506.05869`. 44

[92] Richard S. Wallace. “The Anatomy of A.L.I.C.E.” In: *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. Ed. by Robert Epstein, Gary Roberts, and Grace Beber. Dordrecht: Springer Netherlands, 2009, pp. 181–210. ISBN: 978-1-4020-6710-5. DOI: `10.1007/978-1-4020-6710-5_13`. 1

[93] Shuohang Wang and Jing Jiang. “Machine Comprehension Using Match-LSTM and Answer Pointer.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017. 44

[94] Wenbo Wang et al. “Harnessing Twitter ”Big Data” for Automatic Emotion Identification.” In: *2012 International Conference on Privacy, Security, Risk and Trust, PASSAT 2012, and 2012 International Confernece on Social Computing, SocialCom 2012, Amsterdam, Netherlands, September 3-5, 2012.* 2012, pp. 587–592. DOI: `10.1109/SocialCom-PASSAT.2012.119`. 24

[95] Joseph Weizenbaum. “ELIZA - a computer program for the study of natural language communication between man and machine.” In: *Commun. ACM* 9.1 (1966), pp. 36–45. DOI: `10.1145/365153.365168`. 1

[96] Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R. Zaïane. “Current State of Text Sentiment Analysis from Opinion to Emotion Mining.” In: *ACM Comput. Surv.* 50.2 (2017), 25:1–25:33. DOI: `10.1145/3057270`. 9, 24, 55, 66

[97] Zichao Yang et al. “Hierarchical Attention Networks for Document Classification.” In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. 2016, pp. 1480–1489. 10

[98] Jun Yin et al. “Neural Generative Question Answering.” In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 2016, pp. 2972–2978. 44

[99] Lantao Yu et al. “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.” In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 2017, pp. 2852–2858. 44

[100] Han Zhao, Zhengdong Lu, and Pascal Poupart. “Self-Adaptive Hierarchical Sentence Model.” In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 2015, pp. 4069–4076. 13

[101] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. “Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders.” In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 2017, pp. 654–664. DOI: `10.18653/v1/P17-1061`. 44

[102] Chunting Zhou et al. “A C-LSTM Neural Network for Text Classification.” In: *CoRR* abs/1511.08630 (2015). arXiv: `1511.08630`. 10

[103] Hao Zhou et al. “Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory.” In: *Proc. of the AAAI Conference on Artificial Intelligence*. 2018. 62, 64

[104] Li Zhou et al. “The Design and Implementation of XiaoIce, an Empathetic Social Chatbot.” In: *CoRR* abs/1812.08989 (2018). arXiv: `1812.08989`. 1, 42