Exploiting Unsupervised Data for Emotion Recognition in Conversations

Wenxiang Jiao[†] **Michael R. Lyu**[†] **Irwin King**[†]
[†] Department of Computer Science and Engineering
The Chinese University of Hong Kong, HKSAR, China
[†] {wxjiao,lyu,king}@cse.cuhk.edu.hk

Abstract

Emotion Recognition in Conversations (ERC) aims to predict the emotional state of speakers in conversations, which is essentially a text classification task. Unlike the sentence-level text classification problem, the available supervised data for the ERC task is limited, which potentially prevents the models from playing their maximum effect. In this paper, we propose a novel approach to leverage unsupervised conversation data, which is more accessible. Specifically, we propose the Conversation Completion (ConvCom) task, which attempts to select the correct answer from candidate answers to fill a masked utterance in a conversation. Then, we Pre-train a basic COntext-Dependent Encoder (PRE-CODE) on the ConvCom task. Finally, we fine-tune the PRE-CODE on the datasets of ERC. Experimental results demonstrate that pre-training on unsupervised data achieves significant improvement of performance on the ERC datasets, particularly on the minority emotion classes.[1]

1 Introduction

Emotion recognition in conversations (ERC) has garnered attention recently (Poria et al., 2019), due to its potential in developing practical chatting machines (Zhou et al., 2018a). Unlike traditional text classification that handles context-free sentences, ERC aims to predict the emotional state of each utterance in a conversation (Figure 1). The inherent hierarchical structure of a conversation, i.e., words-to-utterance and utterances-to-conversation, determines that the ERC task should be better addressed by context-dependent models (Poria et al., 2017; Hazarika et al., 2018b; Jiao et al., 2019, 2020).

Despite the remarkable success, context-dependent models suffer from the data scarcity issue. In the ERC task, annotators are required to recognize either obvious or subtle difference between emotions, and tag the instance with a specific emotion label, such that supervised data with human annotations are very costly to collect. In addition, existing datasets for ERC (Busso et al., 2008; Hsu and Ku, 2018; Zahiri and Choi, 2018; Zadeh et al., 2018) contain inadequate conversations, which prevent the context-dependent models from playing their maximum effect.

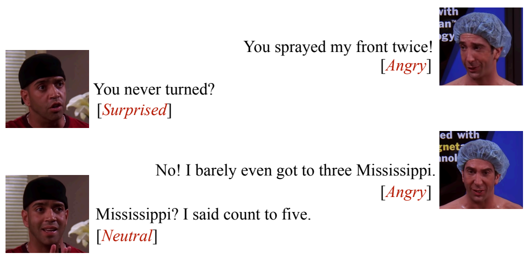
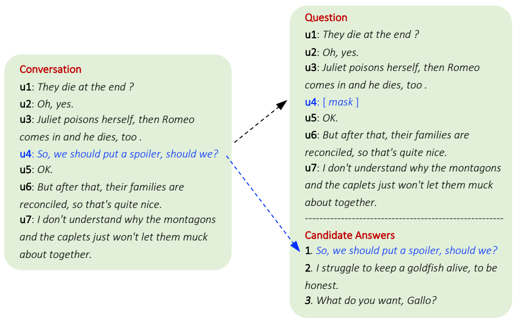

You sprayed my front twice!
[*Angry*]
You never turned?
[*Surprised*]
No! I barely even got to three Mississippi.
[*Angry*]
Mississippi? I said count to five.
[*Neutral*]

Figure 1: A conversation example with emotion labels.

In this paper, we aim to tackle the data scarcity issue of ERC by exploiting the unsupervised data. Specifically, we propose the Conversation Completion (ConvCom) task based on unsupervised conversation data, which attempts to select the correct answer from candidate answers to fill a masked utterance in a conversation. Then, on the proposed ConvCom task, we Pre-train a basic COntext-Dependent Encoder (PRE-CODE). The hierarchical structure of the context-dependent encoder makes our work different from those that focus on universal sentence encoders (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019). Finally, we fine-tune the PRE-CODE on *five* datasets of the ERC task. Experimental results show that the fine-tuned PRE-CODE achieves significant improvement of performance over the baselines, particularly on minority emotion classes, demonstrating the effectiveness of our approach.

[1]The source code is available at https://github.com/wxjiao/Pre-CODE



Conversation

u1: *They die at the end ?*

u2: *Oh, yes.*

u3: *Juliet poisons herself, then Romeo comes in and he dies, too .*

u4: *So, we should put a spoiler, should we?*

u5: *OK.*

u6: *But after that, their families are reconciled, so that's quite nice.*

u7: *I don't understand why the montagons and the caplets just won't let them muck about together.*

Question

u1: *They die at the end ?*

u2: *Oh, yes.*

u3: *Juliet poisons herself, then Romeo comes in and he dies, too .*

u4: [*mask*]

u5: *OK.*

u6: *But after that, their families are reconciled, so that's quite nice.*

u7: *I don't understand why the montagons and the caplets just won't let them muck about together.*

Candidate Answers

1. *So, we should put a spoiler, should we?*
2. *I struggle to keep a goldfish alive, to be honest.*
3. *What do you want, Gallo?*

Figure 2: A data example in the ConvCom task.

Our contributions of this work are as follows: (1) We propose the conversation completion task for the context-dependent encoder to learn from unsupervised conversation data. (2) We fine-tune the pre-trained context-dependent encoder on the datasets of ERC and achieve significant improvement of performance over the baselines.

2 Pre-training Strategy

2.1 Approach

ConvCom Task. We exploit the self-supervision signal in conversations to construct our pre-training task. Formally, given a conversation, $\mathcal{U} = \{u_1, u_2, \cdots, u_L\}$, we mask a target utterance u_l as $\mathcal{U} \backslash u_l = \{\cdots, u_{l-1}, [mask], u_{l+1}, \cdots\}$ to create a question, and try to retrieve the correct utterance u_l from the whole training corpus. The choice of filling the mask involves countless possible utterances, making it infeasible to formulate the task into a multi-label classification task with softmax. We instead simplify the task into a response selection task (Tong et al., 2017) using negative sampling (Mikolov et al., 2013), which is a variant of noise-contrastive estimation (NCE, Gutmann and Hyvärinen, 2010). To achieve so, we sample $N-1$ noise utterances elsewhere, along with the target utterance, to form a set of N candidate answers. Then the goal is to select the correct answer, i.e., u_l, from the candidate answers to fill the mask, conditioned on the context utterances. We term this task "Conversation Completion", abbreviated as ConvCom. Figure 2 shows an example, where the utterance `u4` is masked out from the original conversation and the candidate answers include `u4` and `two` noise utterances.

Context-Dependent Encoder. The context-dependent encoder consists of two parts: an utterance encoder, and a conversation encoder.

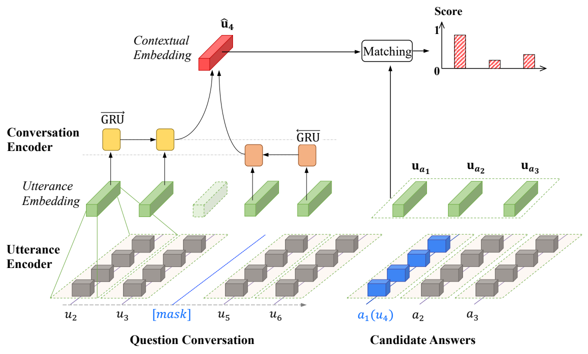


Figure 3: The architecture of the context-dependent encoder with the pre-training objective.

Each utterance is represented by a sequence of word vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$, initialized by the 300-dimensional pre-trained GloVe word vectors[2] (Pennington et al., 2014).

For the utterance encoder, we adopt a BiGRU to read the word vectors of an utterance, and produce the hidden state $\overleftrightarrow{\mathbf{h}}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \in \mathbb{R}^{2d_u}$. We apply *max-pooling* and *mean-pooling* on the hidden states of all words. The pooling results are summed up, followed by a fully-connected layer, to obtain the embedding of the utterance termed $\mathbf{u}_l$:

$$\mathbf{h}_l = \max(\{\overleftrightarrow{\mathbf{h}}_t\}_{t=1}^{T}) + \text{mean}(\{\overleftrightarrow{\mathbf{h}}_t\}_{t=1}^{T}), \quad (1)$$

$$\mathbf{u}_l = \tanh(\mathbf{W}_u \cdot \mathbf{h}_l + \mathbf{b}_u), l \in [1, L], \quad (2)$$

where T denotes the length of the utterance and L is the number of utterances in the conversation.

For the conversation encoder, since an utterance could express different meanings in different contexts, we adopt another BiGRU to model the utterance sequence of a conversation to capture the relationship between utterances. The produced hidden states are termed $\overrightarrow{\mathbf{H}}_l, \overleftarrow{\mathbf{H}}_l \in \mathbb{R}^{d_c}$.

Pre-training Objective. To train the context-dependent encoder on the proposed ConvCom task, we construct a contextual embedding for each masked utterance by combining its context from the history $\overrightarrow{\mathbf{H}}_{l-1}$ and the future $\overleftarrow{\mathbf{H}}_{l+1}$ (see Figure 3):

$$\hat{\mathbf{u}}_l = \tanh(\mathbf{W}_c \cdot [\overrightarrow{\mathbf{H}}_{l-1}; \overleftarrow{\mathbf{H}}_{l+1}] + \mathbf{b}_c). \quad (3)$$

Then, the contextual embedding $\hat{\mathbf{u}}_l$ is matched to the candidate answers to find the most suitable one to fill the mask. To compute the matching score, we adopt dot-product with a sigmoid function as:

$$s(\hat{\mathbf{u}}_l, \mathbf{u}_{a_n}) = \sigma(\hat{\mathbf{u}}_l^\top \mathbf{u}_{a_n}), n \in [1, N], \quad (4)$$

[2] `https://nlp.stanford.edu/projects/glove/`

Model	d_u/d_c	$\mathbf{R}_5@1$	$\mathbf{R}_5@2$	$\mathbf{R}_{11}@1$	$\mathbf{R}_{11}@2$
SMALL	150	70.8	88.0	56.2	72.7
MID	300	73.8	89.7	60.4	76.4
LARGE	450	77.2	91.3	64.2	79.1

Table 1: Test results of CODE on the ConvCom task in three capacities.

where $\sigma(x) = \frac{1}{(1+\exp(-x))} \in (0,1)$ is the sigmoid function, and $\mathbf{u}_{a_n}$ is the embedding of the nth candidate answer. The goal is to maximize the score of the target utterance and minimize the score of the noise utterances. Thus the loss function becomes:

$$\mathcal{F} = -\sum_l \left[\log\sigma(\hat{\mathbf{u}}_l^\top \mathbf{u}_{a_1}) + \sum_{n=2}^{N} \log\sigma(-\hat{\mathbf{u}}_l^\top \mathbf{u}_{a_n}) \right], \quad (5)$$

where a_1 corresponds to the target utterance, and the summation goes over each utterance of all the conversations in the training set.

2.2 Experiment

Dataset. Our unsupervised conversation data comes from an open-source database OpenSubtitle[3] (Lison and Tiedemann, 2016), which contains a large amount of subtitles of movies and TV shows. Specifically, we retrieve the English subtitles throughout the year of 2016, and collect 25,466 html files. After pre-processing, we obtain 58,360, 3,186, 3,297 conversations for the training, validation, and test sets, respectively.

Evaluation. To evaluate the pre-trained model, we adopt the evaluation metric:

$$\mathbf{R}_{N'}@k = \frac{\sum_{i=1}^{k} y_i}{\sum_{i=1}^{N'} y_i}, \quad (6)$$

which is the recall of the true positives among k best-matched answers from N' available candidates for the given contextual embedding $\hat{\mathbf{u}}_k$ (Zhou et al., 2018b). The variate y_i represents the binary label for each candidate, i.e., 1 for the target one and 0 for the noise ones. Here, we report $\mathbf{R}_5@1$, $\mathbf{R}_5@2$, $\mathbf{R}_{11}@1$, and $\mathbf{R}_{11}@2$.

Results. For simplicity, we term the context-dependent encoder as CODE. We train CODE on the created dataset in three different capacities, namely, SMALL, MID, and LARGE, corresponding to different hidden sizes of the BiGRUs. See Appendix A.2 for the training details.

[3] http://opus.nlpl.eu/OpenSubtitles-v2018.php

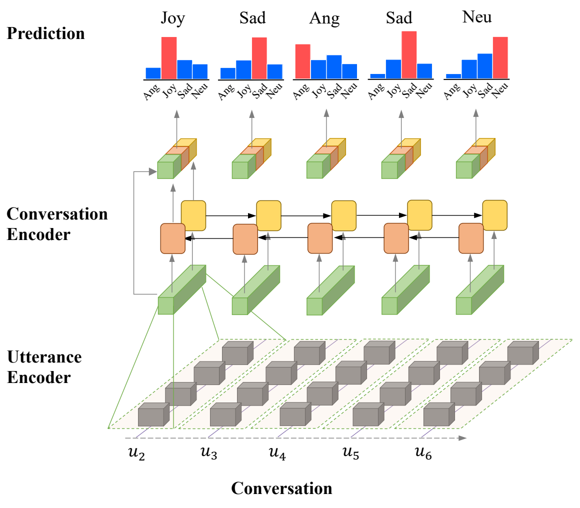


Figure 4: The architecture for the ERC task. Both the utterance encoder and conversation encoder are transferred from the PRE-CODE.

Table 1 lists the results on the test set. For the SMALL CODE, it is able to select the correct answer for 70.8% instances with 5 candidate answers and 56.2% with 11 candidates. The accuracy is considerably higher than random guesses, i.e., 1/5 and 1/11, respectively. By increasing the model capacity to MID and LARGE, we further improve the recalls by several points successively. These results demonstrate that CODE is indeed able to capture the structure of conversations and perform well in the proposed ConvCom task.

3 Fine-tuning Strategy

3.1 Experimental Setup

ERC Architecture. To transfer the pre-trained CODE models, termed PRE-CODE, to the ERC task, we only need to add a fully-connected (FC) layer followed by a softmax function to form the new architecture. Figure 4 shows the resulting architecture, in which we also concatenate the context-independent utterance embeddings to the contextual ones before fed to the FC.

We adopt a weighted categorical cross-entropy loss function to optimize the model parameters:

$$\mathcal{L} = -\frac{1}{\sum_{i=1}^{N} L_i} \sum_{i=1}^{N} \sum_{j=1}^{L_i} \omega(c_j) \sum_{c=1}^{|\mathcal{C}|} \mathbf{o}_j^c \log_2(\hat{\mathbf{o}}_j^c), \quad (7)$$

where $|\mathcal{C}|$ is the number of emotion classes, $\mathbf{o}_j$ is the one-hot vector of the true label, and $\hat{\mathbf{o}}_j$ is the softmax output. The weight $\omega(c)$ is inversely proportional to the ratio of class c in the training set with a power rate of 0.5.

Model	IEMOCAP		EmoryNLP		MOSEI*	
	F1	WA	F1	WA	F1	WA
bcLSTM[1]	–	73.6	–	–	–	–
CMN[2]	–	74.1	–	–	–	–
SCNN[3]	–	–	26.9	**37.9**	–	–
HiGRU-sf[4]	–	82.1	–	–	–	–
bcLSTM	76.6	77.1	25.5	33.5	29.1	56.3
bcGRU	77.6	78.2	26.1	33.1	28.7	56.4
CODE-MID	78.6	79.6	26.7	34.7	29.7	56.6
PRE-CODE	**81.5**	**82.9**	**29.1**	36.1	**31.7**	**57.1**

[1]Poria et al. (2017); [2]Hazarika et al. (2018b); [3]Zahiri and Choi (2018); [4]Jiao et al. (2019).

Table 2: Test results on IEMOCAP, EmoryNLP, and MOSEI*. The implemented bcLSTM performs much better than the original one, possibly because that the original bcLSTM is not trained end-to-end.

Model	Friends		EmotionPush	
	F1	WA	F1	WA
CNN-DCNN[1]	–	67.0	–	75.7
SA-BiLSTM[2]	–	79.8	–	**87.7**
HiGRU[3]	–	74.4	–	73.8
bcLSTM	63.1	79.9	60.3	84.8
bcGRU	62.4	77.6	60.5	84.6
CODE-MID	62.4	78.0	60.3	84.2
PRE-CODE	**65.9**	**81.3**	**62.6**	84.7

[1]Khosla (2018); [2]Luo et al. (2018); [3]Jiao et al. (2019).

Table 3: Test results on Friends and EmotionPush.

Compared Methods. We mainly compare our PRE-CODE with bcLSTM (Poria et al., 2017), CMN (Hazarika et al., 2018b), SA-BiLSTM (Luo et al., 2018), CNN-DCNN (Khosla, 2018), SCNN (Zahiri and Choi, 2018), HiGRU (Jiao et al., 2019), and the following: (1) bcLSTM$_\ddagger$: bcLSTM re-implemented by us following Jiao et al. (2019); (2) bcGRU: A variant of bcLSTM$_\ddagger$ implemented with BiGRUs; (3) CODE without pre-training. Unless otherwise stated, CODE and PRE-CODE are both in the capacity of MID.

ERC Datasets. We conduct experiments on five ERC datasets for the ERC task, namely, IEMOCAP (Busso et al., 2008), Friends (Hsu et al., 2018), EmotionPush (Hsu et al., 2018), EmoryNLP (Zahiri and Choi, 2018), and MOSEI (Zadeh et al., 2018). For MOSEI, we pre-process it to adapt to the ERC task and name the pre-processed dataset as MOSEI* here. See Appendix A.3 for details of the ERC datasets.

Evaluation. To evaluate the performance of our models, we report the macro-averaged F1-score (Zahiri and Choi, 2018) and the weighted

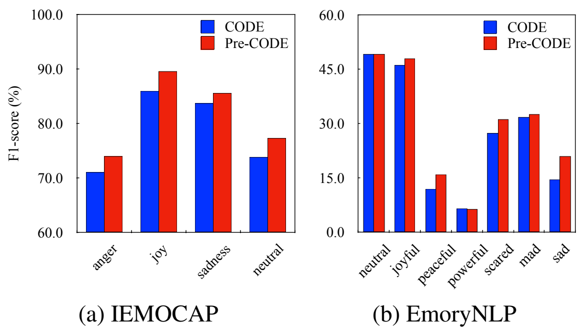


Figure 5: F1-score of emotion classes on IEMOCAP and EmoryNLP.

accuracy (WA) (Hsu and Ku, 2018) of all emotion classes. The F1-score of each emotion class is also presented for discussion.

Results. We train the implemented baselines and fine-tune the PRE-CODE on the five datasets. Each result is the average of 5 repeated experiments. See Appendix A.3 for training details.

We report the main results in Table 2 and Table 3. As seen, our PRE-CODE outperforms the compared methods on all datasets in terms of F1-score by at least 2.0% absolute improvement. We also conduct significance tests by using two-tailed paired t-tests over the F-1 scores of PRE-CODE and CODE-MID. P-values are obtained as 0.0107, 0.0038, 0.0011, 0.0003, and 0.0068 for IEMOCAP, EmoryNLP, MOSEI*, Friends, and EmotionPush, respectively. Therefore, the result for IEMOCAP is statistically significant with a significance level of 0.05 whereas the other four datasets obtain a significance level of 0.01. It demonstrates the effectiveness of transferring the knowledge from unsupervised conversation data to the ERC task.

To inspect which aspects pre-training helps the most, we present the F1-score of each emotion class on IEMOCAP and EmoryNLP in Figure 5. As seen, our PRE-CODE particularly improves the performance on minority emotion classes, e.g., *anger* and *sadness* in IEMOCAP, and *peaceful* and *sad* in EmoryNLP. These results demonstrate that pre-training can ameliorate the issue of imbalanced performance on minority classes while maintaining good performance on majority classes.

3.2 Discussion

Model Capacity. We investigate how the model performance is affected by the number of parameters, as seen in Table 4. We find that: (1)

Model	Capacity	IEMOCAP	Friends
CODE	SMALL	76.5	**62.5**
	MID	**78.6**	62.4
	LARGE	77.6	62.1
PRE-CODE	SMALL	81.2	65.2
	MID	**81.5**	**65.9**
	LARGE	80.3	64.8

Table 4: Ablation study on model capacity.

Layers	IEMOCAP	Friends
PRE-CODE + Re-W	**81.6**	64.5
PRE-CODE	81.5	**65.9**
CODE + Pre-U	80.1	64.8
CODE	78.6	62.4

Table 5: Ablation study on pre-trained layers.

Speaker	Utterance	Truth	CODE	PRE-CODE
Example 1				
Joey	Come on, Lydia, you can do it.	Neu	Neu	Neu
Joey	Push!	Joy	Ang	Ang
Joey	Push 'em out, push 'em out, harder, harder.	Joy	Neu	Neu
Joey	Push 'em out, push 'em out, way out!	Joy	Ang	Joy
Joey	Let's get that ball and really move, hey, hey, ho, ho.	Joy	Neu	Joy
Joey	Let's... I was just... yeah, right.	Joy	Neu	Neu
Joey	Push!	Joy	Ang	Ang
Joey	Push!	Joy	Ang	Ang
Example 2				
Sp1	It's so hard not to cry	Sad	Ang	Sad
Sp2	What happened	Neu	Neu	Neu
Sp1	I lost another 3 set game	Sad	Neu	Sad
Sp2	It's ok person_145	Neu	Neu	Neu
Sp1	Why does it hurt so much	Sad	Neu	Sad
Sp2	Everybody loses	Neu	Neu	Neu

Table 6: Qualitative comparison between CODE and PRE-CODE by two examples.

PRE-CODE consistently outperforms CODE in all cases, suggesting that pre-training is an effective method to boost the model performance of ERC regardless of the model capacity. (2) PRE-CODE shows better performance in the capacities of SMALL and MID, we speculate that the datasets for ERC are so scarce that they are incapable of transferring the pre-trained parameters of the LARGE PRE-CODE to optimal ones for ERC.

Layer Effect. We study how different pre-trained layers affect the model performance, as seen in Table 5. CODE+Pre-U denotes that only the parameters of utterance encoder are initialized by PRE-CODE. From CODE to CODE+Pre-U and then to PRE-CODE, we conclude that pre-training results in better utterance embeddings and helps the model to capture the utterance-level context more effectively. In addition, PRE-CODE+Re-W represents that we re-train PRE-CODE for 10 more epochs to adjust the originally fixed word embeddings. The results suggest that pre-training word embeddings does not improve the model performance necessarily but may corrupt the learned utterance and conversation encoders.

Qualitative Study. In Table 6, we provide two examples for a comparison between CODE and PRE-CODE. The first example is from Friends with consecutive utterances from Joey. It shows that CODE tends to recognize the utterances with exclamation marks "!" as Angry, while those with periods "." as Neutral. The problem also appears on PRE-CODE for short utterances, e.g., "Push!", which contains little and misleading information. This issue might be alleviated by adding other features like audio and video. Still, PRE-CODE performs better than CODE on longer utterances. The other example is from EmotionPush, which are messages with few punctuations. The CODE model predicts almost all utterances as Neutral, which may be because most of the training utterances are Neutral. However, PRE-CODE can identify the minor classes, e.g., Sad, demonstrating that pre-training can alleviate the class imbalance issue.

4 Conclusion

In this work, we propose a novel approach to leverage unsupervised conversation data to benefit the ERC task. The proposed conversation completion task is effective for the pre-training of the context-dependent model, which is further fine-tuned to boost the performance of ERC significantly. Future directions include exploring advanced models (e.g., TRANSFORMER) for pre-training, conducting domain matching for the unsupervised data, as well as multi-task learning to alleviate the possible catastrophic forgetting issue in transfer learning.

Acknowledgments

This work is partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14210717, General Research Fund; CUHK 2410021, Research Impact Fund, No. R5034-18). We thank Xing Wang and the anonymous reviewers for their insightful suggestions on various aspects of this work.

References

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth Narayanan. 2008. IEMOCAP: interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42(4):335–359.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304.

Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria, and Roger Zimmermann. 2018a. ICON: interactive conversational memory network for multimodal emotion detection. In *EMNLP*, pages 2594–2604.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018b. Conversational memory network for emotion recognition in dyadic dialogue videos. In *NAACL-HLT*, pages 2122–2132.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339.

Chao-Chun Hsu, Sheng-Yeh Chen, Chuan-Chun Kuo, Ting-Hao K. Huang, and Lun-Wei Ku. 2018. Emotionlines: An emotion corpus of multi-party conversations. In *LREC*.

Chao-Chun Hsu and Lun-Wei Ku. 2018. Socialnlp 2018 emotionx challenge overview: Recognizing emotions in dialogues. In *SocialNLP@ACL 2018*, pages 27–31.

Wenxiang Jiao, Michael R. Lyu, and Irwin King. 2020. Real-time emotion recognition via attention gated hierarchical memory network. In *AAAI*, pages 8002–8009.

Wenxiang Jiao, Haiqin Yang, Irwin King, and Michael R. Lyu. 2019. Higru: Hierarchical gated recurrent units for utterance-level emotion recognition. In *NAACL-HLT*, pages 397–406.

Sopan Khosla. 2018. Emotionx-ar: CNN-DCNN autoencoder based emotion classifier. In *SocialNLP@ACL 2018*, pages 37–44.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *LREC*.

Linkai Luo, Haiqing Yang, and Francis Y. L. Chin. 2018. Emotionx-dlc: Self-attentive bilstm for detecting sequential emotions in dialogues. In *SocialNLP@ACL 2018*, pages 32–36.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*, pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *ACL*, pages 873–883.

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard H. Hovy. 2019. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *IEEE Access*, 7:100943–100953.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating human trafficking with multimodal deep models. In *ACL*, pages 1547–1556.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018. Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph. In *ACL*, pages 2236–2246.

Sayyed M. Zahiri and Jinho D. Choi. 2018. Emotion detection on TV show transcripts with sequence-based convolutional neural networks. In *AAAI*, pages 44–52.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*, pages 730–739.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018b. Multi-turn response selection for chatbots with deep attention matching network. In *ACL*, pages 1118–1127.

A Appendix

A.1 Related Work

Pre-training on unsupervised data has been an active area of research for decades. Mikolov et al. (2013) and Pennington et al. (2014) lead the heat on learning dense word embeddings over raw text for downstream tasks. Melamud et al. (2016) propose to learn word embeddings in the context with the use of LSTM, which is able to eliminate word-sense ambiguity. More recently, ELMo (Peters et al., 2018) extracts context-sensitive features through a language model and integrates the features into task-specific architectures, achieving state-of-the-art results on several major NLP tasks. Unlike these feature-based approaches, another trend is to pre-train some architecture through a language model objective, and then fine-tune the architecture for supervised downstream tasks (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019). With trainable parameters, this kind of approaches are more flexible, attaining better performance than their feature-based counterparts.

However, the idea of pre-training a context-dependent encoder using unsupervised conversation data for the ERC task has never been explored. On one hand, existing works on ERC focus on modeling the speakers, context, and emotion evolution (Poria et al., 2017; Hazarika et al., 2018a,b; Jiao et al., 2019, 2020). No prior work has tried to solve the issue of data scarcity. On the other hand, existing works on transfer learning focus on pre-training universal sentence encoders, e.g., ELMo, GPT, and BERT. But our PRE-CODE, beyond sentence level, is dedicated for sentence sequences from conversations or speeches. As a result, the pre-training task needs to be customized, for which we propose the ConvCom task. Partially inspired by Word2vec (Mikolov et al., 2013) and response selection task (Tong et al., 2017), our ConvCom task differs in that it should model the order of context meanwhile both historical and future context are provided. In contrast, Word2vec neglects the order of context words, and response selection task usually provides only historical context.

A.2 Pre-training Strategy

Dataset Creation. Our unlabeled conversation data comes from an open-source database named OpenSubtitle[4] (Lison and Tiedemann, 2016), which contains a large amount of subtitles of movies and TV shows. Specifically, We retrieve the English subtitles throughout the year of 2016, including 25466 `.html` files. We extract the text subtitles from all the `.html` files and pre-process them as below:

- For each episode, we remove the first and the last *ten* utterances in case they are instructions but conversations, especially in TV shows;
- We split the conversations in each episode randomly into shorter ones with *five* to *one hundred* utterances, following a uniform distribution;
- A short conversation is removed if over half of its utterances contain less than *eight* words each. This is done to force the conversation to capture more information;
- All the short conversations are randomly split into a training set, a validation set, and a test set, following the ratio of 90:5:5.

Set	#Conversation	Avg. #Utterance	Avg. #Word
Train	58360	41.3	10.1
Val	3186	41.0	10.1
Test	3297	40.8	10.1

Table 7: Statistics of the created datasets for the ConvCom task.

Table 7 lists the statistics of resulting sets, where #Conversation denotes the number of conversations in a set, Avg. #Utternace is the average number of utterances in a conversation, and Avg. #Word is the average number of tokens in an utterance. Totally, there are over 2 million of utterances in over 60k conversations, which is at least 100 times more than those datasets for ERC (see Table 8).

Noise Utterances. We randomly sample *ten* noise utterances for each utterance in the training set, validation set, and test set. In each set, a conversation shares the *ten* noise utterances sampled from elsewhere within the set. During training, we can either use the pre-selected noise utterances or sample an arbitrary number of noise utterances dynamically. We use the validation set to choose model parameters, and evaluate the model performance on the test set.

[4]`http://opus.nlpl.eu/OpenSubtitles-v2018.php`

Model	#Conversation			#Utterance		
	Train	Val	Test	Train	Val	Test
IEMOCAP	96	24	31	3,569	721	1,208
Friends	720	80	200	10,561	1,178	2,764
EmotionPush	720	80	200	10,733	1,202	2,807
EmoryNLP	713	99	85	9,934	1,344	1,328
MOSEI*	2,250	300	676	16,331	1,871	4,662

Table 8: Statistics of the datasets for ERC.

Training Details. We choose Adam (Kingma and Ba, 2015) as the optimizer with an initial learning rate of 2×10^{-4}, which is decayed with a rate of 0.75 once the validation recall $\mathbf{R}_{11}@1$ stops increasing. We use a dropout rate of 0.5 for the utterance encoder and the conversation encoder, respectively. Gradient clipping with a norm of 5 is also applied to avoid gradient explosion. Each conversation in the training set is regarded as a batch, where each utterance plays the role of target utterance by turns. We randomly sample 10 noise utterances for each conversation during training and validate the model every epoch. The CODE is pre-trained for at most 20 epochs, and early stopping with a patience of 3 is adopted to choose the optimal parameters. Note that, we fix the word embedding layer during pre-training to focus on the utterance encoder and the conversation encoder.

A.3 Fine-tuning Strategy

ERC Datasets. Our PRE-CODE and the implemented baselines are fine-tuned on five ERC datasets, namely, IEMOCAP[5] (Busso et al., 2008), Friends[6] (Hsu et al., 2018), EmotionPush[7] (Hsu et al., 2018), EmoryNLP[8] (Zahiri and Choi, 2018), and MOSEI[9] (Zadeh et al., 2018). For MOSEI, we pre-process it to adapt to the ERC task and name the pre-processed dataset as MOSEI* here. Specifically, we utilize the raw transcripts of MOSEI, where over 14k utterances are not annotated, and others are labeled with one or more emotion labels. For the unlabeled utterances, we just remove them from the dataset. For the utterance with more than one emotion label, we determine its primary emotion by the majority vote or the highest emotion intensity sum if there are more than one majority votes. For the utterances that obtain zero vote for all emotion classes, we annotate them as *other*.

For the first three datasets, we follow previous work (Poria et al., 2017; Hsu et al., 2018) to consider only four emotion classes, i.e., *anger*, *joy*, *sadness*, and *neutral*. We consider all the emotion classes for EmoryNLP as in (Zahiri and Choi, 2018) and six emotion classes (without *neutral*) for MOSEI*. All the datasets contain the training set, validation set, and test set, except for IEMOCAP. So, we follow (Poria et al., 2017) to use the first four sessions of transcripts as the training set, and the last one as the test set. The validation set is extracted from the randomly-shuffled training set with the ratio of 80:20. We present the statistic details of datasets in Table 8.

Training Details. We still choose Adam as the optimizer and tune the learning rate for the implemented baselines. Generally, the learning rate of 2×10^{-4} works well for all the datasets except MOSEI*, on which we find 5×10^{-5} works better. For the fine-tuning of PRE-CODE, we use the learning rate of the baselines or its half and report the better results here. We monitor the macro-averaged F1-score of validation set and decay the learning rate once the F1-score stops increasing. The decay rate and patience of early stopping are 0.75 and 6 for all the datasets except IEMOCAP. Since IEMOCAP has much fewer conversations, we change the decay rate and patience of early stopping to 0.95 and 10, respectively.

[5]https://sail.usc.edu/iemocap/
[6]http://doraemon.iis.sinica.edu.tw/emotionlines
[7]http://doraemon.iis.sinica.edu.tw/emotionlines
[8]https://github.com/emorynlp/emotion-detection/
[9]http://immortal.multicomp.cs.cmu.edu/raw_datasets/